开始之前先简单说明下cmd文件和bat文件的区别：在本质上两者没有区别，都是简单的文本编码方式，都可以用记事本创建、编辑和查看。两者所用的命令行代码也是共用的，只是cmd文件中允许使用的命令要比bat文件多。cmd文件只有在windows2000以上的系统中才能运行，而bat文件则没有这个限制。从它们的文件描述中也可以看出以上的区别：cmd文件的描述是“windows nt命令脚本”， bat文件的描述是“ms dos批处理文件”

    如果没有一定的相关知识恐怕不容易看懂和理解批处理文件，也就更谈不上自己动手编写了.批处理文件是无格式的文本文件，它包含一条或多条命令。它的文件扩展名为 .bat 或 .cmd。在命令提示下键入批处理文件的名称，或者双击该批处理文件，系统就会调用cmd.exe按照该文件中各个命令出现的顺序来逐个运行它们。使用批处理文件（也被称为批处理程序或脚本），可以简化日常或重复性任务。当然我们的这个版本的主要内容是介绍批处理在入侵中一些实际运用，例如我们后面要提到的用批处理文件来给系统打补丁、批量植入后门程序等。下面就开始我们批处理学习之旅吧。

1.  简单批处理内部命令简介

1)   echo 命令

       打开回显或关闭请求回显功能，或显示消息。如果没有任何参数，echo 命令将显示当前回显设置。  
语法  
echo [{ on|off }] [message]  
Sample 1 ：

       @echooff

       echohello world

Sample 2 ：

@echo　off

echo 1

echo.

echo 2

echo on

echo 3

pause

在实际应用中我们会把这条命令和重定向符号（也称为管道符号，一般用> >> ^）结合来实现输入一些命令到特定格式的文件中.这将在以后的例子中体现出来。

c:\>dir \*.txt > 1.txt

c:\>dir \*.txt >> 1.txt

2)   @ 命令

    表示不显示@后面的命令，在入侵过程中（例如使用批处理来格式化敌人的硬盘）自然不能让对方看到你使用的命令啦。  
Sample：@echo off

3)   goto 命令

指定跳转到标签，找到标签后，程序将处理从下一行开始的命令。  
语法：goto label （label是参数，指定所要转向的批处理程序中的行。）  
Sample：  
if { %1 }=={ } goto noparms  
if { %2 }=={ } goto noparms（如果这里的if、%1、%2你不明白的话，先跳过去，后面会有详细的解释。）  
@Rem check parameters if null show usage  
:noparms  
echo Usage: monitor.bat ServerIP PortNumber  
goto end  
标签的名字可以随便起，但是最好是有意义的字母啦，字母前加个：用来表示这个字母是标签，goto命令就是根据这个：来寻找下一步跳到到那里。最好有一些说明这样你别人看起来才会理解你的意图啊。

4)   rem 命令

注释命令，在C语言中相当与/\*--------\*/,它并不会被执行，只是起一个注释的作用，便于别人阅读和你自己日后修改。  
Rem Message  
Sample：@Rem Here is the description.

5) pause 命令

运行 Pause 命令时，将显示下面的消息：  
Press any key to continue . . .  
Sample：  
@echo off  
:begin  
copy a:\*.\* d：\\back  
echo Please put a new disk into driver A  
pause  
goto begin  
在这个例子中，驱动器 A 中磁盘上的所有文件均复制到d:\\back中。显示的注释提示您将另一张磁盘放入驱动器 A 时，pause 命令会使程序挂起，以便您更换磁盘，然后按任意键继续处理。

6) call 命令

从一个批处理程序调用另一个批处理程序，并且不终止父批处理程序。call 命令接受用作调用目标的标签。如果在脚本或批处理文件外使用 call，它将不会在命令行起作用

语法  
call [[Drive:][Path] FileName [BatchParameters]] [:label [arguments]]  
参数  
[Drive: }[Path] FileName  
指定要调用的批处理程序的位置和名称。filename 参数必须具有 .bat 或 .cmd 扩展名。

7) start 命令

Start

启动单独的“命令提示符”窗口来运行指定程序或命令。如果在没有参数的情况下使用，start 将打开第二个命令提示符窗口。

语法

start ["title"] [/dPath] [/min] [/max] [{/separate |/shared}] [{/low | /normal | /high | /realtime | /abovenormal | belownormal}][/wait] [/B] [FileName] [parameters]

参数

"title" 指定在“命令提示符”窗口标题栏中显示的标题。

/dpatch 指定启动目录。

/i 将 Cmd.exe 启动环境传送到新的“命令提示符”窗口。

/min 启动新的最小化窗口。

/max 启动新的最大化窗口。

/separate 在单独的内存空间启动 16 位程序。

/shared 在共享的内存空间启动 16 位程序。

/low 以空闲优先级启动应用程序。

/normal 以一般优先级启动应用程序。

/high 以高优先级启动应用程序。

/realtime 以实时优先级启动应用程序。

/abovenormal 以超出常规优先级的方式启动应用程序。

/belownormal 以低出常规优先级的方式启动应用程序。

/wait 启动应用程序，并等待其结束。

/b 启动应用程序时不必打开新的“命令提示符”窗口。除非应用程序启用 CTRL+C，否则将忽略 CTRL+C 操作。使用 CTRL+BREAK 中断应用程序。

非执行文件只要将文件名作为命令键入，即可通过其文件关联运行该文件。有关使用 assoc 和 ftype 在命令脚本中创建这些关联的详细信息，请参阅“”。

在运行的命令的第一个标记为“CMD”字符串但不包括扩展名或路径限定符时，“CMD”将被 COMSPEC 变量的值取代。这样可以防止用户从当前目录选取 cmd。

当您运行 32 位图形用户界面 (GUI) 应用程序时，cmd 不会在返回到命令提示符之前等待应用程序退出。如果从命令脚本运行应用程序，则不会发生这种新情况。在运行的命令中第一个符号不包括扩展名的情况下，Cmd.exe 使用 PATHEXT 环境变量的值确定要查找的扩展名以及查找顺序。PATHEXT 变量的默认值为：COM;.EXE;.BAT;.CMD（语法与 PATH 变量相同，使用分号分开不同元素）。当您搜索可执行文件且在任何扩展名上都没有匹配项时，start 将搜索目录名。

具体例子：

说明：如果你所在程序的路径中带有空格，那么必须用“”把路径括起来，否则系统会提示找不到XX文件，另外，在运行某些程序时，需在路径的前面加一对空白的“”，表示创建一个空白的窗口，它指向的程序是XXXXXXXX。还有就是别忘了空格。

当我想运行位于“D:\draw\”的“photoshop.exe”使，应该使用以下命令：

start “”“D:\draw\photoshop.exe” 表示以常规窗口运行程序

如果想让程序以最大化窗口运行，则使用以下命令：

start /max“”“D:\draw\photoshop.exe” 表示以最大化窗口运行程序

最小化这是这样：

start /min "" "D:\draw\photoshop.exe" 表示以最小化窗口运行程序

等待某个程序允许完毕，也就是窗口关闭后，再打开下一个程序这可以这样：

start /w "" "D:\draw\photoshop.exe"

start "" cmd.exe

start /min “” “e:\t.cmd”

8) choice 命令

choice 使用此命令可以让用户输入一个字符，从而运行不同的命令。使用时应该加/c:参数，c:后应写提示可输入的字符，之间无空格。它的返回码为1234……  
如: choice /c:dme defrag,mem,end  
将显示  
defrag,mem,end[D,M,E]?  
Sample：  
Sample.bat的内容如下:  
@echo off  
choice /c:dme defrag,mem,end  
if errorlevel 3 goto defrag （应先判断数值最高的错误码）  
if errorlevel 2 goto mem  
if errotlevel 1 goto end

:defrag  
c:\\dos\\defrag  
goto end  
:mem  
mem  
goto end  
:end  
echo good bye

此文件运行后，将显示 defrag,mem,end[D,M,E]? 用户可选择d m e ，然后if语句将作出判断，d表示执行标号为defrag的程序段，m表示执行标号为mem的程序段，e表示执行标号为end的程序段，每个程序段最后都以goto end将程序跳到end标号处，然后程序将显示good bye，文件结束。

9) If 命令

if 表示将判断是否符合规定的条件，从而决定执行不同的命令。 有三种格式:  
1、if "参数" == "字符串" 　待执行的命令  
参数如果等于指定的字符串，则条件成立，运行命令，否则运行下一句。(注意是两个等号）  
如if "%1"=="a" format a:  
if { %1 }=={ } goto noparms  
if { %2 }=={ } goto noparms

2、if exist 文件名　 待执行的命令  
如果有指定的文件，则条件成立，运行命令，否则运行下一句

如if exist config.sys edit config.sys

3、if errorlevel / if not errorlevel 数字　 待执行的命令  
如果返回码等于指定的数字，则条件成立，运行命令，否则运行下一句。  
如if errorlevel 2 goto x2 　  
DOS程序运行时都会返回一个数字给DOS，称为错误码errorlevel或称返回码，常见的返回码为0、1。

10)for 命令

FOR这条命令基本上都被用来处理文本,但还有其他一些好用的功能!

看看他的基本格式(这里我引用的是批处理中的格式,直接在命令行只需要一个%号)   
FOR 参数 %%变量名 IN (相关文件或命令) DO 执行的命令

参数:FOR有4个参数 /d   /l   /r   /f   他们的作用我在下面用例子解释   
%%变量名 :这个变量名可以是小写a-z或者大写A-Z,他们区分大小写,FOR会把每个读取到的值给他;   
IN:命令的格式,照写就是了;   
(相关文件或命令) :FOR要把什么东西读取然后赋值给变量,看下面的例子   
do:命令的格式,照写就是了!   
执行的命令:对每个变量的值要执行什么操作就写在这.

可以在CMD输入for /?看系统提供的帮助!对照一下   
FOR %%variable IN (set) DO command [command-parameters]

%%variable 指定一个单一字母可替换的参数。   
(set)      指定一个或一组文件。可以使用通配符。   
command    指定对每个文件执行的命令。   
command-parameters   
             为特定命令指定参数或命令行开关。

现在开始讲每个参数的意思

/d   
仅为目录   
如果 Set (也就是我上面写的 "相关文件或命令") 包含通配符（\* 和 ?），将对与 Set 相匹配的每个目

录（而不是指定目录中的文件组）执行指定的 Command。

系统帮助的格式:FOR /D%%variable IN (set) DO command   
他主要用于目录搜索,不会搜索文件,看这样的例子

@echo off   
for /d %%i in (\*) do @echo %%i   
pause

把他保存放在C盘根目录执行,就会把C盘目录下的全部目录名字打印出来,而文件名字一个也不显示!   
在来一个,比如我们要把当前路径下文件夹的名字只有1-3个字母的打出来

@echo off   
for /d %%i in (???) do @echo %%i   
pause

这样的话如果你当前目录下有目录名字只有1-3个字母的,就会显示出来,没有就不显示了

思考题目:

@echo off   
for /d %%i in (window?) do @echo %%i   
pause

保存到C盘下执行,会显示什么呢?自己看吧!   
/D参数只能显示当前目录下的目录名字,这个大家要注意!

/R   
递归   
进入根目录树[Drive:]Path，在树的每个目录中执行for 语句。如果在 /R 后没有指定目录，则认为是

当前目录。如果 Set 只是一个句点 (.)，则只枚举目录树。   
系统帮助的格式:FOR /R[[drive:]path] %%variable IN (set) DO command

上面我们知道,/D只能显示当前路径下的目录名字,那么现在这个/R也是和目录有关,他能干嘛呢?放心他比

/D强大多了!   
他可以把当前或者你指定路径下的文件名字全部读取,注意是文件名字,有什么用看例子!

@echo off   
for /r c:\ %%i in (\*.exe) do @echo %%i   
pause

咋们把这个BAT保存到D盘随便哪里然后执行,我会就会看到,他把C盘根目录,和每个目录的子目录下面全部

的EXE文件都列出来了,这里的c:\就是目录了。

再来一个   
@echo off   
for /r %%i in (\*.exe) do @echo %%i   
pause

参数不一样了，这个命令前面没加那个C:\也就是搜索路径,这样他就会以当前目录为搜索路径,比如你这

个BAT你把他防灾d:\test目录下执行,那么他就会把D:\test目录和他下面的子目录的全部EXE文件列出

来!!!

/L   
迭代数值范围   
使用迭代变量设置起始值(Start#)，然后逐步执行一组范围的值，直到该值超过所设置的终止值 (End#)

。/L 将通过对 Start# 与 End# 进行比较来执行迭代变量。如果 Start# 小于 End#，就会执行该命令。

如果迭代变量超过 End#，则命令解释程序退出此循环。还可以使用负的 Step# 以递减数值的方式逐步执

行此范围内的值。例如，(1,1,5)生成序列 1 2 3 4 5，而 (5,-1,1) 则生成序列 (5 4 3 2 1)。语法是：

系统帮助的格式:for /L%% Variable in (Start#,Step#,End#) do Command

例如：

@echo off   
for /l %%i in (1,1,5) do @echo %%i   
pause

保存执行看效果,他会打印从1 2 3 4 5 这样5个数字   
(1,1,5)这个参数也就是表示从1开始每次加1直到5终止!

再看这个例子   
@echo off   
for /l %%i in (1,1,5) do start cmd   
pause

执行后是不是吓了一跳,怎么多了5个CMD窗口,呵呵!如果把那个 (1,1,5)改成 (1,1,65535)会有什么结果,

我先告诉大家,会打开65535个CMD窗口....这么多你不死机算你强!

当然我们也可以把那个startcmd改成md %%i 这样就会建立指定个目录了!!!名字为1-65535

看完这个被我赋予破坏性质的参数后,我们来看最后一个参数

/f

含有/F的for详细说明

含有/F的for有很大的用处，在批处理中使用的最多，用法如下：  
格式：   
FOR /F ["options"] %%i IN (file) DOcommand

FOR/F ["options"] %%i IN ("string") DO command

FOR/F ["options"] %%i IN ('command') DO command

这个可能是最常用的，也是最强的命令，主要用来处理文件和一些命令的输出结果。

file代表一个或多个文件

string 代表字符串

command代表命令

["options"]可选

对于FOR /F %%i IN (file) DO command

file为文件名，按照官方的说法是，for会依次将file中的文件打开，并且在进行到下一个文件之前将每个文件读取到内存，按照每一行分成一个一个的元素，忽略空白的行，看个例子。

假如文件a.txt中有如下内容：

第1行第1列第1行第2列第1行第3列  
第2行第1列第2行第2列第2行第3列  
第3行第1列第3行第2列第3行第3列

你想显示a.txt中的内容，会用什么命令呢？当然是type，type a.txt

for也可以完成同样的命令：

for /f %%i in(a.txt) do echo %%i

还是先从括号执行，因为含有参数/f,所以for会先打开a.txt，然后读出a.txt里面的所有内容，把它作为一个集合，并且以每一行作为一个元素，所以会产生这样的集合，

{"第1行第1列第1行第2列第1行第3列"， //第一个元素

"第2行第1列第2行第2列第2行第3列"， //第二个元素

"第3行第1列第3行第2列第3行第3列"}   //第三个元素

集合中只有3个元素，同样用%%i依次代替每个元素，然后执行do后面的命令。

具体过程：

用%%i代替"第1行第1列第1行第2列第1行第3列"，执行do后面的echo %%i，显示"第1行第1列第1行第2列第1行第3列"，

用%%i代替"第2行第1列第2行第2列第2行第3列"，执行echo %%i，显示"第2行第1列第2行第2列第2行第3列"，

依次，直到每个元素都代替完为止。

为了加强理解/f的作用，请执行一下两个命令，对比即可明白：

for/f %%i in (a.txt) do echo %%i //这个会显示a.txt里面的内容，因为/f的作用，会读出a.txt中  
的内容。

for%%i in (a.txt) do echo %%i //而这个只会显示a.txt这个名字，并不会读取其中的内容。

通过上面的学习，我们发现for /f会默认以每一行来作为一个元素，但是如果我们还想把每一行再分解更小的内容，该怎么办呢？不用担心，for命令还为我们提供了更详细的参数，使我们将每一行分为更小的元素成为可能。

它们就是：delims和tokens

delims 用来告诉for每一行应该拿什么作为分隔符，默认的分隔符是空格和tab键

比如，还是上面的文件，我们执行下面的命令：

for/f "delims= " %%i in (a.txt) do echo %%i

显示的结果是：

第1行第1列  
第2行第1列  
第3行第1列

为什么是这样的呢。因为这里有了delims这个参数，=后面有一个空格，意思是再将每个元素以空格分割，默认是只取分割之后的第一个元素。

执行过程是：

将第一个元素"第1行第1列第1行第2列第1行第3列"分成三个元素："第1行第1列" "第1行第2列" "第1行第3列"，它默认只取第一个，即"第1行第1列"，然后执行do后面的命令，依次类推。

但是这样还是有局限的，如果我们想要每一行的第二列元素，那又如何呢？

这时候，tokens跳出来说，我能做到。

它的作用就是当你通过delims将每一行分为更小的元素时，由它来控制要取哪一个或哪几个。

还是上面的例子，执行如下命令：

for/f "tokens=2 delims= " %%i in (a.txt) do echo %%i

执行结果：

第1行第2列  
第2行第2列  
第3行第2列

如果要显示第三列，那就换成tokens=3。

同时tokens支持通配符\*，以及限定范围。

如果要显示第二列和第三列，则换成tokens=2,3或tokens=2-3,如果还有更多的则为：tokens=2-10之类的。

此时的命令为：

for/f "tokens=2,3 delims= " %%i in (a.txt) do echo %%i %%j

怎么多出一个%%j？

这是因为你的tokens后面要取每一行的两列，用%%i来替换第二列，用%%j来替换第三列。

并且必须是按照英文字母顺序排列的，%%j不能换成%%k，因为i后面是j

执行结果为：

第1行第2列第1行第3列  
第2行第2列第2行第3列  
第3行第2列第3行第3列

对以通配符\*，就是把这一行全部或者这一行的剩余部分当作一个元素了。

比如：

for/f "tokens=\* delims= " %%i in (a.txt) do echo %%i

执行结果为：

第1行第1列第1行第2列第1行第3列  
第2行第1列第2行第2列第2行第3列  
第3行第1列第3行第2列第3行第3列

其实就跟for /f %%i in (a.txt) do echo %%i的执行结果是一样的。

再如：

for/f "tokens=2,\* delims= " %%i in (a.txt) do echo %%i %%j

执行结果为：

第1行第2列第1行第3列  
第2行第2列第2行第3列  
第3行第2列第3行第3列

用%%i代替第二列，用%%j代替剩余的所有

最后还有skip合eol，这俩个简单，skip就是要忽略文件的前多少行，而eol用来指定当一行以什么符号开始时，就忽略它。

比如：

for/f "skip=2 tokens=\*" %%i in (a.txt) do echo %%i

结果为:

第3行第1列第3行第2列第3行第3列

用skip来告诉for跳过前两行。

如果不加tokens=\*的话，执行结果为：

第3行第1列

不知道怎么回事。

再如，当a.txt内容变成：

.第1行第1列第1行第2列第1行第3列  
**.**第2行第1列第2行第2列第2行第3列  
第3行第1列第3行第2列第3行第3列

执行for /f "eol=. tokens=\*"%%i in (a.txt) do echo %%i结果是：

第3行第1列第3行第2列第3行第3列

用eol来告诉for忽略以"."开头的行。

同样也必须加tokens=\*，否则只会显示"第3行第1列"

2.  如何在批处理文件中使用参数

批处理中可以使用参数，一般从1%到 9%这九个，当有多个参数时需要用shift来移动，这种情况并不多见，我们就不考虑它了。  
sample1：fomat.bat  
@echo off  
if "%1"=="a" format a:  
:format  
@format a:/q/u/auotset  
@echo please insert another disk to driver A.  
@pause  
@goto fomat  
这个例子用于连续地格式化几张软盘，所以用的时候需在dos窗口输入fomat.bata，呵呵,好像有点画蛇添足了～^\_^

sample2：  
当我们要建立一个IPC$连接地时候总要输入一大串命令，弄不好就打错了，所以我们不如把一些固定命令写入一个批处理，把肉鸡地ip password username 当着参数来赋给这个批处理，这样就不用每次都打命令了。  
@echo off  
@net use \\\\1%\\ipc$ "2%" /u:"3%" 注意哦，这里PASSWORD是第二个参数。  
@if errorlevel 1 echo connection failed  
怎么样,使用参数还是比较简单的吧？你这么帅一定学会了^\_^.

3.  如何使用组合命令(Compound Command)

1.&

Usage：第一条命令 & 第二条命令 [& 第三条命令...]

用这种方法可以同时执行多条命令，而不管命令是否执行成功

Sample：  
C:\\>dir z: & dir c:\\Ex4rch  
The system cannot find the path specified.  
Volume in drive C has no label.  
Volume Serial Number is 0078-59FB

Directory of c:\\Ex4rc

2002-05-14 23:51 <DIR> .  
2002-05-14 23:51 <DIR> ..  
2002-05-14 23:51 14 sometips.gif

2.&&

Usage：第一条命令 && 第二条命令[&& 第三条命令...]

用这种方法可以同时执行多条命令，当碰到执行出错的命令后将不执行后面的命令，如果一直没有出错则一直执行完所有命令；

Sample：  
C:\\>dir z: && dir c:\\Ex4rch  
The system cannot find the path specified.

C:\\>dir c:\\Ex4rch && dir z:  
Volume in drive C has no label.  
Volume Serial Number is 0078-59FB

Directory of c:\\Ex4rch

2002-05-14 23:55 <DIR> .  
2002-05-14 23:55 <DIR> ..  
2002-05-14 23:55 14 sometips.gif  
1 File(s) 14 bytes  
2 Dir(s) 768,671,744 bytes free  
The system cannot find the path specified.

在做备份的时候可能会用到这种命令会比较简单，如：  
dir file://192.168.0.1/database/backup.mdb&& copy file://192.168.0.1/database/backup.mdbE:\\backup  
如果远程服务器上存在backup.mdb文件，就执行copy命令，若不存在该文件则不执行copy命令。这种用法可以替换IF exist了 ：）

3.||

Usage：第一条命令 || 第二条命令 [|| 第三条命令...]

用这种方法可以同时执行多条命令，当碰到执行正确的命令后将不执行后面的命令，如果没有出现正确的命令则一直执行完所有命令；

Sample：  
C:\\Ex4rch>dir sometips.gif || del sometips.gif  
Volume in drive C has no label.  
Volume Serial Number is 0078-59FB

Directory of C:\\Ex4rch

2002-05-14 23:55 14 sometips.gif  
1 File(s) 14 bytes  
0 Dir(s) 768,696,320 bytes free

组合命令使用的例子：  
sample：  
@copy trojan.exe \\\\%1\\admin$\\system32 && if not errorlevel 1 echoIP %1 USER %2 PASS %3 >>victim.txt

4.  管道命令的使用

1. | 命令

Usage：第一条命令 | 第二条命令 [| 第三条命令...]  
将第一条命令的结果作为第二条命令的参数来使用，记得在unix中这种方式很常见。

sample：  
time /t>>D:\\IP.log  
netstat -n -p tcp|find ":3389">>D:\\IP.log  
start Explore

看出来了么？用于终端服务允许我们为用户自定义起始的程序，来实现让用户运行下面这个bat，以获得登录用户的IP。

2. >、>>输出重定向命令

将一条命令或某个程序输出结果的重定向到特定文件中, > 与 >>的区别在于，>会清除调原有文件中的内容后写入指定文件，而>>只会追加内容到指定文件中，而不会改动其中的内容。

sample1：  
echo hello world>c:\\hello.txt (stupid example?)

sample2:  
时下DLL木马盛行，我们知道system32是个捉迷藏的好地方，许多木马都削尖了脑袋往那里钻，DLL马也不例外，针对这一点我们可以在安装好系统和必要的应用程序后，对该目录下的EXE和DLL文件作一个记录：  
运行CMD--转换目录到system32--dir\*.exe>exeback.txt & dir \*.dll>dllback.txt,  
这样所有的EXE和DLL文件的名称都被分别记录到exeback.txt和dllback.txt中,  
日后如发现异常但用传统的方法查不出问题时,则要考虑是不是系统中已经潜入DLL木马了.  
这时我们用同样的命令将system32下的EXE和DLL文件记录到另外的exeback1.txt和dllback1.txt中,然后运行:  
CMD--fc exeback.txt exeback1.txt>diff.txt & fc dllback.txtdllback1.txt>diff.txt.(用FC命令比较前后两次的DLL和EXE文件,并将结果输入到diff.txt中),这样我们就能发现一些多出来的DLL和EXE文件,然后通过查看创建时间、版本、是否经过压缩等就能够比较容易地判断出是不是已经被DLL木马光顾了。没有是最好，如果有的话也不要直接DEL掉，先用regsvr32 /u trojan.dll将后门DLL文件注销掉,再把它移到回收站里，若系统没有异常反映再将之彻底删除或者提交给杀毒软件公司。

3. < 、>& 、<&

< 从文件中而不是从键盘中读入命令输入。  
>& 将一个句柄的输出写入到另一个句柄的输入中。  
<& 从一个句柄读取输入并将其写入到另一个句柄输出中。  
这些并不常用，也就不多做介绍。

5.  如何用批处理文件来操作注册表

在入侵过程中经常会操作注册表的特定的键值来实现一定的目的，例如:为了达到隐藏后门、木马程序而删除Run下残余的键值。或者创建一个服务用以加载后门。当然我们也会修改注册表来加固系统或者改变系统的某个属性，这些都需要我们对注册表操作有一定的了解。下面我们就先学习一下如何使用.REG文件来操作注册表.(我们可以用批处理来生成一个REG文件)  
关于注册表的操作，常见的是创建、修改、删除

1).创建

创建分为两种，一种是创建子项(Subkey)

我们创建一个文件，内容如下：

Windows Registry Editor Version 5.00

[HKEY\_LOCAL\_MACHINE\\SOFTWARE\\Microsoft\\hacker]

然后执行该脚本，你就已经在HKEY\_LOCAL\_MACHINE\\SOFTWARE\\Microsoft下创建了一个名字为“hacker”的子项。

另一种是创建一个项目名称  
那这种文件格式就是典型的文件格式，和你从注册表中导出的文件格式一致，内容如下：

Windows Registry Editor Version 5.00

[HKEY\_LOCAL\_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run]  
"Invader"="Ex4rch"  
"Door"=C:\\\\WINNT\\\\system32\\\\door.exe  
"Autodos"=dword:02

这样就在[HKEY\_LOCAL\_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run]下  
新建了:Invader、door、about这三个项目  
Invader的类型是“String Value”  
door的类型是“REG SZ Value”  
Autodos的类型是“DWORD Value”

2).修改

修改相对来说比较简单，只要把你需要修改的项目导出，然后用记事本进行修改，然后导入（regedit /s）即可。

3).删除

我们首先来说说删除一个项目名称，我们创建一个如下的文件：

Windows Registry Editor Version 5.00

[HKEY\_LOCAL\_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run]  
"Ex4rch"=-

执行该脚本，[HKEY\_LOCAL\_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run]下的"Ex4rch"就被删除了；

附1： WIN2000下的9个批处理命令

批处理文件是将一系列命令按一定的顺序集合为一个可执行的文本文件，其扩展名为BAT。

**1)   REM**

REM 是个注释命令一般是用来给程序加上注解的，该命令后的内容在程序执行的时候将不会被显示和执行。例：   
REM 你现在看到的就是注解，这一句将不会被执行。在以后的例子中解释的内容都REM 会放在REM后面。请大家注意。

**2)   ECHO**

ECHO 是一个回显命令主要参数有OFF和 ON,一般用ECHO message来显示一个特定的消息 。例：   
Echo off   
Rem 以上代表关闭回显即不显示所执行的命令   
Echo 这个就是消息。   
Rem 以上代表显示"这就是消息"这列字符   
执行结果：   
C:\>ECHO.BAT   
这个就是消息。

**3)   GOTO**

GOTO 即为跳转的意思。在批处理中允许以"：XXX"来构建一个标号然后用GOTO ：标号直接来执行标号后的命令。例   
:LABEL   
REM 上面就是名为LABEL的标号。   
DIR C:\   
DIR D:\   
GOTO LABEL   
REM 以上程序跳转标号LABEL处继续执行。

4)   CALL

CALL 命令可以在批处理执行过程中调用另一个批处理，当另一个批处理执行完后再继续执行原来的批处理。例：   
批处理2.BAT内容如下：   
ECHO 这就是2的内容   
批处理1.BAT内容如下：   
ECHO 这是1的内容   
CALL 2.BAT   
ECHO 1和2的内容全部显示完成   
执行结果如下：   
C:\>1.BAT   
这是1的内容   
这就是2的内容   
1和2的内容全部显示完成

5)   PAUSE

PAUSE 停止系统命令的执行并显示下面的内容。例：   
C:\> PAUSE   
请按任意键继续 . . .

**6)   IF**

IF 条件判断语句，语法格式如下：   
IF [NOT] ERRORLEVEL number command   
IF [NOT] string1==string2 command   
IF [NOT] EXIST filename command   
说明：   
[NOT] 将返回的结果取反值即"如果没有"的意思。   
ERRORLEVEL 是命令执行完成后返回的退出值   
Number 退出值的数字取值范围0~255。判断时值的排列顺序应该又大到小。返回的值大于或等于指定的值时条件成立。   
string1==string2 string1和string2都为字符的数据，英文字符的大小写将看做不同，这个条件中的等于号必须是2个（绝对相等），条件想等后即执行后面的 command  
EXIST filename 为文件或目录存在的意思。   
IF ERRORLEVEL这条语句必须放在某一个命令后面。执行命令后由IF ERRORLEVEL来判断命令的返回值。   
例：   
1、 IF [NOT] ERRORLEVEL number command   
检测命令执行完后的返回值做出判断。   
echo off   
dir z:   
rem 如果退出代码为1（不成功）就跳至标题1处执行   
IF ERRORLEVEL 1 goto 1   
rem 如果退出代码为0（成功）就跳至标题0处执行   
IF ERRORLEVEL 0 goto 0   
:0   
echo 命令执行成功！   
Rem 程序执行完毕跳至标题exit处退出   
goto exit   
:1   
echo 命令执行失败！   
Rem 程序执行完毕跳至标题exit处退出   
goto exit   
:exit   
Rem 这里是程序的出口   
2、 IF string1==string2 command   
检测当前变量的值做出判断   
ECHO OFF   
IF %1==2 goto no   
Echo 变量相等！   
Goto exit   
:no   
echo 变量不相等   
goto exit   
:exit   
大家可以这样看效果 C:\>test.bat 数字

3、 IF [NOT] EXIST filename command   
发现特定的文件做出判断   
echo off   
IF not EXIST autoexec.bat goto 1   
echo 文件存在成功！   
goto exit   
:1   
echo 文件不存在失败！   
goto exit   
:exit   
这个批处理大家可以放在c盘和d盘分别执行看看效果。

**7)   FOR**

FOR这个命令比较特殊是一个循环执行命令的命令，同时FOR的循环里面还可以套用FOR在进行循环。这篇我们介绍基本的用法就不做套用的循环了，后面再来讲解套用的循环。在批处理中FOR的命令如下：  
FOR [%%c] IN (set) DO [command] [arguments]   
在命令行中命令如下：   
FOR [%c] IN (set) DO [command] [arguments]   
常用参数：   
/L 该集表示以增量形式从开始到结束的一个数字序列。因此，(1,1,5) 将产生序列 1 2 3 4 5，(5,-1,1) 将产生序列 (5 4 3 2 1)。   
/D 如果集中包含通配符，则指定与目录名匹配，而不与文件名匹配。

/F 从指定的文件中读取数据作为变量   
eol=c - 指一个行注释字符的结尾(就一个)   
skip=n - 指在文件开始时忽略的行数。   
delims=xxx - 指分隔符集。这个替换了空格和跳格键的默认分隔符集。   
tokens=x,y,m-n - 指每行的哪一个符号被传递到每个迭代的 for 本身。这会导致额外变量名称的分配。m-n格式为一个范围。通过 nth 符号指定 mth。如果符号字符串中的最后一个字符星号，那么额外的变量将在最后一个符号解析之后分配并接受行的保留文本。  
usebackq - 指定新语法已在下类情况中使用:在作为命令执行一个后引号的字符串并且一个单引号字符为文字字符串命令并允许在 filenameset中使用双引号扩起文件名称。  
下面来看一个例子：   
FOR /F "eol=; tokens=2,3\* delims=, " %i in (myfile.txt) do @echo %i%j %k   
会分析 myfile.txt 中的每一行，忽略以分号打头的那些行，将每行中的第二个和第三个符号传递给 for 程序体；用逗号和/或空格定界符号。请注意，这个 for 程序体的语句引用 %i 来取得第二个符号，引用 %j 来取得第三个符号，引用 %k来取得第三个符号后的所有剩余符号。对于带有空格的文件名，您需要用双引号将文件名括起来。为了用这种方式来使用双引号，您还需要使用 usebackq 选项，否则，双引号会被理解成是用作定义某个要分析的字符串的。  
%i 专门在 for 语句中得到说明，%j 和 %k 是通过tokens= 选项专门得到说明的。您可以通过 tokens= 一行指定最多 26 个符号，只要不试图说明一个高于字母 'z' 或'Z' 的变量。请记住，FOR变量名分大小写，是通用的；而且，同时不能有 52 个以上都在使用中。  
您还可以在相邻字符串上使用 FOR /F 分析逻辑；方法是，用单引号将括号之间的 filenameset 括起来。这样，该字符串会被当作一个文件中的一个单一输入行。最后，您可以用 FOR /F 命令来分析命令的输出。方法是，将括号之间的 filenameset 变成一个反括字符串。该字符串会被当作命令行，传递到一个子 CMD.EXE，其输出会被抓进内存，并被当作文件分析。因此，以下例子:  
FOR /F "usebackq delims==" %i IN (`set`) DO @echo %i   
会枚举当前环境中的环境变量名称。   
以下列举一个简单的例子，他将说明参数/L和没有参数的区别：   
删除文件1.TXT 2.TXT 3.TXT 4.TXT 5.TXT   
例：   
ECHO OFF   
FOR /L %%F IN (1,1,5) DO DEL %%F.TXT   
或   
FOR %%F IN (1,2,3,4,5) DO DEL %%F.TXT   
以上2条命令执行的结果都是一样的如下：   
C:\>DEL 1.TXT   
C:\>DEL 2.TXT   
C:\>DEL 3.TXT   
C:\>DEL 4.TXT   
C:\>DEL 5.TXT

**8)   SETLOCAL**

开始批处理文件中环境改动的本地化操作。在执行 SETLOCAL 之后   
所做的环境改动只限于批处理文件。要还原原先的设置，必须执   
行 ENDLOCAL。 达到批处理文件结尾时，对于该批处理文件的每个   
尚未执行的 SETLOCAL 命令，都会有一个隐含的ENDLOCAL 被   
执行。例：   
@ECHO OFF   
SET PATH /\*察看环境变量PATH   
PAUSE   
SETLOCAL   
SET PATH=E:\TOOLS /\*重新设置环境变量PATH   
SET PATH   
PAUSE   
ENDLOCAL   
SET PATH   
从上例我们可以看到环境变量PATH第1次被显示得时候是系统默认路径。被设置成了E:\TOOLS后显示为E:\TOOLS但当ENDLOCAL后我们可以看到他又被还原成了系统的默认路径。但这个设置只在该批处理运行的时候有作用。当批处理运行完成后环境变量PATH将会还原。

9)   SHIFT

SHIFT命令可以让在命令上的的命令使用超过10个（%0~%9）以上的可替代参数例：   
ECHO OFF   
ECHO %1 %2 %3 %4 %5 %6 %7 %8 %9   
SHIFT   
ECHO %1 %2 %3 %4 %5 %6 %7 %8 %9   
SHIFT   
ECHO %1 %2 %3 %4 %5 %6 %7 %8 %9   
执行结果如下：   
C::\>SHIFT.BAT 1 2 3 4 5 6 7 8 9 10 11   
1 2 3 4 5 6 7 8 9   
2 3 4 5 6 7 8 9 10   
3 4 5 6 7 8 9 10 11   
以上就是基于WIN2000下的9个批处理命令。

附2：特殊的符号与批处理

在命令行下有些符号是不允许使用的但有些符号却有着特殊的意义。

1)   符号(@)

@在批处理中的意思是关闭当前行的回显。我们从上面知道用命令echo off可以关掉整个批处理的命令回显但却不能不显示echo off这个命令。现在我们在这个命令前加上@这样echo off这一命令就被@关闭了回显从而达到所有命令均不回显得要求。

2)   符号(>)

>的意思是传递并覆盖。他所起的作用是将运行后的回显结果传递到后面的范围（后面可是文件也可是默认的系统控制台）例：   
文件1.txt的文件内容为：   
1+1   
使用命令c:\>dir \*.txt >1.txt  
这时候1.txt的内容如下   
驱动器 C 中的卷没有标签。   
卷的序列号是 301A-1508   
C:\ 的目录   
2003-03-11 14:04 1,005 FRUNLOG.TXT   
2003-04-04 16:38 18,598,494 log.txt   
2003-04-04 17:02 5 1.txt   
2003-03-12 11:43 0 aierrorlog.txt   
2003-03-30 00:35 30,571 202.108.txt   
5 个文件 18,630,070 字节   
0 个目录 1,191,542,784 可用字节   
>将命令执行的结 哺橇嗽 嫉奈募 谌荨?   
在传递给控制台的时候程序将不会有任何回显（注意：这里的回显跟echo off关掉的回显不是同一概念。Echo off关掉的是输入命令的回显，这里的回显是程序执行中或后的回显）例：  
C:\>dir \*.txt >nul   
程序将没有任何显示也不会产生任何痕迹。

3)   符号(>>)

符号>>的作用与符号>相似，但他们的区别在于>>是传递并在文件末尾追加，>>也可将回显传递给控制台（用法同上）例：   
文件1.txt内同为：   
1+1   
使用命令c:\>dir \*.txt >>1.txt   
这时候1.txt的内容如下   
1+1   
驱动器 C 中的卷没有标签。   
卷的序列号是 301A-1508   
C:\ 的目录   
2003-03-11 14:04 1,005 FRUNLOG.TXT   
2003-04-04 16:38 18,598,494 log.txt   
2003-04-04 17:02 5 1.txt   
2003-03-12 11:43 0 aierrorlog.txt   
2003-03-30 00:35 30,571 202.108.txt   
5 个文件 18,630,070 字节   
0 个目录 1,191,542,784 可用字节   
>>将命令执行的结果追加在了原始的文件内容后面。

4) 符号(|)

|是一个管道传输命令意思是将上一命令执行的结果传递给下一命令去处理。例：   
C:\>dir c:\|find "1508"   
卷的序列号是 301A-1508   
以上命令的意思为查找c:\的所有并发现1508字符串。Find的用法请用 find /?自行查看   
在不使用format的自动格式化参数的时候我是这样来自动格式化盘片的   
echo y|fornat a: /s /q /v:system   
用过format命令的人都知道format有一个交互对化过程，要使用者输入y来确定当前的命令是否被执行。在这个命令前加上echo y并用管道传输符|将echo执行的结果y传递给format从而达到手工输入y的目的（这条命令有危害性，测试的时候请谨慎）

5)   符号(^)

^ 是对特殊符号 > 、<、 &、的前导字符。在命令中他将以上的3个符号的特殊动能去掉仅仅只吧他们当成符号而不使用他们的特殊意义。例：   
c:\>echo test ^> 1.txt   
test > 1.txt   
从上面可以看出并没有把test写入文件1.txt而是将test >1.txt 当字符串显示了出来。这个符号在远程构建批处理的时候很有效果。

6)   符号(&)

&符号允许在一行中使用2个以上不同的命令，当第一个命令执行失败将不影响第2个命令的执行。例：   
c:\> dir z:\ &dir y:\ &dir c:\   
以上的命令将会连续显示z: y: c:盘内的内容不理会该盘符是否存在。

7)   符号(&&)

&&符号也是允许在一行中使用2个以上不同的命令，当第一个命令执行失败后后续的命令将不会再被执行。例：   
c:\> dir z:\ &&dir y:\ &&dir c:\   
以上的命令将会提示检查是否存在z:盘如果存在则执行，如果不存在则停止执行所有的后续命令

8) 符号("")

" "符号允许在字符串中包含空格。进入一个特殊的目录可以用如下方法例:   
c:\>cd "Program Files"   
c:\>cd progra~1   
c:\>cd pro\*   
以上方法都可以进入Program Files目录

9)   符号（,）

,符号相当于空格。在某些特殊的情况下可以用,来代替空格使用。例：   
c:\>dir,c:\

10)  符号(;)

;符号当命令相同的时候可以将不同的目标用;隔离开来但执行效果不变。如执行过程中发生错误则只返回错误报告但程序还是会继续执行。例：   
DIR C:\;D:\;E:\F:\   
以上的命令相当于   
DIR C:\   
DIR D:\   
DIR E:\   
DIR F:\   
当然还有些特殊的符号但他们的使用范围很小我就不再这里一一的说明了