

# 【应急响应】 windows 入侵检查流程

应急响应的时候，我们需要判断一个系统是否有被黑客入侵，本篇给大家介绍一些在应急响应时 Windows 入侵检查的一些知识点。

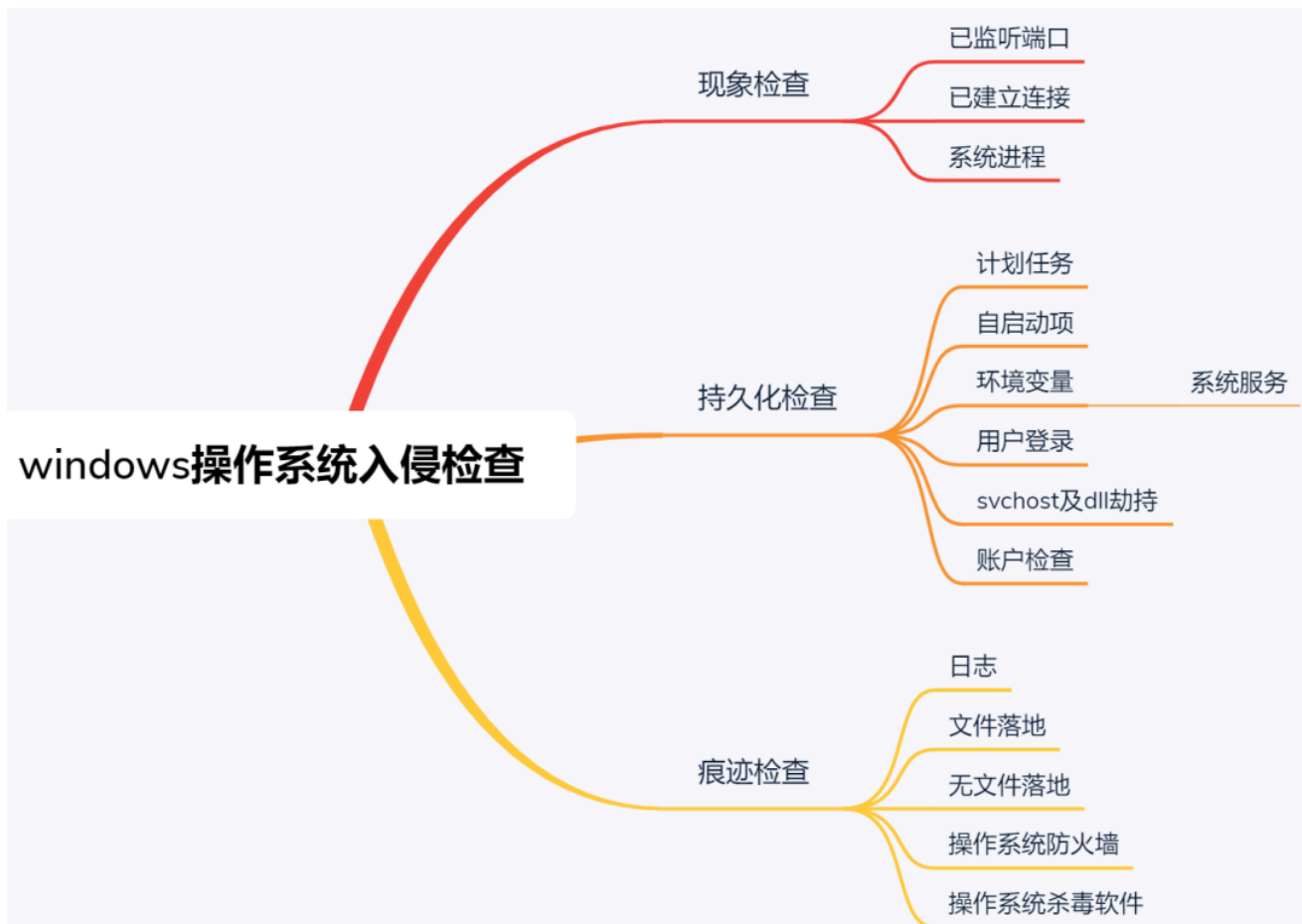
## 检查概述

由于无法站在攻击者视角审视其做过哪些攻击行为，因此标准化的检查内容可以规避非标准化的风险。

例如操作系统虽然没有异常登录日志，但如果不检查操作系统用户即会存在遗漏从而产生风险，同时也可规避上次检查 ab 内容，本次检查 bc 内容的非标准化风险。

因此无论每个人的标准是否统一，取长补短逐渐完善自己的标准化是建议进行的。

windows 操作系统入侵检查流程图如下所示：



## 现象检查

可通过监测告警、日常巡检等主动机制发现存在的异常事件，如果没有主动发现，则只能在安全事件发生后被动发现。以下详细讲解入侵检查时可能存在的异常现象进行检查。

安全事件发生后被动发现。以下说明在被入侵后对可能存在的异常现象进行检查。

现象检查发现的异常程序不可直接删除，应先验证异常进程是否存在自我守护机制，否则安全事件无法得到根除。

## 1.1 已监听端口

已监听端口并非一个独立的对象，而是和进程相关联，进程如果需要对外提供访问接口，则必须通过监听端口的方式对外开放，常用于在内网中部署正向后门程序。

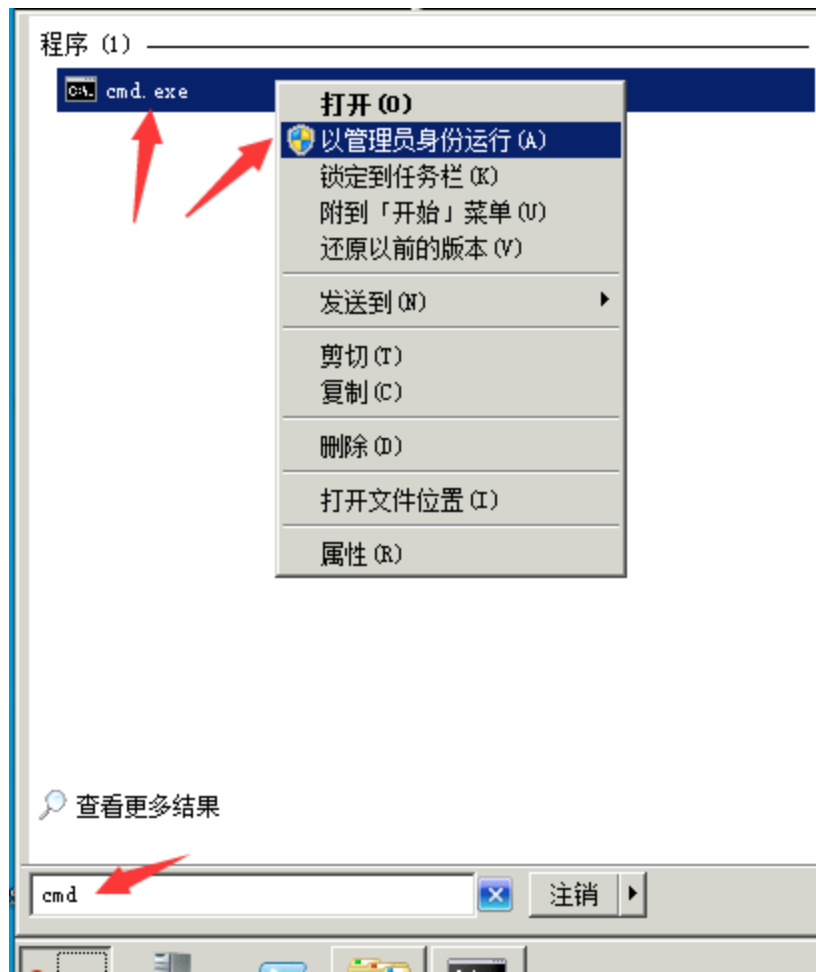
注意点：

- 1 在操作系统初始化正常运行后，建议记录已监听端口的基线值，供日常巡检使用；
- 2 受操作系统、关键路径中的网络层访问控制影响。

例如检查已监听端口是否存在异常。则运行 cmd 命令行，使用 `netstat -ano | findstr LIST` 命令检查已监听端口。

示例：

点击【开始菜单】，搜索框中输入【cmd】，右键点击【cmd.exe】程序，选择【以管理员身份运行】。



使用 netstat -ano | findstr LIST 命令检查已监听端口。含义如下：

1 左 1 列，程序协议；

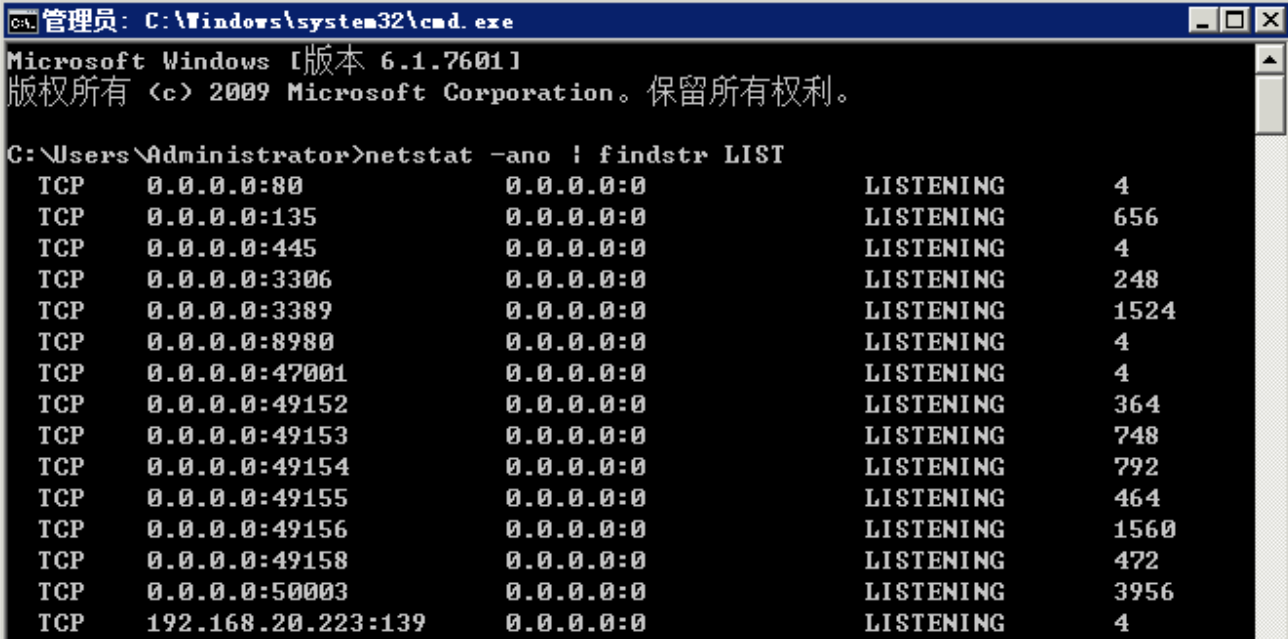
2 左 2 列，本地监听地址和端口；

3 左 3 列，外部地址（留空）；

4 左 4 列，状态为监听；

5 左 5 列，程序 pid。

可根据已知程序不会监听的端口进行判断是否存在异常，并根据该链接的 pid 进行深入分析。



```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>netstat -ano | findstr LIST
TCP      0.0.0.0:80           0.0.0.0:0           LISTENING      4
TCP      0.0.0.0:135          0.0.0.0:0           LISTENING      656
TCP      0.0.0.0:445          0.0.0.0:0           LISTENING      4
TCP      0.0.0.0:3306         0.0.0.0:0           LISTENING      248
TCP      0.0.0.0:3389         0.0.0.0:0           LISTENING      1524
TCP      0.0.0.0:8980         0.0.0.0:0           LISTENING      4
TCP      0.0.0.0:47001        0.0.0.0:0           LISTENING      4
TCP      0.0.0.0:49152        0.0.0.0:0           LISTENING      364
TCP      0.0.0.0:49153        0.0.0.0:0           LISTENING      748
TCP      0.0.0.0:49154        0.0.0.0:0           LISTENING      792
TCP      0.0.0.0:49155        0.0.0.0:0           LISTENING      464
TCP      0.0.0.0:49156        0.0.0.0:0           LISTENING      1560
TCP      0.0.0.0:49158        0.0.0.0:0           LISTENING      472
TCP      0.0.0.0:50003        0.0.0.0:0           LISTENING      3956
TCP      192.168.20.223:139   0.0.0.0:0           LISTENING      4
```

## 1.2 已建立连接

已建立连接分为入站连接和出站连接，入站意为访问操作系统本地的方向，出站意为操作系统访问外部的方向。

注意点：

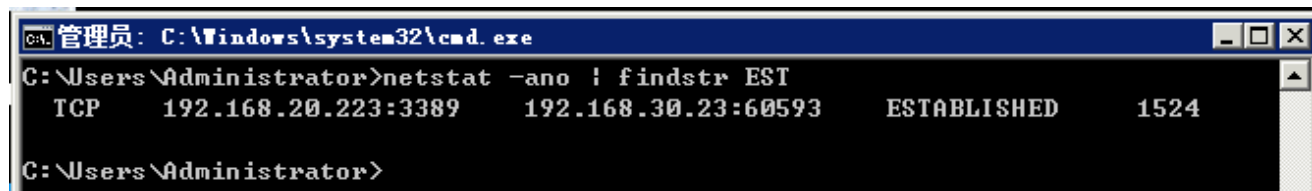
1 受操作系统、关键路径中的网络层访问限制影响；

2 服务端如存在主动外联则需要重点检查。

例如检查已建立连接是否存在异常。则使用 netstat -ano | findstr EST 命令检查已建立连接。

示例：

查询结果，可根据非常规连接判断是否存在异常，并根据该链接的 pid 进行深入分析：



```
C:\Windows\system32\cmd.exe
C:\Users\Administrator>netstat -ano | findstr EST
TCP 192.168.20.223:3389 192.168.30.23:60593 ESTABLISHED 1524
C:\Users\Administrator>
```

### 1.3 系统进程

cpu 资源被占满、异常的已监听端口、异常的已建立连接在深入分析时都会检查系统进程。

注意点：不建议使用任务管理器进行系统进程检查，因为可供分析的维度较少，且容易被进程名欺骗，操作系统允许相同名称但不同执行路径的进程同时存在。

例如检查系统进程是否存在异常，使用以下命令获取系统进程详细信息。

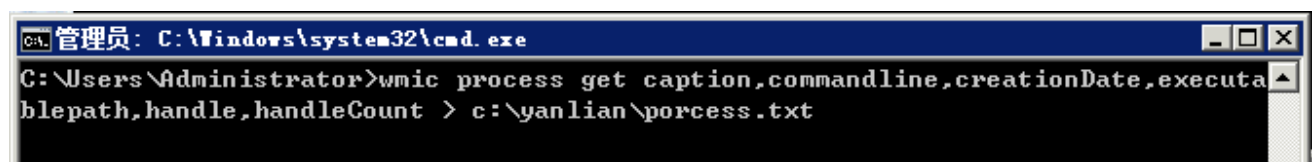
```
wmic process get
```

```
caption,commandline,creationDate,executablepath,handle,handleCount
```

```
> c:\yanlian\porcess.txt
```

示例：

在 cmd 命令行中复制以上命令并回车执行。



```
C:\Windows\system32\cmd.exe
C:\Users\Administrator>wmic process get caption,commandline,creationDate,executablepath,handle,handleCount > c:\yanlian\porcess.txt
```

打开 C:\yanlian\process.txt，可看到 6 列内容，含义如下所示：

1 caption：进程名；

2 commandline：进程名、程序执行路径、进程执行参数；

3 creationDate：进程启动时间（格式为：年月日时分秒）；

4 executablepath：程序执行路径；

5 handle：进程 pid；

6 handleCount：该进程的父进程 pid。

可根据进程名、进程执行参数、进程启动时间、程序执行路径判断是否存在异常，并根据异常点进行深入分析。

| 1  | Caption             | CommandLine                                    |
|----|---------------------|--|
| 2  | System Idle Process |  |
| 3  | System              |  |
| 4  | smss.exe            | \SystemRoot\System32\smss.exe                  |
| 5  | csrss.exe           | %SystemRoot%\system32\csrss.exe ObjectDirector |
| 6  | wininit.exe         | wininit.exe                                    |
| 7  | csrss.exe           | %SystemRoot%\system32\csrss.exe ObjectDirector |
| 8  | winlogon.exe        | winlogon.exe                                   |
| 9  | services.exe        | C:\Windows\system32\services.exe               |
| 10 | lsass.exe           | C:\Windows\system32\lsass.exe                  |
| 11 | lsm.exe             | C:\Windows\system32\lsm.exe                    |
| 12 | svchost.exe         | C:\Windows\system32\svchost.exe -k DcomLaunch  |
| 13 | svchost.exe         | C:\Windows\system32\svchost.exe -k RPCSS       |

| 1  | CreationDate              | ExecutablePath                   | Handle | HandleCount |
|----|---------------------------|----------------------------------|--------|-------------|
| 2  |                           |                                  | 0      | 0           |
| 3  | 20210122142501.632409+480 |                                  | 4      | 487         |
| 4  | 20210122142501.648009+480 |                                  | 240    | 35          |
| 5  | 20210122142502.287610+480 | C:\Windows\system32\csrss.exe    | 312    | 404         |
| 6  | 20210122142502.490410+480 | C:\Windows\system32\wininit.exe  | 364    | 86          |
| 7  | 20210122142502.506010+480 | C:\Windows\system32\csrss.exe    | 376    | 72          |
| 8  | 20210122142502.552810+480 | C:\Windows\system32\winlogon.exe | 408    | 77          |
| 9  | 20210122142502.740011+480 | C:\Windows\system32\services.exe | 464    | 214         |
| 10 | 20210122142502.818011+480 | C:\Windows\system32\lsass.exe    | 472    | 797         |
| 11 | 20210122142502.818011+480 | C:\Windows\system32\lsm.exe      | 480    | 246         |
| 12 | 20210122142503.067611+480 | C:\Windows\system32\svchost.exe  | 576    | 356         |
| 13 | 20210122142503.239211+480 | C:\Windows\system32\svchost.exe  | 656    | 261         |

## 持久化检查

如通过现象检查发现异常程序，则可以通过停止运行该进程的方式，判断其是否会重新启动。

### 1.1 任务计划

任务计划可以将任何脚本或程序定时启动。如被黑客利用则会充当恶意程序的守护机制。

注意点：不建议使用图形化任务计划程序进行检查，因为数量、层级较多不方便检查。

检查任务计划是否存在异常的方法：

1 使用 `schtasks /query /fo LIST /v`

>c:\yanlian\schtasks.txt 命令获取任务计划；

2 使用正则 (Folder|TaskName|Status|Author|Task To Run|Scheduled Task State|Start Time|Start Date)(.\*) 过滤任务计划关键字段；

3 使用正则 (Folder|TaskName|Status|Author|Task To Run|Scheduled Task State|Start Time|Start Date)(.\*) 过滤任务计划关键字段；

3 使用正则 Start Date(.\*) 和 Start Date\$0\n 分割不同任务计划。

示例 1:

导出任务计划，提示错误。

```
C:\Users\Administrator>schtasks /query /fo LIST /v > c:\yanlian\schtasks.txt
错误: 无法加载列资源。
C:\Users\Administrator>
```

查看当前活动代码页为 936，将其修改为 437。

```
C:\Users\Administrator>chcp
活动代码页: 936
C:\Users\Administrator>chcp 437
```

再次导出任务计划。但导出的任务计划无关信息过多，需要过滤。

```
管理员: C:\Windows\system32\cmd.exe
Active code page: 437
C:\Users\Administrator>schtasks /query /fo LIST /v > c:\yanlian\schtasks.txt
C:\Users\Administrator>
```

示例 2:

复制以下正则表达式。

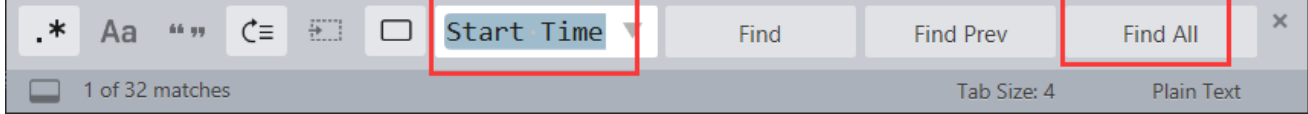
(Folder|TaskName|Status|Author|Task To

Run|Scheduled Task State|Start Time|Start Date)(.\*)

将正则表达式复制到搜索框中，点击【find all】，再【ctrl+c】复制匹配到的内容。

```
untitled • - Sublime Text (UNREGISTERED)
文件(F) 编辑(E) 选择(S) 查找(I) 视图(V) 跳转(G) 工具(T) 项目(P) 首选项(N) 帮助(H)

1
2 Folder: \
3 HostName: WIN-NE1AJP8FADJ
4 TaskName: \??
5 Next Run Time: 2021/2/3 17:18:45
6 Status: Ready
7 Logon Mode: Interactive/Background
8 Last Run Time: N/A
9 Last Result: 1
10 Author: WIN-NE1AJP8FADJ\Administrator
11 Task To Run: C:\Windows\winsxs\wow64_mi
```



同时新建一个文档将复制的内容进行粘贴，但所有任务计划未分割不方便检查，因此还需要对过滤后的任务计划进行分割。

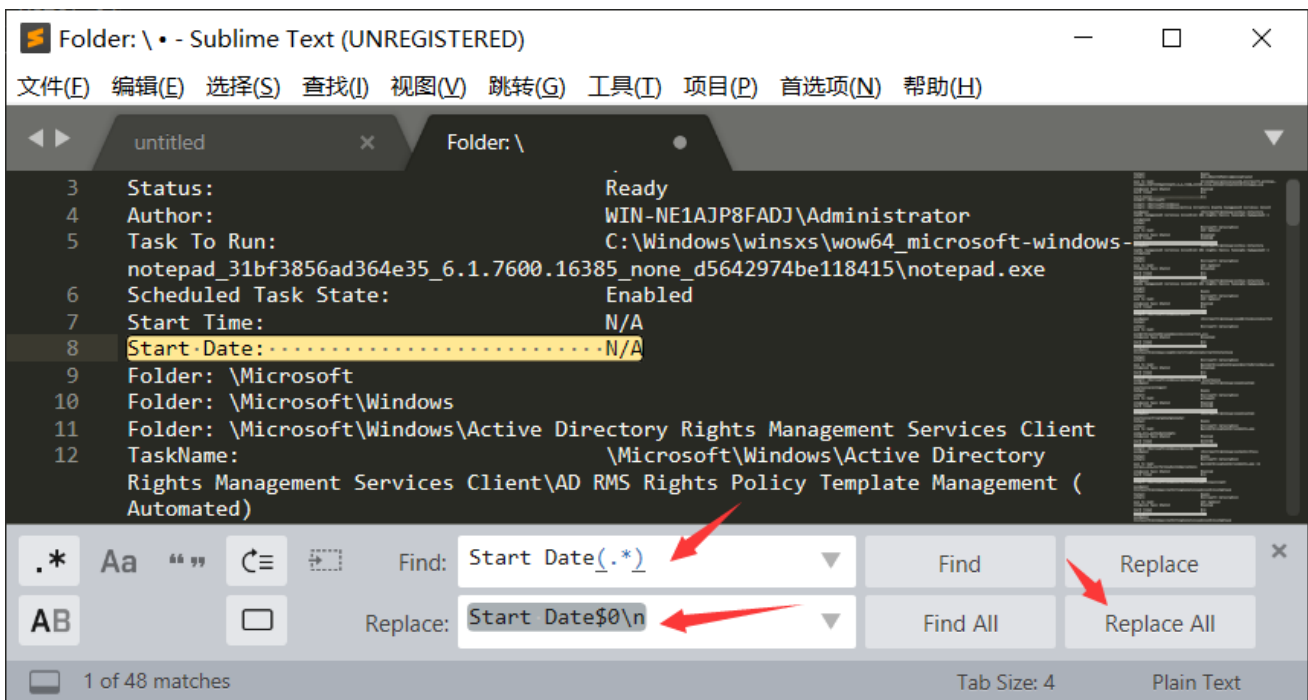
示例 3:

复制以下正则表达式。

Start Date(.\*)

Start Date\$0\n

按【ctrl+h】将以上正则进行粘贴并替换所有。



最后形成如下文档：只记录任务计划名称，运行状态，创建者，程序路径，计划状态，启动时间，以方便对可能存在异常的对象进行检查。

```
1 Folder: \
2 TaskName: \??
3 Status: Ready
4 Author: WIN-NE1AJP8FADJ\Administrator
5 Task To Run: C:\Windows\winsxs\wow64_microsoft
1.7600.16385_none_d5642974be118415\notepad.exe
6 Scheduled Task State: Enabled
7 Start Time: N/A
8 Start DateStart Date: N/A
9
10 Folder: \Microsoft
11 Folder: \Microsoft\Windows
12 Folder: \Microsoft\Windows\Active Directory Rights Management Service
13 TaskName: \Microsoft\Windows\Active Directory
Client\AD RMS Rights Policy Template Management (Automated)
14 Status:
15 Author: Microsoft Corporation
16 Task To Run: COM handler
17 Scheduled Task State: Disabled
18 Start Time: 3:00:00
19 Start DateStart Date: 2006/11/9
```

## 1.2 自启动项

自启动项可在系统启动时自动运行相关程序，恶意程序的第二个自启机制。

注意点：不建议使用图形化 msconfig 工具进行检查，因为名称、路径较长则不方便取证。

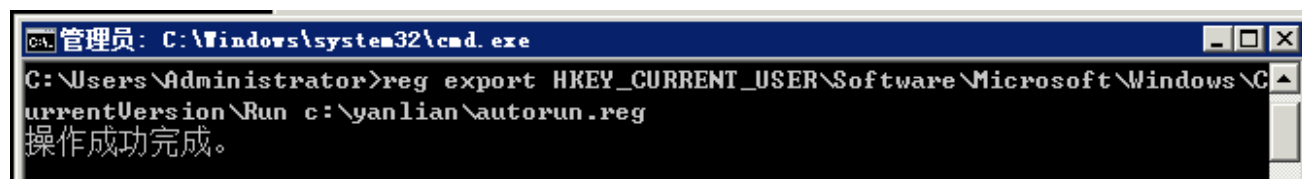
使用以下命令将自启动项导出检查。

```
reg export
```

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
c:\yanlian\autorun.reg
```

示例：

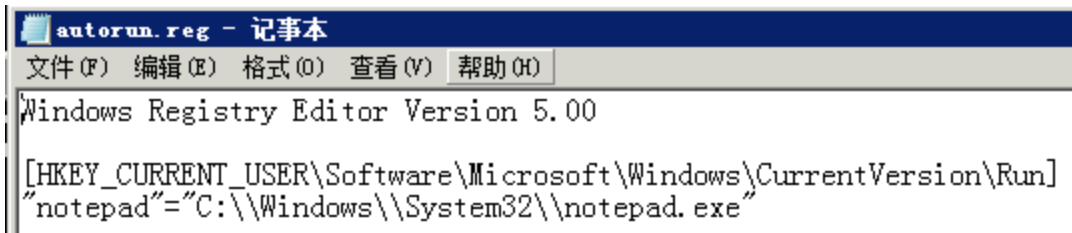
输入命令将自启动项配置文件导出。



```
管理员: C:\Windows\system32\cmd.exe
C:\Users\Administrator>reg export HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run c:\yanlian\autorun.reg
操作成功完成。
```

检查导出的自启动项配置是否存在异常。





```
autorun.reg - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
Windows Registry Editor Version 5.00
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run]
"notepad"="C:\\Windows\\System32\\notepad.exe"
```

### 1.3 环境变量

环境变量用于将系统路径变量化，如被黑客利用则会以最高权限运行恶意程序，例如将环境变量 %systemroot% 变更为其他路径，同时建立 system32 文件夹并将恶意程序通过服务启动。

注意点：环境变量 %systemroot% 修改后需进行恢复，否则系统无法正常重启。

使用 set 命令将环境变量导出检查。

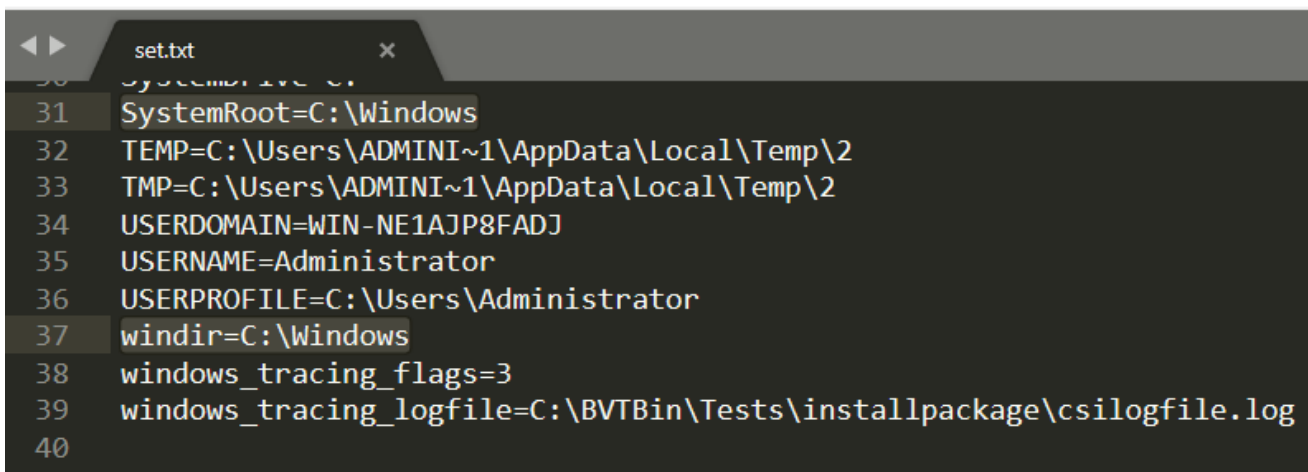
示例：

输入命令将环境变量配置文件导出。



```
C:\Users\Administrator>set > c:\yanlian\set.txt
```

检查导出的环境变量配置是否存在异常。



```
SystemRoot=C:\Windows
TEMP=C:\Users\ADMINI~1\AppData\Local\Temp\2
TMP=C:\Users\ADMINI~1\AppData\Local\Temp\2
USERDOMAIN=WIN-NE1AJP8FADJ
USERNAME=Administrator
USERPROFILE=C:\Users\Administrator
windir=C:\Windows
windows_tracing_flags=3
windows_tracing_logfile=C:\BVTBin\Tests\installpackage\csilogfile.log
```

### 1.4 系统服务

服务可在系统启动时自动运行相关程序或启动后延迟运行相关程序，是恶意程序的第三个自启机制。

注意点：不建议使用图形化 services.msc 程序进行检查，因为数量、层级较多不方便检查。

1 使用命令将服务配置文件导出检查；

2 过滤包含 Description、ImagePath、ServiceDll 的字段；

3 过滤包含 (\*.\*)(\.dll|.exe)(.\*) 的字段；

4 删除 Description REG\_SZ, Description

REG\_EXPAND\_SZ, ImagePath

REG\_EXPAND\_SZ, ServiceDll

REG\_EXPAND\_SZ 无关字符

5 将 / Processid(\*) 替换为空；

6 将, -(\*) 替换为空；

7 将 @替换为空；

8 根据环境变量检查结果对 %systemroot%, %windir% 进行替换；

9 将 ^[a-z]\*\.dll\n 替换为空；

10 排序、统一小写和去重后进行服务检查。

示例 1:

选择一条命令将服务配置文件导出。

```
reg query
```

```
"HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services" /s
```

```
> c:\yanlian\service_001.txt
```

```
reg query
```

```
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services" /s
```

```
> c:\yanlian\service_set.txt
```



```
es" /s > c:\yanlian\service001.txt
```

服务配置文件路径说明：

- 1 ControlSet001：系统真实的服务配置信息；
- 2 ControlSet002：最后一次成功启动的服务配置信息；
- 3 CurrentControlSet：系统运行时的服务配置信息；
- 4 系统启动时，从 ControlSet001 复制到 CurrentControlSet 中；
- 5 系统运行时，修改的都是 CurrentControlSet 中的信息；
- 6 系统重启时，从 CurrentControlSet 复制到 ControlSet001 中；
- 7 系统正常启动时，从 ControlSet001、CurrentControlSet 复制到 ControlSet002；
- 8 开机选择“最近一次正确配置”时，从 ControlSet002 复制到 CurrentControlSet 中。

服务配置说明：

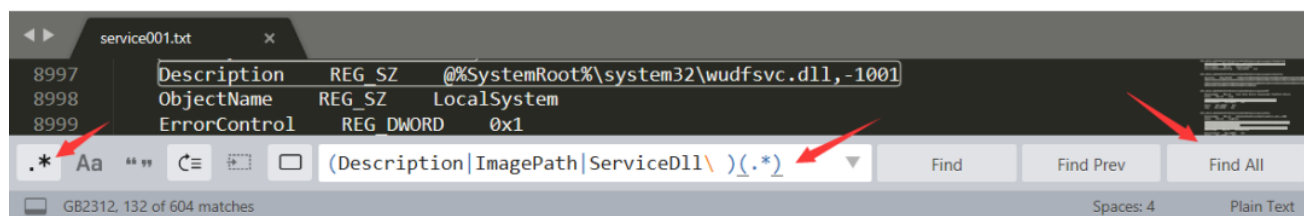
- 1 ImagePath：服务所启动程序的路径；
- 2 Parametes\servicedll：程序调用的真实 dll 文件路径；
- 3 Start：0/boot，1/system，2 / 自动，3 / 手动，4 / 禁用；
- 4 DelayedAutostart：1 / 延迟启动；
- 5 Type：程序类型。

示例 2：

输入以下正则表达式进行内容过滤。

`(Description|ImagePath|ServiceDll\)(.*)`

点击 `【.*】` 启用正则匹配，输入正则表达式，点击 `【find all】` 后复制。

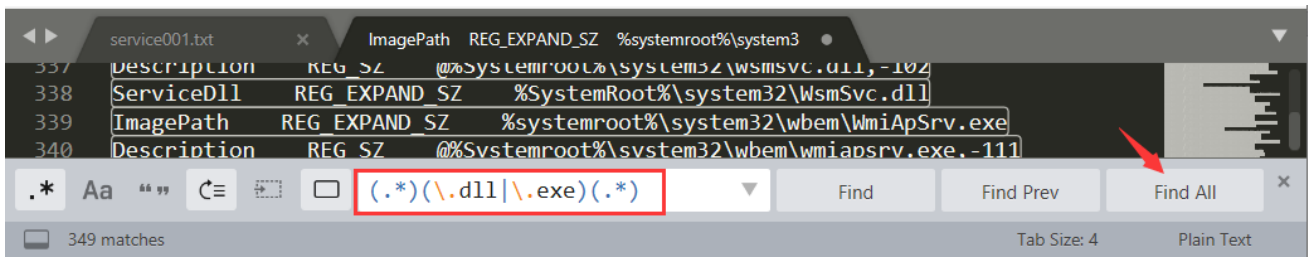


示例 3：

输入以下正则表达式进行内容过滤。

`(.*)"(\.dll|\.exe)(.*)`

点击【find all】后复制。



示例 4:

复制以下单行内容并逐个替换。

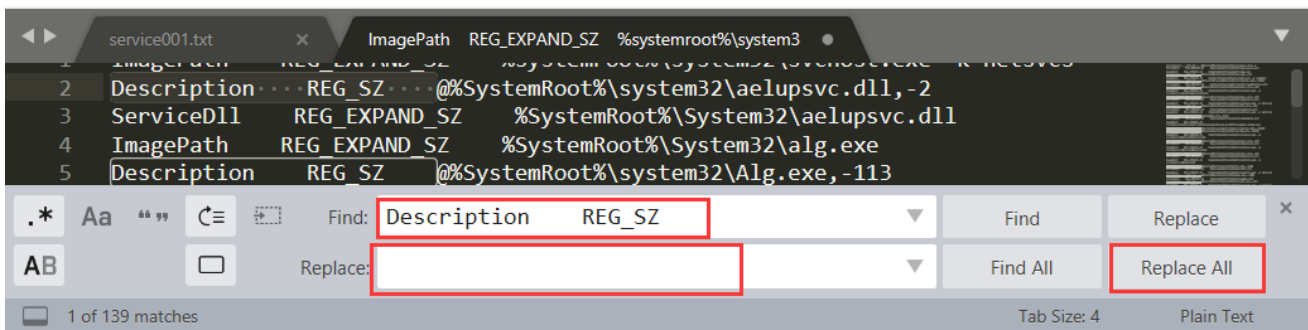
Description REG\_SZ

Description REG\_EXPAND\_SZ

ImagePath REG\_EXPAND\_SZ

ServiceDll REG\_EXPAND\_SZ

替换方法为在【find】框中粘贴，【replace】框中内容为空，点击【replace all】进行替换。

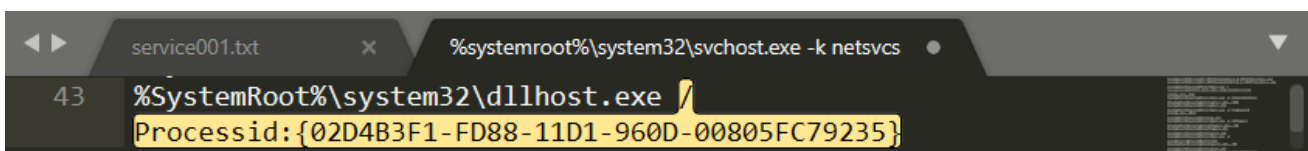


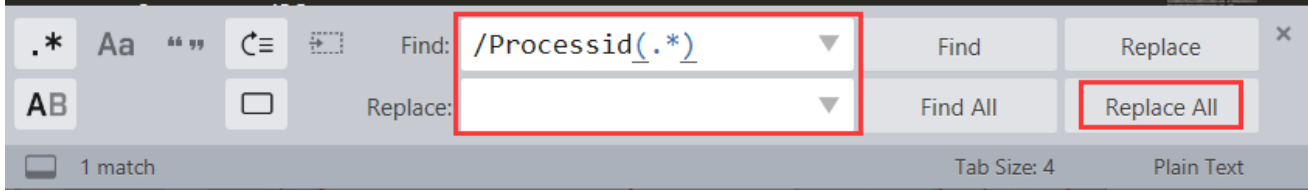
示例 5:

复制以下内容并替换。

`/Processid(.*)`

替换方法为在【find】框中粘贴，【replace】框中内容为空，点击【replace all】进行替换。



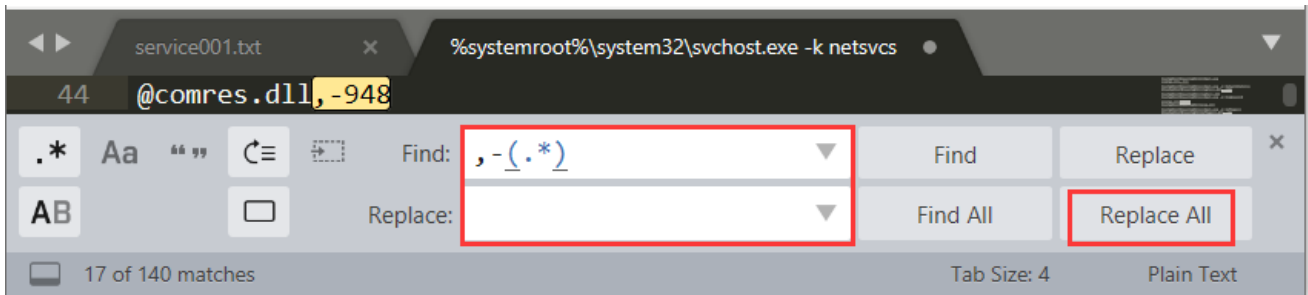


示例 6:

复制以下内容并替换。

,-(.\*)

替换方法为在【find】框中粘贴，【replace】框中内容为空，点击【replace all】进行替换。

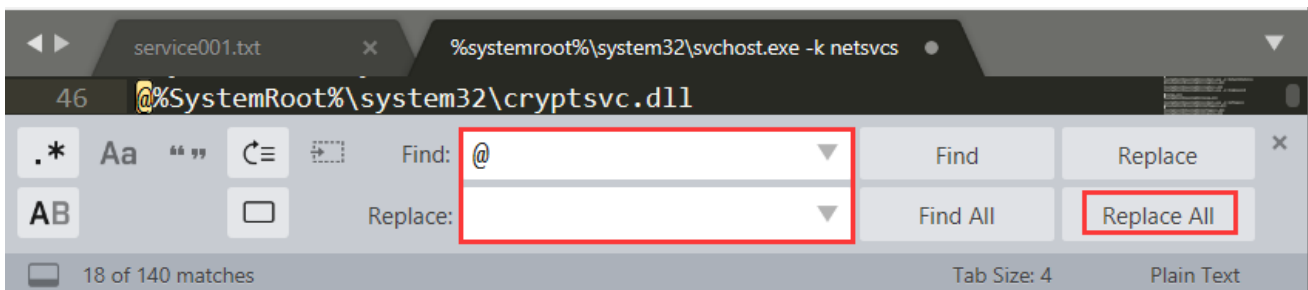


示例 7:

复制以下内容并替换。

@

替换方法为在【find】框中粘贴，【replace】框中内容为空，点击【replace all】进行替换。



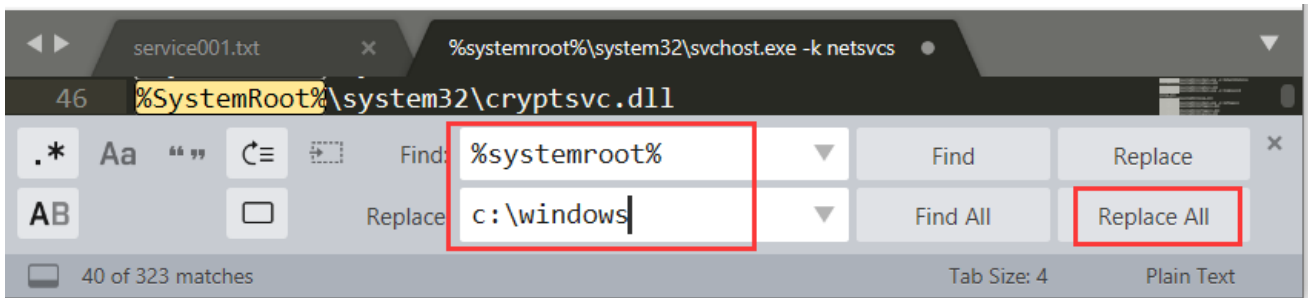
示例 8:

复制以下内容并替换。

%systemroot% c:\windows

%windir% c:\windows

替换方法为在【find】框中粘贴，【replace】框中内容为空，点击【replace all】进行替换。

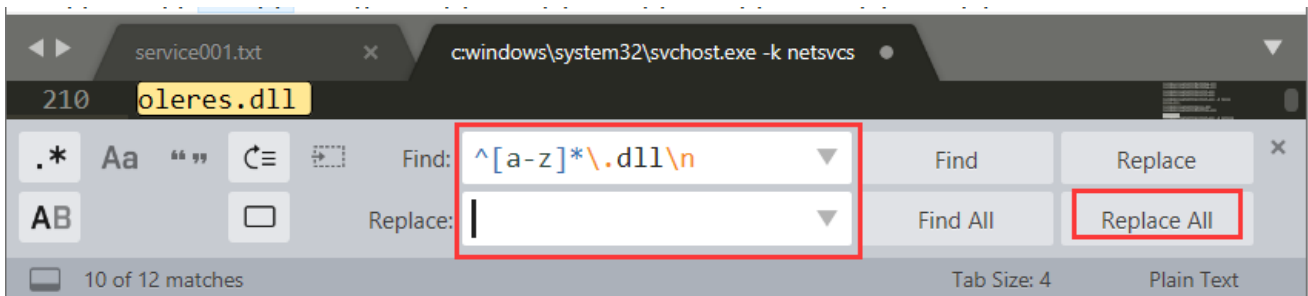


示例 9:

复制以下内容并替换。

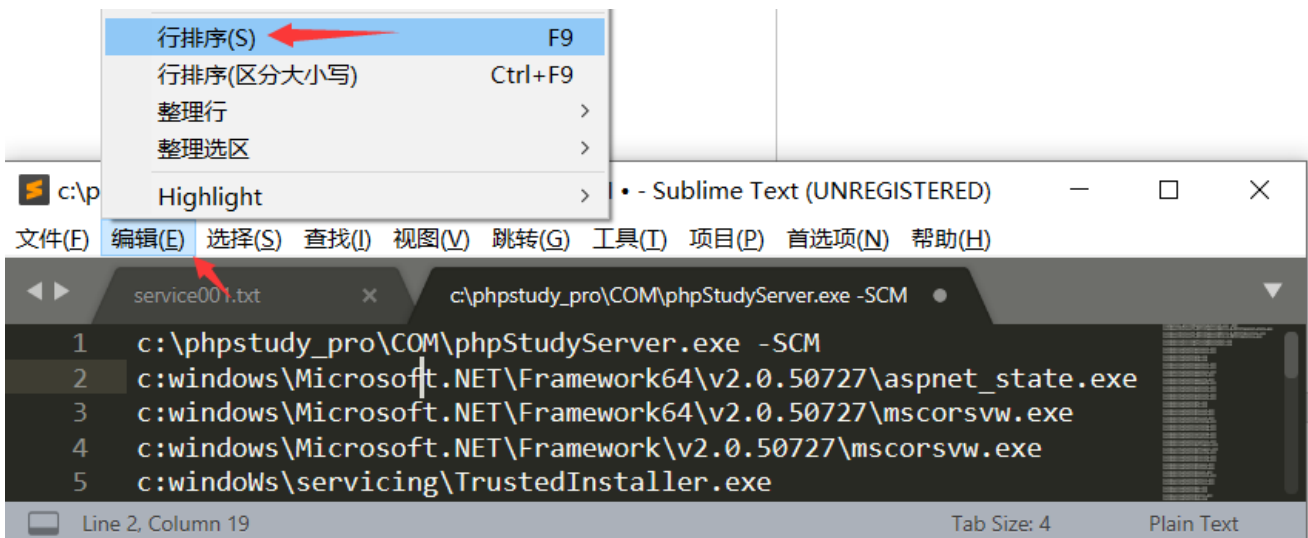
```
^[a-z]*\.dll\n
```

替换方法为在【find】框中粘贴，【replace】框中内容为空，点击【replace all】进行替换。

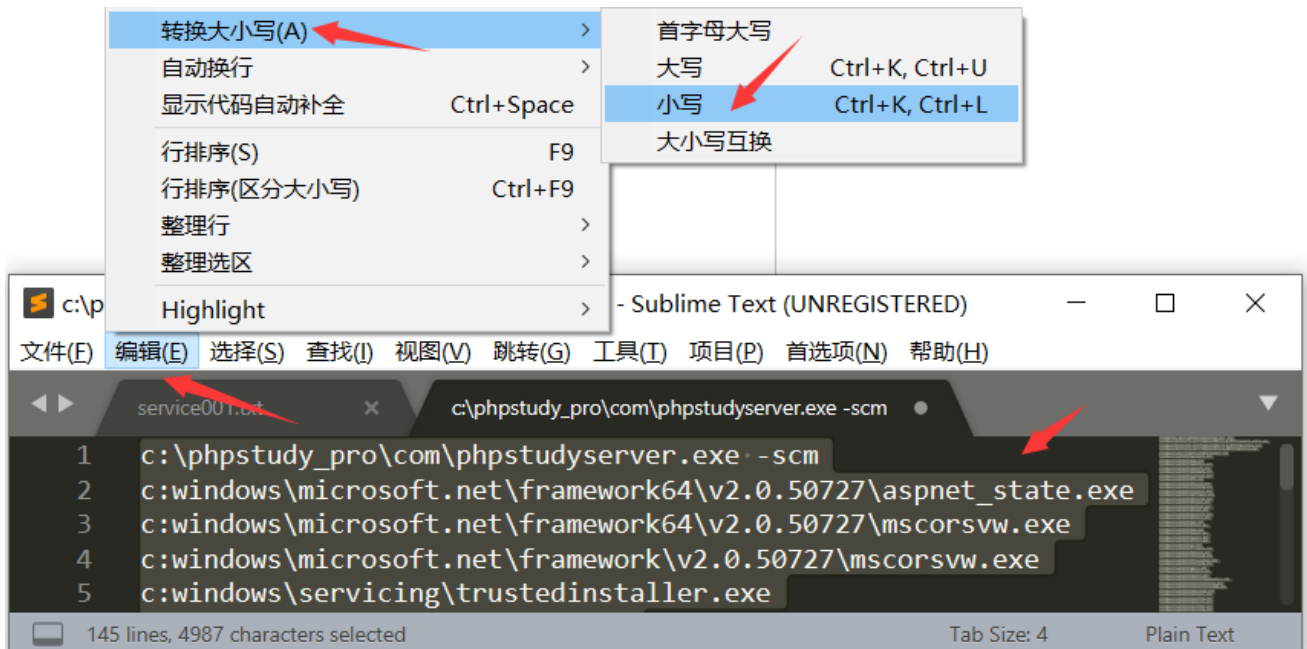


示例 10:

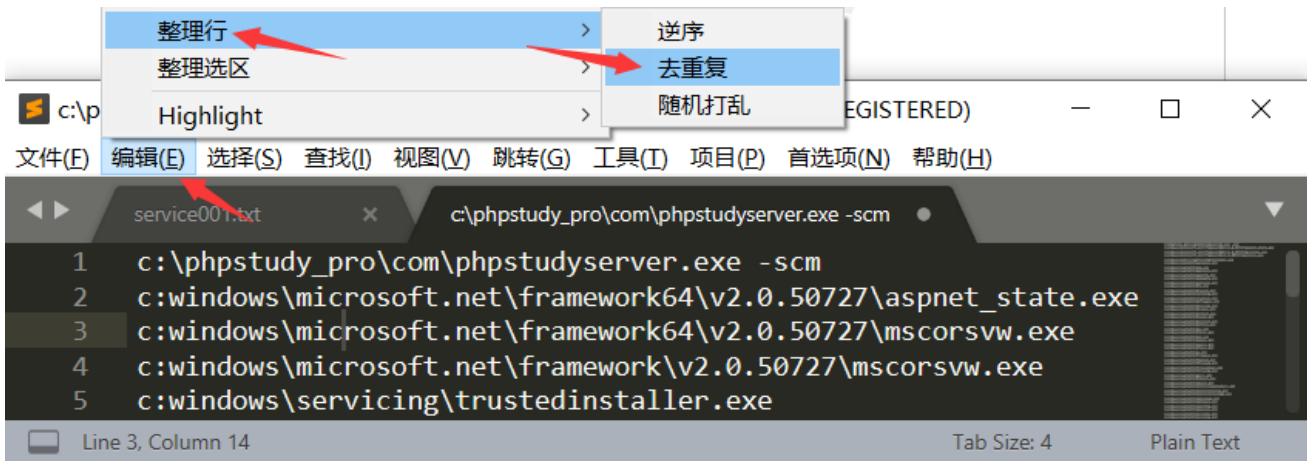
对过滤后的程序路径进行排序。



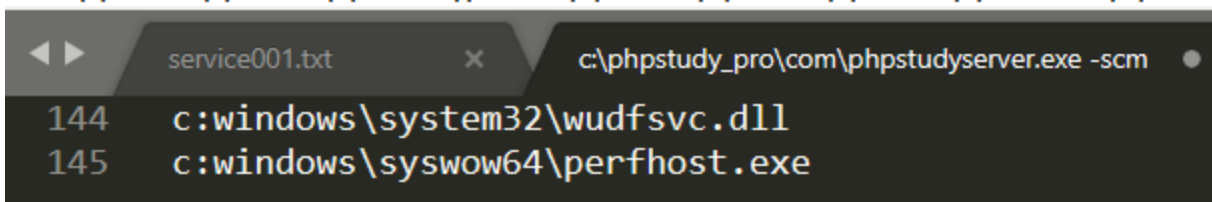
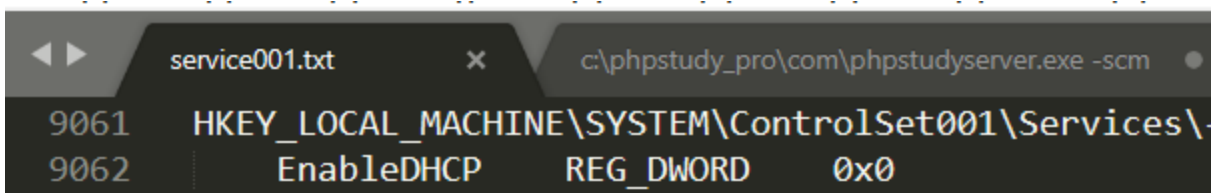
全选所有内容将所有大写字母转换为小写。



对过滤后的程序路径进行去重。



根据去重后的结果可直观的对服务所调用的程序进行检查。例如异常的程序路径，程序名称。同时工作量也会大大减少，因为原先需要分析 9062 行，现在只需要分析 145 行即可，如在 145 行中发现异常则可查看导出的服务配置文件进行深入分析。



## 1.5 用户登录

用户登录可在系统启动登录、注销登录时自动运行相关程序，是恶意程序的第四个自启机制。

注意点：操作系统中有两处配置文件可用于在登录时启动相关程序。

使用命令将用户登录配置文件导出检查。

示例 1:

复制以下命令:

```
reg query HKEY_CURRENT_USER\Environment
```

```
/v UserInitMprLogonScript >
```

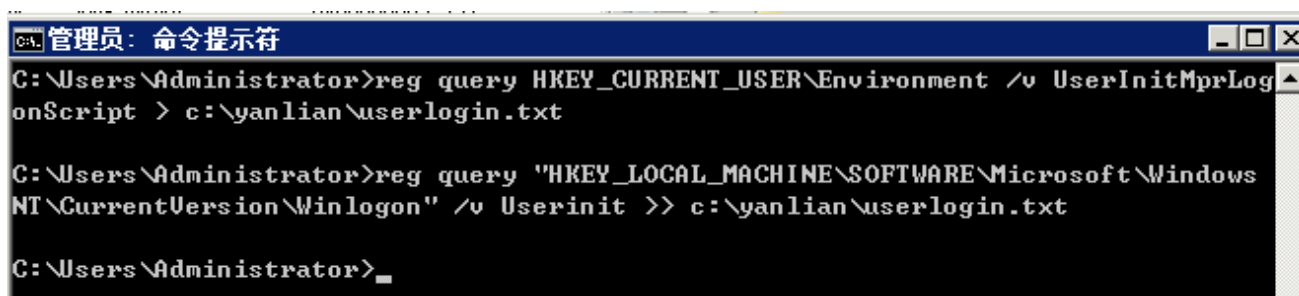
```
c:\yanlian\userlogin.txt
```

```
reg query
```

```
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
```

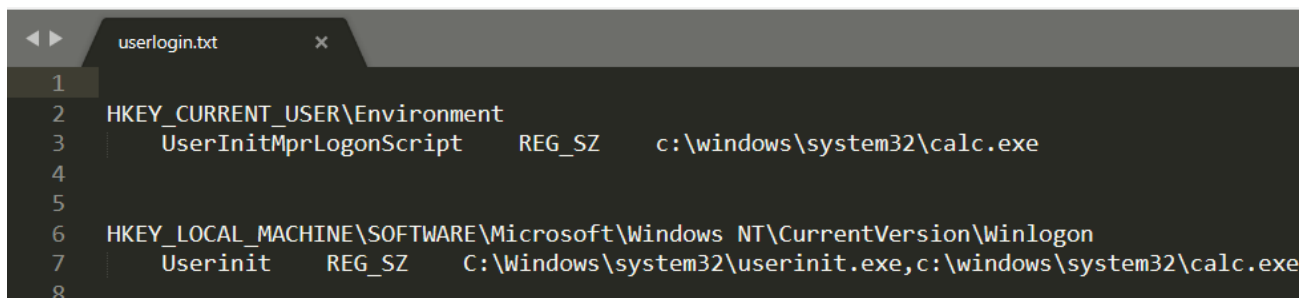
```
NT\CurrentVersion\Winlogon" /v Userinit >> c:\yanlian\userlogin.txt
```

在命令行中粘贴将用户登录配置文件导出检查。



```
管理员: 命令提示符
C:\Users\Administrator>reg query HKEY_CURRENT_USER\Environment /v UserInitMprLogonScript > c:\yanlian\userlogin.txt
C:\Users\Administrator>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v Userinit >> c:\yanlian\userlogin.txt
C:\Users\Administrator>_
```

根据导出的登录配置进行检查。



```
userlogin.txt
1
2 HKEY_CURRENT_USER\Environment
3   UserInitMprLogonScript    REG_SZ    c:\windows\system32\calc.exe
4
5
6 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
7   Userinit    REG_SZ    C:\Windows\system32\userinit.exe,c:\windows\system32\calc.exe
8
```

## 1.6 svchost 及 dll 劫持

svchost.exe 主要作用是将动态链接库 (后缀为 .dll 的文件) 以服务的方式运行。svchost.exe



对系统的正常运行非常重要，是不能被结束的。通过服务、dll、com 均可劫持注入到程序中启动。

注意点：

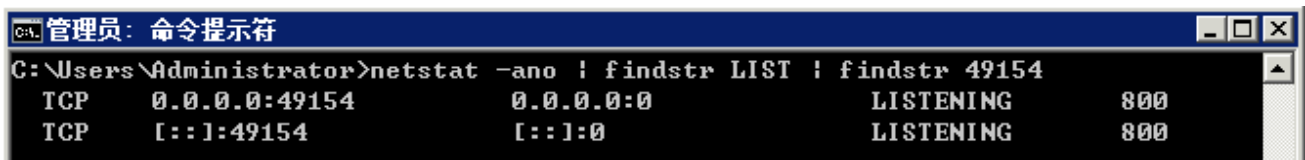
1 在入侵检查方向，更倾向于将 svchost.exe 作为一个单独的持久化检查项目，而非系统服务，因为检查方法完全不同；

2 同时 svchost.exe 不作为常规检查项，一般根据【现象检查】的结果寻找恶意程序。

存在异常监听端口，pid 指向 svchost.exe 程序，需检查是否存在异常。通过第三方工具 ProcessExplorer（简称 pe）检查 svchost.exe 程序。

示例：

通过检查已监听端口，发现“异常”监听，pid 指向 800。

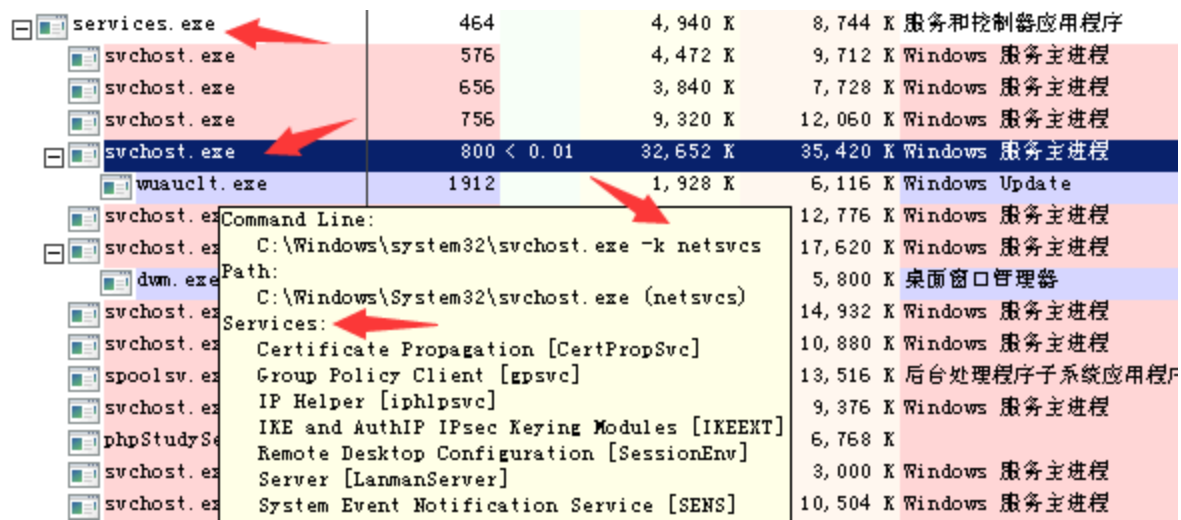


```
管理员：命令提示符
C:\Users\Administrator>netstat -ano | findstr LIST | findstr 49154
TCP    0.0.0.0:49154    0.0.0.0:0      LISTENING      800
TCP    [::]:49154     [::]:0         LISTENING      800
```

打开任务管理器，发现 pid 800 是 svchost.exe 程序，由于该程序的特性，需使用第三方工具 pe 进行检查。

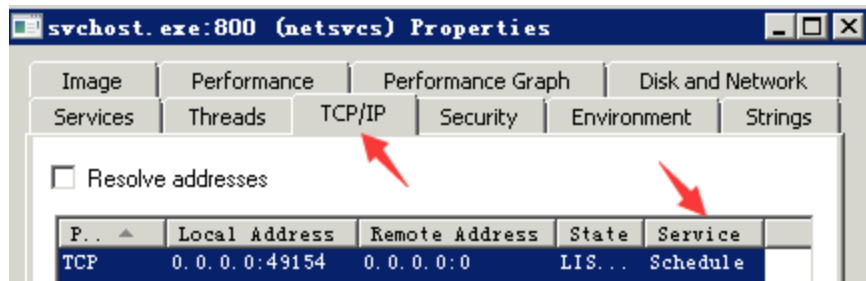


通过 pe 工具可发现 pid 800 是由服务启动，根据启动路径可发现该程序是由正常路径启动，但程序是否被替换未知，启动的服务是 netsvcs（小白杀手，当初虐了我 n 久），该服务下挂 12 个子服务。



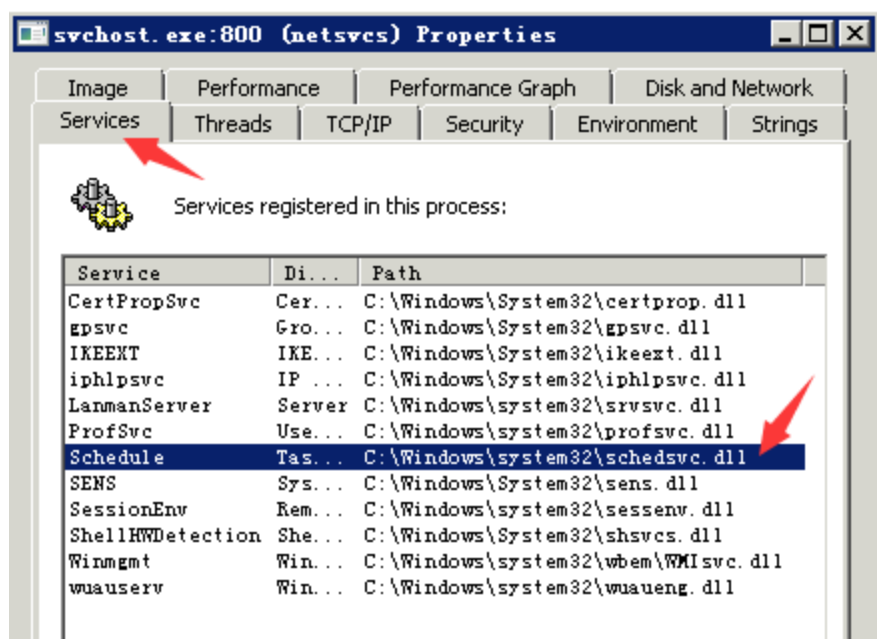
|             |  |                             |
|-------------|--|-----------------------------|
| svchost.exe | Shell Hardware Detection [ShellHWDetection]  | 8,864 K Windows 服务主进程       |
| rdpclip.exe | Task Scheduler [Schedule]                    | 5,952 K RDP Clip 监视程序       |
| svchost.exe | User Profile Service [ProfSvc]               | 5,484 K Windows 服务主进程       |
| sppsvc.exe  | Windows Update [wuauserv]                    | 12,756 K Microsoft 软件保护平台.. |
|             | Windows Management Instrumentation [Winmgmt] | 4,700 K Microsoft 服务主进程     |

在 pe 中双击该程序，点击【tcp/ip】，可发现 49154 端口是服务 schedule 监听。

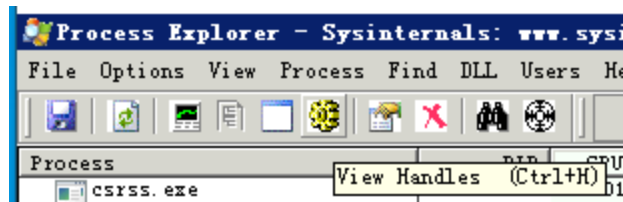


点击【services】，可发现 schedule 服务启动的动态链接库绝对路径是

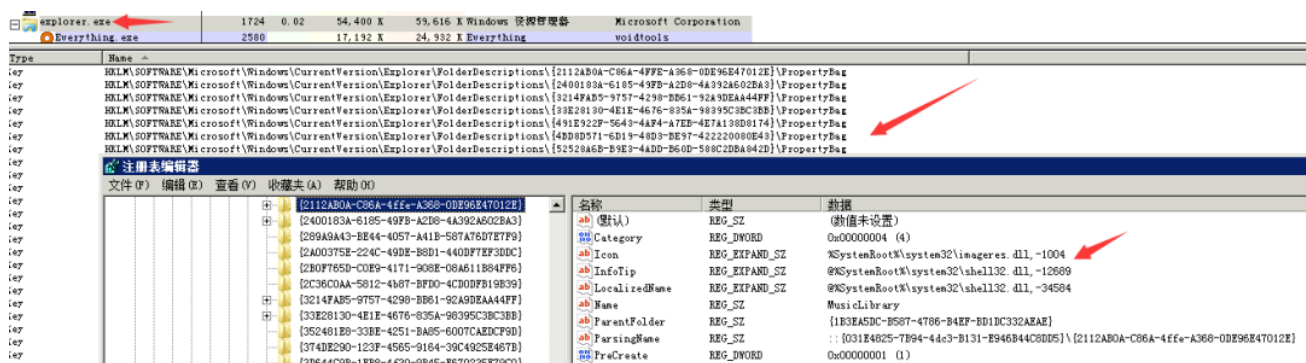
c:\windows\system32\schedsvc.dll，从而发现 49154 端口是哪个程序正在监听。



通过点击【view handles】可显示该程序所调用的 clsid。



如存在启动、点击某个程序后恶意进程重新启动的情况，则可以对 com 劫持进行检查。





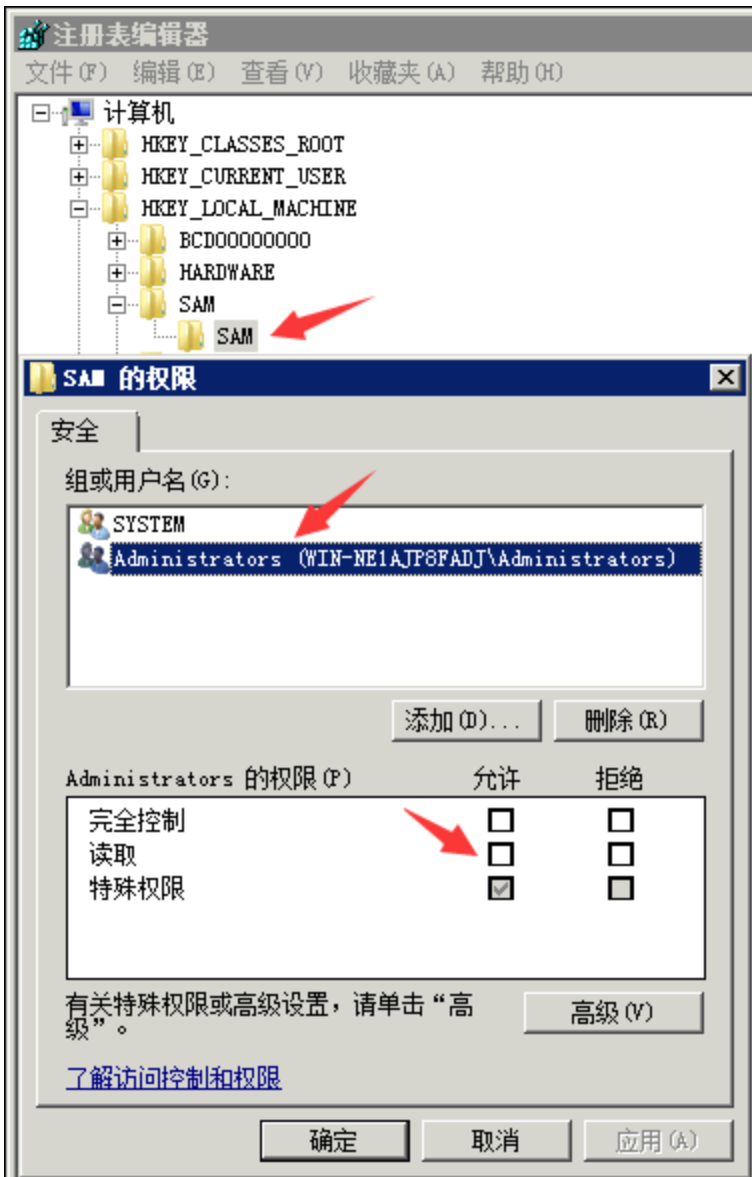
示例 1:

在 cmd 命令行中输入 regedit 打开注册表。



在注册表

【HKEY\_LOCAL\_MACHINE\SAM\SAM】处右键选择权限，点击【administrators】，勾选【读取】，并点击确定。



复制以下命令。

```
reg export
```

```
HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users c:\yanlian\user.reg
```

在命令行中粘贴将账户配置文件导出检查。

```
管理员: 命令提示符
C:\Users\Administrator>reg export HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users c:\yanlian\user.reg
操作成功完成。
```

根据导出的账户信息进行检查。

```
user.reg
139 [HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\Names]
140 @=hex(0):
141
142 [HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\Names\Administrator]
143 @=hex(1f4):
144
145 [HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\Names\Guest]
146 @=hex(1f5):
147
148 [HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\Names\WDAGUtilityAccount]
149 @=hex(3e9):
150
151 [HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\Names\yjy1]
152 @=hex(3e8):
153
154 [HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\Names\YunWei$]
155 @=hex(3ee):
```

记录该路径下的账户名称 16 进制数。

HKEY\_LOCAL\_MACHINE\SAM\SAM\Domains\Account\Users\Names

例如 YunWei\$ 是 3ee。可在以下注册表中检查其 F 值和 V 值

HKEY\_LOCAL\_MACHINE\SAM\SAM\Domains\Account\Users\000003EE

其中 F 为权限值，如将 administrator 的 F 值复制给 guest，并启用免密码登录，则会形成克隆账户的隐藏现象，V 为密码值。



```
user.txt
1
2 \\WIN-NE1AJP8FADJ 的用户帐户
3
4 -----
5 Administrator..... Guest..... WDAGUtilityAccount
6 yjy1...
7 命令成功完成。
8
9 别名      administrators
10 注释      管理员对计算机/域有不受限制的完全访问权
11
12 成员
13
14 -----
15 Administrator
16 Guest
17 WDAGUtilityAccount
18 命令成功完成。
19
```

## 痕迹检查

### 1.1 日志

操作系统日志中记录着攻击成功前和攻击后的相关痕迹。

注意点：日志默认记录 20Mb，达到最大值时优先覆盖旧事件，同时网络安全法要求日志至少保存 6 个月以上。

将安全日志和系统日志导出检查。

示例：

操作系统安全日志和系统日志默认存储路径：

%SystemRoot%\System32\Winevt\Logs\System.evtx

%SystemRoot%\System32\Winevt\Logs\Security.evtx

使用以下命令将日志导出。

```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>copy %SystemRoot%\System32\Winevt\Logs\System.evtx c:\yanlian\
已复制          1 个文件。

C:\Users\Administrator>copy %SystemRoot%\System32\Winevt\Logs\Security.evtx c:\yanlian\
已复制          1 个文件。
```

安全日志典型分析方法：大量事件 id 4625 后第一次 4624，则为操作系统密码暴力破解成功。其中源 ip 可能为空，则需要硬件资产管理系统支持，定位源工作站的名称，即主机名，特定情况下 windows 可通过 ping 主机名携带 -4 参数返回 ipv4 地址。或通过不对互联网映射操作系统远程桌面管理端口以及使用堡垒机进行管理可规避操作系统密码暴力破解攻击。

系统日志典型分析方法：id 为 7036 是服务启动 / 关闭的系统事件，在异常时间启动的服务需要重点检查，例如 wmi 服务。

## 1.2 文件落地

恶意程序保存在硬盘的文件系统中，例如后门程序，后门程序属于非授权的远程管理通道，黑客可通过该通道未经授权管理被入侵的主机，以及随后门文件同时生成的其他恶意程序。

注意点：文件落地不单独进行手工检查，一般根据【现象检查】或【持久化检查】的结果在硬盘中寻找恶意程序，否则工作量会过大及质量较差，常规恶意程序检查建议通过操作系统杀毒软件进行。

例如在 1999 年 01 月 01 日发现异常程序对其相关处置后，检查当天是否生成其他异常内容。

1 在 %temp% 目录下创建并修改 test.txt 文件的创建时间为 1999 年 01 月 01 日 00 点 00 分 00 秒，修改时间为 1999 年 01 月 02 日 00 点 00 分 00 秒、访问时间为 1999 年 01 月 03 日 00 点 00 分 00 秒；

2 通过 everything 工具进行检查。

示例 1:

修改文件时间：C:\Windows\system32\cmd.exe



修改文件时间 bat 脚本内容如下所示：

```
ChangeDate.bat
```

```
@echo off
```

```
powershell.exe -command "ls'%temp%\test.txt'| foreach-object {$_.CreationTime  
='01/01/1999 00:00:00'; $_.LastWriteTime ='01/02/1999 00:00:00'; $_.LastAccessTime  
='01/03/1999 00:00:00'}"
```

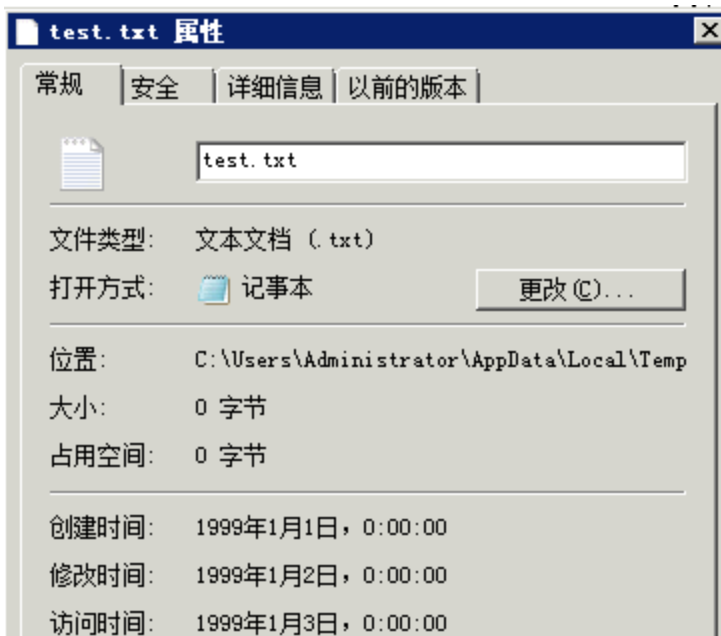
```
pause
```

```
#CreationTime 创建时间
```

```
#LastWriteTime 修改时间
```

```
#LastAccessTime 访问时间
```

修改后各项时间如下所示：



示例 2：

dc: 后跟日期，检查创建时间为 1999 年 01 月 01 日，后缀为 .txt 的文件。



dm: 后跟日期，检查修改时间为 1999 年 01 月 02 日，名称包含 te 和 . 的文件。



| 名称       | 路径  | 大小   | 创建时间          | 修改时间          | 访问时间          |
|----------|---|------|---------------|---------------|---------------|
| test.txt | C:\Users\Administrator\AppData\Local\Temp\2 | 0 KB | 1999/1/1 0:00 | 1999/1/2 0:00 | 1999/1/3 0:00 |

da: 后跟时间，检查访问时间为 1999 年 01 月 03 日的文件。

| 名称       | 路径  | 大小   | 创建时间          | 修改时间          | 访问时间          |
|----------|---|------|---------------|---------------|---------------|
| test.txt | C:\Users\Administrator\AppData\Local\Temp\2 | 0 KB | 1999/1/1 0:00 | 1999/1/2 0:00 | 1999/1/3 0:00 |

通过以上案例可知，在入侵取证时 windows 文件的所有时间完全不可信（linux 只是两项时间不可信），因此文件时间仅作为参考依据。

### 1.3 无文件落地

wmi 全称 windows 管理规范，其提供大量 api 接口供程序调用；同时也是恶意程序无文件落地的关键所在，黑客将恶意载荷存储于 wmi 中，并通过持久化机制在内存中通过 powershell 直接调用存储于 wmi 中的恶意载荷，从而实现无文件落地。

注意点：

1 可以将 Windows Management Instrumentation 服务关闭，并将其作为一个基线对象定期检查是否被恶意开启。因为 wbemtest.exe 或 powershell 均是通过该服务调用【wmi class】【对象】【属性】中的【值】；

2 当服务关闭时，wbemtest.exe 可打开 wmi 测试器，但无法连接命名空间，powershell 可正常运行，但不会执行 wmi class 中的载荷；

3 无文件落地不单独进行手工检查，一般根据【现象检查】或【持久化检查】的结果在 wmi class 中寻找恶意载荷，否则工作量会过大及质量较差，常规恶意程序检查建议通过操作系统杀毒软件进行。

检查 wmi class 是否存在异常：

1 搭建无文件落地场景；

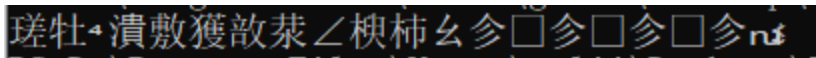
2 对无文件落地场景进行检查。

示例 1：

对以下字符串进行 base64 编码：

```
start powershell "ping 114.114.114.114 -t"
```

不能使用网页版 base64 编码工具进行编码，因为编码方式不同会导致 powershell 无法识别，上述命令通过网页版 base64 工具编码后再通过 powershell 识别，其结果如下所示：



因此可通过以下 powershell 工具将其 base64 编码。将命令复制到【\$string】中，当存在【\$】【"】符号时需要使用【`】进行转义，多行内容可直接回车。

```
ps_string_base64.ps1
```

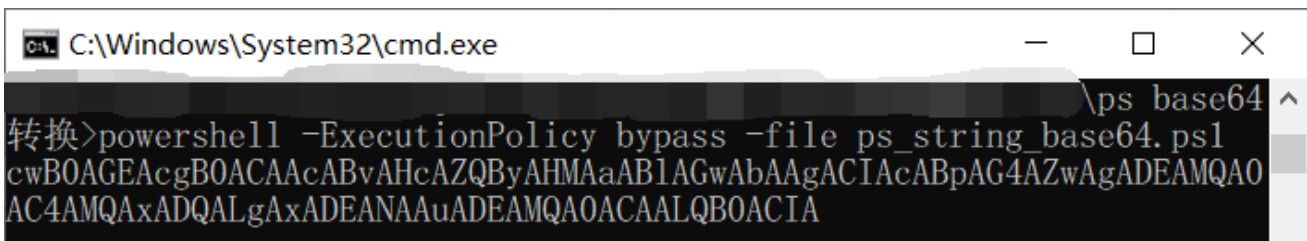
```
$string = "start powershell `ping 114.114.114.114 -t`"
```

```
$bytes = [System.Text.Encoding]::Unicode.GetBytes($string)
```

```
$encoded = [Convert]::ToBase64String($bytes)
```

```
echo $encoded
```

当前目录下打开 cmd 执行 powershell -executionpolicy bypass -file ps\_string\_base64.ps1 即可获得 base64 后的字符串。



打开 powershell 输入以下命令（可以将以下内容 base64 编码化，从 cc 主控端拉取执行，执行完毕即从内存中释放，这可能也是在被入侵主机中没有发现此类样本的原因），以下命令通过 powershell 调用 .net 的方法将 base64 字符串写入 wmiclass 同时执行。

```
#要连接的 wmi 命名空间及类对象
```

```
$SaveClass
```

```
= [System.Management.ManagementClass]
```

```
('root\default:StdRegProv')
```

```
#添加对象属性 (名称, 类型, 非数组)
```

```
$SaveClass.Properties.Add('ping',[System.Management.CimType]::String,$False)
```

```
#修改... 属性的值为... 64 字符串
```

#修改 ping 属性的值为 base64 字符串

```
$SaveClass.SetPropertyValue('ping','cwB0AGEAcgB0ACAAcABvAHcAZQByAHMAaABIAGw  
AbAAgACIAcABpAG4AZwAgADEAMQA0AC4AMQAxADQALgAxADEANAAuADEAMQA0AC  
AALQB0ACIA')
```

#保存

```
$SaveClass.Put()
```

#查询 ping 属性的值

```
$SaveClass.GetPropertyValue('ping')
```

#取载荷执行后退出

```
set ping
```

```
([WmiClass]'root\default:StdRegProv').Properties['ping'].Value;powershell -E $ping;exit
```

.net 帮助文档参考链接如下所示：

[https://docs.microsoft.com/zh-](https://docs.microsoft.com/zh-cn/dotnet/api/system.management.managementbaseobject.setpropertyvalue?view=dotnet-plat-ext-5.0#System_Management_ManagementBaseObject_SetPropertyValue_System_String_System_Object_)

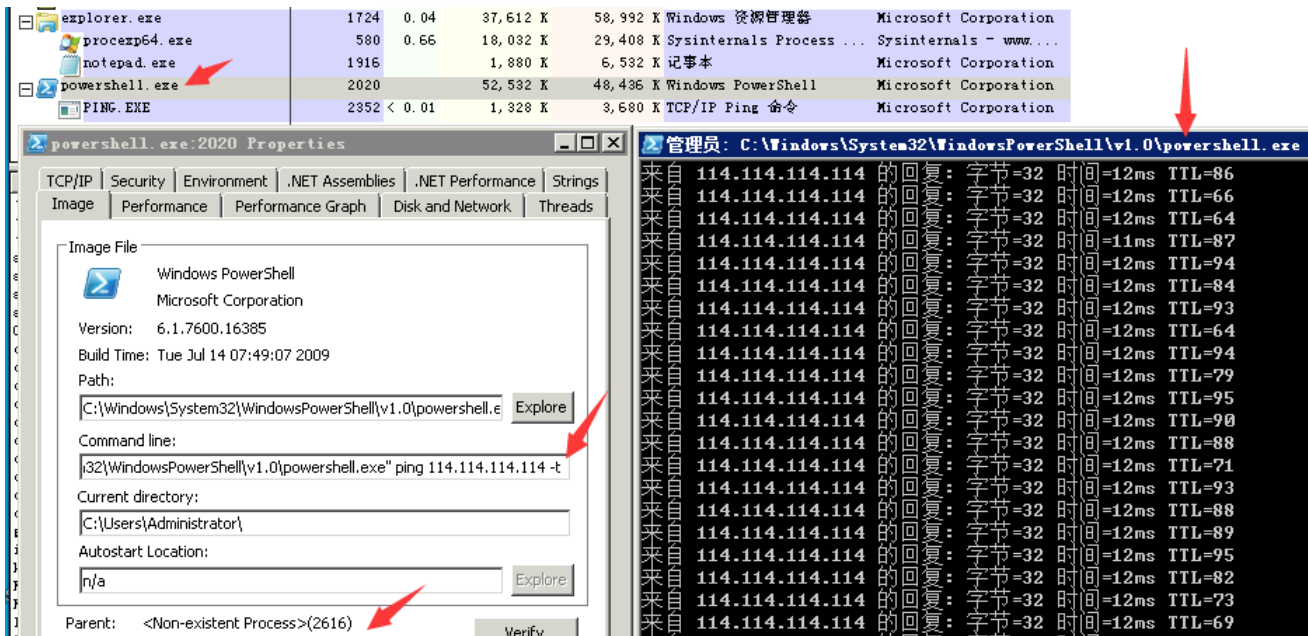
[cn/dotnet/api/system.management.managementbaseobject.setpropertyvalue?](https://docs.microsoft.com/zh-cn/dotnet/api/system.management.managementbaseobject.setpropertyvalue?view=dotnet-plat-ext-5.0#System_Management_ManagementBaseObject_SetPropertyValue_System_String_System_Object_)

[view=dotnet-plat-ext-](https://docs.microsoft.com/zh-cn/dotnet/api/system.management.managementbaseobject.setpropertyvalue?view=dotnet-plat-ext-5.0#System_Management_ManagementBaseObject_SetPropertyValue_System_String_System_Object_)

[5.0#System\\_Management\\_ManagementBaseObject\\_SetPropertyValue\\_System\\_String\\_System\\_Object\\_](https://docs.microsoft.com/zh-cn/dotnet/api/system.management.managementbaseobject.setpropertyvalue?view=dotnet-plat-ext-5.0#System_Management_ManagementBaseObject_SetPropertyValue_System_String_System_Object_)

通过此类方法可发现存储于 wmiclass 中的载荷已被执行，但却不见其父进程，严重影响到溯源的逻辑性，即无法根据痕迹溯源该程序如何启动，只能通过经验判断可能的自我守护机制。

如持久化机制未被发现，则无法根除，因此在【现象检查】伊始即强调，在未确认是否存在自我守护机制前，不可先行删除异常程序。逆向思维考虑：如结束进程后频繁自启动，则一定存在自我守护机制。

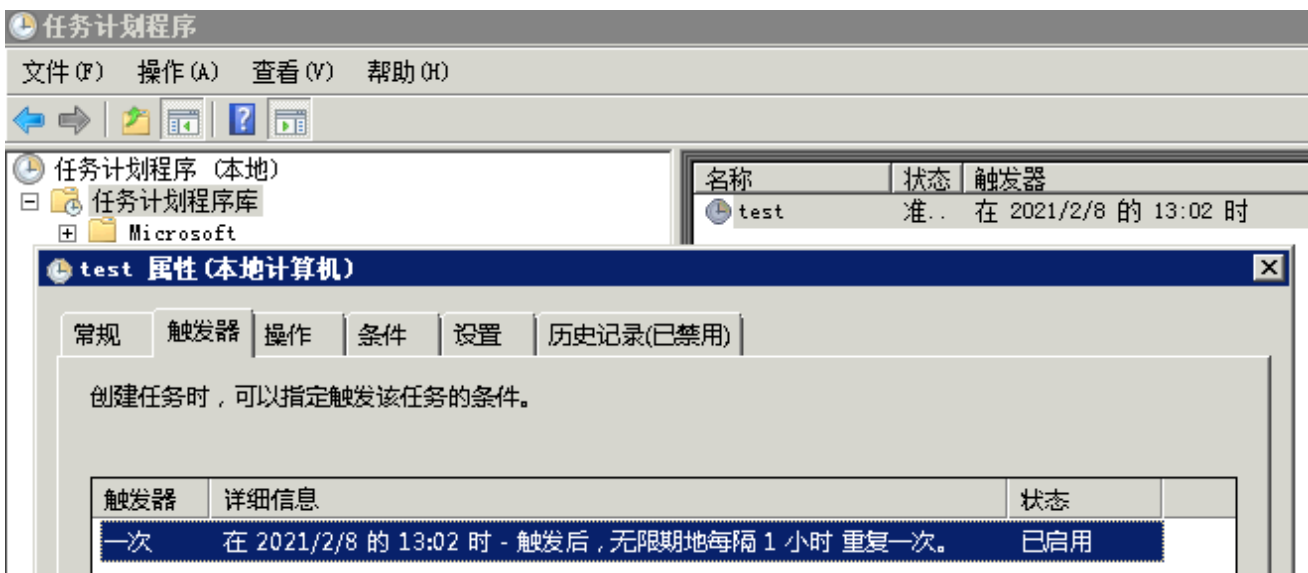


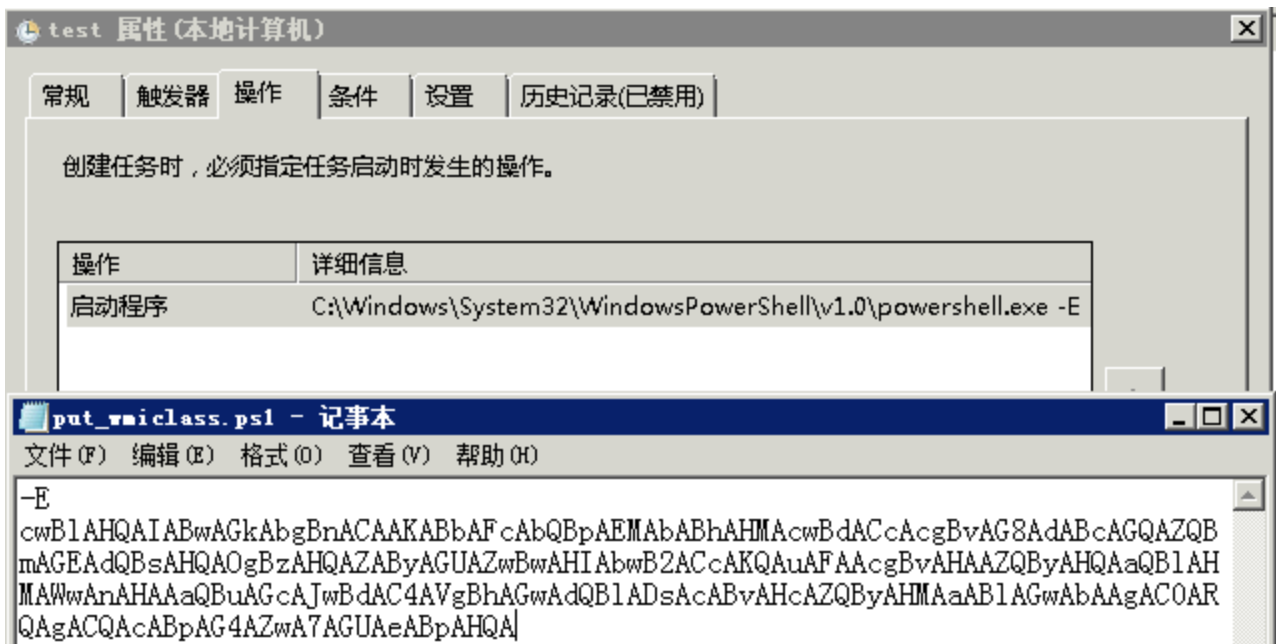
以下命令可在前台运行时将载荷带入后台运行，诸如任务计划运行的程序默认在后台运行：

```
start powershell -NoP -Nonl -W Hidden "ping 114.114.114.114 -t"
```

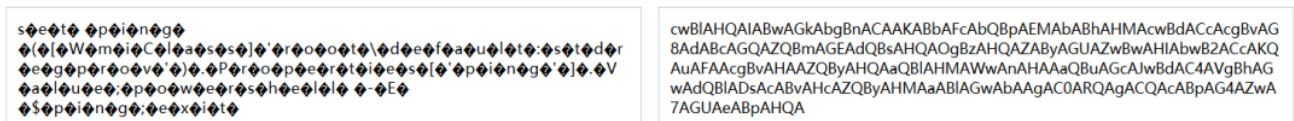
示例 2：

根据 pe 检查结果无法判断该进程的父进程，即无法确认该进程如何启动，仔细检查任务计划时发现异常：





在解码时不建议使用网页版 base64 解码工具，因为大部分网页工具默认会按多字节解码，导致每个字符后会再补一个字节内容，如下所示：



因此可通过以下 powershell 工具将其 base64 解码。将 base64 字符串复制到 **【\$string】** 中，GetString() 方法建议使用 Unicode，如改成 utf8，则会在每个字符后面补一个空格。

```
ps_base64_string.ps1
```

```
$string =
```

```
"cwB1AHQAIABwAGkAbgBnACAABbAFcAbQBpAEMAbABhAHMAcwBdACcAcgBvAG8AdABcAGQAZQBmAGEAdQBsAHQAQgBzAHQAZABYAGUAZwBwAHIAbwB2ACcAKQAuAFAAacgBvAHAZQByAHQAaQB1AHMAWwAnAHAAaQBuAGcAJwBdAC4AVgBhAGwAdQB1ADsAcABvAHcAZQByAHMAaAB1AGwAbAAgAC0ARQAgACQAcABpAG4AZwA7AGUAeABpAHQA"
```

```
$bytes =
```

```
[System.Convert]::FromBase64String($string);
```

```
$decoded =
```

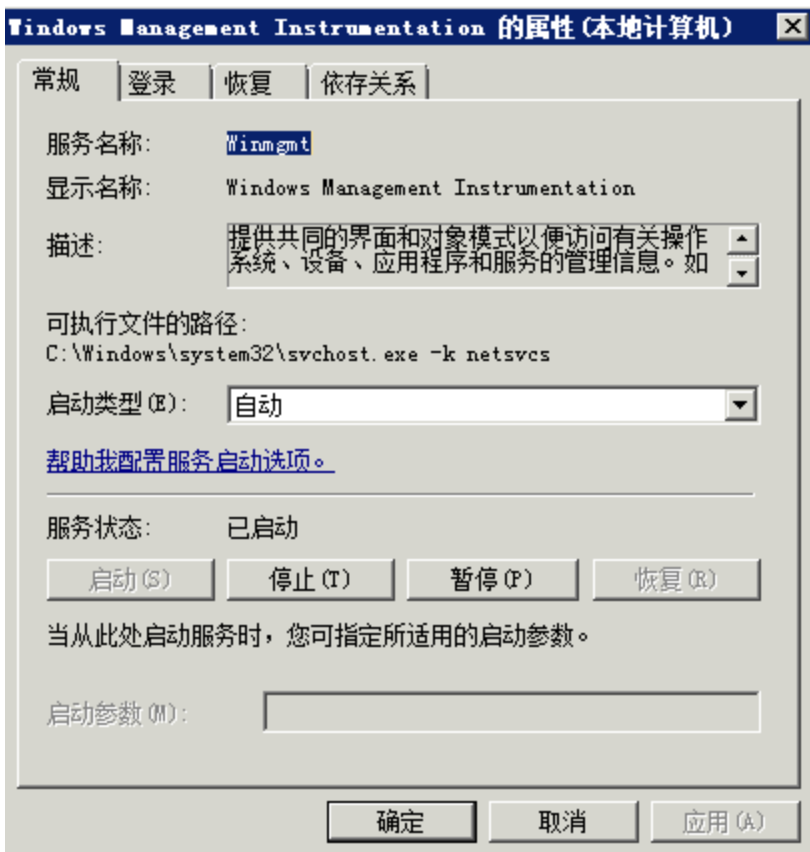
```
[System.Text.Encoding]::Unicode.GetString($bytes);
```

echo \$decoded

当前目录下打开 cmd 执行 powershell -executionpolicy bypass -file ps\_base64\_string.ps1 即可发现解码后的内容。

```
.\ps_base64转换> .\ps_base64_string.ps1  
set ping ([WmiClass]'root\default:stdregprov').Properties['ping'].Value;powershell -E $ping;exit
```

检查 wmi 服务，发现其已被未授权启动（默认为自动启动，建议关闭后建立基线值作为日常自动化巡检的对象）。



需判断 root\default 命名空间的 stdregprov 类对象的 ping 值是否存在异常。可打开 powershell 通过以下命令进行检查。

```
$SaveClass  
  
= [System.Management.ManagementClass]  
  
(root\default:StdRegProv)  
  
$SaveClass.GetPropertyValue('ping')
```

检查结果如下所示：

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

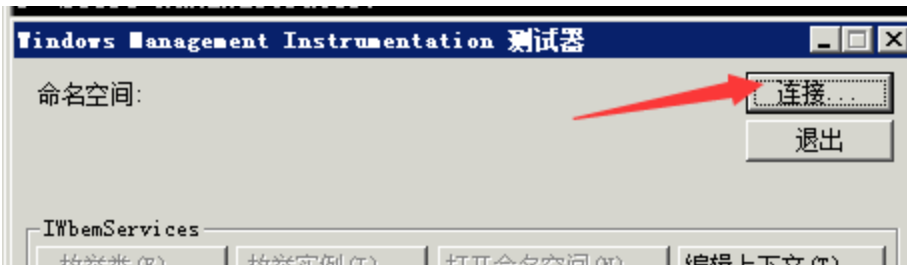
PS C:\Users\Administrator> $$SaveClass = [System.Management.ManagementClass]'root\default:StdRegProv'
PS C:\Users\Administrator> $$SaveClass.GetPropertyValue('ping')
cwB0A6EAcgB0ACAACABvAHcAZQByAHMAaAB1AGwAbAAgACIAcABpAG4AZwAgADEAMQA0AC4AMQAxADQALgAxADEANAAuADEAMQA0ACAALQB0ACIA
PS C:\Users\Administrator>
```

也可以打开 wbemtest.exe 工具进行检查。

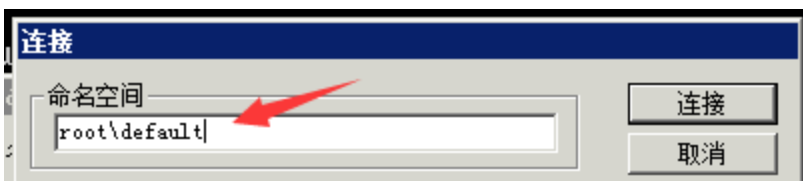
```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>wbemtest.exe
```

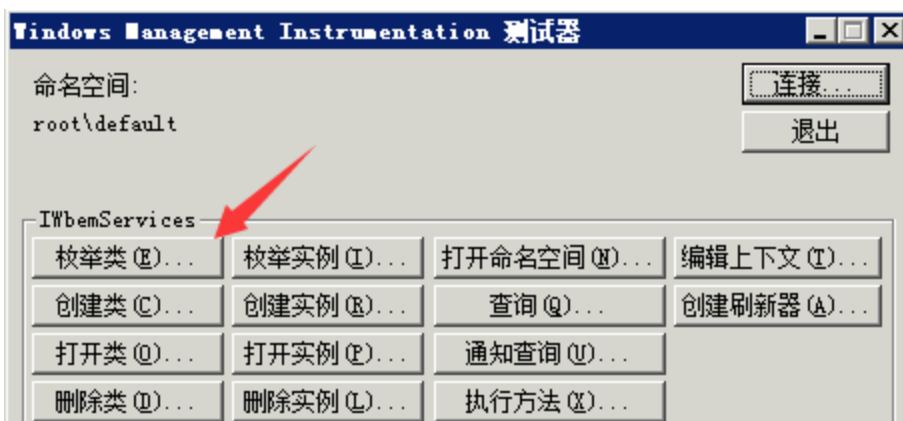
点击【连接】。



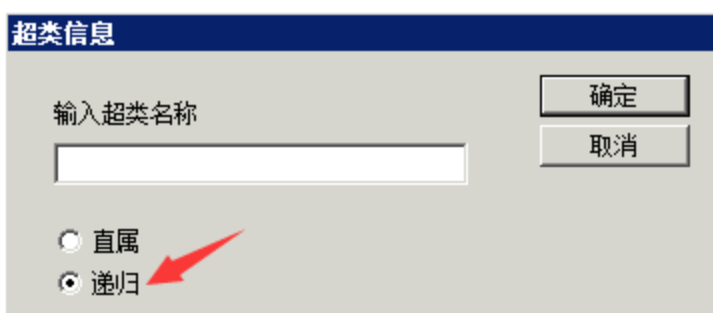
根据残留痕迹，输入【root\default】并点击【连接】。



点击【枚举类】。

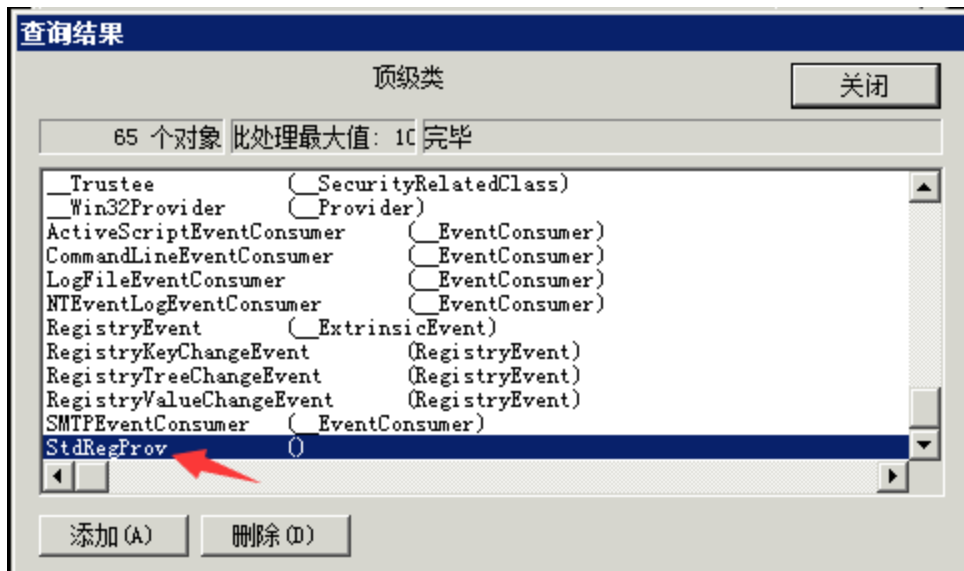


勾选【递归】并【确定】。

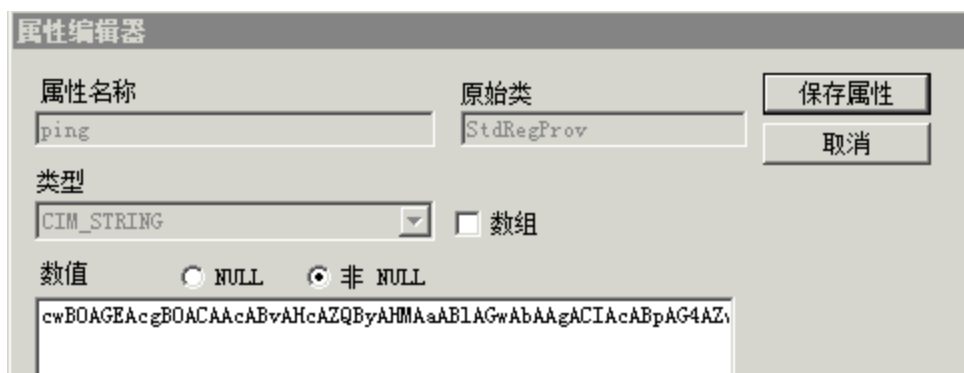
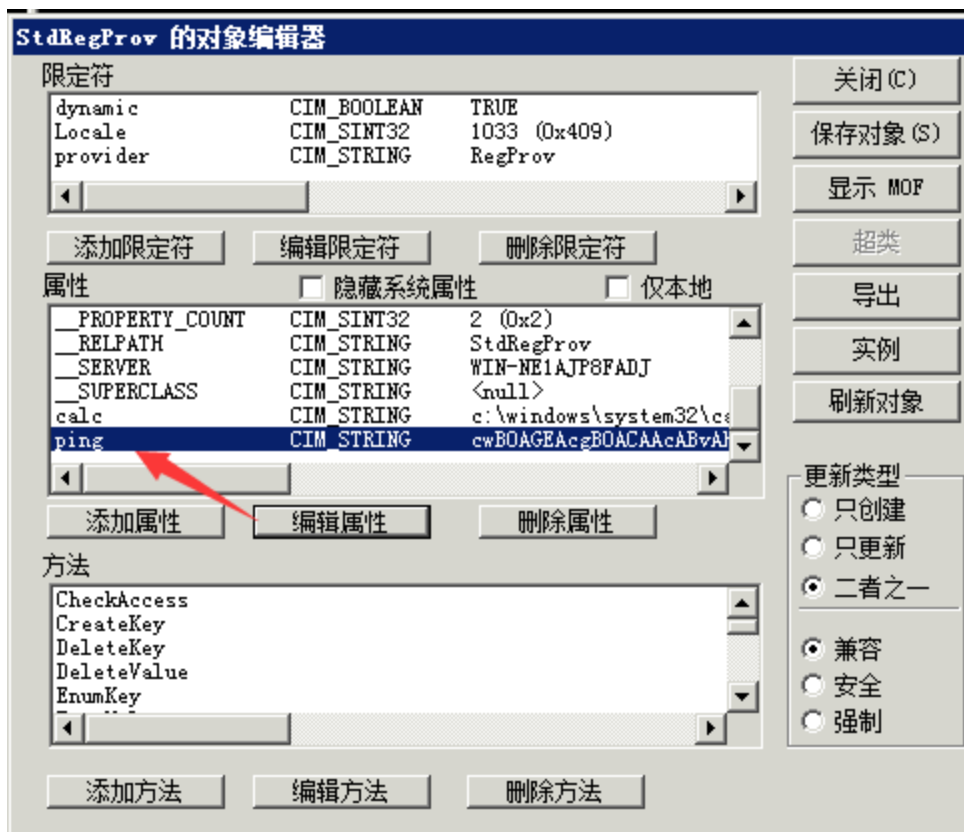


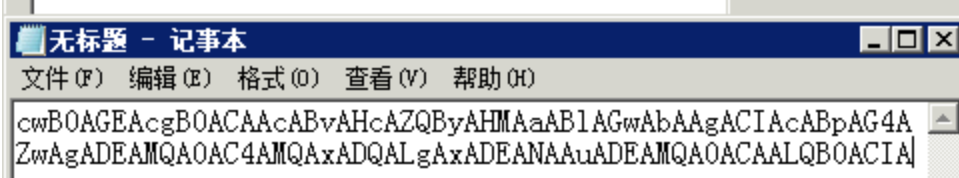


双击【stdregprov】类对象。



在【属性】中找到 ping 属性，双击编辑属性





最终对其进行解码，发现其功能和异常现象一致。

```
start powershell "ping 114.114.114.114 -t" \ps base64转换> .\ps_base64_string.psl
```

## 1.4 操作系统防火墙

windows 操作防火墙配置着允许 / 拒绝通行的入站 / 出站规则。可根据防火墙配置判断该主机在同网段内的可攻击面积。

注意点：检查配置前应先检查防火墙是否已启用。

检查操作系统防火墙是否存在异常：

1 通过 netsh 命令将防火墙状态导出检查；

2 通过 netsh 命令将防火墙所有入站、出站规则导出检查；

示例 1：

复制以下命令在 cmd 中执行。

```
netsh firewall show state > c:\yanlian\firewall-state.txt
```

```
netsh advfirewall firewall show rule name=all dir=in > c:\yanlian\firewall-in.txt
```

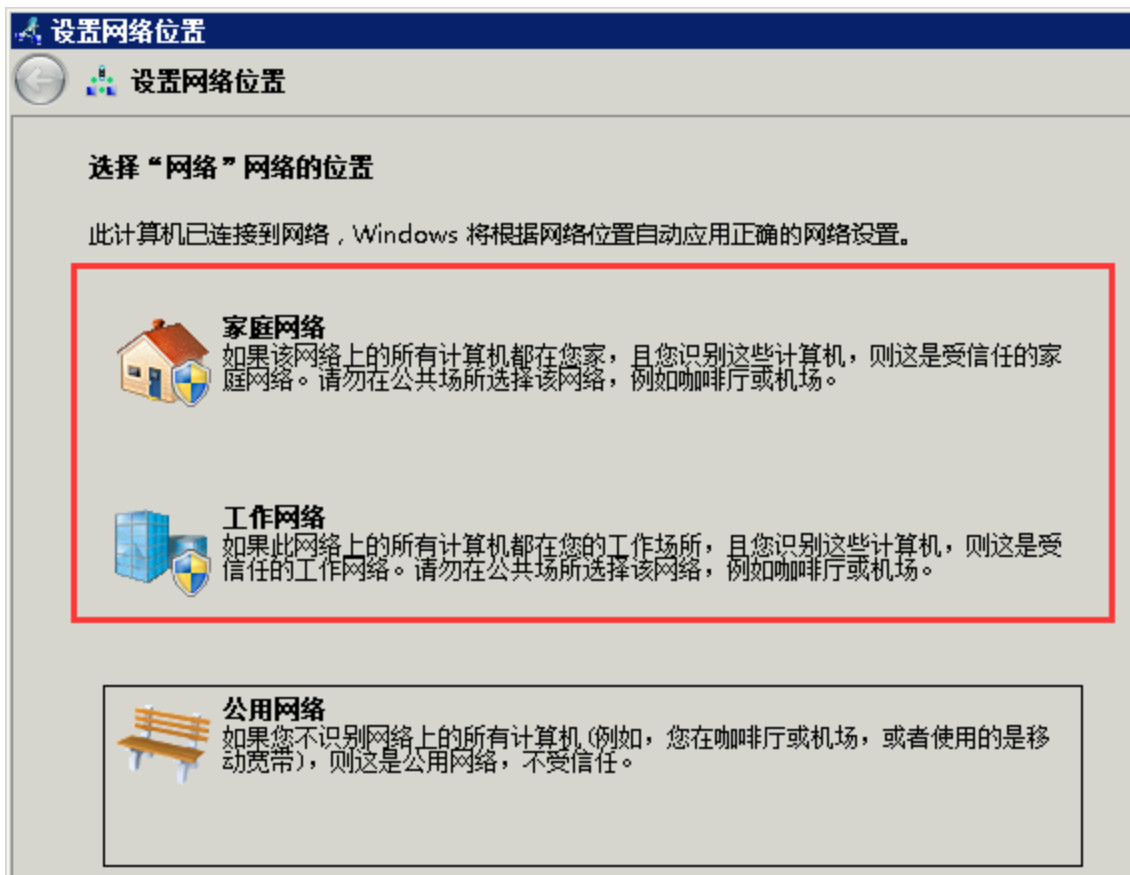
```
netsh advfirewall firewall show rule name=all dir=out > c:\yanlian\firewall-out.txt
```

【操作模式】为【禁用】意为防火墙已关闭，即已监听的端口同网段所有主机均可访问。

```
firewall-state.txt
1  防火墙状态:
2  -----
3  配置文件                = 标准
4  操作模式                = 禁用
5  例外模式                = 启用
6  多播/广播响应模式      = 启用
7  通知模式                = 启用
8  组策略版本              = Windows 防火墙
9  远程管理模式          = 禁用
10
11  所有网络接口上的端口当前均为打开状态:
12  端口  协议  版本  程序
13  -----
14  当前没有在所有网络接口上打开的端口。
```

【操作模式】的启用和禁用随网卡的网络位置变更而变更。

例如网卡的网络位置是【家庭网络】或【工作网络】，则匹配家庭或工作（专用）网络位置防火墙的状态。或网卡的网络位置是【公用网络】，则匹配公用网络位置防火墙的状态。

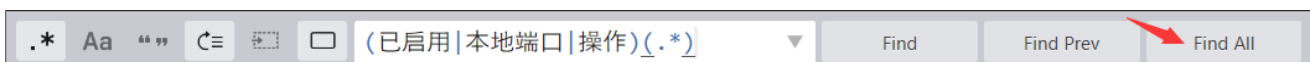




示例 2:

复制以下正则进行搜索和结果复制:

(已启用 | 本地端口 | 操作)(.\*)



复制以下正则进行结果替换。

操作 (.)

操作 \$1\n



即可对入站、出站规则进行检查分析，且工作量也会大大减少。

|      |        |                          |
|------|--------|--------------------------|
| 1297 | 规则名称:  | 远程桌面 - RemoteFX (TCP-In) |
| 1298 | -----  | -----                    |
| 1299 | 已启用:   | 是                        |
| 1300 | 方向:    | 入                        |
| 1301 | 配置文件:  | 域, 专用, 公用                |
| 1302 | 分组:    | 远程桌面 - RemoteFX          |
| 1303 | 本地 IP: | 任何                       |
| 1304 | 远程 IP: | 任何                       |
| 1305 | 协议:    | TCP                      |
| 1306 | 本地端口:  | 3389                     |
| 1307 | 远程端口:  | 任何                       |
| 1308 | 边缘遍历:  | 否                        |
| 1309 | 操作:    | 允许                       |
| 1310 | 确定。    |                          |
| 1311 |        |                          |

|     |       |      |
|-----|-------|------|
| 351 | 已启用:  | 是    |
| 352 | 本地端口: | 3389 |
| 353 | 操作:   | 允许   |
| 354 |       |      |
| 355 | 已启用:  | 是    |
| 356 | 本地端口: | 3389 |
| 357 | 操作:   | 允许   |
| 358 |       |      |

windows 终端进站规则应根据实际工作需要通过网络层访问控制，windows 服务端出站规则应根据实际工作需要通过网络层访问控制。

### 1.5 操作系统杀毒软件

杀毒软件可用于防护恶意程序的存储、运行等恶意行为，作为主管单位对下辖单位进行攻击溯源时需进行检查。

注意点：

- 1 应检查杀毒软件的安装时间；
- 2 应检查杀毒软件的补丁更新情况；
- 3 应检查杀毒软件的特征库是否最新；
- 4 应检查杀毒软件最近的杀毒记录；

应检查杀毒软件的白名单。

