原文地址:http://drops.wooyun.org/papers/4918

0x00 前言

这篇文章讲述了微软最新修补的漏洞CVE-2015-0057的细节。

原文：《One-Bit To Rule Them All: Bypassing Windows' 10 Protections using a Single Bit》

链接：http://breakingmalware.com/vulnerabilities/one-bit-rule-bypassing-windows-10-protections-using-single-bit/

今天微软发布了最新的补丁。这个补丁修复了漏洞CVE-2015-0057，这是我们几个月前负责任地披露给微软的一个重要级别的可利用漏洞。作为研究的一部分，我们发现这个权限提升漏洞，利用该漏洞，可使攻击者完全控制权Windows系统。换句话说，获得Windows访问权的攻击者（例如，通过网络钓鱼运动）可以利用此漏洞绕过Windows所有的安全措施，击败缓解措施，如沙箱，内核隔离和内存随机化。 有趣的是，漏洞利用仅需要修改Windows操作系统一个比特。

我们已经证实了这一漏洞对所有支持的Windows桌面版本都是可用的，包括Windows10技术预览版。

下面开始漏洞的细节。起初，我们似乎无法利用它。但是经过一番努力，我们成功地利用了它，我们将描述整个过程。作为分析的一部分，我们也展示了利用视频。最后，通过分享我们认为有趣的死代码bug来总结文章。

负责任的披露：尽管这篇博客是技术性的，但我们不会发布任何代码，或完整的细节，以防止任何人重现漏洞。

0x01 背景

在过去的几年里，权限提升漏洞对于漏洞利用来说变得更加重要了，因为它使恶意代码可以在内核权限上运行。因此，攻击者利用权限提升漏洞可以绕过安全防护机制，如应用程序沙箱。

伴随着攻击技术的发展，微软做了大量的努力来保护内核。其理由是，即使漏洞存在，利用也是很困难的，但不是不可能。 例如，这里仅仅是少数存在于Windows 8.1的内核保护机制：

内核DEP - 确保大多数内核数据区不能被执行

KASLR - 随机化内核地址空间，以避免内核模块位置固定

完整性等级 - 限制非特权应用程序泄漏内核相关信息的能力

常见攻击向量的缓解 – 常被滥用的结构的保护（如Win32K WND PROC域）

SMEP - 防止执行内核模式到用户模式的控制转移

NULL解引用保护 - 禁止用户空间前64K数据的映射

尽管存在这些保护机制，在过去的一年中，我们已经看到了一些演示，展示了绕过这些保护机制的技术。

我们在本文中描述的漏洞，就是新近披露的绕过这些保护的可利用的权限提升漏洞。

漏洞：Win32k.sys中的模块的一个漏洞

这一漏洞出现在微软Windows内核GUI组件中，即Win32k.sys模块中。

本文假设对Win32k.sys模块有很强的技术认识。此模块的详细信息，请参考Tajei Mandt的Kernel Attacks through UserMode Callbacks，Gilad Bakas的Windows Kernel Vulnerability Research and Exploitation。

0x02 了解窗口滚动条

WIN32K模块实际上管理着windows滚动条。这些滚动条 - 无论是水平或垂直 – 都是为每个窗口设置的。

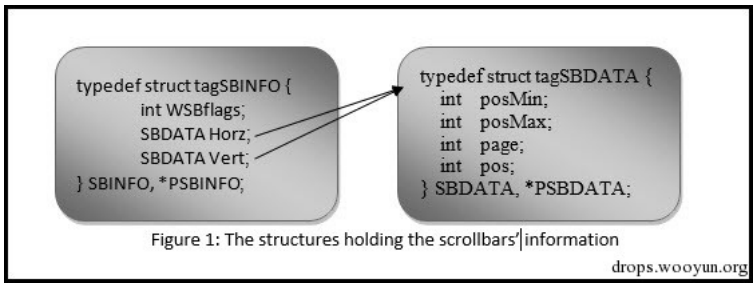


图1

正如图1中看到的，每个SBDATA结构定义关于一个滚动条的信息。

WSBflags是一个位掩码，决定了滚动条的状态。

为了启用和禁用一个窗口滚动条，可以使用xxxEnableWndSBArrows函数。通过简单的函数调用，这个函数就可以改变滚动条的状态。 漏洞正是存在于此功能中。

0x03 深入探究 xxxEnableWndSBArrows

xxxEnableWndSBArrows原型如下：

wnd - 一个指向相关窗口的指针

wSBflags - 滚动条类型（例如水平或垂直）

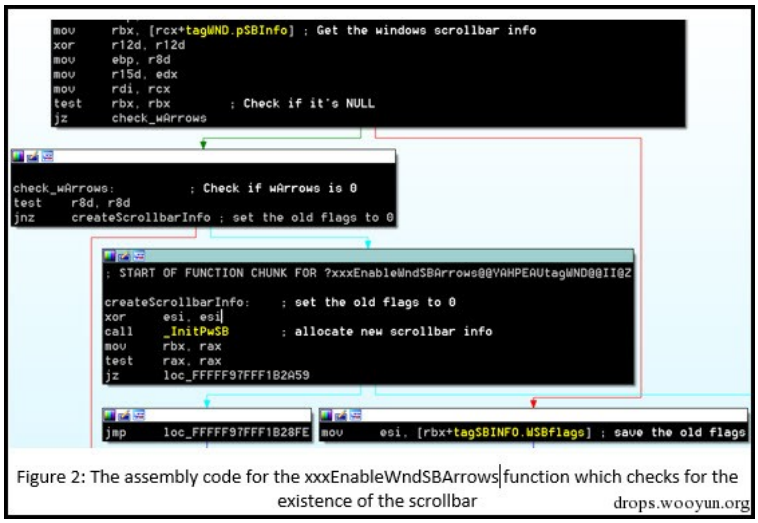
wArrows - 指定滚动条的箭头是否启用或禁用，并指示那个箭头启用或禁用。

为了描述漏洞，我们就来看看在xxxEnableWndSBArrows功能的第一部分，可细分为3个逻辑部分的：

1. - 分配一个新的滚动条（如果需要）

该功能首先通过检查该窗口是否已经存在滚动信息，并根据需要分配一个新的滚动条信息结构。

从技术上讲，该函数读取pSBInfo域（这个字段指向tagSBINFO结构），测试指针是否为NULL。如果该字段为null和wArrows参数不是NULL，则为窗口分配一个tagSBINFO结构，滚动条的旧标志被设置为0，否则从现有的窗口滚动条的信息中复制旧的标志。该代码可以在图2中找到。



2. 设置水平滚动条的状态

流程继续测试是否应该改变水平滚动的状态。

根据设定的wArrows参数是什么，函数启用或禁用的箭头（图3）。

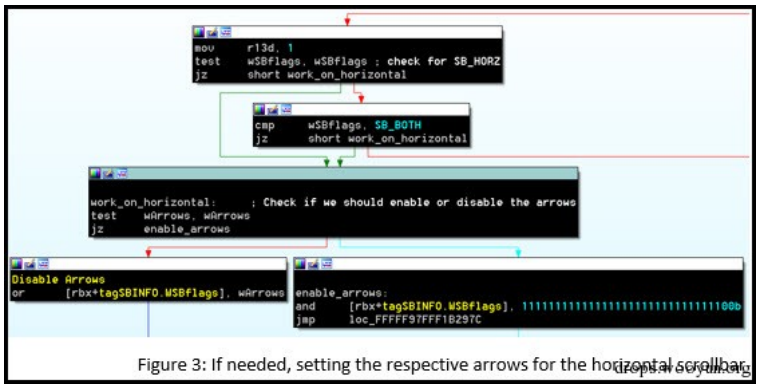


Figure 3: If needed, setting the respective arrows for the horizontal scrollbar

3. 测试滚动箭头的状态

流程继续检查箭头的状态是否发生了变化。

从技术上讲，这是通过检查箭头的标志来做的（注意，有几个标志的检查 - 但这些对于我们的目的来说都不是很有趣）。如果标志已经改变，同时该窗口可见，则xxxDrawScrollbar被调用。

这正是事情变得有趣的地方。

当深入研究代码，似乎xxxDrawScrollBar可能会导致用户模式回调（图4）。

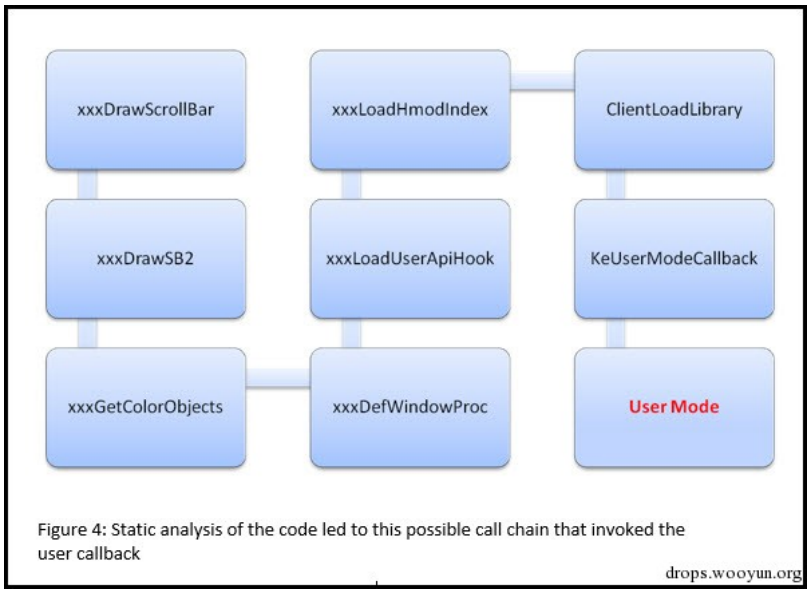


Figure 4: Static analysis of the code led to this possible call chain that invoked the user callback

调用链中的关键函数是ClientLoadLibrary。这个函数执行回调到用户模式函数__ClientLoadLibrary。

现在，让我们回到xxxEnableWndSBArrows的代码。

我们的检查表明tagSBINFO指针在回调后，不任何验证就使用了。最终，这可能会导致使用后释放（UAF）漏洞，因为该函数可能继续使用被释放了的滚动条信息（图5）。

Figure 5: This is the flow that could lead to a Use After Free (UAF) vulnerability

0x04 漏洞利用：操纵窗户属性

回调之后，xxxEnableWndSBArrows函数继续并且改变垂直滚动条的状态。

在这个阶段中，函数试图启用或禁止标志。然而，由于该结构已经被释放，我们可以用它来将位于释放的缓冲区的第一个DWORD与0xC按位或（如果我们禁止箭头）或者清除第3位和第4（如果启用箭头）。见图6。

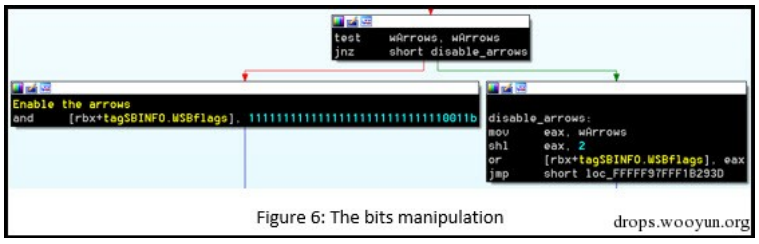


Figure 6: The bits manipulation

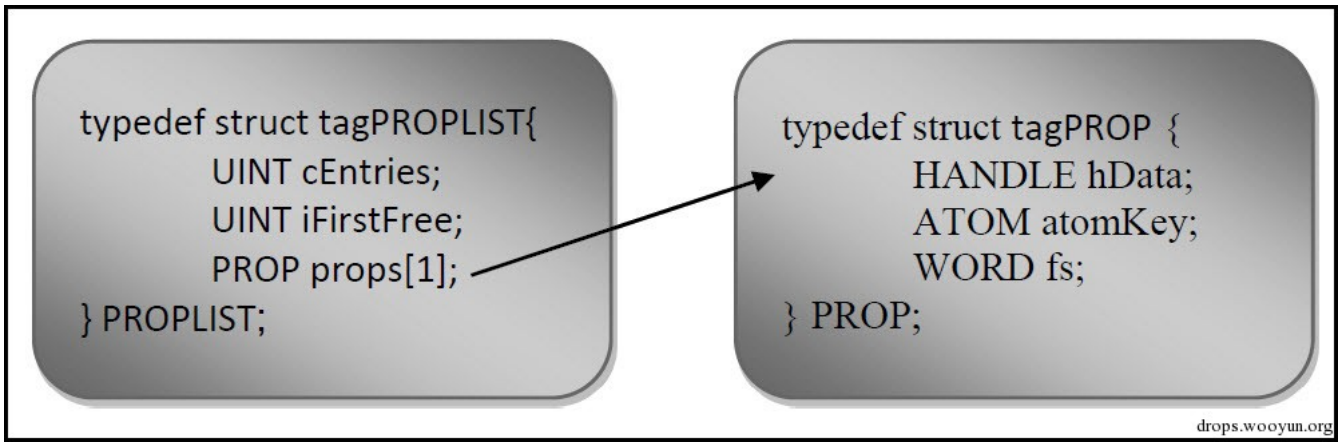
为了简便起见，我们将展示如何操纵2个比特位来完全控制系统。其实，操纵其中之一就足够了。

似乎位操作起初还不足以造成任何显著影响，但我们还是决定继续尝试。最需要去尝试的事情是，或者增加一些缓冲（使用按位OR）的大小或者减少一定的参考计数器（使用按位AND）。

短暂的搜索后，我们发现了满足第一个要求的对象。这个对象是一个窗口的属性列表。

在窗口属性列表

每个窗口都有一个属性列表。通常GUI应用程序可以使用这些属性来存储任意值， Win32K也使用属性列表以便存储内部数据。



用于保存窗口的属性的数据结构可以在图7中看到。第一个域cEntries，是属性数组中的条目数; iFirstFree是属性列表中的第一空闲单元的索引;和 props 是PROP数组。

应用程序可以使用SetProp API设置窗口的属性。该函数的原型如下：

```
BOOL WINAPI SetProp(HWND hWnd, LPCTSTR lpString, HANDLE hData);
```


hWnd - 窗口句柄。

lpString – 属性或ATOM。

HDATA – 要存储的数据。

添加属性到一个窗口是通过CreateProp函数来完成，它在Win32K模块中。

如图8所示它的分配算法是相当简单的。如果没有空间可用于属性列表中的新条目，则函数分配多一个条目的新列表。该函数然后将旧属性的缓冲区复制到新的缓冲区，释放旧的缓冲区，并增加条目计数。

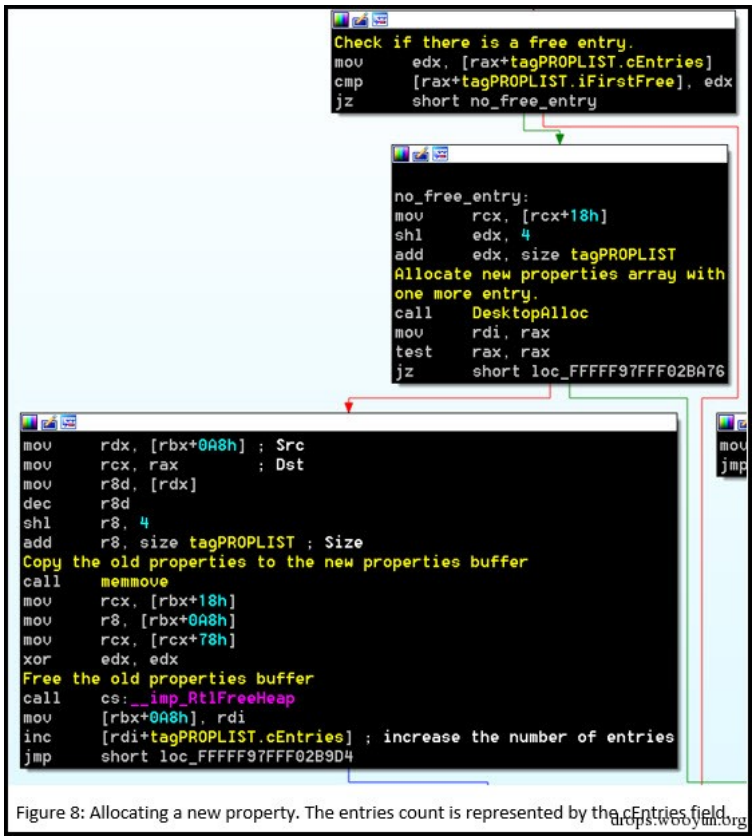


Figure 8: Allocating a new property. The entries count is represented by the cEntries field.

这段代码中还有几个重要的点要注意： 首先，属性是从桌面堆中分配（使用DesktopAlloc）。tagSBINFO也从该堆中分配。如果我们想用UAF

漏洞来改变属性结构，这是至关重要的。 第二，每一个新的条目都触发缓冲区的重新分配。这意味着我们可以很容易地触发缓冲区的重新分配，来达到tagSBINFO结构的大小。这样做增加了缓冲区将被分配在释放了的tagSBINFO结构的地址的机会。 第三，也是最重要的是，cEntries域位于该结构的第一个DWORD。这意味着我们可以增加它的大小（用按位OR）。增加属性数组的大小后，我们基本上实现了经典的缓冲区溢出。

0x05 概念证明视频

上面研究了权限提升漏洞的利用。我们就此停止，同时避免发布任何敏感代码。 我们在64位Windows10技术预览版上提供了概念证明。

视屏地址：https://www.youtube.com/watch?v=ukAr6MiA788

0x06 总结

经过一番工作，我们成功地创建针对Windows所有版本的一个稳定利用 - Windows XP到Windows10预览都可用（SMEP和其他保护都打开）。我们已经表明即使是微小的缺陷也可以用来完全控制任何Windows操作系统。

尽管如此，我们认为微软在使其操作系统更安全方面做的努力起到了显著作用，编写可靠的攻击代码比以往更难了。 不幸的是，这些措施不能完全阻止攻击者。我们预计，攻击者将继续整合漏洞利用到他们的犯罪工具包，使得妥协在所难免。

最后旁注：有趣的代码

检查xxxEnableWndSBArrows函数的代码，可以看到有调用xxxWindowEvent函数。

乍一看，似乎这两个函数会比xxxDrawScrollbar函数更容易用于漏洞利用，具体如上所述。

但是，在深入研究代码之后，很快就明白，在代码中的横向滚动条部分调用xxxWindowEvent实际上是死代码（图9）。

Figure 9: A bug in xxxEnableWndSBArrows which leads to dead-code drops.wooyun.org

看代码，有两个条件调用xxxWindowEvent函数。仅当滚动条信息的旧标记与新的标志不同时，这些调用才执行。然而，在这些条件出现时，旧的标志的值与新的标志总是相等的。因此，调用xxxWindowEvent的条件是永远不会满足的。这实际上意味着，这个死码在那里大约15年没有做任何事。