**CVE-2018-8174漏洞利用分析**

**1. POC**

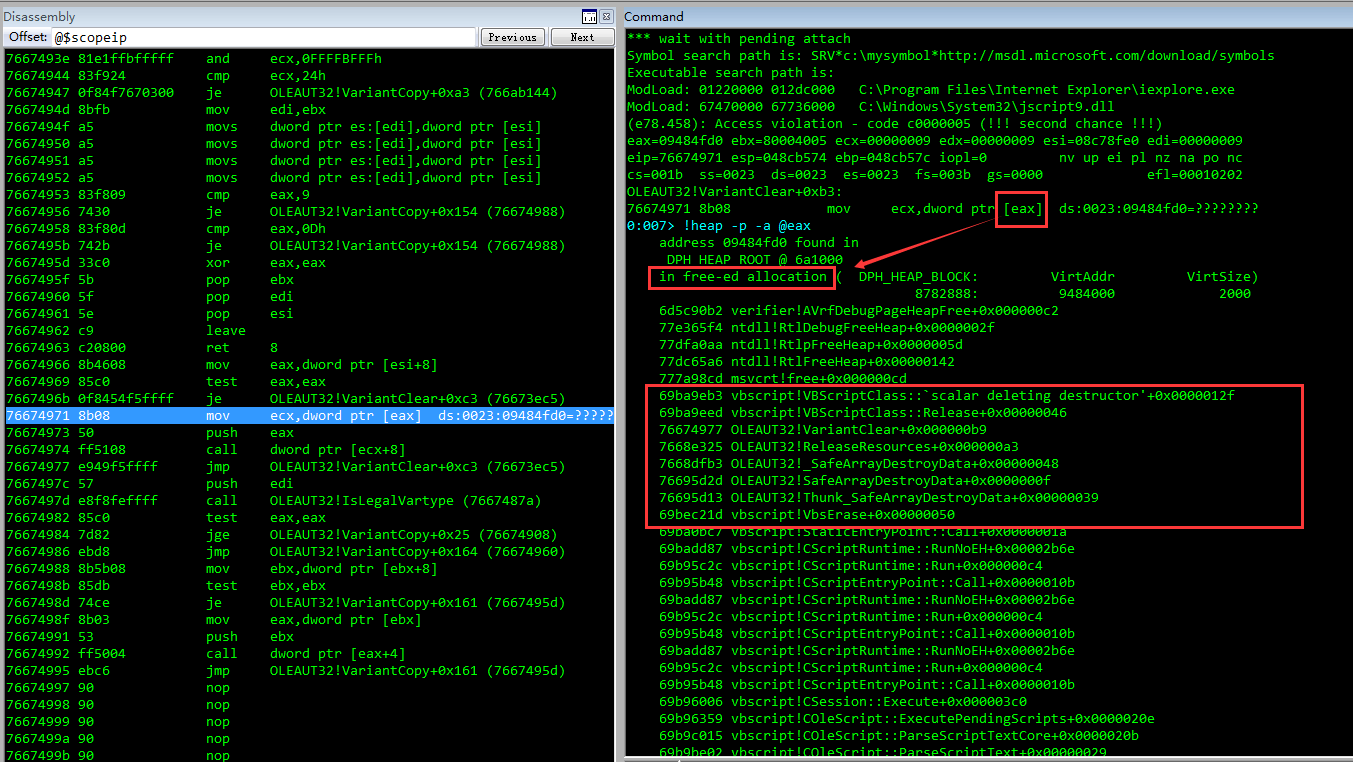
|  |
| --- |
| <!doctype html>  <html lang="en">  <head>  <meta http-equiv="x-ua-compatible" content="IE=10">  </head>  <body>  <script language="vbscript">  Dim a  Dim b(1)  Class MyClass  Private Sub Class\_Terminate()  Set b(0) = a(1)  a(1) = 1  End Sub  End Class  Sub PrepareVuln  ReDim a(1)  Set a(1) = New MyClass  Erase a  End Sub  Sub TriggerVuln  b(0) = 0  End Sub  Sub StartExploit  PrepareVuln  TriggerVuln  End Sub  StartExploit  </script>  </body>  </html> |

Poc首先定义两个数组a b并声明了一个类MyClass，MyClass中重载了析构函数Class\_Terminate，在PrepareVuln中，创建了一个MyClass的实例赋值给a(1)，通过Erase a清空a中的元素，这时候会触发Class\_Terminate的调用，在Class\_Terminate中增加了一个b(0)对MyClass实例的引用，再通过a(1) = 1删除a(1) 对MyClass实例的引用，这时候MyClass实例会被释放，但是b(0)仍然保留了一个MyClass实例的引用，最终在TriggerVuln中通过b(0) = 0触发漏洞。

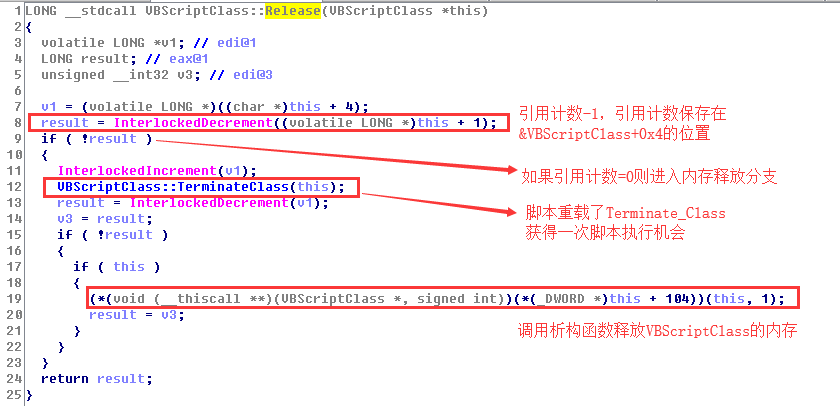
**2. 漏洞原理**

该漏洞原理和CVE-2018-1004类似，都是通过重载类析构函数Class\_Terminate，在自定义析构函数中对即将被释放的内存进行操作，而造成的内存破坏。

开启hpa和ust，观察漏洞现场：



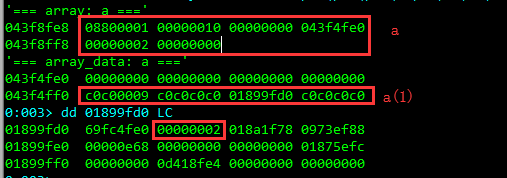
显然eax已经在vbscript!VbsErase中被释放了，通过ust可以看出eax保存的可能是VBScriptClass对象也就是脚本中的MyClass实例的地址。类似mshtml.dll，微软通过引用计数来进行垃圾回收，这里对应的函数是VBScriptClass::Release：



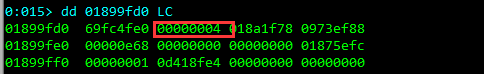
VBScriptClass::Release中首先对VBScriptClass的引用计数-1（&VBScriptClass+0x4），如果引用计数=0（vbscript!VbsErase后会多次调用VBScriptClass::Release递减VBScriptClass的引用计数）则调用VBScriptClass::TerminateClass，调用VBScriptClass::TerminateClass时因为在脚本中重载了Class\_Terminate函数，所以获得了一次脚本执行的机会，这里就可以在释放VBScriptClass的内存前将即将释放的内存地址保存脚本控制的变量中(Set b(0) = a(1))，并通过a(1) = 1平衡引用计数，使得最终释放内存。

观察VBScriptClass的引用计数的变化：

1. Set a(1) = New MyClass 后MyClass的引用计数为2：

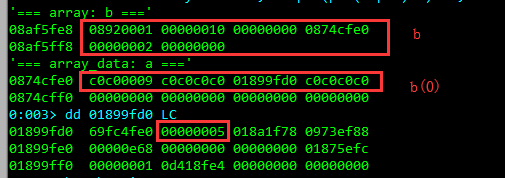


1. Erase a后MyClass的引用计数为4：

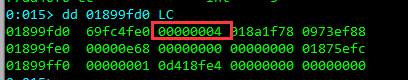


1. 进入Erase a后根据之前的分析会递减VBScriptClass的引用计数，最终在引用计数=0时进入TerminateClass：

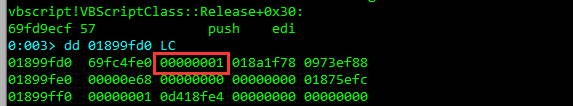
Set b(0) = a(1) 后MyClass的引用计数为5：



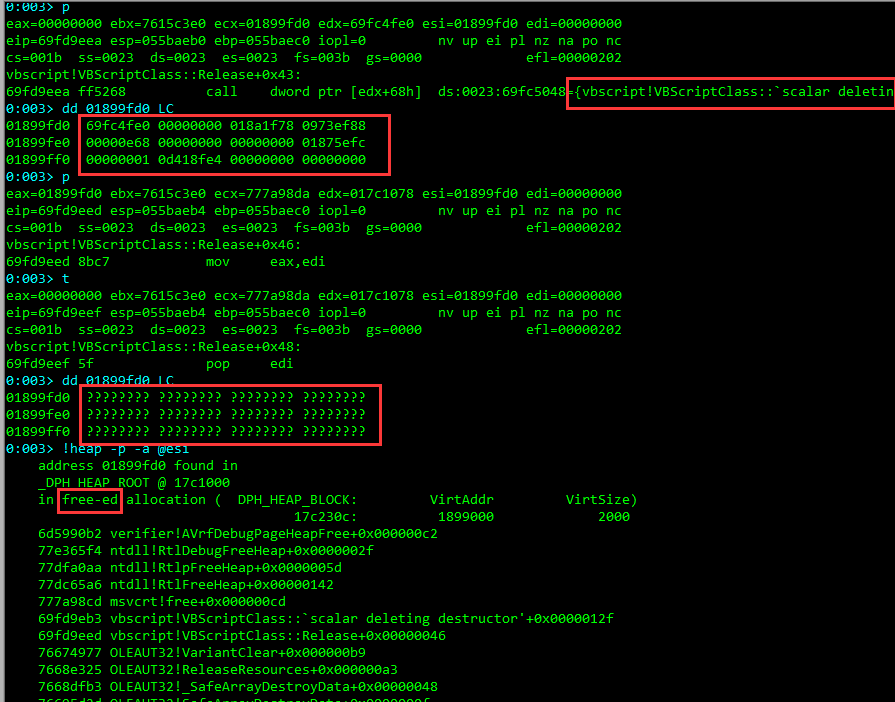
a(1) = 1后又将MyClass的引用计数还原为4：



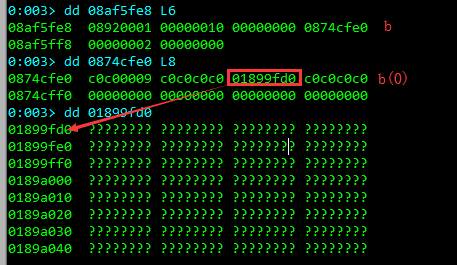
TerminateClass返回后MyClass的引用计数为1



最终调用析构函数释放内存：



此时b(0)中仍然保存了这块地址:



最终造成UAF漏洞。