**CVE-2018-1004漏洞利用分析**

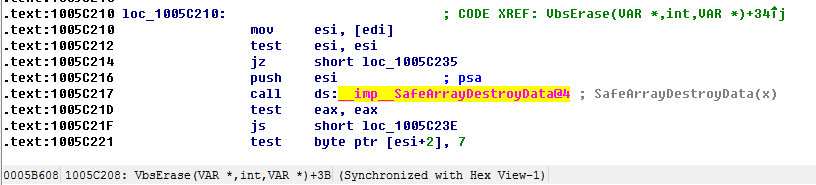
**1. POC**

|  |
| --- |
| <html>  <meta http-equiv="x-ua-compatible" content="IE=8">  <meta http-equiv="Expires" content="-1">  <script language="VBScript">  redim a(&h400000)  Class MyClass  Private Sub Class\_Terminate  ReDim Preserve a(100)  End Sub  End Class  Set a(0) = new MyClass  Erase a  </script>  </html> |

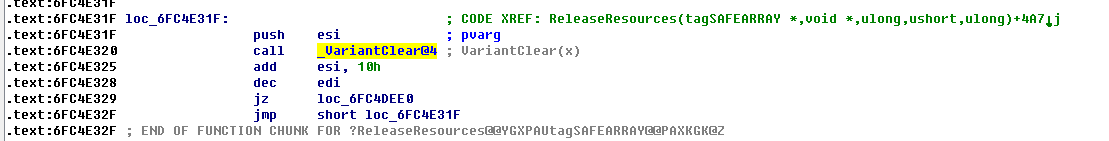
poc比较简单，大概流程是创建一个长度为0x400000的数组a，并将a的第一个元素指向MyClass的一个实例，MyClass注册了析构函数Class\_Terminate会将a的长度修改为100，最后通过Erase a清空a内的数据。

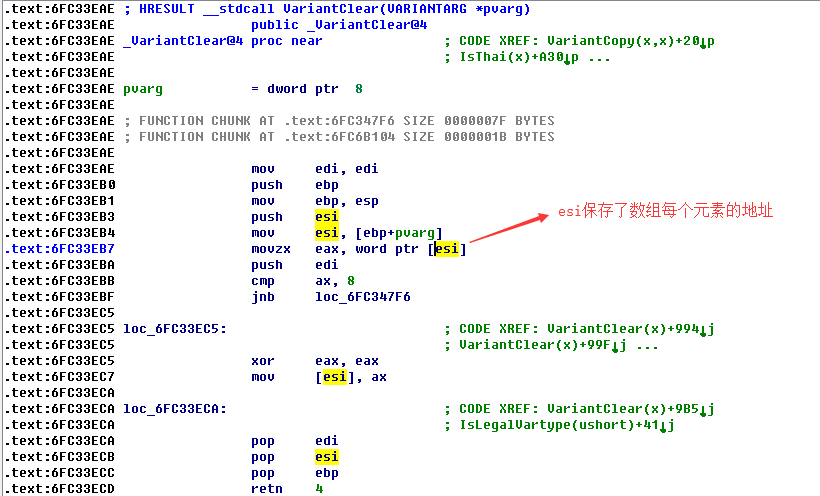
**2. 漏洞原理**

“Erase a”代码会调用vbscript!VbsErase函数，然后调用OLEAUT32! \_SafeArrayDestroyData计算释放数组大小再调用OLEAUT32!ReleaseResources释放数组元素的数据，最终调用OLEAUT32!VariantClear释放数组的每个元素：

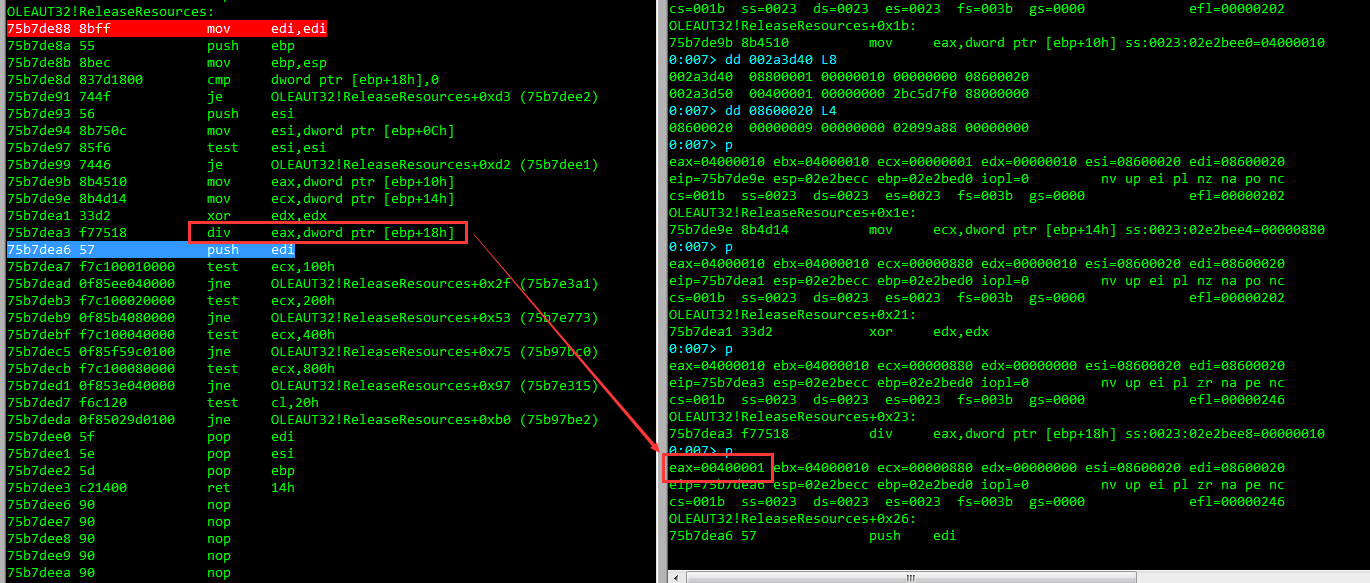


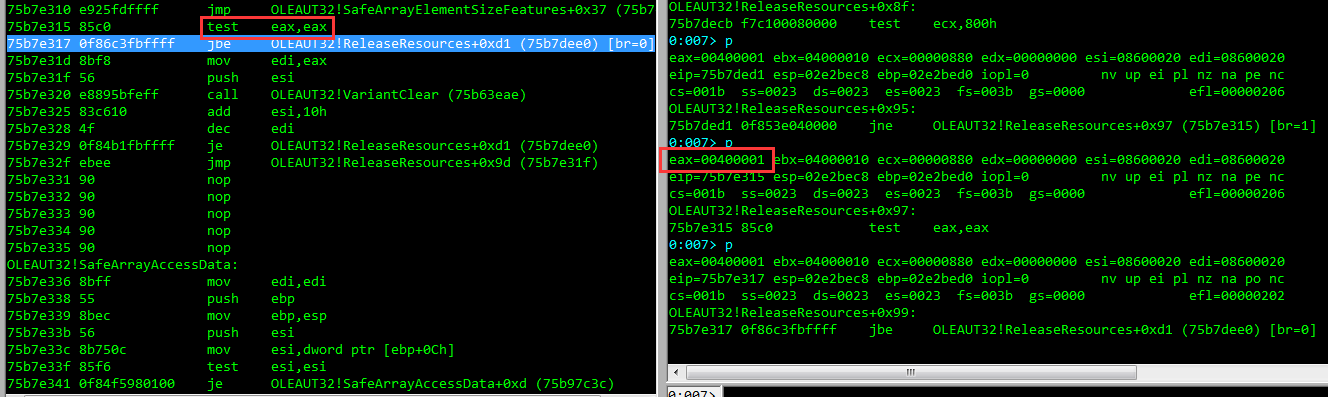






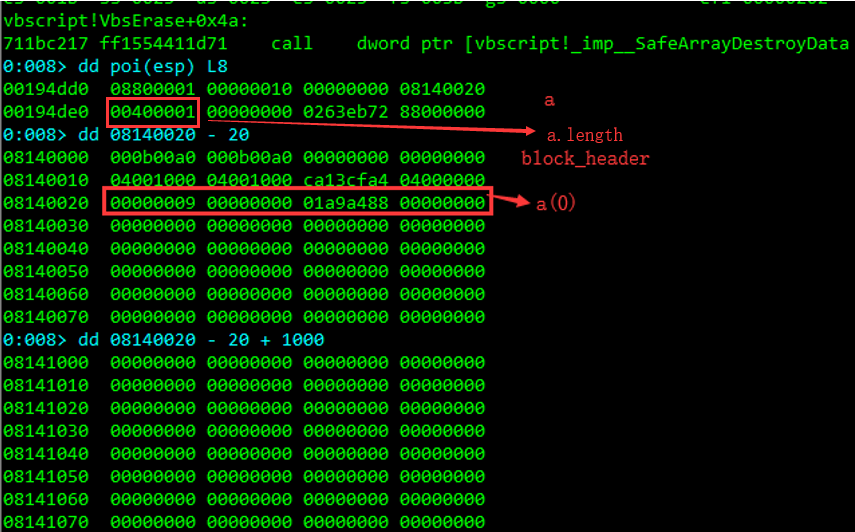
因为OLEAUT32!VariantClear是逐个释放数据元素，当释放a(0)时会触发MyClass的Class\_Terminate调用，从而修改了数组a的大小，然而OLEAUT32!ReleaseResources中依然保存了a长度的副本，



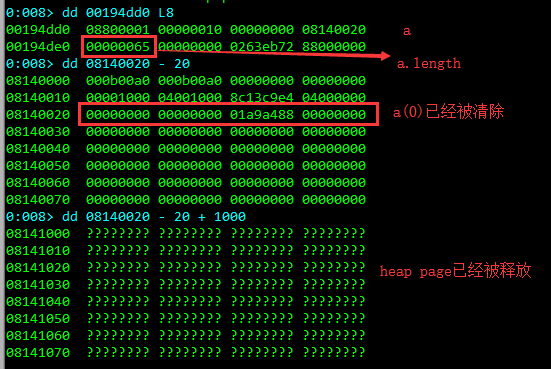


从而造成的数组越界读取，最后在OLEAUT32!VariantClear释放越界数组元素时，造成了内存破坏：

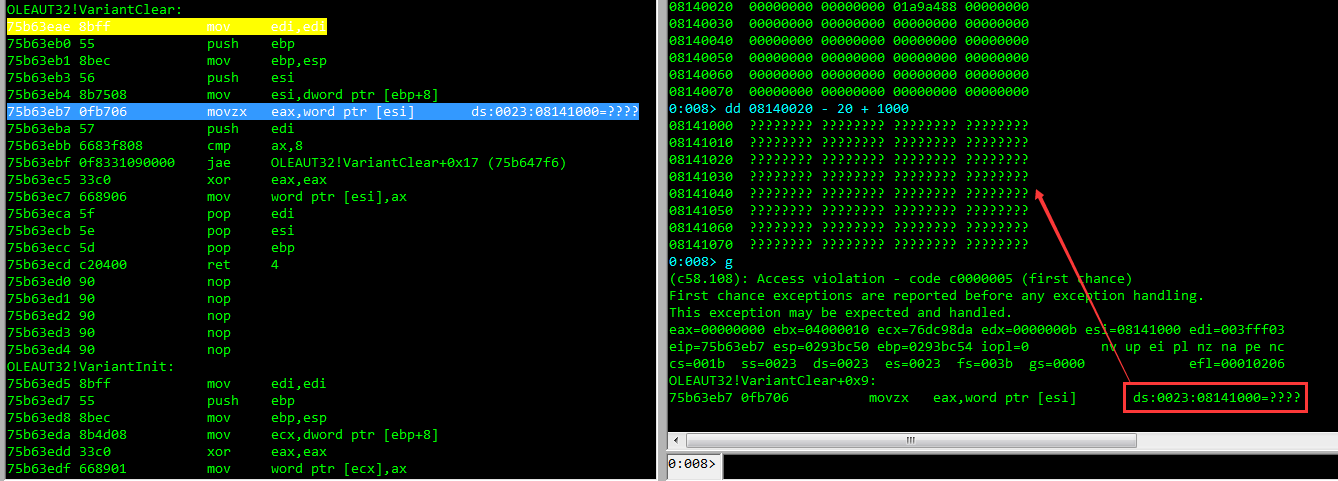
Set a(0) = new MyClass后的a：



ReDim Preserve a(100)后的a:



显然a+0x1000开始的内存已经被释放，最终造成数据的越界修改漏洞：



利用这个漏洞可以造成内存破坏，但是考虑到OLEAUT32!VariantClear只是将esi指向内存前两个Byte改为0，目前还没有想到可以利用的内存布局方法。可以考虑上一条sourcing rule，如果有REC的exploit，由于利用稳定，应该会被EK广泛使用。