使用VC2017 + IDA模拟复现CVE-2017-11882缓冲区溢出

前言

缓冲区溢出是一种非常普遍的漏洞,广泛存在于操作系统和各种应用软件中。通过缓冲区溢出攻击,可以完成执行非授权命令、执行恶意shellcode等高危攻击操作。

在过往的缓冲区漏洞利用案例中,CVE-2017-11882是我印象比较深也是觉得比较有意思的一个。攻击者只需给目标发送一个制作好的word文档并且这个文档被打开即可完成攻击,够简单暴力。

这个漏洞竟然隐藏10余年后才被发现,惊叹之余便有了一点小反思: 这样的"漏洞代码"作者倒底是如何写出来的。

网上已经有很多关于11882的漏洞分析、复现的文章,本文从另外一个角度来复现:通过VC写一个小段程序,模拟展示这个缓冲区漏洞是如何产生和利用的。

演示代码是两年前写的,想想还是该配上篇文章,混点Star好过年。本文适合于开发新手阅读,需要一点基本的汇编和C基础,大牛请飘过...

实验效果

CVE-2017-11882的利用过程简要概述如下,详细分析文章可查阅本文末尾相关网页

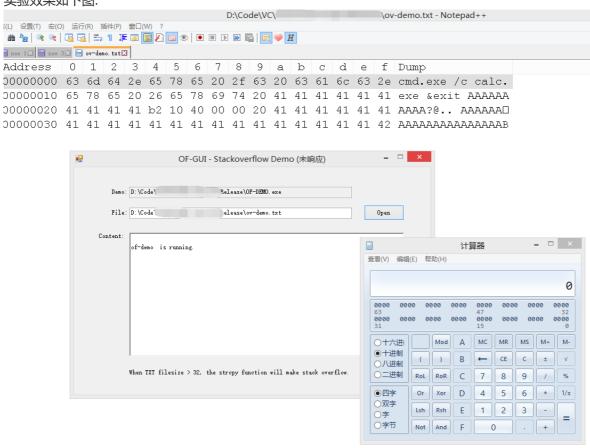
- 1.用word打开一个包含攻击代码的Doc文档
- 2.Word启动公式编辑器组件EQNEDT32.EXE
- 3.EQNEDT32.EXE在处理OLE对象的数据流时发生溢出,跳转 .text:00430C12 调用winExec执行命令或 shellcode

为模拟复现上述过程,编写构造2个小程序和1个txt文档,操作步骤如下

Word应用软件 -> OF-GUI.exe EQNEDT32.EXE ==> OF-Demo.exe Doc文档 ==> txt文档

- 1.用OF-GUI.exe打开txt文档
- 2.OF-GUI.exe启动OF-Demo.exe, 加载处理txt文档
- 3.OF-Demo.exe发生溢出后,执行txt中的命令

实验效果如下图:



演示代码下载: https://github.com/ekgg/Overflow-Demo-CVE-2017-11882

操作步骤

1、用IDA打开EQNEDT32.EXE, 找开调用WinExec的那个函数sub 430C00

```
.text:00430C00
Function name
                                                   f sub_430C00
f sub_430C50
                                                                                                                   ; CODE XREF: Eqr
; EqnFrameWinPrc
                                                   .text:00430C00
   sub_430CAA
                                                   .text:00430C00
f sub_430D10
f sub_430D44
f sub_430D57
f sub_430D88
f sub_430DB1
f sub_430DD0
                                                   .text:00430C00 Buffer
                                                                                       = bute ptr -100h
                                                   .text:00430C00 lpCmdLine
                                                                                       = dword ptr
                                                   .text:00430C00
                                                   .text:00430C00
                                                                                       push
                                                   .text:00430C01
                                                                                       mnv
                                                                                                ebp,
                                                                                                      esn
   sub_430DD0
                                                   .text:00430C03
                                                                                       sub
                                                                                                      100h
                                                                                                esp,
f
f
   sub_430E78
                                                   .text:00430C09
                                                                                       push
   sub_430EBB
                                                   .text:00430C0A
                                                                                       push
                                                                                                esi
f
f
f
   sub_430F1D
sub_430F70
                                                   .text:00430C0B
                                                                                       push
                                                                                                edi
                                                   .text:00430C0C
                                                                                       .
push
                                                                                                                    ; uCmdShow
   sub_430F99
                                                   .text:00430C0E
                                                                                                eax, [ebp+lpCmdLine]
   sub_4310D2
                                                                                                                   ; 1pCmdLine
                                                   .text:00430C11
                                                                                       push
                                                                                                eax
f
f
   sub 4310F8
                                                   .text:00
   sub_43110F
                                                   .text:00430C18
                                                                                                eax, 20h
                                                                                       cmp
f
f
   sub_431153
                                                   .text:00430C1B
                                                                                       jnb
                                                                                                loc_430C43
   sub_431279
                                                                                                eax, [ebp+Buffer]
                                                   .text:00430C21
                                                                                       1ea
f
f
   sub_4312BE
                                                                                                                     1pBuffer
                                                   .text:00430C27
                                                                                       push
                                                                                                eax
   sub_4312E0
                                                   .text:00430C28
                                                                                                6 0h
                                                                                       push
                                                                                                                     _int16
   sub_431340
                                                   .text:00430C2A
                                                                                       call
                                                                                                sub 427680
f
f
   sub_4313C9
sub_431414
                                                   .text:00430C2F
                                                                                       add
                                                                                                esp, 8
                                                                                                                     __int16
                                                   .text:00430C32
                                                                                       push
   sub_4314BD
                                                   .text:00430C34
                                                                                                eax, [ebp+Buffer]
   sub_4316DD
                                                   .text:00430C3A
                                                                                       push
                                                                                                eax
                                                                                                                    ; 1pMessageText
   sub 431794
                                                                                                sub 418544
                                                   .text:00430C3B
                                                                                       call
   sub_431803
                                                   .text:00430C40
                                                                                       add
                                                                                                esp, 8
```

2、按F5转为C代码,

```
Library function Data Regular function Unexplored Instruction External symbol
                          □ & ×
                                     📜 IDA View-A 🗵
                                                       📳 Pseudocode-A 🛛
                                                                         Mex View-A
Functions window
                                   UINT cdecl sub 430C00(LPCSTR lpCmdLine)
Function name
f sub_430C00
                                     UINT result; // eax@1
f sub_430C50
                                     CHAR Buffer; // [sp+Ch] [bp-100h]@2
f sub_430CAA
f sub_430D10
                                     result = WinExec(lpCmdLine, 1u);
  sub_430D44
                                     if ( result < 0x20 )
  sub 430D57
                                     {
  sub 430D88
                                        sub_427680(96, &Buffer);
  sub_430DB1
                                       result = sub_418544(&Buffer, 1);
  sub_430DD0
                                      3
  sub_430E78
                                     return result;
  sub 430EBB
f sub 430F1D
```

3. 在Visual Studio中创建一个控制台程序工程,复制上面这个sub_430C00函数,微调一下,去掉无关代码

```
UINT sub_430C00(const char* lpCmdLine)
{
    UINT result;
    char Buffer[256];

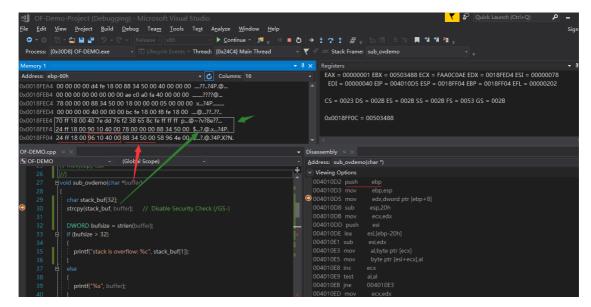
    result = WinExec(lpCmdLine, 1);
    if (result < 32) // 1 = SW_NORMAL
    {
        FatalAppExit(0, TEXT("WinExec Failure"));
    }
    return result;
}</pre>
```

4. 编写一个典型的缓冲区溢出函数, 使用strcpy这个不安全拷贝函数, 定义一个32字节的局部变量, 传入超过32字节的字符串, 就会发生溢出

```
void sub_ovdemo(char *buffer)
{
    char stack_buf[32];//传入超过32字节的字符串,就会发生溢出

    strcpy(stack_buf, buffer); // Disable Security Check (/GS-)
    DWORD bufsize = strlen(buffer);
    if (bufsize > 32)
    {
        printf("stack is overflow: %c", stack_buf[1]);
    }
    else
    {
        printf("%s", buffer);
    }
}
```

5. 编译工程后,在strcpy行设置断点,启动调试



如上图直观可见,进入sub_ovdemo函数代码空间后,EBP=0X18FF04,

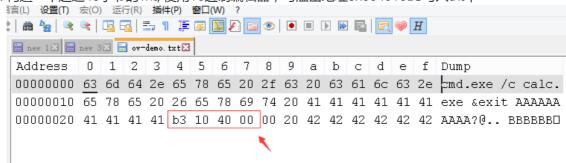
- 。 红色箭头所指,ebp+8是调用sub_ovdemo的参数buffer
- 。 ebp+4 = 0x00401096 (友情提示:低位在前高位在后)是函数ret时,跳转的下一条指令的地址
- 。 绿色箭头所指的32字节,就是局部变量 char stack_buf[32]的内存区域

到现在就很明了,只要构造一个32+4+4 = 40个字节的缓存,即可溢出覆盖掉ebp+4 ,跳到想要达到的地方

6. 在VS中或者用IDA查看上面的sub_430C00函数对WinExec的调用,可以得到想要跳转的地址: 004010B3

因此用004010B3溢出覆盖掉 上面的ebp+4 (00401096),即可达到用WinExec执行payload命令的目的。

7. 构造一个超过40字节的Txt, 使用16进制编辑器,写溢出地址0x004010B3写入txt中



- 8. 只要将上面txt文件的内容传入sub_ovdemo函数,即可溢出跳转执行命令。
- 9. 实验完成,收工。

后记

- 1、手写代码模拟漏洞复现,确实有点意思,可以更加理解漏洞产生的来龙去脉。因为这个漏洞利用起来实在是太方便了,都不用写shellcode,直接跳转调用WinExec执行命令。从阴谋论上来说,实在难以排除有开发人员故意留下这后门的可能性哈(开个小玩笑)。
- 2、为了完成实验,前后不断调整Visual Studio的编译优化选项,才使得生成类似的汇编代码。

```
Project setting:

c/c++

Optimization

Optimization: Maximum Optimization (Favor Size) (/O1)

Favor Size of Speed: Favor small code (/Os)

Whole Program Optimization: no

Code Generation

Basic Runtime Checks: Default

Security Check: Disable Security Check (/GS-)

Linker

Advanced

Randomized Base Address: No (/DYNAMICBASE:NO)

Data Execution Prevention(DEP): No (/NXCOMPAT:NO)
```

在实际工作中,推荐使用高版本的开发环境(VS2015+),打开安全检查选项和运行时检查,开启了动态随机基址(ALSR)和DEP,使生成的二进制程序不出现这类简单的溢出漏洞。

附:

CVE-2017-11882介绍漏洞分析

1.小白学习CVE-2017-11882漏洞过程 https://bbs.pediy.com/thread-247740.htm

2.Office 0day (CVE-2018-0802与2017-11882) 漏洞分析与利用 https://bbs.pediy.com/thread-2297
17.htm