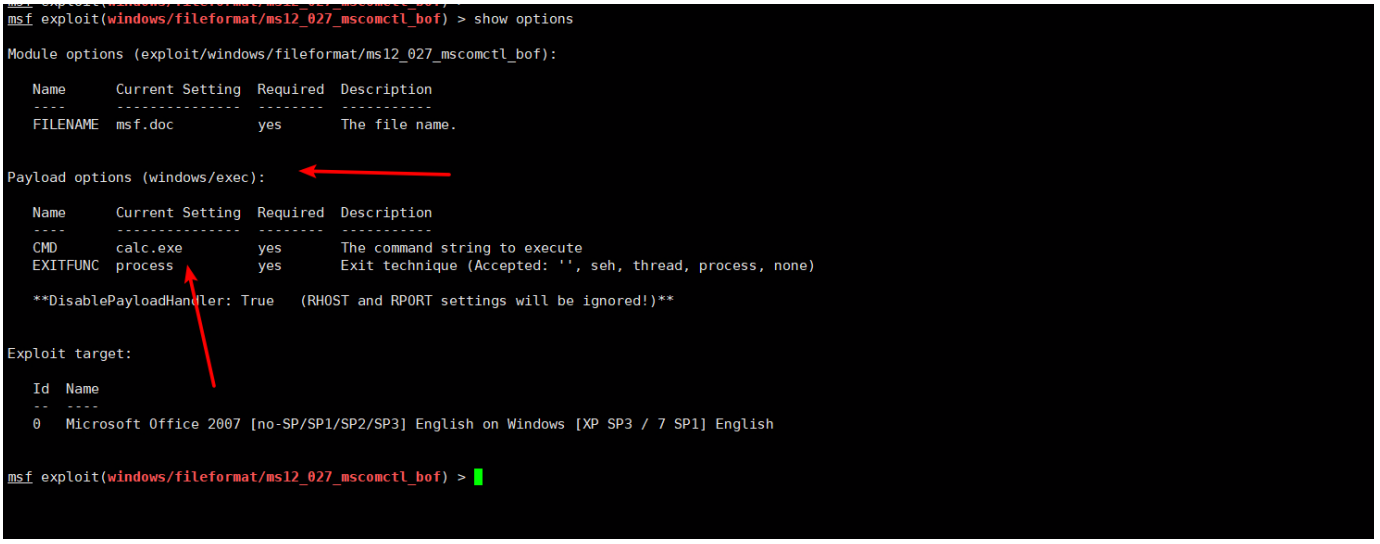
CVE-2012-0158 分析

生成样本

用 `msf` 生成一个 执行 计算器的 样本



拖到 `office 2007` 的 `word` 里面打开即可弹出计算器。

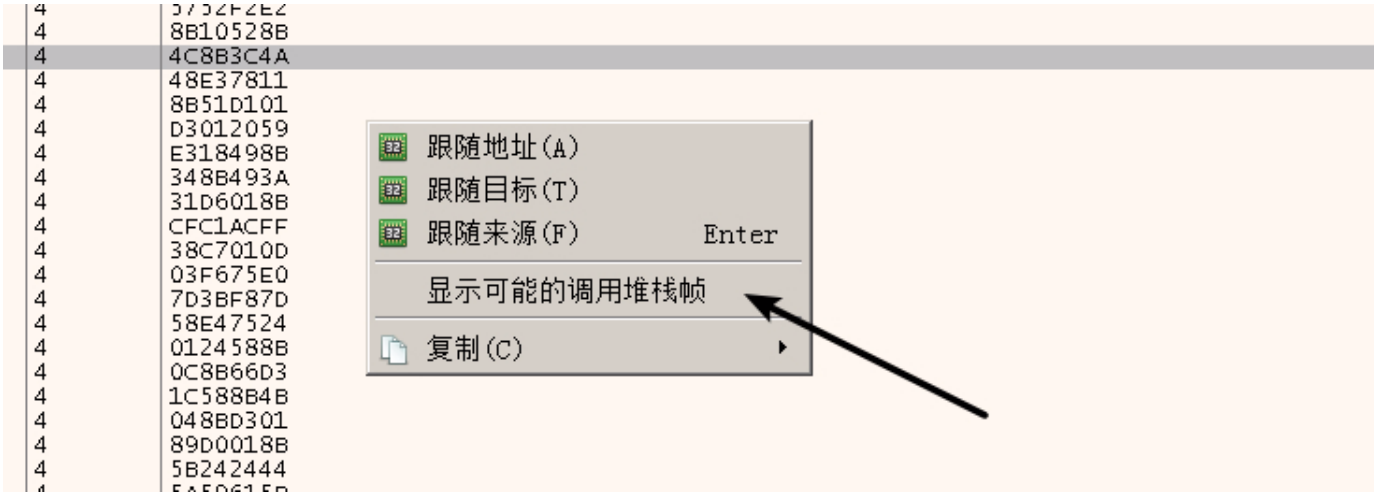
漏洞分析

弹计算器一般是执行系统命令实现， 这里使用的是 `kernel32.dll` 里面的 `WinExec` 实现的。所以可以在这个函数下个断点， 然后运行样本看看。

PS： `x64dbg` 可以选择下载指定模块的符号。

在 `WinExec` 断下来后，查看栈回溯，发现栈的数据已经被破坏。

此时可以在 调用堆栈窗口 ， 右键选择 **显示可能的调用堆栈帧**



把结果到出来，在文本编辑器里面搜索，发现有两处 `mscomctl` 的地址。

```
002E4A2C 275C8800          24    返回到 mscomctl.275C8800
002E4A50 275C8A0A 275C876D 8404 返回到 mscomctl.275C8A0A
```

通过 `rtfobj` 工具可以发现 `msf.doc` 中用到的 `ole` 对象，对应的控件的 `CLSID`为 `BDD1F04B-858B-11D1-B16A-00C0F0283628`

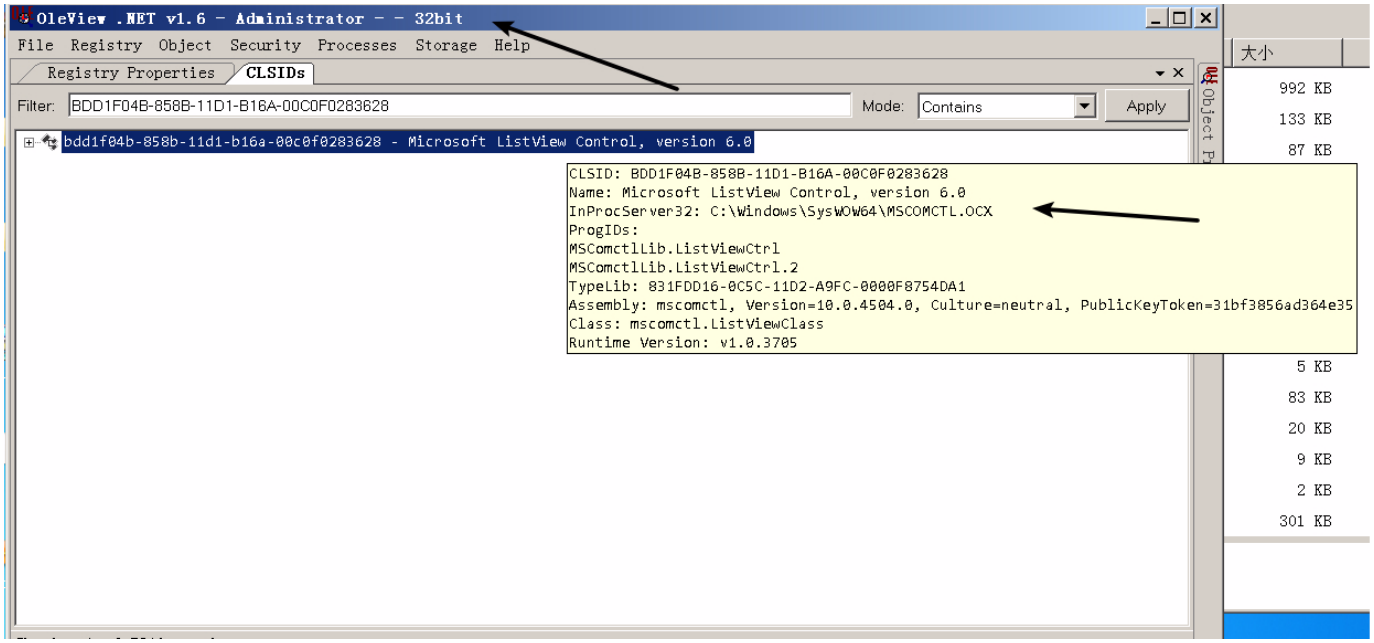
```
root@kali:~/vm_share/tmp# rtfobj msf.doc
rtfobj 0.53.1 on Python 2.7.15 - http://decalage.info/python/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

===============================================================================
File: 'msf.doc' - size: 10296 bytes
---+---------+---------------------------------------------------------------
id |index    |OLE Object
---+---------+---------------------------------------------------------------
```

```
0   |000000A0h |format_id: 2 (Embedded)
    |          |class name: 'MSComctlLib.ListViewCtrl.2'
    |          |data size: 3584
    |          |CLSID: BDD1F04B-858B-11D1-B16A-00C0F0283628
    |          |MSCOMCTL.ListViewCtrl (may trigger CVE-2012-0158)
---+----------+------------------------------------------------------------
```

然后使用

https://github.com/tyranid/oleviewdotnet

来查找 CLSID 对应的控件。



PS: 由于 word 是 32 位的 所以需要使用 32 位的 oleview 来查找。

可以看到漏洞用到的控件就是 mscomctl.ocx，拿 ida 打开它，跳到刚刚发现的两个地址。

002E4A2C 275C8800 24 返回到 mscomctl.275C8800

002E4A50 275C8A0A 275C876D 8404 返回到 mscomctl.275C8A0A

275C8800 传入的 参数为

arg1: 读取数据保存位置

arg2: 一个 CExposedStream 对象指针

arg3: 需要读取的大小

```
int __cdecl read_data_to_dst(void *dst, LPVOID lpMem, SIZE_T size)
{
  LPVOID v3; // ebx
  int result; // eax
  LPVOID v5; // eax
  int v6; // esi
  int v7; // [esp+Ch] [ebp-4h]
  void *src; // [esp+1Ch] [ebp+Ch]

  v3 = lpMem;
  result = (*(*lpMem + 12))(lpMem, &v7, 4, 0);
  if ( result < 0 )
    return result;
  if ( v7 == size )
  {
    v5 = HeapAlloc(hHeap, 0, size);
    src = v5;
    if ( v5 )
    {
      v6 = (*(*v3 + 12))(v3, v5, size, 0);      // CExposedStream::Read
      if ( v6 >= 0 )
      {
        qmemcpy(dst, src, size);
        v6 = (*(*v3 + 12))(v3, &unk_27632368, ((size + 3) & 0xFFFFFFFC) - size, 0);
      }
      HeapFree(hHeap, 0, src);
      result = v6;
    }
```

函数的作用大概为 利用 `CExposedStream` 对象 从 `objstream` 里面读取数据到 `dst`， 大小为 `size`.

下面在看看它的上层调用 `275C8A0A` 。

这里就是漏洞的位置了首先从 `stream` 数据读出 `0xc` 字节到 `dst` . 这是我定义的一个结构体，结构体定义为。

```
struct stack_struct
{
  int flag;
  int field_4;
  int next_size;
};
```

函数首先判断 `dst->flag` 为 `0x6A626F43` 和 `dst->next_size >= 8` . 通过这个判断后，会再次利

用 CExposedStream 读取 stream 数据到 v6 ，大小为 dst->next_size 。

在 ida 中分析可以发现 v6 是栈上的一个变量， 大小为 8 个字节， 而 dst->next_size 则为我们的输入数据是在 doc 文件里面设置的， 栈溢出！。

可以看看样本里面的相关字段。

首先使用 rtfobj 导出 ole 对象。

```
root@kali:~/vm_share/tmp# rtfobj msf.doc -s all
rtfobj 0.53.1 on Python 2.7.15 - http://decalage.info/python/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

================================================================================
File: 'msf.doc' - size: 10296 bytes
---+----------+--------------------------------------------------------------
id |index     |OLE Object
---+----------+--------------------------------------------------------------
0  |000000A0h |format_id: 2 (Embedded)
   |          |class name: 'MSComctlLib.ListViewCtrl.2'
   |          |data size: 3584
   |          |CLSID: BDD1F04B-858B-11D1-B16A-00C0F0283628
   |          |MSCOMCTL.ListViewCtrl (may trigger CVE-2012-0158)
---+----------+--------------------------------------------------------------
Saving file embedded in OLE object #0:
  format_id  = 2
  class name = 'MSComctlLib.ListViewCtrl.2'
  data size  = 3584
  saving to file msf.doc_object_000000A0.bin
```

然后拿 010editor 打开。

可以看到这里 size 设置为了 0x00008282 。

下面在 275C8A05 处下个断点，然后加载文档。

可以看到 `next_size` 的值为 `0x8282`.

运行完之后，返回地址被修改，单步到 `ret` 指令。

`27583C30` 是一条 `jmp esp` 指令。

然后继续运行就会利用 `jmp esp` 跳转到栈里面的 `shellcode`.

一些 rtf 文件格式的东西

rtf 文件可以使用

```
{\object\objocx
```

来调用控件，控件的数据保存在

```
{\*\objdata
```

保存的方式为 16 进制字符串。

其中 D0CF11E0A1B11AE1 为 ole 数据的签名，表明从这里开始就是 ole 的数据。

更多详细可以看

http://www.infocomm-journal.com/cjnis/CN/abstract/abstract156911.shtml#1

参考

http://www.infocomm-journal.com/cjnis/CN/abstract/abstract156911.shtml#1

https://www.anquanke.com/post/id/91643#h2-6

来源： https://www.cnblogs.com/hac425/p/10003567.html