ctf.360.cn第二届，逆向部分writeup——第五题

转载

weixin_33894640 于 2014-12-04 16:56:43 发布 58 收藏
原文链接：http://blog.51cto.com/cugou/1586369
版权
题目：见附件

这是最后一题，难度加倍。

> **提示：**请先搭建该EXE的执行环境
> 1. 在本地安装appserv[如果不知道appserv是啥，请自行so.com一下]，配置HTTP端口为80
> 2. 将您修改后的Exploit.html以及压缩包中的shell.dat复制到www目录下
> 3. 运行MFC_ASM.exe，如果您成功修复了该Exploit.html，会弹出key，否则，崩溃

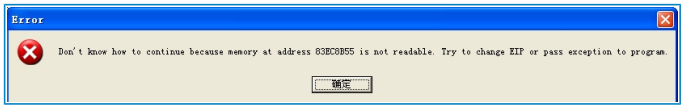
看题目意思，应该是模拟一个下载者，从网上下载一个shellcode然后执行。要求修复这个shellcode然后获得key。

OD载入MFC_ASM.exe，F9出现程序界面。

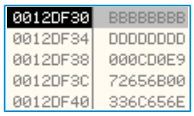
由于平时写C程序访问网络都直接用socket，找了一下竟然没有发现socket调用，看了导入表，发现原来使用wininet.dll提供的http封装。

（这里不知道有没有更好的方法能让OD跟踪到button按钮的用户代码调用，还请大牛不吝赐教）

直接按button，出现错误：

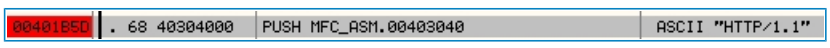


此时的栈情况：



BBBBBBBB和DDDDDDDD数据和Exploit.html文件里面开头的是一致的，可见程序已经将Exploit.html下载并拷贝到了堆栈地址0012DF30。

为了跟踪到button按钮的处理函数，还是从字符串入手，随便翻翻发现：



此处有http协议的引用，下断点后，按button能停下来。一路F8，可以跟到00401519处，此处的retn指令返回到了shellcode中的错误地址。

需要注意的是，shellcode的起始地址是0012DF30，但00401518处的leave指令，会将esp指向0012E130。这个地址里面的值，就是前面错误对话框中出现的值。

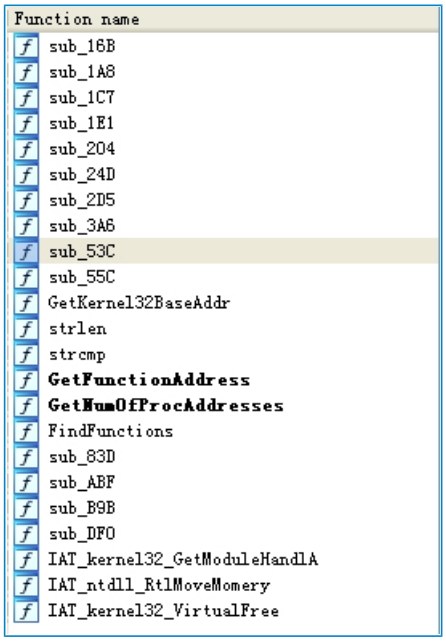
之所以要指出这个，是因为当前的esp指向了shellcode内部，如果执行shellcode代码，会覆盖掉shellcode的一些区域，导致出错。这就就是需要修正的第一个有问题的地方。

下面就是要分析Exploit.html这个文件了，载入IDA，进行静态分析。由于对windows下的shellcode不是很熟悉，这里花了非常多的时间。关键点就是如大牛提示的，需要一个一个函数分析，一段一段shellcode中的字节寻找。由于这个shellcode的复杂度，修复实际上就是寻找到一个合适的入口点，然后在OD中将EIP指向它，直接执行就可以弹出key。

函数还是比较容易寻找的，一般以push ebp开头，后面跟着sub esp（或add esp）。IDA能分析出一些函数，许多还是要自己去寻找。

这里IDA的Xrefs功能（快捷键X）实在是太有用了。

最终分析下来的函数列表：



这里几个关键函数我都做了注释，里面有非常多的代码值得学习，比如fs:[eax]这种获取kernel32基址的方法，以及建立shellcode的导入函数表，运行时动态获取当前指令地址，从而实现在shellcode中寻址与shellcode加载地址无关。

这里几乎找全了shellcode中的所有函数，通过Xrefs分析，还是没有哪里调用了FindFunctions等关键函数，也就是没有找到入口。只能在IDA代码text视图下，一点点的看，终于在00000CDD处找到一段没有定义的数据字节。按C，定义为code后：

```
seg000:00000CDD                 call    FindFunctions
seg000:00000CE2                 nop
seg000:00000CE3                 nop
seg000:00000CE4                 nop
seg000:00000CE5                 sub     esp, 64h
seg000:00000CE8                 mov     esi, esp
seg000:00000CEA                 lea     eax, [esp]
seg000:00000CED                 mov     byte ptr [eax], 31h ; '1'
seg000:00000CF0                 mov     byte ptr [eax+1], 32h ; '2'
seg000:00000CF4                 mov     byte ptr [eax+2], 37h ; '7'
seg000:00000CF8                 mov     byte ptr [eax+3], 2Eh ; '.'
seg000:00000CFC                 mov     byte ptr [eax+4], 30h ; '0'
seg000:00000D00                 mov     byte ptr [eax+5], 2Eh ; '.'
seg000:00000D04                 mov     byte ptr [eax+6], 30h ; '0'
seg000:00000D08                 mov     byte ptr [eax+7], 2Eh ; '.'
seg000:00000D0C                 mov     byte ptr [eax+8], 31h ; '1'
seg000:00000D10                 mov     byte ptr [eax+9], 0
seg000:00000D14                 mov     byte ptr [eax+0Ah], 0
```

Great！入口点应该就在这里了。因为其他地方出了字符串表和00字节，没有多余的疑似未定义代码了。

OD载入，断点在00401518处，修改EBP为0012DF30。F8到00401519，修改当前栈上0012DF34处的值为0012DF30+CDD。然后F9，直接弹出对话框，key就在这里啦！

转载于:https://blog.51cto.com/cugou/1586369