



イマーシブリーダーをお試しいたき、ありがとうございます。フィードバックをお寄せください。  

×

【逆向工程】在PE结构空白区段插入代码

最近在学PE结构，遇到了个比较有意思的实验，学过PE结构的都知道因为文件对齐FileAlignment和区块对齐SectionAlignment的缘故，在磁盘中的PE文件的块与块表 块与块之间有许多空白间隙，这些间隙以0为填充，在文件执行中这些空白是没有意义的，所以我们就可以利用这段空白插入我们想执行的一些代码。

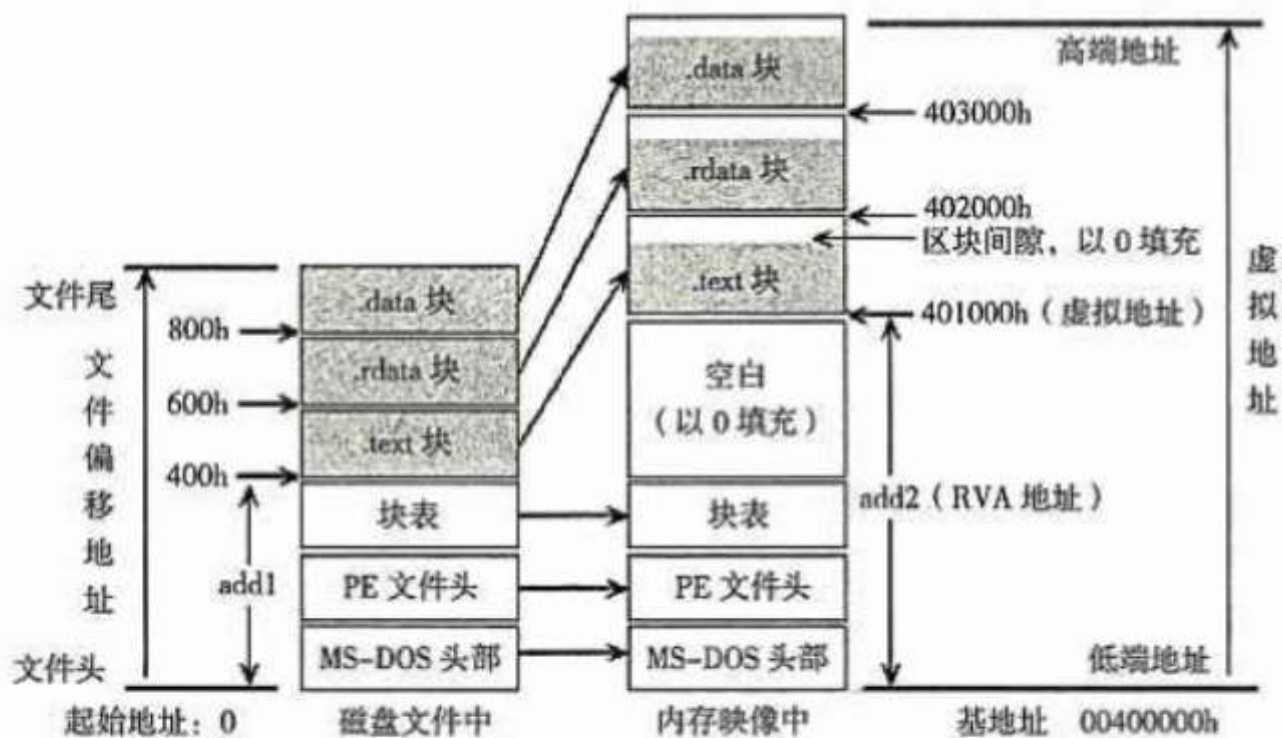


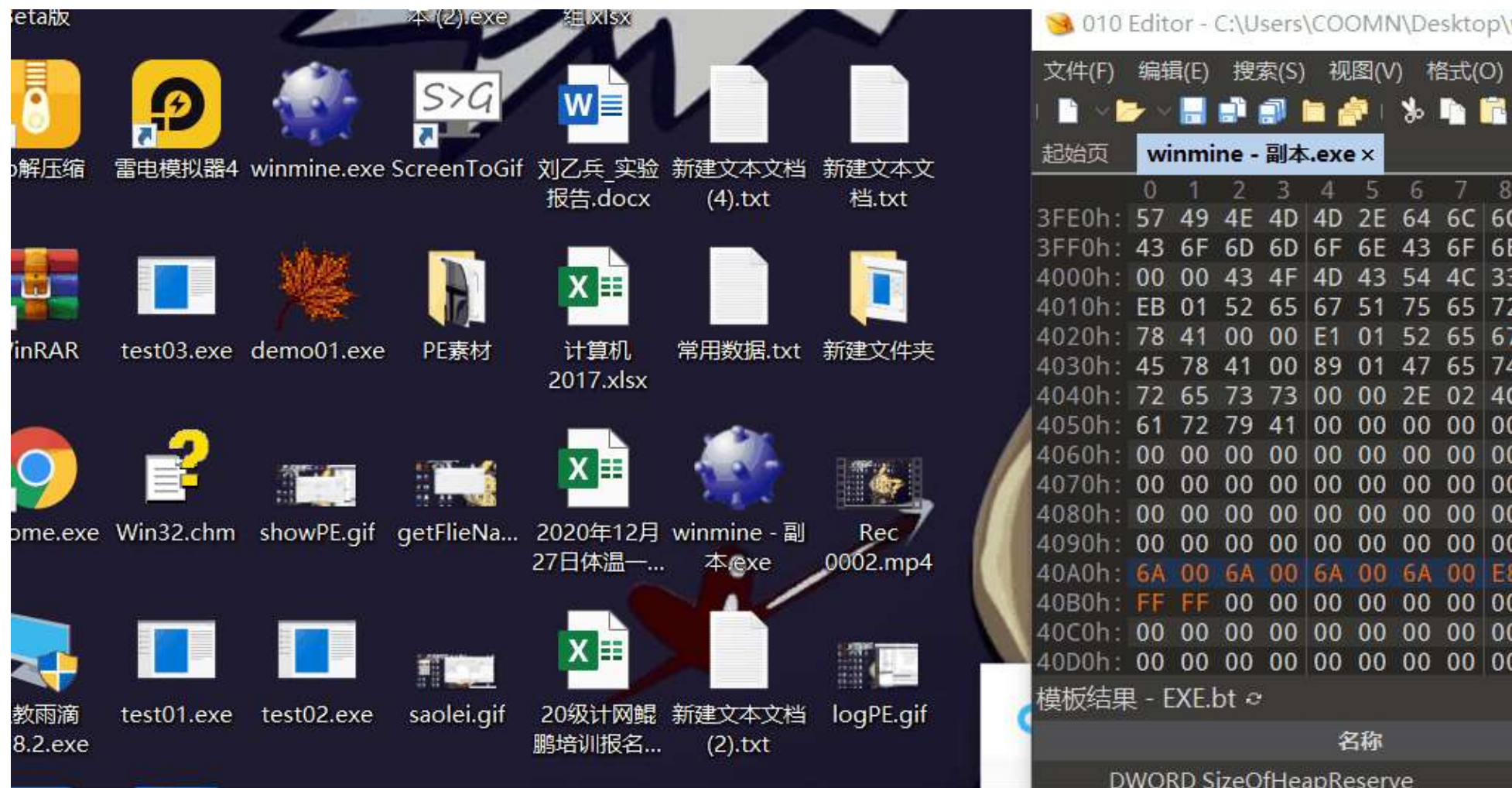
图 11.11 应用程序加载映射示意图

演示程序 经典扫雷

演示dll user32.dll.MessageBoxW

演示所用工具 X64dbg, PEGAME, 010Editor

演示目标 实现修改程序使其弹出一个MessageBox消息对话框



查看硬编码

这里用user32.dll中的**MessageBoxW**函数为例，MessageBoxA也可以，无非一个是宽字节版一个是单字节版，可以在VC中调用此接口然后调试查看反汇编代码。

获取硬编码如下

```
PUSH 6A 00  
CALL E8 00 00 00 00  
JMP E9 00 00 00 00
```

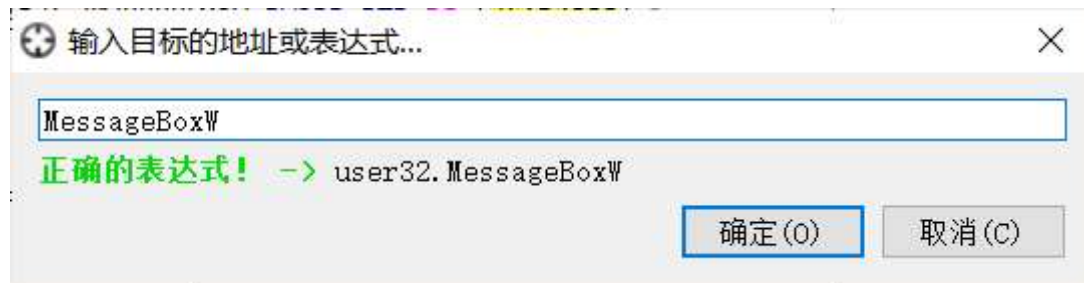
PUSH 指令通常用于在函数调时传递参数，会在函数调用时将参数压如栈中

CALL 通常用来调用函数 后面的00 00 00 00为跳转步长

JMP 无条件跳转指令 使用该指令无条件跳转到我们想要的指令处

查看所引用的DLL在程序中的VA地址

注意该程序必须导入了user32.dll这个动态链接库，否则将不会又MessageBox这个函数接口，也就无法找到该函数地址
将演示程序加载进动态调试工具X64dbg，Ctrl+G定位到该程序中MessageBox的函数，

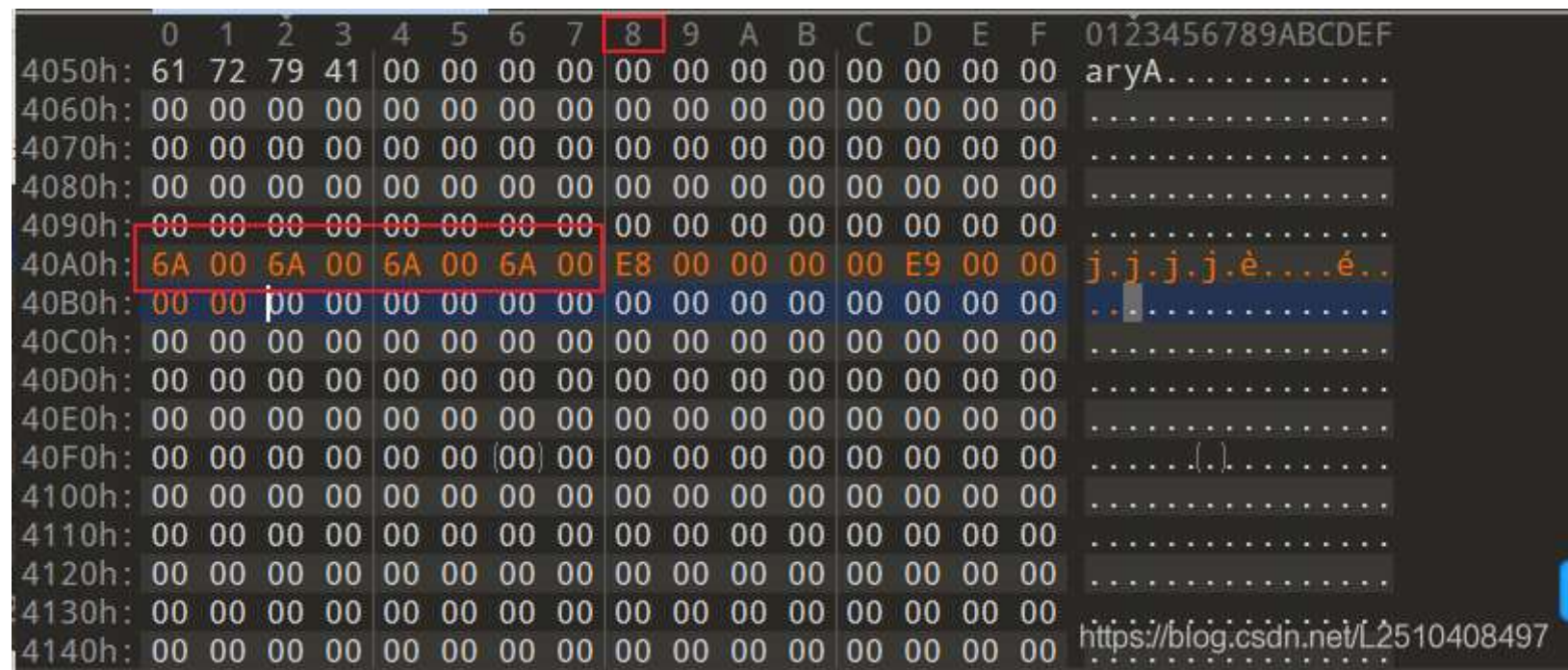


获取该VA地址为**76AB1E50**

76AB1E50	8BFF	MOV EDI,EDI	MessageBoxW
76AB1E52	55	PUSH FRP	

将演示程序加载入十六进制编辑工具，选择一个空白区段，我这里选的是第一个区块和第二个区块之间的空白区段。我们若想插入MessageBox函数必须用PUSH指令先传递所需参数，CALL指令调用，JMP指令再回到原本程序正常执行处，即AddressOfEntryPoint处。

查阅WIN32API 可知MessageBox的四个参数都可为0



CALL 指令调用，因此我们需要计算下CALL指令的步长，有两种方法

步长=要跳转的目的地址-当前指令地址-指令长度

步长=要跳转的目的地址-当前指令的下一条指令地址

我们采用第一个公式

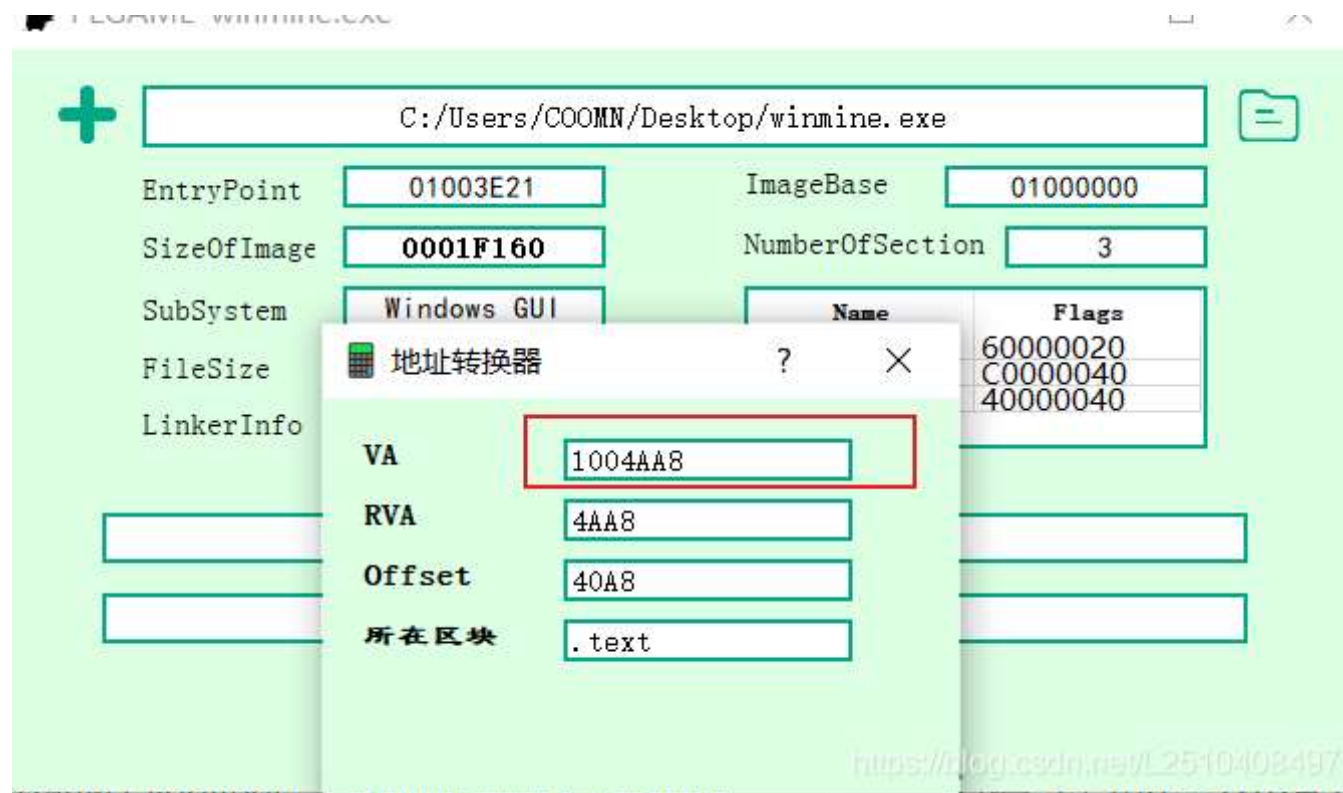
在上面我们已经得到了MessageBoxW的地址即要跳转的目的地址为**76AB1E50**

当前指令的地址即CALL指令E8的地址为**40A8**

但要注意的是76AB1E50为VA即虚拟地址，40A8为文件偏移地址Fileoffset，因此需要将40A8转化为虚拟地址。
这里可以手算，因为**插入硬编码的地址在区块中**，所以计算公式为

$$VA = Fileoffset - ImageBase - (\text{所在区块.VirtualAddress} - \text{所在区块.PointerToRawData})$$

也可以借助带有地址转换工具的PE结构查看器，如LordPE，我这里用我自己写的PE结构查看工具，感兴趣的话可以移步至[【逆向工程】QT编写PE结构分析工具](#)



查看到其VA地址为1004AA8

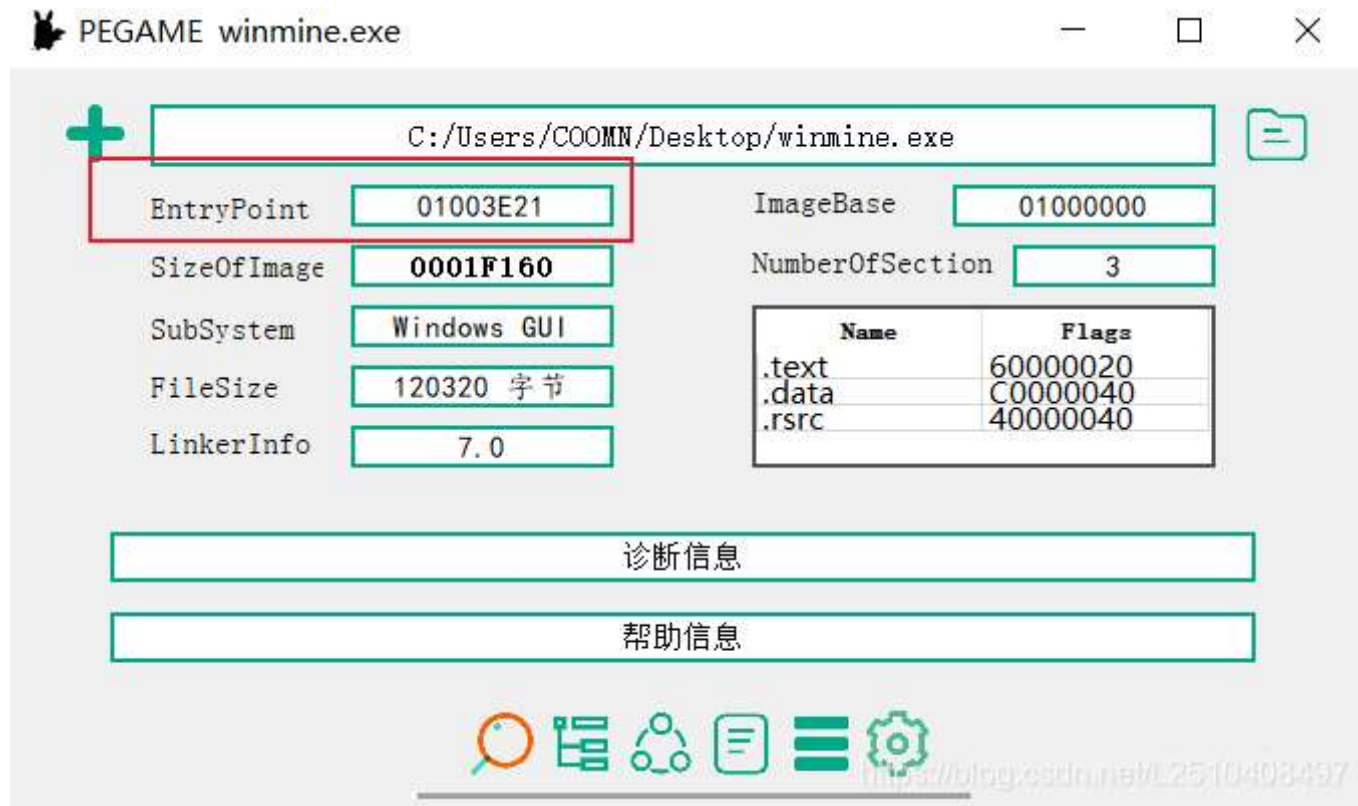
E8 00 00 00 00指令长度为五字节

则步长=76AB1E50-1004AA8-5=75AA D3A3

因为PE文件是小端存储所以 应该插入A3 D3 AA 75

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
3FE0h:	57	49	4E	4D	4D	2E	64	6C	6C	00	4C	00	49	6E	69	74	WINMM.dll.L.Init
3FF0h:	43	6F	6D	6D	6F	6E	43	6F	6E	74	72	6F	6C	73	45	78	CommonControlsEx
4000h:	00	00	43	4F	4D	43	54	4C	33	32	2E	64	6C	6C	00	00	..COMCTL32.dll..
4010h:	EB	01	52	65	67	51	75	65	72	79	56	61	6C	75	65	45	ë.RegQueryValueE
4020h:	78	41	00	00	E1	01	52	65	67	4F	70	65	6E	4B	65	79	xA..á.RegOpenKey
4030h:	45	78	41	00	89	01	47	65	74	50	72	6F	63	41	64	64	ExA.%.GetProcAdd
4040h:	72	65	73	73	00	00	2E	02	4C	6F	61	64	4C	69	62	72	ress....LoadLibr
4050h:	61	72	79	41	00	00	00	00	00	00	00	00	00	00	00	00	aryA.....
4060h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4070h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00{.}.....
4080h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4090h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40A0h:	6A	00	6A	00	6A	00	6A	00	E8	A3	D3	AA	75	E9	6F	F3	j.j.j.j.èfÓuéoó
40B0h:	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	00	00	ÿÿ.....
40C0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40D0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

我们调用完MessageBoxW后还需要让程序正常执行，因此还要JMP指令跳转到程序入口处，通过PE查询工具可以看到程序入口，这里还是使用我自己编写的PE结构查询工具查看。



则JMP指令步长=1003E21-1004AAD-5=FFFF F36F

同样的小端存储格式 E9 6D F3 FF FF

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
3FE0h:	57	49	4E	4D	4D	2E	64	6C	6C	00	4C	00	49	6E	69	74	WINMM.dll.L.Init
3FF0h:	43	6F	6D	6D	6F	6E	43	6F	6E	74	72	6F	6C	73	45	78	CommonControlsEx
4000h:	00	00	43	4F	4D	43	54	4C	33	32	2E	64	6C	6C	00	00	..COMCTL32.dll..
4010h:	EB	01	52	65	67	51	75	65	72	79	56	61	6C	75	65	45	ë.RegQueryValueE
4020h:	78	41	00	00	E1	01	52	65	67	4F	70	65	6E	4B	65	79	xA..á.RegOpenKey
4030h:	45	78	41	00	89	01	47	65	74	50	72	6F	63	41	64	64	ExA.%.GetProcAdd
4040h:	72	65	73	73	00	00	2E	02	4C	6F	61	64	4C	69	62	72	ress....LoadLibr
4050h:	61	72	79	41	00	00	00	00	00	00	00	00	00	00	00	00	aryA.....
4060h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4070h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4080h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4090h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40A0h:	6A	00	6A	00	6A	00	6A	00	E8	A3	D3	AA	75	E9	6F	F3	j.j.j.j.èfÓ*uéóó
40B0h:	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	00	00	ÿÿ.....
40C0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	https://blog.csdn.net/L2510408497
40D0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

修改程序入口AddressOfEntryPoint

操作完如上步骤后，程序并不会执行我们的代码，因此还需要修改程序的入口地址为我们插入代码的地址，这里直接用十六进制编辑工具修改，**这里填入的数字应该是RVA地址**

白菜信安网