

# 实验吧 该题不简单 OD详解 (flag 和分析内部算法)

原创

[pipixia233333](#) 于 2018-08-17 20:53:54 发布 1313 收藏

分类专栏: [逆向之旅](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/qq\\_41071646/article/details/81782365](https://blog.csdn.net/qq_41071646/article/details/81782365)

版权



[逆向之旅](#) 专栏收录该内容

128 篇文章 2 订阅

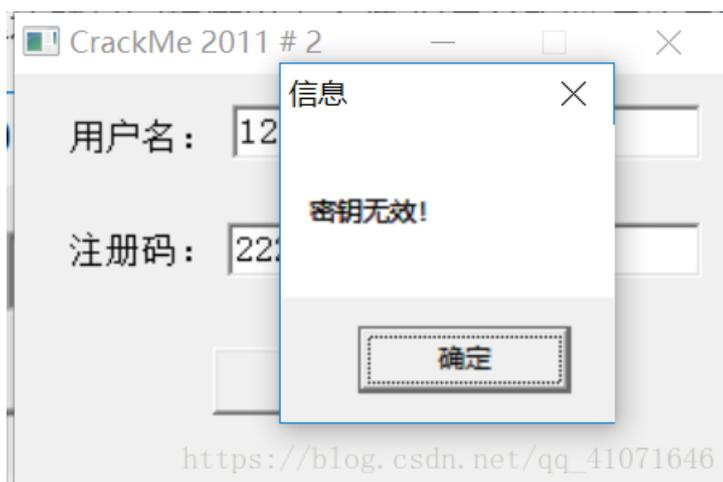
订阅专栏

这个题是偶然在实验吧想起来做的题 本来打算的是分析一晚上的 但是发现经过这一段时间的学习 自己进步很大 然后在半小时的时候就做出来了这一道题 然后 分享给大家

[实验吧原题网址](#) 这道题其实想做出来不难 但是分析算法还是需要一些基础的



点开就是这个样子 随便输入一下 就是这个样子



知道了程序大概是什么样子 然后开始分析算法 (题目要求的flag 是 用户名为 hello 的注册码)

然后打开OD 这里有几种方法断到我们想分析的地方 有一点就是 API `GetDlgItemTextA` 这个API 就是获取控件中的用户输入 然后我们先输入后查看 字符串

地址	反汇编	文本字符串
0040105A	push CrackMe1.00406048	ASCII "信息"
004010F0	mov dword ptr ss:[esp+0x58],CrackMe1.00406050	ASCII "myWindowClass"
00401126	push CrackMe1.00406050	ASCII "CrackMe 2011 # 2"
0040112B	push CrackMe1.004050E8	ASCII "myWindowClass"
0040122D	call esi	(初始 CPU 选择)
004017CD	mov esi,CrackMe1.0040855C	ASCII "C:\Users\Lenovo\Desktop\CrackMe1.exe"
00401D5C	push CrackMe1.0040511C	ASCII "_MSVCRT_HEAP_SELECT"
00401D9B	push CrackMe1.00405104	ASCII "_GLOBAL_HEAP_SELECTED"
0040213A	push CrackMe1.0040540C	ASCII "<program name unknown>"
0040217C	push CrackMe1.00405408	ASCII "..."
00402190	push CrackMe1.004053EC	ASCII "Runtime Error!\n\nProgram: "
004021AE	push CrackMe1.004053E8	ASCII "\n\n"
004021D6	push CrackMe1.004053C0	ASCII "Microsoft Visual C++ Runtime Library"
00402375	mov edi,CrackMe1.004086E0	ASCII "粒豕"
004023C3	mov edi,CrackMe1.004086E0	ASCII "粒豕"
00402478	mov edi,CrackMe1.004086E0	ASCII "粒豕"
00403E71	push CrackMe1.00405454	ASCII "user32.dll"
00403E88	push CrackMe1.00405448	ASCII "MessageBoxA"
00403E99	push CrackMe1.00405438	ASCII "GetActiveWindow"
00403EA1	push CrackMe1.00405424	ASCII "GetLastActivePopup"

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

我们点开信息那一行

00401048	~ 75 41	jnz short CrackMe1.0040108B	
0040104A	- 66:817C24 0C	cmp word ptr ss:[esp+0xC],0x3EA	
00401051	~ 75 38	jnz short CrackMe1.0040108B	
00401053	- E8 78010000	call CrackMe1.004011D0	
00401058	- 6A 00	push 0x0	
0040105A	- 68 48604000	push CrackMe1.00406048	Style = MB_OK MB_APPLMODAL Title = "信息"
0040105F	- 85C0	test eax,eax	
00401061	~ 75 16	jnz short CrackMe1.00401079	
00401063	- A1 AC864000	mov eax,dword ptr ds:[0x4086AC]	
00401068	- 68 3C604000	push CrackMe1.0040603C	Text = "密钥正确! "
0040106D	- 50	push eax	hOwner = NULL
0040106E	- FF15 D0504000	call dword ptr ds:[&USER32.MessageBoxA]	MessageBoxA
00401074	- 33C0	xor eax,eax	
00401076	- C2 1000	ret 0x10	
00401079	> 8B0D AC864000	mov ecx,dword ptr ds:[0x4086AC]	
0040107F	- 68 30604000	push CrackMe1.00406030	Text = "密钥无效! "
00401084	- 51	push ecx	hOwner = NULL
00401085	- FF15 D0504000	call dword ptr ds:[&USER32.MessageBoxA]	MessageBoxA
0040108B	> 33C0	xor eax,eax	
0040108D	- C2 1000	ret 0x10	
00401090	~ 83EC 4C	sub esp,0x4C	
00401093	- 53	push ebx	

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

我们可以初步判定这是我们的响应信息后的代码 我找到那个函数 其实有点运气成分 因为我是往下看的时候正好看见了 这个函数

004011F1	. 8D7C24 49	lea edi,dword ptr ss:[esp+0x49]	
004011F5	. C64424 48 00	mov byte ptr ss:[esp+0x48],0x0	
004011FA	. F3:AB	rep stos dword ptr es:[edi]	
004011FC	. 66:AB	stos word ptr es:[edi]	
004011FE	. AA	stos byte ptr es:[edi]	
004011FF	. B9 07000000	mov ecx,0x7	
00401204	. 33C0	xor eax,eax	
00401206	. 8D7C24 29	lea edi,dword ptr ss:[esp+0x29]	
0040120A	. C64424 28 00	mov byte ptr ss:[esp+0x28],0x0	
0040120F	. F3:AB	rep stos dword ptr es:[edi]	
00401211	. 8B0D B086400	mov ecx,dword ptr ds:[0x4086B0]	
00401217	. 8B35 A450400	mov esi,dword ptr ds:[&USER32.GetDlgItemTextA]	user32.GetDlgItemTextA
0040121D	. 66:AB	stos word ptr es:[edi]	
0040121F	. AA	stos byte ptr es:[edi]	
00401220	. 8D4424 08	lea eax,dword ptr ss:[esp+0x8]	
00401224	. 6A 10	push 0x10	Count = 10 (16.)
00401226	. 50	push eax	Buffer = 000016C8
00401227	. 68 E8030000	push 0x3E8	ControlID = 3E8 (1000.)
0040122C	. 51	push ecx	hWnd = 00000007
0040122D	. FFD6	call esi	GetDlgItemTextA
0040122F	. 83F8 05	cmp eax,0x5	
00401232	. 73 0B	jnb short CrackMe1.0040123F	

[https://blog.csdn.net/qq\\_41071646](https://blog.csdn.net/qq_41071646)

其实真正的做法应该是 用API断点 我用的吾爱破解的OD 有断点插件 插件就是第二个项 然后点下面的就可以了



然后我们断下来后 可以 为所欲为了（能断下来剩下的分析就很好说了）

然后 我们先不管算法 随便输入试试 然后往下走看看

00401290	- 49	dec ecx	
00401291	- 3BF1	cmp esi,ecx	
00401293	^ 72 D0	jb short CrackMe1.00401265	
00401295	> 8B15 64604000	mov edx,dword ptr ds:[0x406064]	
0040129B	- 66:A1 68604000	mov ax,word ptr ds:[0x406068]	
004012A1	- 8A0D 6A604000	mov cl,byte ptr ds:[0x40606A]	
004012A7	- 895424 08	mov dword ptr ss:[esp+0x8],edx	
004012AB	- 66:894424 0C	mov word ptr ss:[esp+0xC],ax	
004012B0	- 8D5424 28	lea edx,dword ptr ss:[esp+0x28]	
004012B4	- 8D4424 08	lea eax,dword ptr ss:[esp+0x8]	
004012B8	- 52	push edx	
004012B9	- 50	push eax	StringToAdd = "ssssssssss" ConcatString = "Happy@!GA0U"
004012BA	- 884C24 16	mov byte ptr ss:[esp+0x16],cl	
004012BE	- FF15 04504000	call dword ptr ds:[<&KERNEL32.lstrcatA]	lstrcatA
004012C4	- 8D4C24 08	lea ecx,dword ptr ss:[esp+0x8]	
004012C8	- 8D5424 48	lea edx,dword ptr ss:[esp+0x48]	
004012CC	- 51	push ecx	
004012CD	- 52	push edx	String2 = "Happy@!GA0U" String1 = "ssssssssss"
004012CE	- FF15 00504000	call dword ptr ds:[<&KERNEL32.lstrcpA]	lstrcpA
004012D4	- F7D8	neg eax	
004012D6	- 1BC0	sbb eax,eax	
004012D8	- 5F	pop edi	

到了这一步应该有所警觉 是不是到了比较的部分 然后我们试着输入 看着flag对不对 然后输入了Happy@!GA0U 发现对了 我们可以知道我们地方找对了 然后就是算法分析部分了（大家应该有一个好习惯 要保持一定的好奇心） 然后我们往上走 发现了

00401260	- F7D1	not ecx	
00401262	- 49	dec ecx	
00401263	~ 74 30	je short CrackMe1.00401295	
00401265	> 0FBF4434 08	movsx eax,byte ptr ss:[esp+esi+0x8]	可疑点
0040126A	- 0FAFC0	imul eax,eax	
0040126D	- 0FAFC6	imul eax,esi	
00401270	- 03C6	add eax,esi	
00401272	- 33D2	xor edx,edx	
00401274	- B9 42000000	mov ecx,0x42	
00401279	- 8D7C24 08	lea edi,dword ptr ss:[esp+0x8]	
0040127D	- F7F1	div ecx	
0040127F	- 83C9 FF	or ecx,-0x1	
00401282	- 33C0	xor eax,eax	
00401284	- 80C2 21	add dl,0x21	
00401287	- 885434 28	mov byte ptr ss:[esp+esi+0x28],dl	
0040128B	- 46	inc esi	
0040128C	- F2:AE	repne scas byte ptr es:[edi]	
0040128E	- F7D1	not ecx	
00401290	- 49	dec ecx	
00401291	- 3BF1	cmp esi,ecx	
00401293	^ 72 D0	jb short CrackMe1.00401265	
00401295	> 8B15 64604000	mov edx,dword ptr ds:[0x406064]	

这里是可疑点 然后经过分析 大概每一步是这样的

0040125E	- F2:AE	repne scas byte ptr es:[edi]	
00401260	- F7D1	not ecx	
00401262	- 49	dec ecx	
00401263	~ 74 30	je short CrackMe1.00401295	
00401265	> 0FBF4434 08	movsx eax,byte ptr ss:[esp+esi+0x8]	循环的是我们输入的名字
0040126A	- 0FAFC0	imul eax,eax	eax*eax
0040126D	- 0FAFC6	imul eax,esi	eax *esi
00401270	- 03C6	add eax,esi	eax+esi
00401272	- 33D2	xor edx,edx	edx=0
00401274	- B9 42000000	mov ecx,0x42	ecx=0x42
00401279	- 8D7C24 08	lea edi,dword ptr ss:[esp+0x8]	edi= name
0040127D	- F7F1	div ecx	eax/ecx
0040127F	- 83C9 FF	or ecx,-0x1	ecx=-1
00401282	- 33C0	xor eax,eax	eax=0
00401284	- 80C2 21	add dl,0x21	edx+0x21
00401287	- 885434 28	mov byte ptr ss:[esp+esi+0x28],dl	dl存的是eax/ecx的商 dl存放一个空间
0040128B	- 46	inc esi	循环 索引
0040128C	- F2:AE	repne scas byte ptr es:[edi]	找到a1的值或者cx=0 每次循环cx-1 又因为cx=-1 所以就是找a1的值
0040128E	- F7D1	not ecx	在这里是ecx是从还剩多少字母
00401290	- 49	dec ecx	ecx=-ecx
00401291	- 3BF1	cmp esi,ecx	ecx实际上就是我们name的长度
00401293	^ 72 D0	jb short CrackMe1.00401265	

其中的循环 为什么 按位取反然后-1为啥等于相反数 这个可以自己慢慢体会 其实这就是esi是个索引值 ecx就是 len (name) 然后

004012B0	- 8D5424 28	lea edx,dword ptr ss:[esp+0x28]	
004012B4	- 8D4424 08	lea eax,dword ptr ss:[esp+0x8]	
004012B8	- 52	push edx	
004012B9	- 50	push eax	StringToAdd = "!GA0U"
004012BA	- 884C24 16	mov byte ptr ss:[esp+0x16],cl	ConcatString = "Happy@"
004012BE	- FF15 0450400	call dword ptr ds:[<&KERNEL32.lstrcatA>]	lstrcatA
004012C4	- 8D4C24 08	lea ecx,dword ptr ss:[esp+0x8]	
004012C8	- 8D5424 48	lea edx,dword ptr ss:[esp+0x48]	

eax=0019F79C, (ASCII "Happy@")

[https://blog.csdn.net/qq\\_41071040](https://blog.csdn.net/qq_41071040)

这个是将与我们得到的合为一个字符串 那么 我们可以用python来表达这个过程

```
lists="Happy@"
name="hello"

for i in range(len(name)):
    lists+=chr((i+i*ord(name[i])*ord(name[i]))%0x42+0x21)

print(lists)
```

感觉这个题也不是很难