

VisualSite Designer破解教程

0x1, 次数破解:

1, 首先

在原文件中



有Number字符, 看来是因为次数会限制, 那么我们就先来破解它。

2, 由于无壳, 就直接上OD了:

004BD497	. E8 74000000	call VisualSi.004BD510	
004BD49C	. 8945 98	mov [local.26],eax	

下断点:

3, ctrl+F2, F9到断点, F7进入函数

004BD51C	. FF7424 10	push dword ptr ss:[esp+0x10]	Visua
004BD520	. E8 43000000	call <jmp.&MFC42.#1576>	
004BD525	. C2 1000	retn 0x10	

4, 老规矩, 继续:

0FC86241	8BCF	mov ecx,edi	
0FC86243	FFD6	call esi	
0FC86245	85C0	test eax,eax	

5, 再次就有点意思了, eax与1比较, 然后跳转, 这是关键所在, 严谨其间还是在进去一次吧;

00489912	. E8 132B0300	call <jmp.&MFC42.#2514>	
00489917	. 83F8 01	cmp eax,0x1	

6, 又找到了一个:

0FCB3507	50	push eax
0FCB3508	E8 53CFFEFF	call mfc42.#5718
0FCB350D	395F 20	cmp dword ptr ds:[edi+0x20],ebx

The screenshot shows a debugger window with assembly code. A dialog box is open, allowing the user to insert instructions at a specific address. The dialog box contains the following elements:

- Address:** 00489912
- Instruction:** `mov eax, 1`
- Options:** ☒ 使用 NOP 填充
- Buttons:** 汇编 (Compile), 取消 (Cancel)

The assembly code in the background includes instructions like `lea ecx, dword ptr ss:[esp+0x20C]`, `mov byte ptr ss:[esp+0x8778], 0xB`, `call <jmp.&MFC42.#2514>`, `cmp eax, 0x1`, and `mov byte ptr ss:[esp+0x8778], 0xB`.

0x2, 广告弹窗:

1, 在OD中运行程序, 再关掉, 在OD中点击暂停:

然后进入堆栈段查看调用:

地址	堆栈	函数过程 / 参数	调用来自	结构
0019E9FC	76B48CE5	win32u.NtUserGetMessage	user32.76B48CDF	0019EA38
0019EA3C	0F5A3843	user32.GetMessageA	mfc42.0F5A383D	0019EA38
0019EA40	0058B264	pMsg = VisualSi.0058B264		
0019EA44	00000000	hWnd = NULL		
0019EA48	00000000	MsgFilterMin = 0x0		
0019EA4C	00000000	MsgFilterMax = 0x0		
0019EA60	0F5C056B	包含 mfc42.0F5A3843	mfc42.0F5C056B	0019EA5C
0019EA8C	0F5D350D	mfc42.#5718	mfc42.0F5D3508	0019EA88
0019EAD6	00480C29	? <jmp.&MFC42.#2514>	VisualSi.00480C24	0019EAD4
0019EB48	0F5A83D5	包含 VisualSi.00480C29	mfc42.0F5A83D3	0019EB4C
0019EB50	0F59F73B	mfc42.0F5A83C0	mfc42.0F59F736	0019EB4C
0019EBF0	0F59FE40	包含 mfc42.0F59F73B	mfc42.0F59FE3E	0019EBEC
0019EC1C	0F59A44B	包含 mfc42.0F59FE40	mfc42.0F59A449	0019EC18
0019EC9C	0F59A2FF	mfc42.#1109	mfc42.0F59A2FA	0019EC98
0019ECE4	76B5BF1B	包含 mfc42.0F59A2FF	user32.76B5BF19	0019ECE0
0019ED10	76B583EA	user32.76B5BEF0	user32.76B583E5	0019ED0C
0019EDF8	76B57F8A	user32.76B58040	user32.76B57F85	0019EDF4
0019EE5C	76B5A6D9	包含 user32.76B57F8A	user32.76B5A6D7	0019EE58
0019EE9C	7795CD3D	包含 user32.76B5A6D9	ntdll.7795CD3B	0019EE98

在所选位置，我们看到其他都是mfc的系统调用，只有所选位置是程序自身的，点击进去相应位置：

00480C1C	. C74424 68 00	mov dword ptr ss:[esp+0x68],0x0
00480C24	. E8 01B80300	call <jmp.&MFC42.#2514>
00480C29	. 8D4C24 00	lea ecx,dword ptr ss:[esp]
00480C2D	. C74424 68	
00480C35	. E8 DEBA03	
00480C3A	. 8B4C24 60	
00480C3E	. 64:890D 0	
00480C45	. 83C4 6C	
00480C48	. C3	
00480C49	. 90	

汇编于此处: 00480C24

nop|

☒ 使用 NOP 填充

汇编

取消

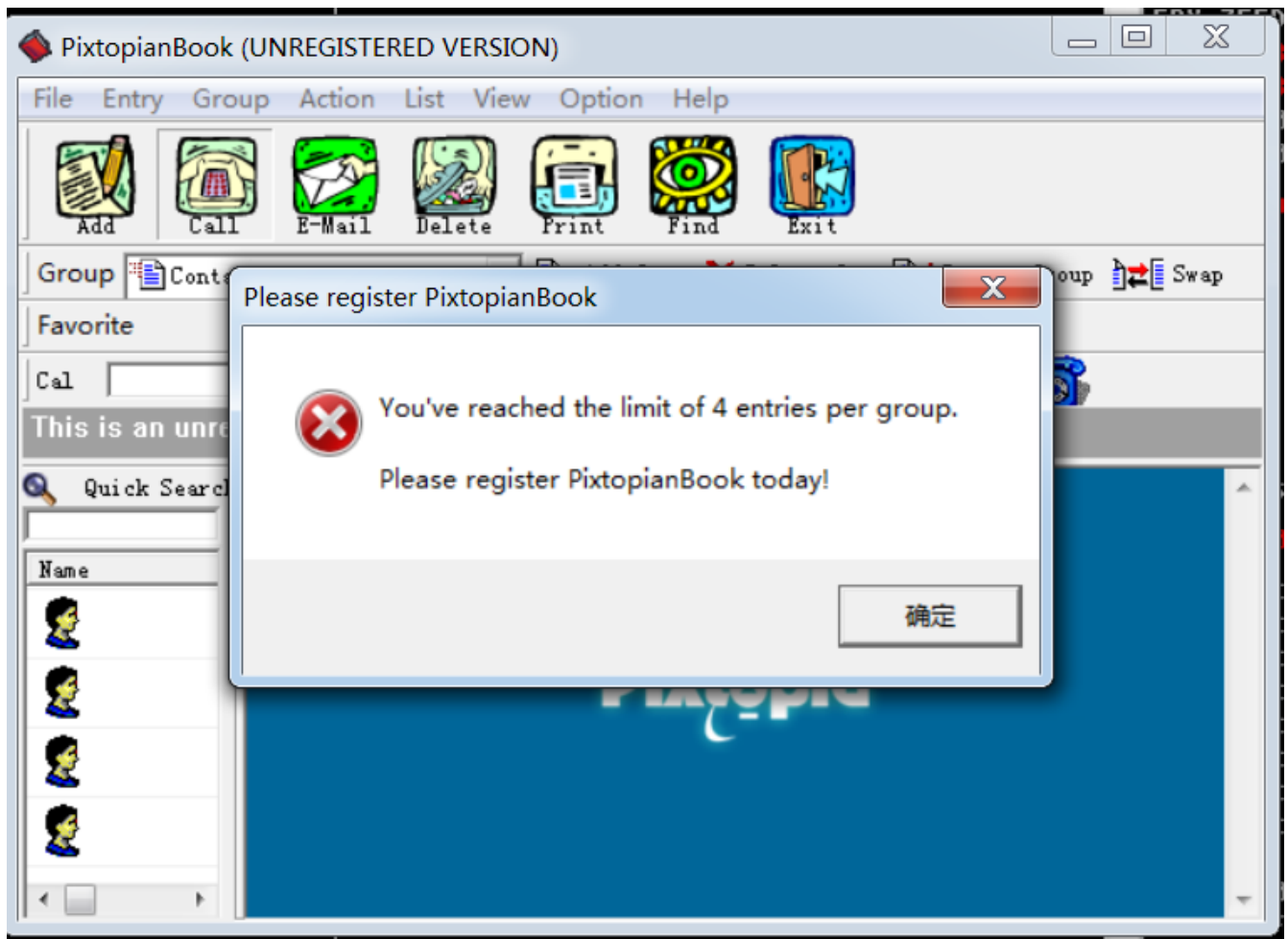
nop填充，保存文件。

这后，就没有广告了。

pixtopianbook107破解教程

0x1，添加联系人限制：

1，原始程序只能添加四个联系人，现在突破限制；



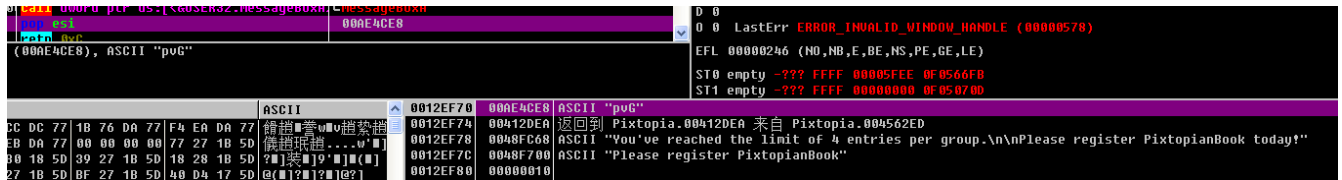
2, 现在OD中载入程序, 运行程序, 添加练习人, 当人数到弹窗后在OD中暂停, 进入堆栈段:

调用堆栈: 主线程					
地址	堆栈	函数过程 / 参数	调用来自	结构	
0012EA24	77D193F5	包含 ntdll.KiFastSystemCallRet	user32.77D193F3	0012EA58	
0012EA28	77D3EA24	user32.WaitMessage	user32.77D3EA1F	0012EA58	
0012EA5C	77D2688A	user32.77D3E895	user32.77D26885	0012EA58	
0012EA84	77D3B7C5	user32.77D267D4	user32.77D3B7C0	0012EA80	
0012ED44	77D3B12B	user32.SoftModalMessageBox	user32.77D3B126	0012ED40	
0012EE94	77D65FDF	user32.77D3AFB6	user32.77D65FDA	0012EE90	
0012EEEC	77D66084	user32.MessageBoxTimeoutW	user32.77D6607F	0012EE88	
0012EF20	77D50598	? user32.MessageBoxTimeoutA	user32.77D50593	0012EF1C	
0012EF40	77D50550	? user32.MessageBoxExA	user32.77D5054B	0012EF3C	
0012EF44	000701C4	hOwner = 000701C4 (class= '#32770'			
0012EF48	0048FC68	Text = "You've reached the limit			
0012EF4C	0048F700	Title = "Please register Pixtopia			
0012EF50	00000010	Style = MB_OK MB_ICONHAND MB_APPL			
0012EF54	00000000	LanguageID = 0x0 (LANG_NEUTRAL)			
0012EF5C	0045631B	? user32.MessageBoxA	Pixtopia.00456315	0012EF58	
0012EF60	000701C4	hOwner = 000701C4 (class= '#32770'			
0012EF64	0048FC68	Text = "You've reached the limit			
0012EF68	0048F700	Title = "Please register Pixtopia			
0012EF6C	00000010	Style = MB_OK MB_ICONHAND MB_APPL			
0012EF74	00412DEA	? Pixtopia.004562ED	Pixtopia.00412DE5		

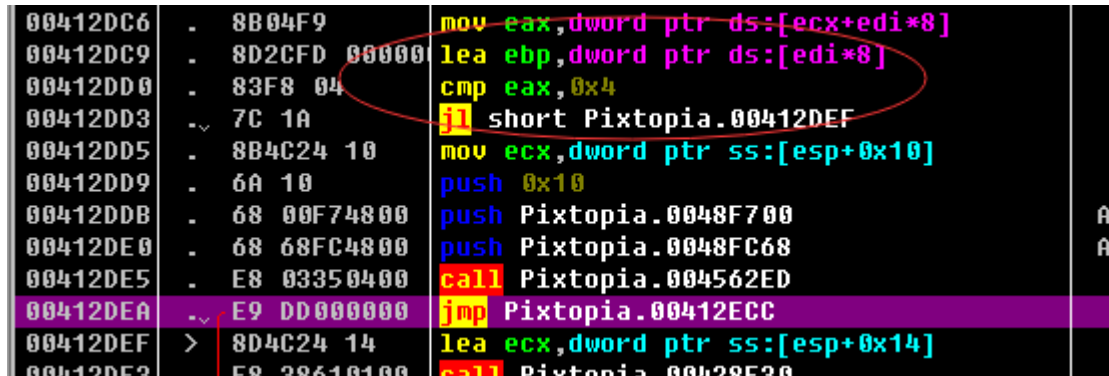
在用户段看到调用弹窗的地址, 重新加载OD, 找到对应位置:

00456315	. FF15 0456470	call dword ptr ds:[&USER32.MessageBoxA]	LMessageBoxA
0045631B	. 5E	pop esi	
0045631C	. C2 0C00	retn 0xC	
0045631F	. 55	push ebp	

3, 看反汇编窗口跟随

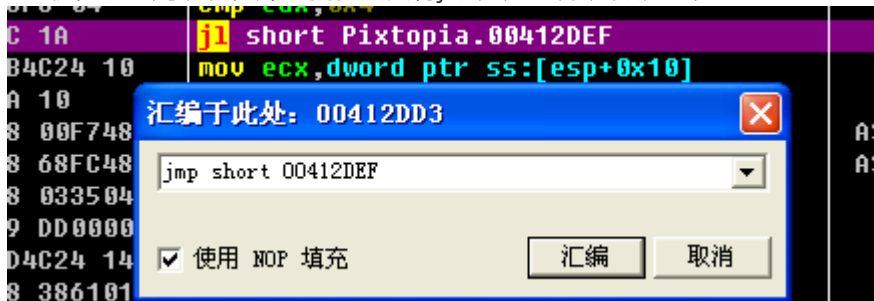


找到此API的位置, 现在到此位置来看看,

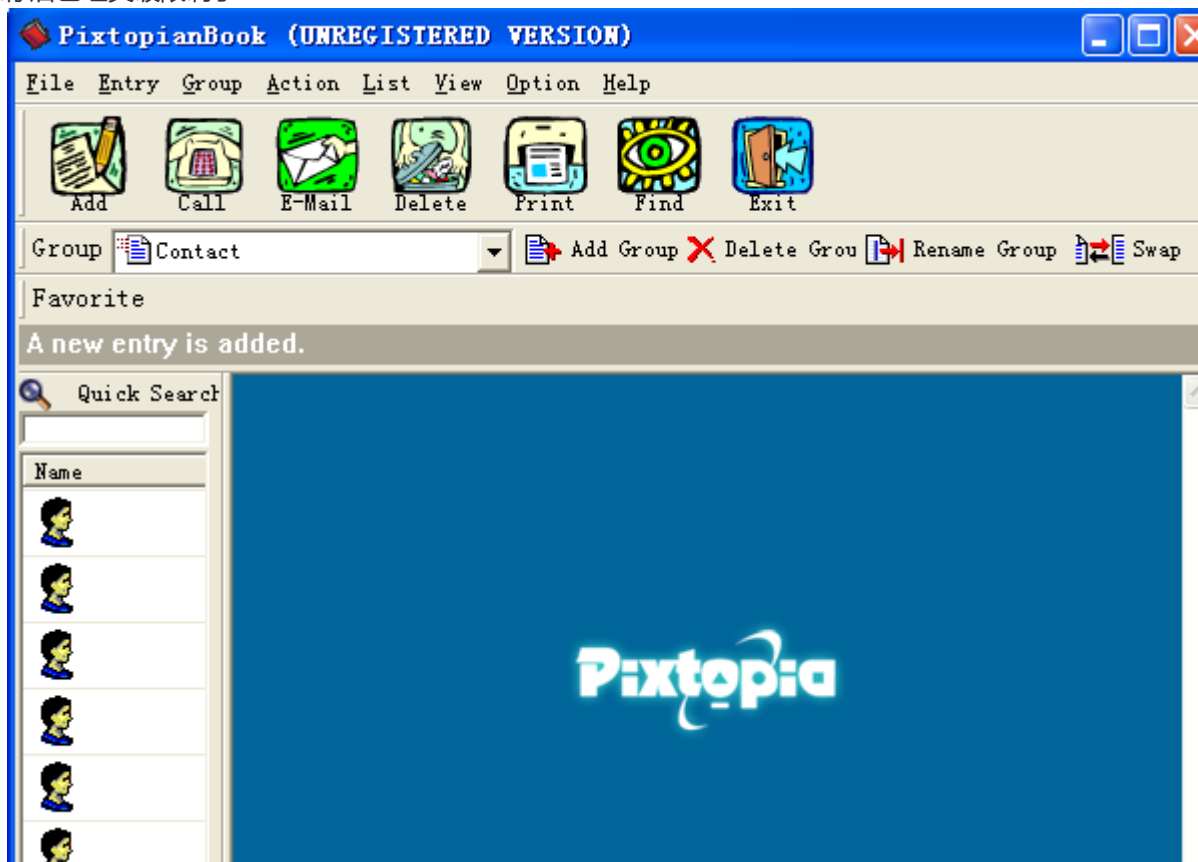


看到这有比较, 与

4比较, 小于则跳转, 那么我们只要将j改为无条件跳转就可以了:



4, 保存后已经突破限制了:



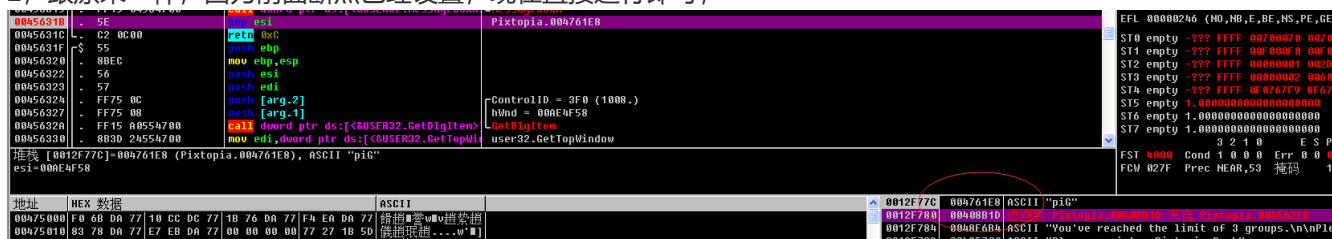
0x2, Add Group限制

1, 现在只能添加两组:

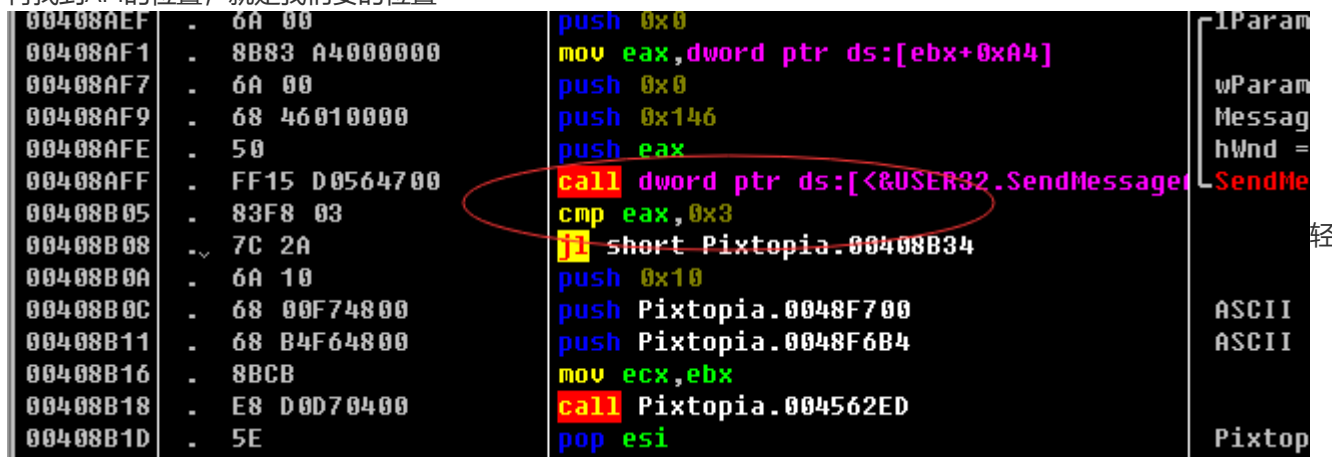


根据上次的来试一试吧;

2, 跟原来一样, 因为前面断点已经设置, 现在直接运行即可,



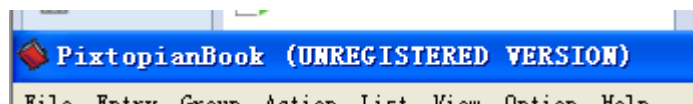
再找到API的位置, 就是我们要的位置



车熟路的改成无条件跳转即可, 此后Add Group没有限制;

0x3, 美观注册破解

1,



尽管大多部分已经破解完毕, 但是有这种未注册的字样, 现在我们来破解他们;

2, 我们去内存找一下,

地址	大小	属主	区段	包含	类型	访问	初始访问	已映射为
00250000	00006000				Priv	RW	RW	
00260000	00003000				Map	RW	RW	
00270000	00016000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\system32\
00290000	0003D000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\system32\
002D0000	00041000							\Device\HarddiskVolume1\WINDOWS\system32\
00320000	00008000							\Device\HarddiskVolume1\WINDOWS\system32\
00330000	00041000							
00380000	00008000							
00390000	00008000							
003A0000	00001000							
003B0000	00001000							
003C0000	00003000							
003D0000	00004000							\Device\HarddiskVolume1\WINDOWS\system32\
003E0000	00002000							
003F0000	00003000							
00400000	00001000							
00401000	00074000							
00475000	0001A000							
0048F000	0000E000							
0049D000	0004C000							
004F0000	00003000				map	R E	R E	
005B0000	00002000				Map	R E	R E	
005C0000	00103000				Map	R	R	

输入要查找的二进制字符串

ASCII: U.N.R.E.G.I.S.T.E.R.E.D.

UNICODE: UNREGISTERED

HEX +18: 55 00 4E 00 52 00 45 00 47 00 49 00 53 00 54 00 45 00 52 00 45 00 44 00

☒ 整个块 ☐ 区分大小写

确定 取消

运行

0017CAFA	55 00 4E 00	52 00 45 00	47 00 49 00	53 00 54 00	U.N.R.E.G.I.S.T.	
0017CB0A	45 00 52 00	45 00 44 00	29 00 00 00	00 00 49 00	E.R.E.D.)...I.	
0017CB1A	53 00 54 00	45 00 52 00	45 00 44 00	29 00 00 00	S.T.E.R.E.D.)...	
0017CB2A	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
0017CB3A	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
0017CB4A	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
0017CB5A	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
0017CB6A	00 00 00 00	00 00 3E 00	A6 00 F0 01	0C 00 00 00>??...	
0017CB7A	00 00 70 01	00 00 04 00	00 00 20 00	00 00 10 00	..pr... ..+	
0017CB8A	00 00 00 00	00 00 00 00	00 00 00 01	00 00 00 00	

现在知道地址了, 在dump

面板跳转;

3,

地址	HEX 数值	ASCII
004D4830	55 00 6E 00 72 00 65 00 67 00 69 00 73 00 74 00	U.n.r.e.g.i.s.t.
004D4840	65 00 72 00 65 00 64 00 20 00 76 00 65 00 72 00	e.r.e.d. .v.e.r.
004D4850	73 00 69 00 6F 00 6E 00 20 00 76 00 31 00 2E 00	s.i.o.n. .v.1...

4, 这句话

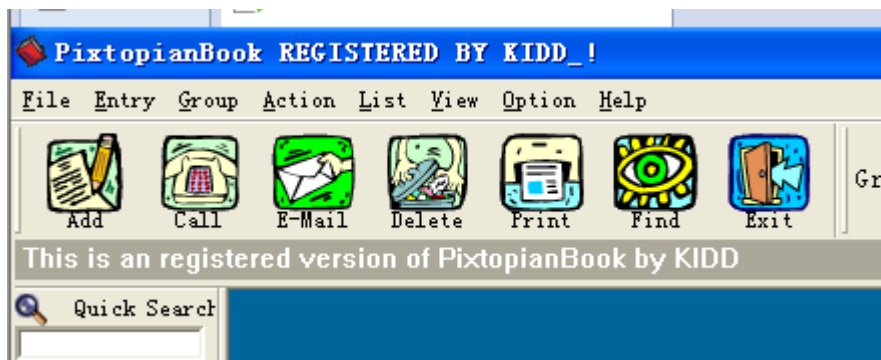


根据

0040C002	push Pixtopia.0046FD08	有MH
0040C237	push Pixtopia.0046F974	This is an unregistered version of PixtopianBook. Please register today!
0040C3B8	push Pixtopia.0046FD73	欲MH
0040C712	push Pixtopia.0046FDE7	欲MH
0040CD48	push Pixtopia.0046FEA8	赴OH

得到地址

然后更改即可:



解决。