

前言  
软件概况  
分析程序  
栈回溯分析About点击事件  
获取必要的API  
添加区段  
    配置区段  
    添加区段数据  
植入代码  
修改目标函数  
校验结果

【软件名称】： defiler.1.exe

【软件大小】： 287KB

【下载地址】： 自行搜索下载

【加壳方式】： 无壳

【保护方式】： 无保护

【编译语言】： Delphi

【调试环境】： W7 32

【使用工具】： OD + IDA + Darkde

【破解日期】： 2019年5月15日

【破解目的】： 纯属兴趣

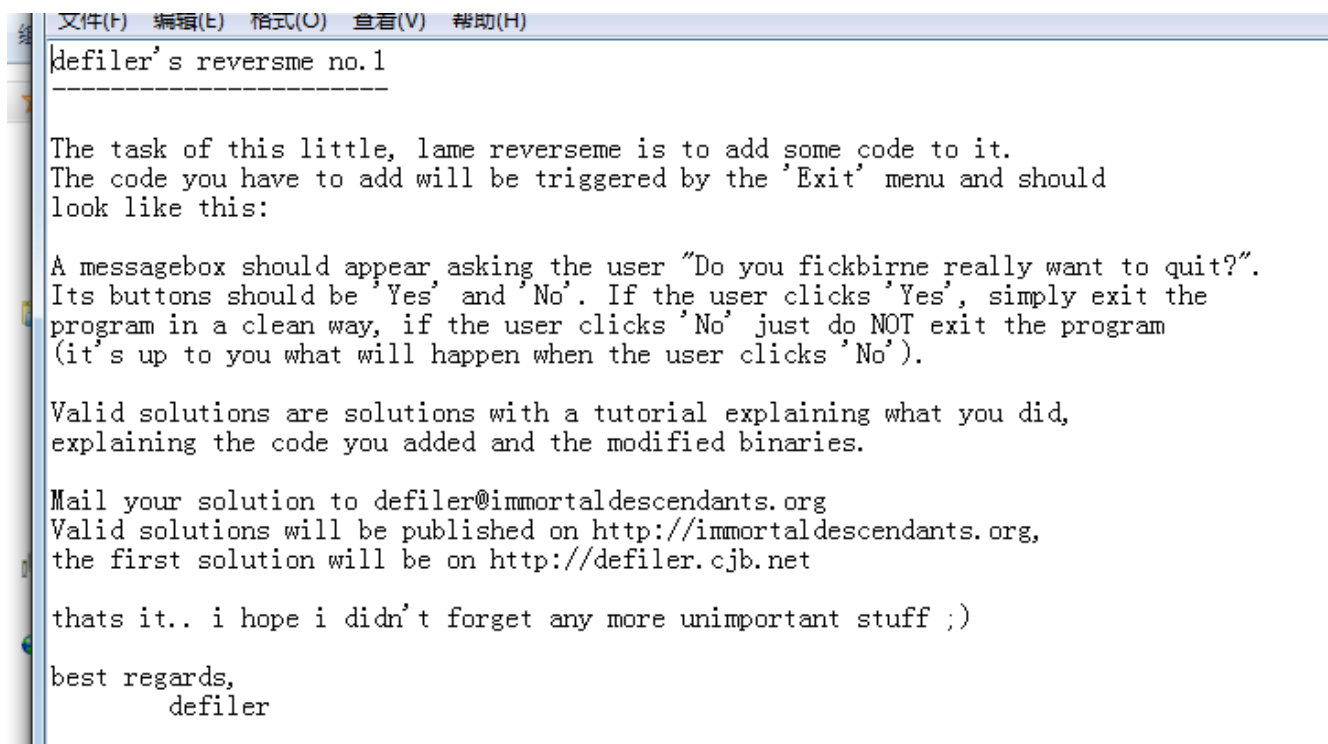
## 前言

---

本文参考自吾爱破解论坛zbnysjwsnd8的文章: <https://www.52pojie.cn/thread-654237-1-1.html>

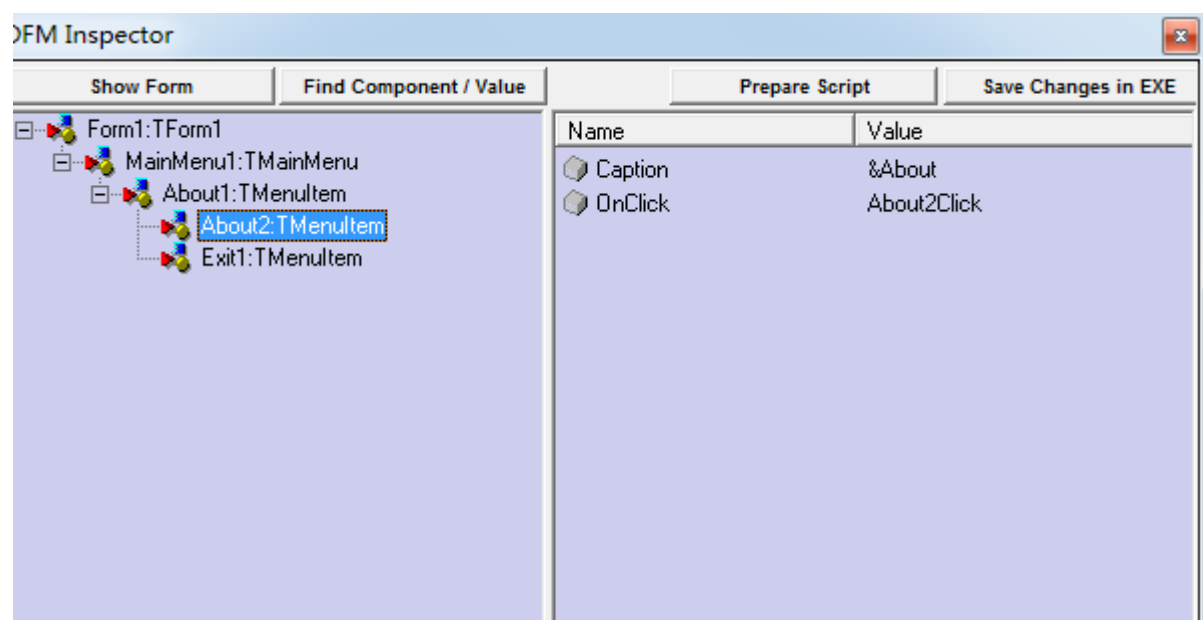
## 软件概况

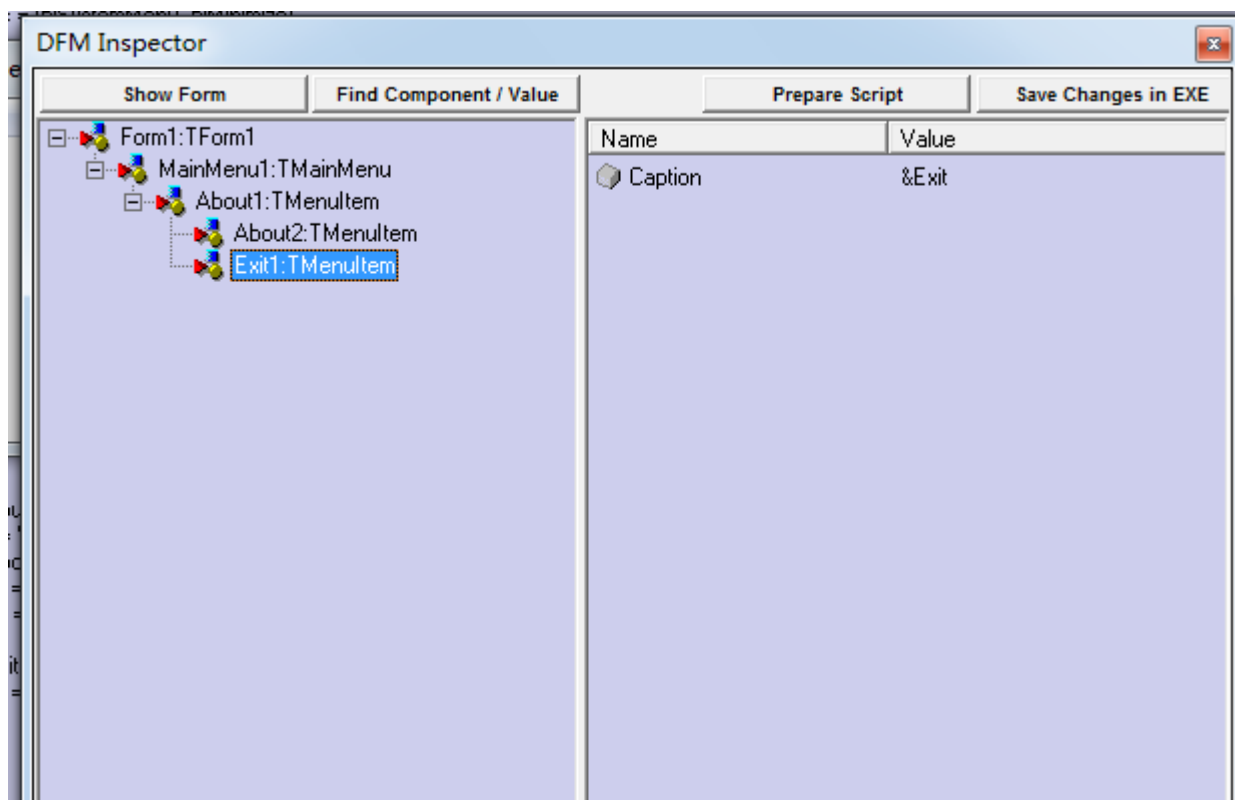
---



这个Crackme的要求大概是要我们往Exit菜单中添加代码，让程序在点击Exit菜单时弹出MessageBox框。这个属于软件重构的范围了。

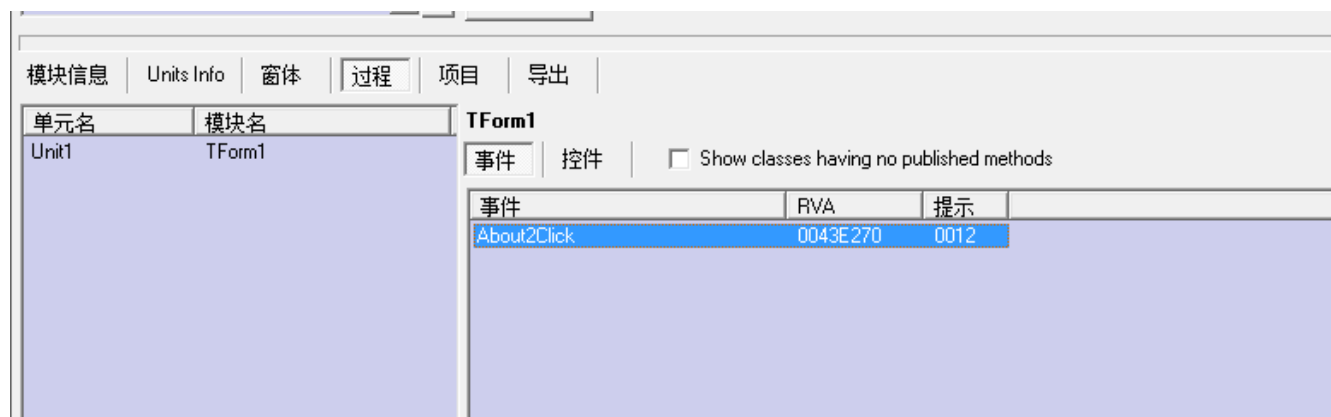
## 分析程序





使用DarkDe分析程序，这两个菜单中，只有about是有点击事件的，Exit菜单是没有点击事件。那么我们先从about菜单事件开始分析

## 栈回溯分析About点击事件



直接找到About的点击事件的RVA，在OD中下断点



about的点击事件中调用了一个MessageBox，然后直接Ctrl+F9返回

0042F411	. FF52 18	call dword ptr ds:[edx+0x18]		ECX 0012F9
0042F414	. EB 18	jmp short <defiler_.loc_42F42E>		EDX 775970
0042F416	> 66:83BB 8200	cmp word ptr ds:[ebx+0x82],0x0	比较[ebx+0x82]是否为0	EBX 011634
0042F41E	. 74 0E	je short <defiler_.loc_42F42E>		ESP 0012F9
0042F420	. 8BD3	mov edx,ebx		EBP 0012FB
0042F422	. 8B83 84000000	mov eax,dword ptr ds:[ebx+0x84]		ESI 0012FB
0042F428	. FF93 80000000	call dword ptr ds:[ebx+0x80]	<defiler_.TForm1_About2Click>	EDI 0012FC
0042F42E	> 5B	pop ebx	loc_42F42E	EIP 0042F4
0042F42F	. C3	ret		C 0 ES 00
0042F430	. 56	push esi	Menus::TMenuItem::IndexOf(Menus::TMenuItem *)	P 1 CS 00
0042F431	. 83C9 FF	or ecx,-0x1		A 0 SS 00
0042F434	. 8B70 50	mov esi,dword ptr ds:[eax+0x50]		Z 1 DS 00
0042F437	. 85F6	test esi,esi		S 0 FS 00
0042F439	. 74 09	je short <defiler_.loc_42F444>		T 0 CS 00
0042F43B	. 9BFC	mov ebx,esi		

然后找到IDA中对应的代码

```
1 int __fastcall Menus::TMenuItem::Click(int result)
2 {
3     int v1; // ebx
4
5     v1 = result;
6     if ( (*_BYTE *)(result + 0x2D) )
7     {
8         if ( (*_WORD *)(result + 0x82)
9             && (result = Menus::TMenuItem::GetAction((Menus::TMenuItem *)result)) != 0
10            && (result = (*_DWORD *) (Menus::TMenuItem::GetAction((Menus::TMenuItem *)v1) + 48), result != (*_DWORD *) (v1 + 128)) )
11        {
12            result = (* (int (__fastcall *) (_DWORD, int)) (v1 + 128)) (*_DWORD *) (v1 + 132), v1;
13        }
14        else if ( (*_BYTE *) (v1 + 0x20) & 0x10 || !(*_DWORD *) (v1 + 56) )
15        {
16            if ( (*_WORD *) (v1 + 0x82) )
17                result = (* (int (__fastcall *) (_DWORD, int)) (v1 + 128)) (*_DWORD *) (v1 + 132), v1; // 调用About菜单响应事件
18        }
19        else
20        {
21            result = (* (int (**)(void)) (**_DWORD *) (v1 + 56) + 24) ();
22        }
23    }
24    return result;
25 }
```

这里就是菜单的响应事件了，然后再次按Ctrl+F9返回

0043052D	. 51 D1D2	mov eax, eax		
0043052E	. 33C9	xor ecx,ecx		
00430530	. E8 53FFFFFF	call <defiler_.Menus::TMenu::FindItem(int, Menus::TFindItemKind)>		
00430535	. 85C0	test eax, eax		
00430537	. 74 07	je short <defiler_.loc_430540>		
00430539	. 8B10	mov ebx, dword ptr ds:[eax]		
0043053B	. FF52 40	call dword ptr ds:[edx+0x40]		
0043053E	. B3 01	mov bl, 0x1		
00430540	> 8BC3	mov eax, ebx	loc_430540	
00430542	. 5B	pop ebx	011617E4	
00430543	. C3	ret		
00430544	. 53	push ebx	Menus::TMenu::DispatchPopup(uint)	
00430545	. 56	push esi		
00430546	. 57	push edi		
00430547	. 8BF0	mov esi, eax		
00430549	. 33DB	xor ebx, ebx		
0043054B	. B1 01	mov cl, 0x1		

```
1 int __fastcall Menus::TMenu::DispatchCommand(Menus::TMenu *this, unsigned __int16 a2)
2 {
3     int v2; // ebx
4     int v3; // eax
5
6     v2 = 0;
7     v3 = Menus::TMenu::FindItem(this, a2, 0);
8     if ( v3 )
9     {
10         (* (void (**)(void)) (*_DWORD *) (v3 + 64)) (); |
11         LOBYTE(v2) = 1;
12     }
13     return v2;
14 }
```

这里是DispatchCommand函数，再往上回溯就是消息循环了，我们直接在0043053B的菜单点击事件下断点，点击About按钮，F7进入函数

地址	HEX 数据	反汇编	注释	寄存器 (3DNow!)
0042F3C0	. 53	push ebx	Menu::TMenuItem::Click(void)	EAX 011634F8
0042F3C1	. 8BD8	mov ebx, eax	ebx=菜单相关结构体	ECX 0012F9D0
0042F3C3	. 807B 2D 00	cmp byte ptr ds:[ebx+0x2D], 0x0	比较[ebx+0x2D]是否为0	EDX 0042BE04 <defiler_.cls_MenuItem
0042F3C7	. 74 65	je short <defiler_.loc_42F42E>		EBX 011634F8
0042F3C9	. 66:83BB 8200	cmp word ptr ds:[ebx+0x82], 0x0	比较[ebx+0x82]是否为0	ESP 0012F9E8
0042F3D1	. 74 2D	je short <defiler_.loc_42F400>		EBP 0012FB40
0042F3D3	. 8BC3	mov eax, ebx		ESI 0012FBA0
0042F3D5	. E8 22FDFFFF	call <defiler_.Menu::TMenuItem::GetAction(void)>		EDI 0012FC2C
0042F3DA	. 85C0	test eax, eax		EIP 0042F3C3 defiler_.0042F3C3
0042F3DC	. 74 22	je short <defiler_.loc_42F400>		C 0 ES 0023 32位 0(FFFFFFFF)
0042F3DE	. 8BC3	mov eax, ebx		P 0 CS 001B 32位 0(FFFFFFFF)
0042F3E0	. E8 17FDFFFF	call <defiler_.Menu::TMenuItem::GetAction(void)>		A 0 SS 0023 32位 0(FFFFFFFF)
0042F3E5	. 8B40 30	mov eax, dword ptr ds:[eax+0x30]		Z 0 DS 0023 32位 0(FFFFFFFF)
0042F3E8	. 3B83 80000000	cmp eax, dword ptr ds:[ebx+0x80]	<defiler_.TForm1_About2Click>	S 0 FS 003B 32位 7FFDF000(FFF)
0042F3EE	. 74 10	je short <defiler_.loc_42F400>		T 0 GS 0000 NULL
0042F3F0	. 8BD3	mov edx, ebx		D 0
0042F3F2	. 8B83 84000000	mov eax, dword ptr ds:[ebx+0x84]		O 0 LastErr ERROR_SUCCESS (00000000)
0042F3F8	. FF93 80000000	call dword ptr ds:[ebx+0x80]	<defiler_.TForm1_About2Click>	EFL 00000202 (NO, NB, NE, A, NS, PO, GE, G
0042F3FE	. EB 2E	jmp short <defiler_.loc_42F42E>		MM0 0.0, 0.0
0042F400	. F643 20 10	test byte ptr ds:[ebx+0x20], 0x10	比较[ebx+0x20]是否为0x10	MM1 0.0, 0.0
0042F404	. 75 10	jnz short <defiler_.loc_42F416>		MM2 0.0, 0.0

地址	数值	注释	地址	数值	注释
011634F8	0042BE04	<defiler_.cls_MenuItem>	0012F9E8	00000000	
011634FC	011617E4		0012F9EC	0043053E	返回到 defiler_.Menu::TMenu::DispatchCommand(ushort)+16
01163500	011634AC	ASCII "About2"	0012F9F0	011617E4	
01163504	00000000		0012F9F4	00438B8C	返回到 defiler_.00438B8C 来自 <defiler_.Menu::TMenu::DispatchCom
01163508	00000000		0012F9F8	0012FBA0	
0116350C	00000000		0012F9FC	011617E4	
01163510	00000000		0012FA00	00420350	返回到 defiler_.Controls::TControl::WndProc(Messages::TMessage &)
01163514	00000000		0012FA04	0012FC2C	
01163518	00010000		0012FA08	0012FBA0	
0116351C	011634C0	ASCII "&about"	0012FA0C	011617E4	
01163520	00000000		0012FA10	00422D47	返回到 defiler_.Controls::TWinControl::WndProc(Messages::TMessage
01163524	02000100		0012FA14	0012FC2C	
01163528	00010002		0012FA18	0012FBA0	
0116352C	FFFFFFFF		0012FA1C	011617E4	
01163530	00000000		0012FA20	0012FB10	
01163534	00000000		0012FA24	00000092	
01163538	00000000		0012FA28	0012FB10	

此时ebx的值为about控件的结构体，有about控件相关的字符串

然后再次点击Exit菜单，F7进入函数

地址	HEX 数据	反汇编	注释	寄存器 (3DNow!)
0042F3C0	. 53	push ebx	Menu::TMenuItem::Click(void)	EAX 01163
0042F3C1	. 8BD8	mov ebx, eax	ebx=菜单相关结构体	ECX 0012F
0042F3C3	. 807B 2D 00	cmp byte ptr ds:[ebx+0x2D], 0x0	比较[ebx+0x2D]是否为0	EDX 0042B
0042F3C7	. 74 65	je short <defiler_.loc_42F42E>		EBX 01163
0042F3C9	. 66:83BB 8200	cmp word ptr ds:[ebx+0x82], 0x0	比较[ebx+0x82]是否为0	ESP 0012F
0042F3D1	. 74 2D	je short <defiler_.loc_42F400>		EBP 0012F
0042F3D3	. 8BC3	mov eax, ebx		ESI 0012F
0042F3D5	. E8 22FDFFFF	call <defiler_.Menu::TMenuItem::GetAction(void)>		EDI 0012F
0042F3DA	. 85C0	test eax, eax		EIP 0042F
0042F3DC	. 74 22	je short <defiler_.loc_42F400>		C 0 ES 0
0042F3DE	. 8BC3	mov eax, ebx		P 1 CS 0
0042F3E0	. E8 17FDFFFF	call <defiler_.Menu::TMenuItem::GetAction(void)>		A 0 SS 0
0042F3E5	. 8B40 30	mov eax, dword ptr ds:[eax+0x30]		Z 0 DS 0
0042F3E8	. 3B83 80000000	cmp eax, dword ptr ds:[ebx+0x80]		S 0 FS 0
0042F3EE	. 74 10	je short <defiler_.loc_42F400>		T 0 GS 0
0042F3F0	. 8BD3	mov edx, ebx		D 0
0042F3F2	. 8B83 84000000	mov eax, dword ptr ds:[ebx+0x84]		O 0 Last
0042F3F8	. FF93 80000000	call dword ptr ds:[ebx+0x80]		EFL 00000
0042F3FE	. EB 2E	jmp short <defiler_.loc_42F42E>		MM0
0042F400	. F643 20 10	test byte ptr ds:[ebx+0x20], 0x10	比较[ebx+0x20]是否为0x10	MM1
0042F404	. 75 10	jnz short <defiler_.loc_42F416>		MM2

地址	数值	注释	地址	数值	注释
01163628	0042BE04	<defiler_.cls_MenuItem>	0012F9E8	00000000	
0116362C	011617E4		0012F9EC	0043053E	返回到 defiler_.Menu::TMenu::DispatchC
01163630	0116361C	ASCII "Exit1"	0012F9F0	011617E4	
01163634	00000000		0012F9F4	00438B8C	返回到 defiler_.00438B8C 来自 <defiler_.
01163638	00000000		0012F9F8	0012FBA0	
0116363C	00000000		0012F9FC	011617E4	
01163640	00000000		0012FA00	00420350	返回到 defiler_.Controls::TControl::Wnf
01163644	00000000		0012FA04	0012FC2C	
01163648	00010000		0012FA08	0012FBA0	
0116364C	01163710	ASCII "&Exit"	0012FA0C	011617E4	
01163650	00000000		0012FA10	00422D47	返回到 defiler_.Controls::TWinControl::V
01163654	02000100		0012FA14	0012FC2C	
01163658	00010002		0012FA18	0012FBA0	
0116365C	FFFFFFFF		0012FA1C	011617E4	
01163660	00000000		0012FA20	0012FB10	
01163664	00000000		0012FA24	00000092	

此时ebx的值为Exit控件的结构体，有Exit控件相关的字符串。

那么我们可以根据ebx的不同，植入一段代码，判断此时ebx值的情况，如果ebx值有Exit字符串的话，就说明Exit按钮被点击，然后执行我们自己的函数，否则继续执行原有函数。相当于变相给菜单添加了点击事件

## 获取必要的API

首先需要拿到MessageBox和ExitProcess的函数地址，查看导入表

[ 输入表 ]					
DllName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	名称	FirstThunk
kernel32.dll	00000000	00000000	00000000	00041B18	00041208
gdi32.dll	00000000	00000000	00000000	00041EA4	000412E0
user32.dll	00000000	00000000	00000000	00042234	000413BC
ole32.dll	00000000	00000000	00000000	00042BA6	00041610
ThunkRVA	ThunkOffset	ThunkValue	Hint	ApiName	
00041158	0003E958	0004186A	0000	FreeLibrary	
0004115C	0003E95C	00041878	0000	FindFirstFileA	
00041160	0003E960	0004188A	0000	FindClose	
00041164	0003E964	00041896	0000	ExitProcess	
00041168	0003E968	000418A4	0000	WriteFile	
0004116C	0003E96C	000418B0	0000	UnhandledExceptionFilter	
00041170	0003E970	000418CC	0000	SetFilePointer	
Number Of Thunks: 24h / 42d (FirstThunk chain)					<input type="checkbox"/> 总是查看 FirstThunk

[ 输入表 ]					
DllName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	名称	FirstThunk
user32.dll	00000000	00000000	00000000	00042234	000413BC
ole32.dll	00000000	00000000	00000000	00042BA6	00041610
comctl32.dll	00000000	00000000	00000000	00042BBE	00041618
ThunkRVA	ThunkOffset	ThunkValue	Hint	ApiName	
00041474	0003EC74	00042548	0000	OffsetRect	
00041478	0003EC78	00042556	0000	OemToCharA	
0004147C	0003EC7C	00042564	0000	MessageBoxA	
00041480	0003EC80	00042572	0000	MapWindowPoints	
00041484	0003EC84	00042584	0000	MapVirtualKeyA	
00041488	0003EC88	00042596	0000	LoadStringA	
0004148C	0003EC8C	000425A4	0000	LoadKeyboardLayoutA	
Number Of Thunks: 94h / 148d (FirstThunk chain)					<input type="checkbox"/> 总是查看 FirstThunk

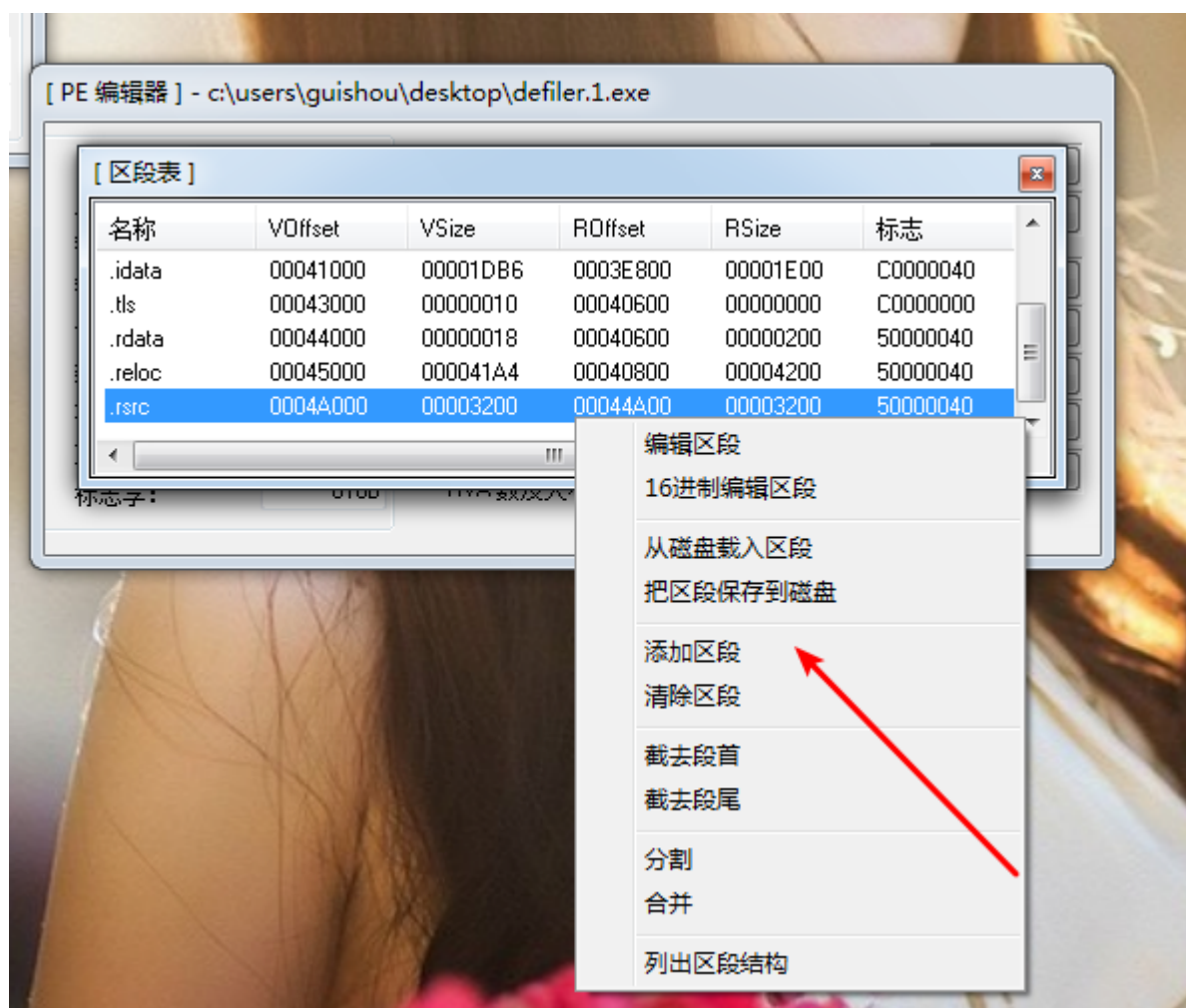
这个程序的导入表已经有这两个API，那么就不需要手工添加了

## 添加区段

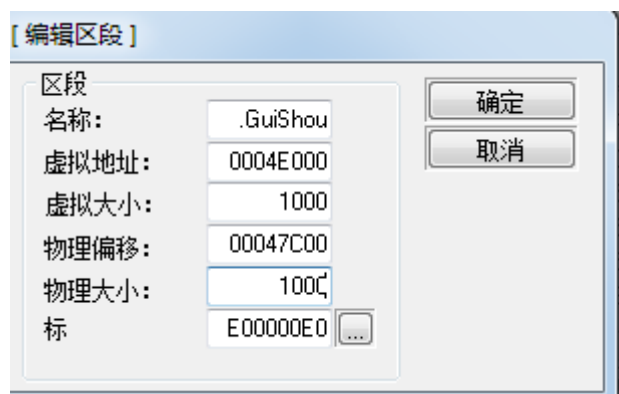
由于植入的代码量较多，所以需要添加一个区段。

## 配置区段

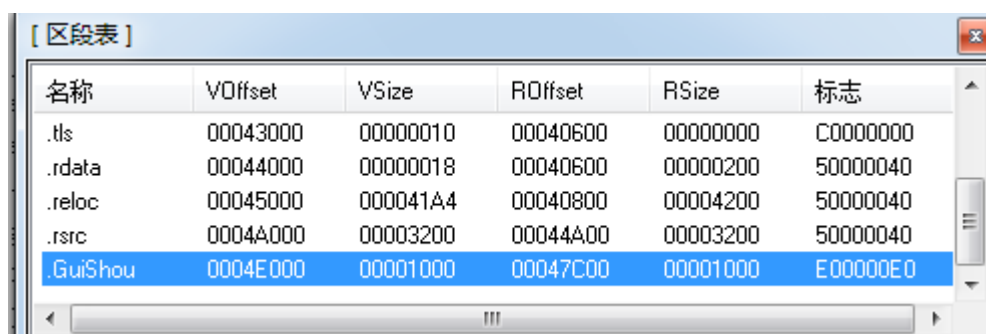
打开LoadPE，加载目标程序，点击右键->添加区段



然后编辑新添加的区段，输入名称和虚拟大小以及物理大小即可，点击保存



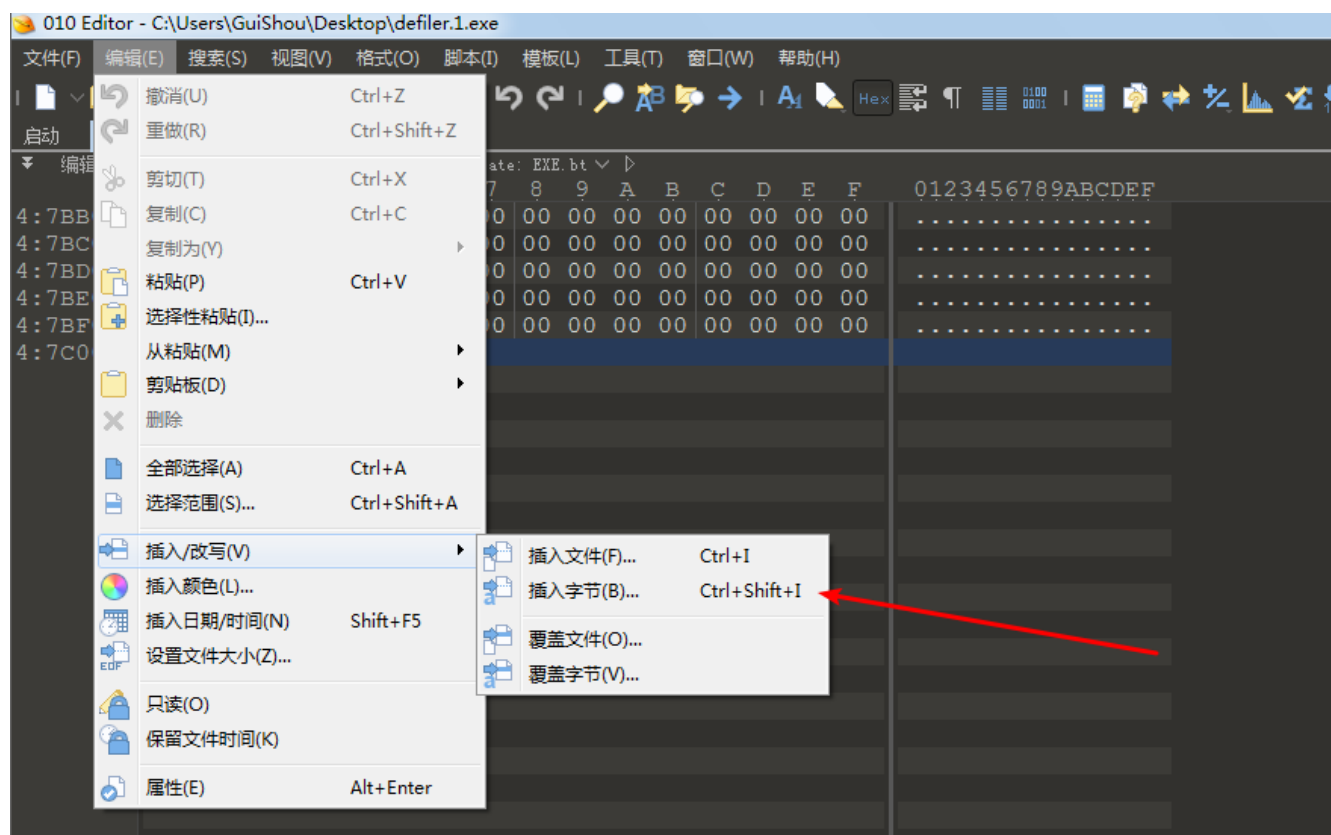
添加完成之后如图，记下新区段的RVA是4E000



## 添加区段数据

区段虽然添加好了,但真正重要的区段数据还需要插入到文件中,以扩充文件的大小. 因为区段只是一个相当于目录的存在,如果只有目录而没有内容,就会造成一个无效的PE文件.

用010Edit打开目标文件, 拉到文件末尾的位置, 点击编辑->插入->插入字节



输入起始地址和大小, 点击确定 然后保存即可

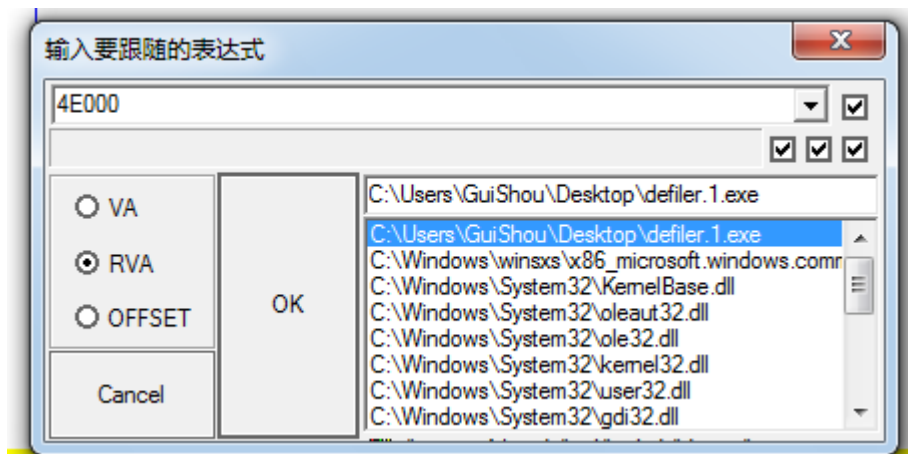


添加区段后如果文件能正常运行, 说明添加成功

## 植入代码



接着将程序载入到OD，直接用RVA来到新区段地址处，我这里是4E000，



然后添加如下代码

```
0044E000  60          pushad
0044E001  90          nop
0044E002  FF73 08     push  dword ptr [ebx+0x8]
0044E005  68 87E04400 push  dword ptr [0044E087] ; ASCII
"Exit1"
0044E00A  E8 28000000 call  0044E037
0044E00F  85C0        test  eax, eax
0044E011  74 1B       je    short 0044E02E
0044E013  FF73 24     push  dword ptr [ebx+0x24]
0044E016  68 8DE04400 push  dword ptr [0044E08D] ; ASCII
"&Exit"
0044E01B  E8 17000000 call  0044E037
0044E020  85C0        test  eax, eax
0044E022  74 0A       je    short 0044E02E
0044E024  C783 80000000 9>mov  dword ptr [ebx+0x80], 0044E093 ; 新函数的地址
0044E02E  90          nop
0044E02F  61          popad
0044E030  68 E4E04400 push  dword ptr [0044E0E4] ; UNICODE "苹
"
0044E035  C3          retn
0044E036  90          nop
0044E037  FF7424 04     push  dword ptr [esp+0x4] ; 字符串比较
0044E03B  E8 33000000 call  0044E073
0044E040  50          push  eax
0044E041  FF7424 08     push  dword ptr [esp+0x8]
0044E045  E8 29000000 call  0044E073
0044E04A  8BD0        mov  edx, eax
0044E04C  58          pop  eax
0044E04D  3BC2        cmp  eax, edx
0044E04F  75 1D       jnz  short 0044E06E
0044E051  33C9        xor  ecx, ecx
0044E053  8B7424 04     mov  esi, dword ptr [esp+0x4]
0044E057  8B7C24 08     mov  edi, dword ptr [esp+0x8]
0044E05B  8A1431      mov  dl, byte ptr [ecx+esi]
0044E05E  3A1439      cmp  dl, byte ptr [ecx+edi]
0044E061  75 0B       jnz  short 0044E06E
```

```

0044E063  41          inc    ecx
0044E064  3BC8       cmp    ecx, eax
0044E066  ^ 75 F3    jnz    short 0044E05B
0044E068  33C0       xor    eax, eax
0044E06A  40         inc    eax
0044E06B  C2 0800    retn   0x8
0044E06E  33C0       xor    eax, eax
0044E070  C2 0800    retn   0x8
0044E073  33C9       xor    ecx, ecx                ; 取字符串长度
0044E075  8B4424 04   mov    eax, dword ptr [esp+0x4]
0044E079  803C08 00   cmp    byte ptr [eax+ecx], 0x0
0044E07D  74 03      je     short 0044E082
0044E07F  41         inc    ecx
0044E080  ^ EB F7    jmp    short 0044E079
0044E082  8BC1       mov    eax, ecx
0044E084  C2 0400    retn   0x4
0044E087  45         inc    ebp
0044E088  78 69     js     short 0044E0F3
0044E08A  74 31     je     short 0044E0BD
0044E08C  0026     add    byte ptr [esi], ah
0044E08E  45         inc    ebp
0044E08F  78 69     js     short 0044E0FA
0044E091  74 00     je     short 0044E093
0044E093  6A 24     push   0x24                ; MB_YESNO |
MB_ICONQUESTION
0044E095  68 DCE04400 push   0044E0DC            ; ASCII
"_KaQqi"
0044E09A  68 B6E04400 push   0044E0B6            ; ASCII "Do
you fickbirne really want to quit?"
0044E09F  6A 00     push   0x0
0044E0A1  FF15 A4114400 call    dword ptr [&user32.MessageBoxA] ;
user32.MessageBoxA
0044E0A7  83F8 06   cmp    eax, 0x6
0044E0AA  75 09     jnz    short 0044E0B5
0044E0AC  6A 00     push   0x0
0044E0AE  2E:FF15 6411440>call    dword ptr cs:[&kernel32.ExitProcess] ;
kernel32.ExitProcess
0044E0B5  C3        retn
0044E0B6  44         inc    esp                ; 提示信息 and 标题
0044E0B7  6f        outs   dx, dword ptr [esi]
0044E0B8  2079 6f   and    byte ptr [ecx+0x6F], bh
0044E0BB  75 20     jnz    short 0044E0DD
0044E0BD  66:6963 6B 6269 imul   sp, word ptr [ebx+0x6B], 0x6962
0044E0C3  72 6E     jb     short 0044E133
0044E0C5  65:2072 65   and    byte ptr gs:[edx+0x65], dh
0044E0C9  61        popad
0044E0CA  6c        ins    byte ptr [edi], dx
0044E0CB  6c        ins    byte ptr [edi], dx
0044E0CC  79 20     jns    short 0044E0EE
0044E0CE  77 61     ja     short 0044E131
0044E0D0  6e        outs   dx, byte ptr [esi]
0044E0D1  74 20     je     short 0044E0F3
0044E0D3  74 6f     je     short 0044E144

```

```

0044E0D5    2071 75      and      byte ptr [ecx+0x75], dh
0044E0D8    69743F 00 5F4B6>imul esi, dword ptr [edi+edi], 0x51614B5F
0044E0E0    71 69      jno      short 0044E14B
0044E0E2    0000      add      byte ptr [eax], al
0044E0E4    66:83BB 8200000>cmp      word ptr [ebx+0x82], 0x0          ; 转移
0044E0EC    - E9 2D13FEFF jmp      0042F41E
0044E0F1    90        nop
0044E0F2    0000      add      byte ptr [eax], al

```

十六进制代码如图

```

60 90 FF 73 08 68 87 E0 44 00 E8 28 00 00 00 85 C0 74 1B FF 73 24 68 8D E0 44 00 E8 17 00
00 00
85 C0 74 0A C7 83 80 00 00 00 93 E0 44 00 90 61 68 E4 E0 44 00 C3 90 FF 74 24 04 E8 33 00
00 00
50 FF 74 24 08 E8 29 00 00 00 8B D0 58 3B C2 75 1D 33 C9 8B 74 24 04 8B 7C 24 08 8A 14 31
3A 14
39 75 0B 41 3B C8 75 F3 33 C0 40 C2 08 00 33 C0 C2 08 00 33 C9 8B 44 24 04 80 3C 08 00 74
03 41
EB F7 8B C1 C2 04 00 45 78 69 74 31 00 26 45 78 69 74 00 6A 24 68 DC E0 44 00 68 B6 E0 44
00 6A
00 FF 15 A4 11 44 00 83 F8 06 75 09 6A 00 2E FF 15 64 11 44 00 C3 44 6F 20 79 6F 75 20 66
69 63
6B 62 69 72 6E 65 20 72 65 61 6C 6C 79 20 77 61 6E 74 20 74 6F 20 71 75 69 74 3F 00 5F 4B
61 51
71 69 00 00 66 83 BB 82 00 00 00 00 E9 2D 13 FE FF 90 00 00

```

直接复制，右键->二进制->二进制粘贴即可

地址	HEX 数据	反汇编	注释
0044E0B6	44	inc esp	
0044E0B7	6f	outs dx,dword ptr ds:[esi]	
0044E0B8	2079 6F	and byte ptr ds:[ecx+0x6F],bh	
0044E0BB	75 20	jnz short defiler_.0044E0DD	
0044E0BD	66:6963 6B 6268	imul sp,word ptr ds:[ebx+0x6B],0x6962	
0044E0C3	72 6E	jb short defiler_.0044E133	
0044E0C5	65:2072 65	and byte ptr gs:[edx+0x65],dh	
0044E0C9	61	popad	
0044E0CA	6c	ins byte ptr es:[edi],dx	
0044E0CB	6c	ins byte ptr es:[edi],dx	
0044E0CC	79 20	jns short defiler_.0044E0EE	
0044E0CE	77 61	ja short defiler_.0044E131	
0044E0D0	6e	outs dx,byte ptr ds:[esi]	
0044E0D1	74 20	je short defiler_.0044E0F3	
0044E0D3	74 6F	je short defiler_.0044E144	
0044E0D5	2071 75	and byte ptr ds:[ecx+0x75],dh	
0044E0D8	69743F 00 5F4B6	imul esi,dword ptr ds:[edi+edi],0x51614B5F	
0044E0E0	71 69	jno short defiler_.0044E14B	
0044E0E2	0000	add byte ptr ds:[eax],al	
0044E0E4	66:83BB 8200000	cmp word ptr ds:[ebx+0x82],0x0	
0044E0EC	- E9 2D13FEFF	jmp defiler_.0042F41E	

修改完成之后保存

## 修改目标函数

接下来我们要修改关键函数，让目标程序点击菜单时，跳转到自己的新添加的区段，直接来到0x0042F216处，

修改为如下代码：

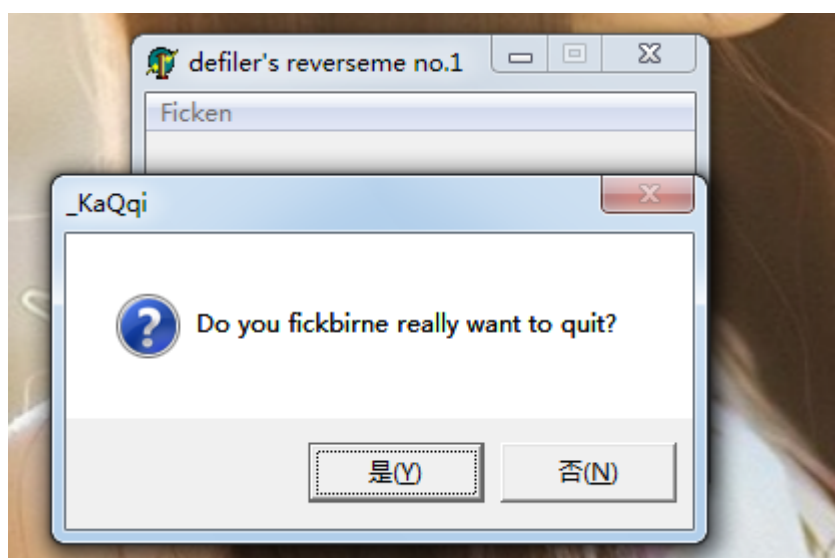
```
0042F416    68 00E04400    push    0044E000
0042F41B    C3            retn
0042F41C    90            nop
0042F41D    90            nop
```

对应的十六进制为

```
68 00 E0 44 00 C3 90 90
```

## 校验结果

修改完成之后来检验一下结果，点击Exit菜单，弹出对话框



点击是，退出程序。Patch完成！

最后，需要相关文件可以到我的Github下载：<https://github.com/TonyChen56/160-Crackme>