

## 第二十七章-Visual Basic 程序的破解-Part2

### Visual Basic 程序破解续

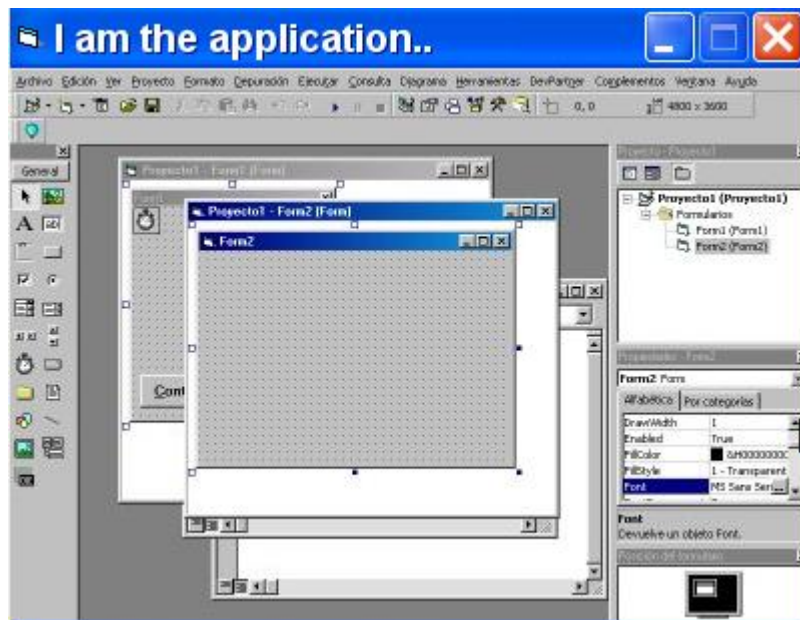
本章我们继续讨论 VB 应用程序破解的话题,上一章我们解决的是一个带 NAG 窗口的 CrackMe。我们是用那个 patch 过的 OD 来弄的,本章我们也会用到,但是本章我们还会介绍另一种更加简便,更加省时的方法。首先我们还是用那个 patch 过的 OD 来分析,弄清楚了具体原理以后,我们再来介绍那种简便的方法。

本章我们实验的对象名称叫做 killme,这个程序启动的时候会弹出一个烦人的 NAG 窗口,我们需要想办法把它剔除掉。

我们直接运行起来的话,可以看到弹出一个 NAG 窗口。



我们可以看到 Continue 按钮是灰色的(不可用),当定时器由 5 减少到 0 后,Continue 按钮就被激活了,我们单击 Continue 按钮。



NAG 关闭了,并且弹出主程序窗口,我们用 OD 加载该程序。

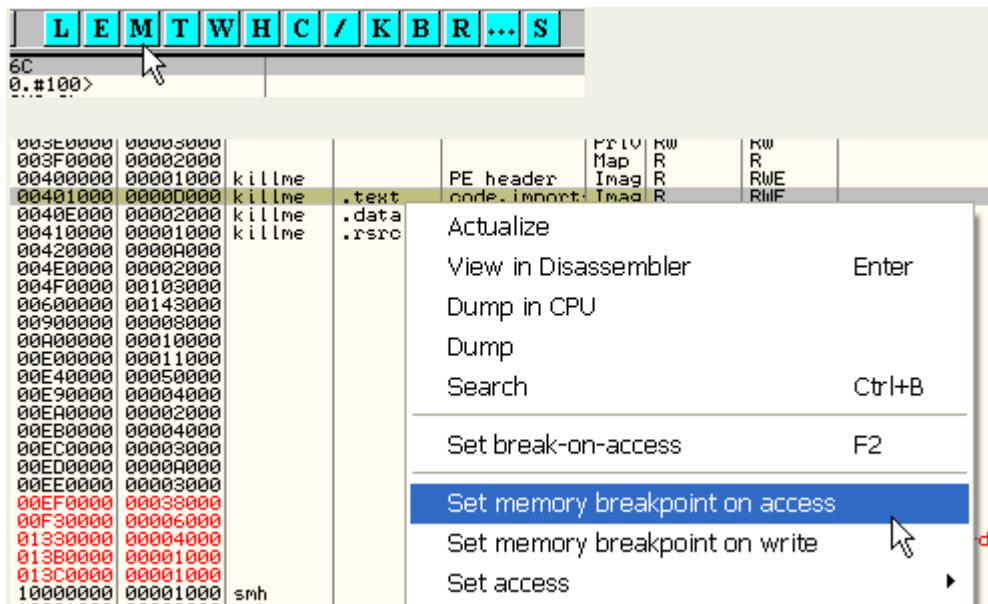
0040120C	68 6C434000	PUSH killme.0040436C
00401211	E8 F0FFFFFF	CALL <JMP.&MSUBUM160.#100>
00401216	0000	ADD BYTE PTR DS:[EAX],AL
00401218	0000	ADD BYTE PTR DS:[EAX],AL
0040121A	0000	ADD BYTE PTR DS:[EAX],AL
0040121C	3000	XOR BYTE PTR DS:[EAX],AL
0040121E	0000	ADD BYTE PTR DS:[EAX],AL
00401220	40	INC EAX
00401221	0000	ADD BYTE PTR DS:[EAX],AL
00401223	0000	ADD BYTE PTR DS:[EAX],AL

我们按 F9 键运行起来。



定时器时间到了后,Continue 按钮就被激活了。

我们可以想象一下,NAG 窗口关闭后,代码中应该会有一个无条件跳转到主窗口代码执行,所以我们给代码段设置内存访问断点 (该内存访问断点实际上只是内存执行断点),接着单击 Continue 按钮。



单击 Continue 按钮后,我们断在了这里。

00404B58	816C24 04 3F00	SUB DWORD PTR SS:[ESP+4],3F
00404B60	✓ E9 2B850000	JMP killme3.0040D090
00404B65	816C24 04 4300	SUB DWORD PTR SS:[ESP+4],43
00404B6D	✓ E9 6E860000	JMP killme3.0040D1E0
00404B72	816C24 04 3700	SUB DWORD PTR SS:[ESP+4],37
00404B7A	✓ E9 D1860000	JMP killme3.0040D250
00404B7F	816C24 04 3B00	SUB DWORD PTR SS:[ESP+4],3B
00404B87	✓ E9 64890000	JMP killme3.0040D4F0
00404B8C	0000	ADD BYTE PTR DS:[EAX],AL
00404B8E	0000	ADD BYTE PTR DS:[EAX],AL
00404B90	40	INC EAX
00404B91	✓ E0 40	LOOPDNE SHORT killme3.00404BD3
00404B93	0028	ADD BYTE PTR DS:[EAX],CH

我们可以看到 404B60 处 JMP 指令会跳转到 40D090 地址处开始执行,也就是说主窗口程序实际上是从 40D090 处开始执行的,这里说主窗口程序可能不太准确,因为我们可以看到这里有好几处 JMP 分支会跳转到程序的不同部分执行。我们给这几处 JMP 指令分别设置断点,然后来看看每个分支跳转分别在什么情况下被触发。

00404B58	816C24 04 3F00	SUB DWORD PTR SS:[ESP+4],3F
00404B60	✓ E9 2B850000	JMP killme3.0040D090
00404B65	816C24 04 4300	SUB DWORD PTR SS:[ESP+4],43
00404B6D	✓ E9 6E860000	JMP killme3.0040D1E0
00404B72	816C24 04 3700	SUB DWORD PTR SS:[ESP+4],37
00404B7A	✓ E9 D1860000	JMP killme3.0040D250
00404B7F	816C24 04 3B00	SUB DWORD PTR SS:[ESP+4],3B
00404B87	✓ E9 64890000	JMP killme3.0040D4F0
00404B8C	0000	ADD BYTE PTR DS:[EAX],AL
00404B8E	0000	ADD BYTE PTR DS:[EAX],AL
00404B90	40	INC EAX

我们重新启动该程序。

00404B7A	✓ E9 D1860000	JMP killme3.0040D250
00404B7F	816C24 04 3B00	SUB DWORD PTR SS:[ESP+4],3B
00404B87	✓ E9 64890000	JMP killme3.0040D4F0
00404B8C	0000	ADD BYTE PTR DS:[EAX],AL
00404B8E	0000	ADD BYTE PTR DS:[EAX],AL
00404B90	40	INC EAX
00404B91	✓ E0 40	LOOPDNE SHORT killme3.00404BD3
00404B93	0028	ADD BYTE PTR DS:[EAX],CH
00404B95	0040 00	DD killme3.00404BD3

刚刚我们看到了单击 NAG 窗口上的 Continue 按钮后,会执行 JMP 40D090 这个分支。现在我们重新启动程序后,断在了 JMP 40D250 这个分支处。NAG 窗口还没有显示。我们继续运行起来。

00404B7A	✓ E9 D1860000	JMP killme3.0040D250
00404B7F	816C24 04 3B00	SUB DWORD PTR SS:[ESP+4],3B
00404B87	✓ E9 64890000	JMP killme3.0040D4F0
00404B8C	0000	ADD BYTE PTR DS:[EAX],AL
00404B8E	0000	ADD BYTE PTR DS:[EAX],AL
00404B90	40	INC EAX
00404B91	✓ E0 40	LOOPDNE SHORT killme3.00404BD3

我们可以看到断在了 JMP 40D4F0 这个分支处。这个时候 NVG 窗口已经显示出来了,上面显示了数字 5。

TestDbg - killme.exe - [CPU - main thread, module killme]
File View Debug Plugins Options Window Help
00404B7A ✓ E9 D1860000 JMP killme.0040D250
00404B7F 816C24 04 3B00 SUB DWORD PTR SS:[ESP+4],3B
00404B87 ✓ E9 64890000 JMP killme.0040D4F0
00404B8C 00 DB 00
00404B8D 00 DB 00
00404B8E 00 DB 00
00404B8F 00 DB 00
00404B90 DD killme.0040E040
00404B94 28D04000 DD killme.0040D028
00404B98 DB FF
00404B99 FF DB FF
00404B9A FF DB FF
00404B9B FF DB FF
00404B9C 00 DB 00

Registers (FPU)
EAX 00404B7F killme.00404B7F
ECX 00000000
EDX 733A7530 MSVBUM60.733A7530
EBX 00000001
ESP 0013FB7C
EBP 0013FB84
ESI 0013FC54
EDI 0013FB84
EIP 00404B87 killme.00404B87
C 0 ES 0023 32bit 0<FFFFFFFF>
P 0 CS 001B 32bit 0<FFFFFFFF>
A 0 SS 0023 32bit 0<FFFFFFFF>

TNT CRACK TEAM
CRACKME!
5
Continue Exit

我们继续运行。

00404B7A	E9 D1860000	JMP killme3.0040D250
00404B7F	816C24 04 3B00	SUB DWORD PTR SS:[ESP+4],3B
00404B87	E9 64890000	JMP killme3.0040D4F0
00404B8C	0000	ADD BYTE PTR DS:[EAX],AL
00404B8E	0000	ADD BYTE PTR DS:[EAX],AL
00404B90	40	INC EAX
00404B91	E0 40	LOOPNE SHORT killme3.00404BD3
00404B93	0028	ADD BYTE PTR DS:[EAX],CH

又断在了 JMP 40D4F0 这个分支处。这个时候 NVG 窗口上的数字变为了 4。我们会发现 JMP 40D4F0 这个分支一共会断下来 5 次。也就是说,这个分支过程与定时器相关。

让我们来跟进到这个分支中一探究竟。

0040D4F0	55	PUSH EBP
0040D4F1	8BEC	MOV EBP,ESP
0040D4F3	83EC 0C	SUB ESP,0C
0040D4F6	68 16114000	PUSH <JMP.&MSUBUM60.__vbaExceptionHandler>
0040D4FB	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
0040D501	50	PUSH EAX
0040D502	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
0040D509	83EC 2C	SUB ESP,2C
0040D50C	53	PUSH EBX
0040D50D	56	PUSH ESI
0040D50E	57	PUSH EDI
0040D50F	8965 F4	MOV DWORD PTR SS:[EBP-C],ESP
0040D512	C745 F8 E81040	MOV DWORD PTR SS:[EBP-8],killme3.0040D101
0040D519	8B75 08	MOV ESI,DWORD PTR SS:[EBP+8]
0040D51C	8BC6	MOV EAX,ESI
0040D51E	83E0 01	AND EAX,1
0040D521	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
0040D524	83E6 FE	AND ESI,FFFFFFFE
0040D527	56	PUSH ESI
0040D528	8975 08	MOV DWORD PTR SS:[EBP+8],ESI
0040D52B	8B0E	MOV ECX,DWORD PTR DS:[ESI]
0040D52D	FF51 04	CALL DWORD PTR DS:[ECX+4]
0040D530	8B16	MOV EDX,DWORD PTR DS:[ESI]
0040D532	33C0	XOR EAX,EAX
0040D534	56	PUSH ESI
0040D535	8945 E8	MOV DWORD PTR SS:[EBP-18],EAX
0040D538	8945 E4	MOV DWORD PTR SS:[EBP-1C],EAX
0040D53B	8945 E0	MOV DWORD PTR SS:[EBP-20],EAX
0040D53E	8945 DC	MOV DWORD PTR SS:[EBP-24],EAX
0040D541	FF92 08030000	CALL DWORD PTR DS:[EDX+308]
0040D547	8B1D 20104000	MOV EBX,DWORD PTR DS:[<&MSUBUM60.__vbaObjSet
0040D54D	50	PUSH EAX
0040D54E	8D45 DC	LEA EAX,DWORD PTR SS:[EBP-24]
0040D551	50	PUSH EAX
0040D552	FFD3	CALL EBX
0040D554	8B0E	MOV ECX,DWORD PTR DS:[ESI]
0040D556	56	PUSH ESI
0040D557	8945 D0	MOV DWORD PTR SS:[EBP-30],EAX
0040D55A	FF91 08030000	CALL DWORD PTR DS:[ECX+308]
0040D560	8D55 E0	LEA EDX,DWORD PTR SS:[EBP-20]
0040D563	50	PUSH EAX
0040D564	52	PUSH EDX
0040D565	FFD3	CALL EBX
0040D567	8BF8	MOV EDI,EAX
0040D569	8D4D E8	LEA ECX,DWORD PTR SS:[EBP-18]
0040D56C	51	PUSH ECX
0040D56D	57	PUSH EDI

没看出什么特别的地方,我们继续按 F8 键跟踪。

我们跟到了一处条件跳转 JE 40D64F 处。我们来分析一下看看这个条件跳转是不是决定定时器停止并激活 Continue 按钮的关键跳转。

0040D634	51	PUSH EAX
0040D635	FF15 68104000	CALL DWORD PTR DS:[<&MSUBUM60.__vbaR8Str
0040D63B	DC1D 08104000	FCDMP QWORD PTR DS:[4010D8]
0040D641	DFF0	FSTSW AX
0040D643	F6C4 40	TEST AH,40
0040D646	74 07	JE SHORT killme3.0040D64F
0040D648	B8 01000000	MOV EAX,1
0040D64D	EB 02	JMP SHORT killme3.0040D651
0040D64F	33C0	XOR EAX,EAX
0040D651	F7D8	NEG EAX

上面是一个浮点比较指令,由于上下文的线索并不太多,所以我们暂时还看不出它的作用是什么。我们给这个 JE 40D64F 设置一个断点,接着运行起来。

0040D641	DFF0	FSTSW AX
0040D643	F6C4 40	TEST AH,40
0040D646	74 07	JE SHORT killme3.0040D64F
0040D648	B8 01000000	MOV EAX,1
0040D64D	EB 02	JMP SHORT killme3.0040D651
0040D64F	33C0	XOR EAX,EAX

断在了 JE 40D64F 这个分支处,并且该跳转将成立,我们多运行几次会发现,计时器减至 0 之前这个跳转一直都是成立的,只有当定

时器显示为 0 时,该跳转才不成立。所以我们可以得知,该条件跳转是用来判断定时器是否为 0 的。我们将这一行 NOP 掉看看会发生什么。

3040D63B	DC1D 08104000	FCOMP QWORD PTR DS:[4010D8]	
3040D641	DFF0	FSTSW AX	
3040D643	F6C4 40	TEST AH,40	
3040D646	90	NOP	
3040D647	90	NOP	
3040D648	B8 01000000	MOV EAX,1	
3040D64D	EB 02	JMP SHORT killme3.0040D651	
3040D64F	33C0	XOR EAX,EAX	
3040D651	F7D8	NEG EAX	
3040D653	8D4D E8	LEA ECX,DWORD PTR SS:[EBP-18]	

好了,已经 NOP 掉了,这样这里就不会跳转到 40D64F 处了,我们运行起来。



我们可以看到,定时器并没有减至 0,而 Continue 按钮已经被激活了。所以我们对于该条件跳转是用来判断定时器是否为 0 的假设是成立的。好了,现在我们这个 NAG 窗口关闭,当当前这个函数返回到 VB 的 DLL 中时,那么该 NAG 窗口将继续显示,所以我们需要定位该函数何时返回到 VB 的模块中,定位到返回指令后,我们可以将返回指令指定返回到显示主窗口程序的分支 40D090 即可,这样 NAG 窗口就会主动关闭而不需要人为的单击 Continue 按钮了。

现在我们重新启动程序。

我们来到刚刚 NOP 掉的那一行。

0040D63B	DC1D 08104000	FCOMP QWORD PTR DS:[4010D8]	
0040D641	DFF0	FSTSW AX	
0040D643	F6C4 40	TEST AH,40	
0040D646	90	NOP	
0040D647	90	NOP	
0040D648	B8 01000000	MOV EAX,1	
0040D64D	EB 02	JMP SHORT killme3.0040D651	
0040D64E	33C0	XOR EAX,EAX	
0040D653	8D4D E8	LEA ECX,DWORD PTR SS:[EBP-18]	
0040D65B	FF15 68104000	CALL DWORD PTR DS:[68104000] ; 0040D65B: 75	
0040D63B	DC1D 08104000	FCOMP QWORD PTR DS:[4010D8]	
0040D641	DFF0	FSTSW AX	
0040D643	F6C4 40	TEST AH,40	
0040D646	90	NOP	
0040D647	90	NOP	
0040D648	B8 01000000	MOV EAX,1	
0040D64D	EB 02	JMP SHORT killme3.0040D651	

我们删掉之前设置的所有断点,接着给 TEST AH,40 这一行设置一个断点,运行起来。

0040D63B	DC1D 08104000	FCOMP QWORD PTR DS:[4010D8]	
0040D641	DFF0	FSTSW AX	
0040D643	F6C4 40	TEST AH,40	
0040D646	90	NOP	
0040D647	90	NOP	
0040D648	B8 01000000	MOV EAX,1	
0040D64D	EB 02	JMP SHORT killme3.0040D651	

断了下来,我们按 F8 键往下跟踪,看看哪里会返回到 VB 的 DLL 中,返回到了 VB 的 DLL 中的话程序就会运行起来。这样 NAG 窗口就会显示出来并等待我们按 Continue 或者 Exit 键。



0040D707	6A 02	PUSH 2	
0040D709	FF15 10104000	CALL DWORD PTR DS:[<&MSUBUM60.__vbaFree	MSUBUM60
0040D70F	83C4 18	ADD ESP,18	
0040D712	C3	RETN	
0040D713	C3	RETN	
0040D714	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
0040D717	50	PUSH EAX	
0040D718	8B08	MOV ECX,DWORD PTR DS:[EAX]	
0040D71A	FF51 08	CALL DWORD PTR DS:[ECX+8]	
0040D71D	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
0040D720	8B4D EC	MOV ECX,DWORD PTR SS:[EBP-14]	
0040D723	5F	POP EDI	
0040D724	5E	POP ESI	
0040D725	64:890D 000000	MOV DWORD PTR FS:[0],ECX	
0040D72C	5B	POP EBX	
Return to 0040D714 (killme3.0040D714)			
Address Hex dump ASCII			

我们到达了 40D713 处的 RETN 指令处,但是我们会发现该处会返回到紧接着的下一行 40D714 处,并不是返回到 VB 的 DLL 中,所以我们继续跟踪。

0040D72C	5B	POP EBX	
0040D72D	8BE5	MOV ESP,EBP	
0040D72F	5D	POP EBP	
0040D730	C2 0400	RETN 4	
0040D733	E9 E439FFFF	JMP <JMP.&MSUBUM60.__vbaFPException>	
0040D738	90	NOP	
0040D739	90	NOP	
0040D73A	90	NOP	
0040D73B	90	NOP	
0040D73C	90	NOP	
0040D73D	90	NOP	
0040D73E	90	NOP	
0040D73F	90	NOP	
0040D740	55	PUSH EBP	
0040D741	8BE5	MOV EBP,ESP	
0040D743	83EC 0C	SUB ESP,0C	
0040D746	68 16114000	PUSH <JMP.&MSUBUM60.__vbaExceptionHandler>	
0040D748	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0040D751	50	PUSH EAX	
Return to 66051FB3 (MSUBUM60.66051FB3)			
Address Hex dump ASCII			
0040E000	00 00 00 00 00 00 00 00	.....	001

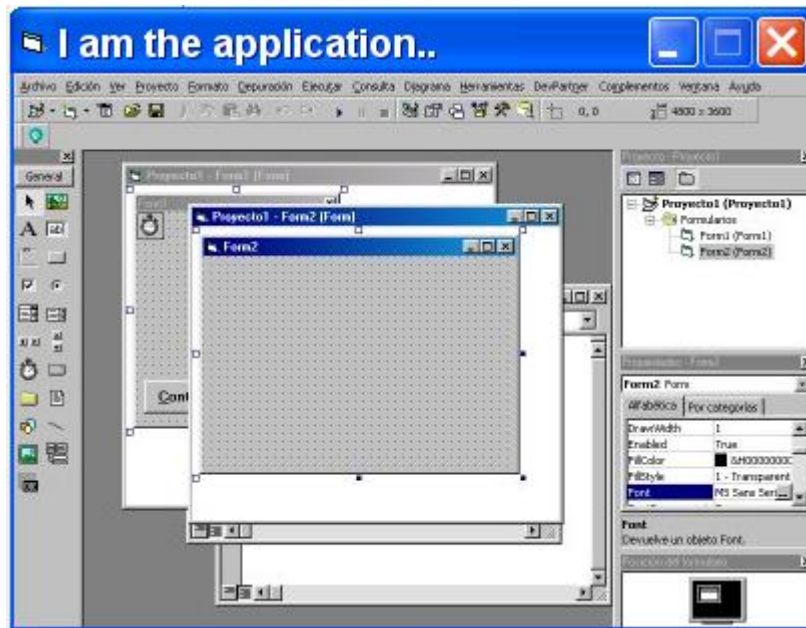
我们跟踪到了 40D730 处的 RETN 4 指令处,从解释窗口中我们可以看出来这里将返回到 MSVBVM60.DLL 中。接着程序将运行起来了。NAG 窗口将继续显示,为了剔除掉 NAG 窗口,这里我们可以尝试将该 RETN 4 修改为 JMP 指令,让其直接跳转到显示主窗口的代码处(40D090)。还有一点需要注意,这里 RETN 4 占 3 个字节,但是 JMP 40D090 占 5 个字节。我们还可以注意到 40D738 起始处有几个字节的 NOP 空间可以利用。所以我们可以先跳转到 40D739 处。

0040D72C	5B	POP EBX	
0040D72D	8BE5	MOV ESP,EBP	
0040D72F	5D	POP EBP	
0040D730	EB 07	JMP SHORT killme3.0040D739	
0040D732	90	NOP	
0040D733	E9 E439FFFF	JMP <JMP.&MSUBUM60.__vbaFPException>	
0040D738	90	NOP	
0040D739	90	NOP	
0040D73A	90	NOP	
0040D73B	90	NOP	
0040D73C	90	NOP	
0040D73D	90	NOP	
0040D73E	90	NOP	
0040D73F	90	NOP	
0040D740	55	PUSH EBP	

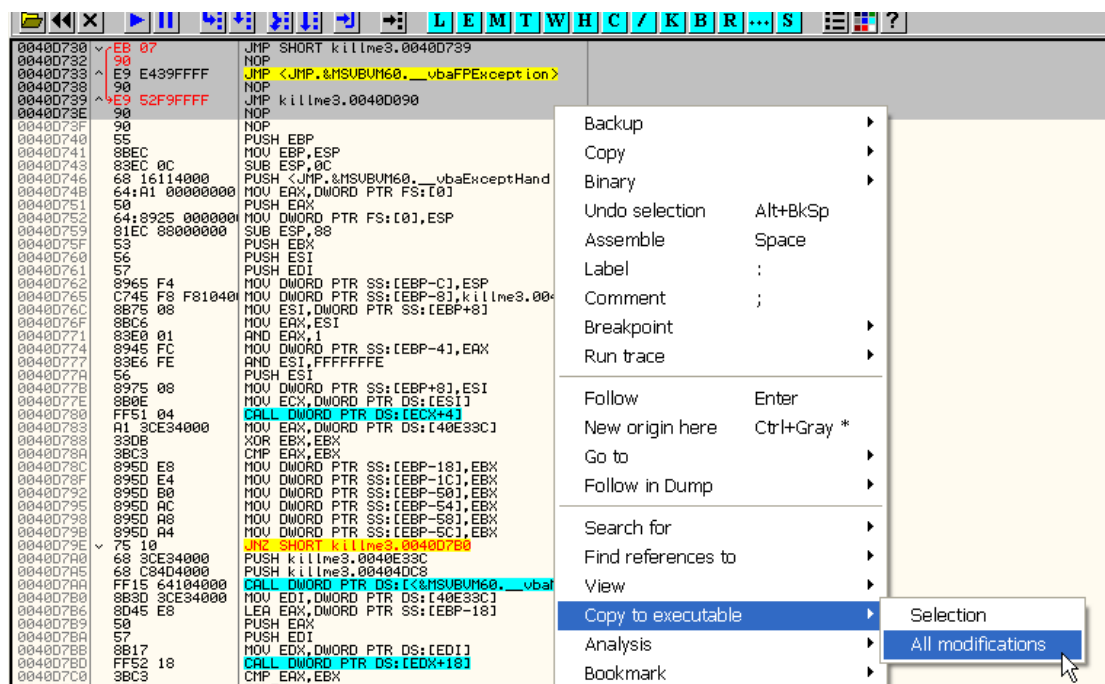
接着我们在 40D739 写入 JMP 40D090 指令即可。这样就可以直接跳转到显示主程序的代码开始执行,NVG 也被关闭了。

File View Debug Plugins Options Window Help			
[Icons]			
0040D730	EB 07	JMP SHORT killme3.0040D739	
0040D732	90	NOP	
0040D733	E9 E439FFFF	JMP <JMP.&MSUBUM60.__vbaFPException>	
0040D738	90	NOP	
0040D739	E9 52F9FFFF	JMP killme3.0040D090	
0040D73A	90	NOP	
0040D73B	90	NOP	
0040D73F	55	PUSH EBP	
0040D740	8BEC	MOV EBP,ESP	

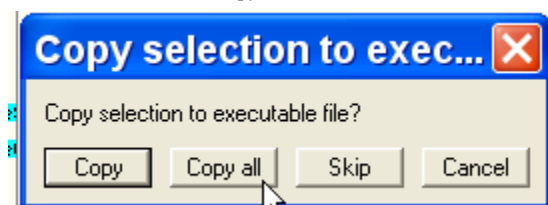
我们直接运行起来看看效果。



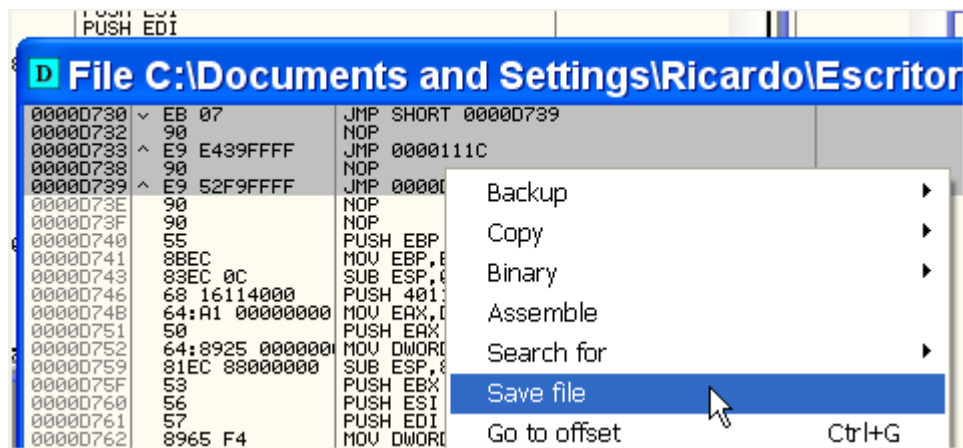
我们可以看到主窗口直接显示出来,NAG 不见了。好,现在我们将之前做的修改(NOP 掉的,两处跳转)保存到文件。



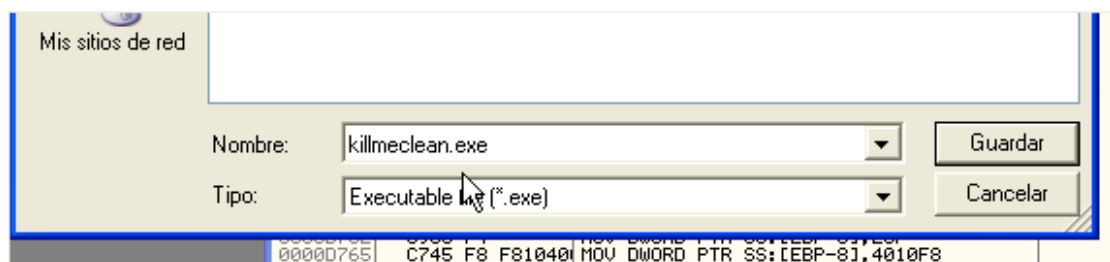
我们单击鼠标右键选择-Copy to executable-All modifications,这样就可以保存所有修改了。



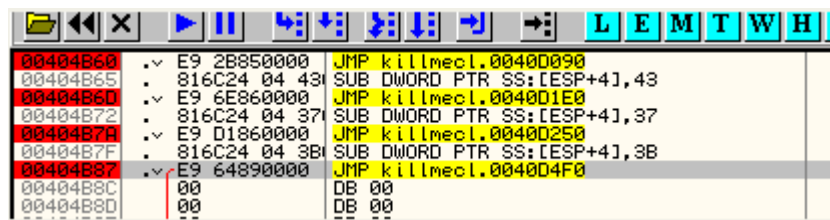
选择 Copy all。



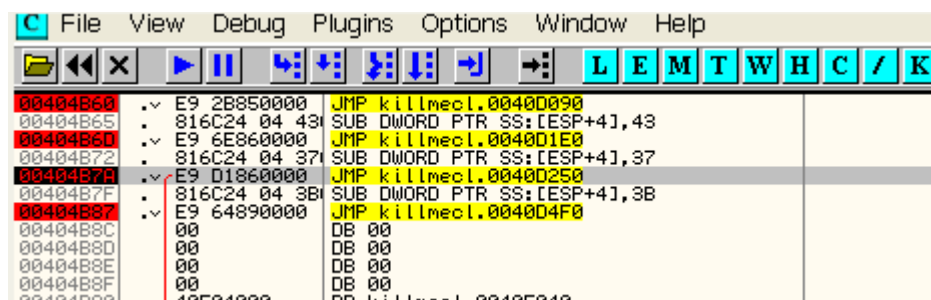
单击鼠标右键选择 Save file。



好了,保存名为 killmecl.exe,直接双击运行起来,我们会发现 NAG 窗口还是出现了,不过几秒钟后就消失了,然后主窗口就弹出来了。就是说 NAG 窗口还没有被完全剔除掉,所以我们还是来到之前分析的多分支处。



运行起来。

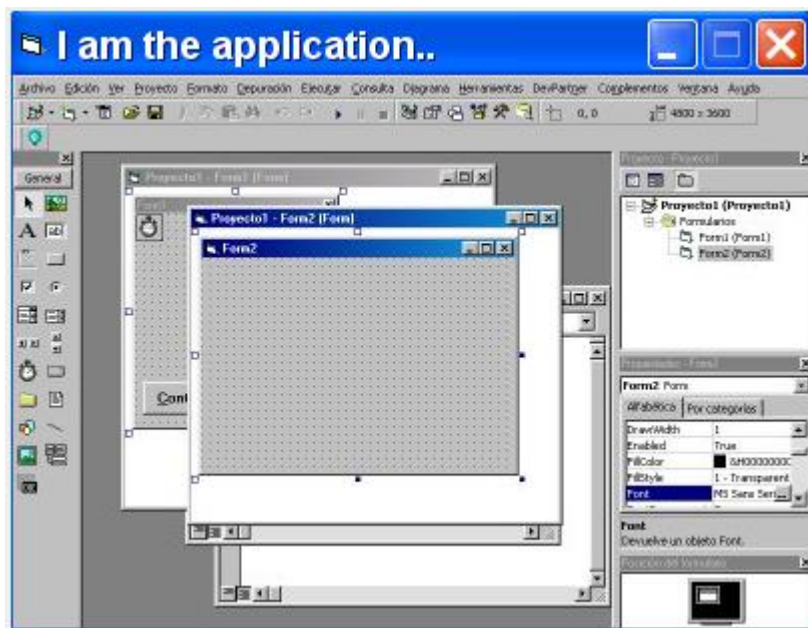


我们可以看到在 NAG 窗口创建之前断在了 404B7A 处,我们运行起来以后,几秒钟后,就断在了 404B87 处,并且这个时候 NAG 窗口被创建并显示出来了。所以 JMP 40D250 这个分支应该是创建并显示 NAG 窗口的分支。我们用 OD 加载刚刚我们 patch 过的 killmecl.exe 这个程序,我们将创建 NAG 窗口的跳转 NOP 掉。

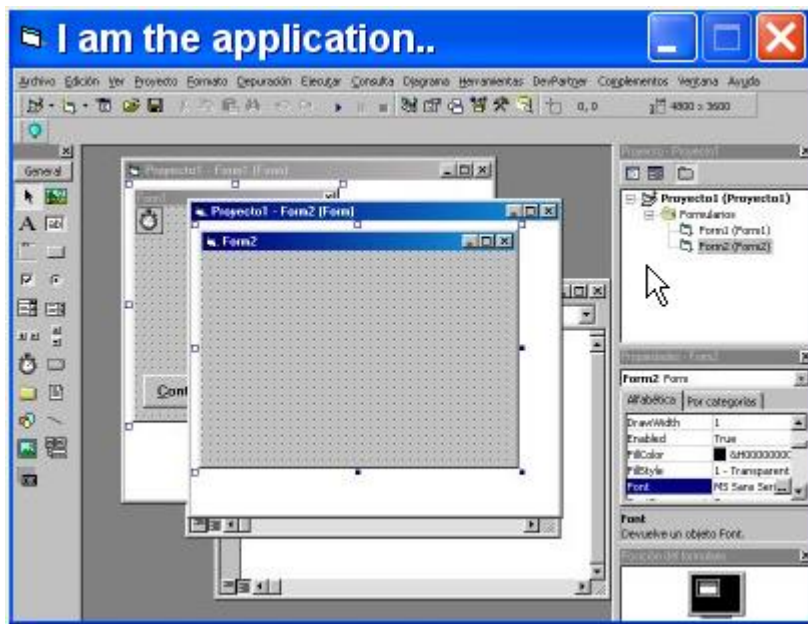


00404B70	816C24 04 37	SUB	DWORD PTR SS:[ESP+4], 37
00404B71	90	NOP	
00404B72	90	NOP	
00404B73	90	NOP	
00404B74	90	NOP	
00404B75	90	NOP	
00404B76	90	NOP	
00404B77	90	NOP	
00404B78	90	NOP	
00404B79	90	NOP	
00404B7A	90	NOP	
00404B7B	90	NOP	
00404B7C	90	NOP	
00404B7D	90	NOP	
00404B7E	90	NOP	
00404B7F	90	NOP	
00404B80	90	NOP	
00404B81	90	NOP	
00404B82	90	NOP	
00404B83	90	NOP	
00404B84	90	NOP	
00404B85	90	NOP	
00404B86	90	NOP	
00404B87	E9 64890000	JMP	killmecl.0040D4F0
00404B88	00	DB	00
00404B89	00	DB	00
00404B8A	00	DB	00

NOP 掉创建 NAG 窗口的分支后将直接转入定时器计时的分支执行,我们按 F9 键运行起来看看效果。



我们可以看到直接弹出了主窗口,并没有显示 NAG 窗口,我们保存所有修改到文件,然后直接双击运行看看效果。



我们可以看到直接就弹出了主窗口,并没有出现 NAG 窗口,嘿嘿。这样我们将手工剔除了这个 NAG 窗口。

4C 法-这样方式更加快捷方便,该法是基于 VB 程序的特性来的。

下面我们就来介绍这种方法,我们用 OD 加载 killme 程序,停在了入口点处。

0040120C	\$	68 6C434000	PUSH killmecl.0040436C
00401211	.	E8 F0FFFFFF	CALL <JMP.&MSUBUM60.#100>
00401216	.	0000	ADD BYTE PTR DS:[EAX],AL
00401218	.	0000	ADD BYTE PTR DS:[EAX],AL
0040121A	.	0000	ADD BYTE PTR DS:[EAX],AL
0040121C	.	3000	XOR BYTE PTR DS:[EAX],AL
0040121F	.	0000	ADD BYTE PTR DS:[EAX],AL

我们会注意到 VB 程序有个特点-入口点处都是一个 PUSH 指令,然后一个 CALL 指令。(如果你遇到的不是这种情况的话,那么该程序可能被加过壳) PUSH 将要压入堆栈的是 40436C,现在我们在数据窗口中定位到这个地址。

Address	Hex dump	ASCII
0040436C	56 42 35 21 F0 1F 56 42	VB5!-!VB
00404374	36 45 53 2E 44 4C 4C 00	6ES.DLL.
0040437C	00 00 00 00 2A 00 00 00	....*....
00404384	00 00 00 00 00 00 00 00	.....
0040438C	00 00 0A 00 0A 0C 00 00	.....
00404394	09 04 00 00 00 00 00 00	.....
0040439C	FC 45 40 00 12 F8 30 00	FE@.00.
004043A4	00 FF FF FF 00 00 00 00	....
004043AC	01 00 00 00 02 00 00 00	0...0...
004043B4	E9 00 00 00 0C 44 40 00	U...D@.
004043BC	64 43 40 00 18 12 40 00	dC@.00.
004043C4	78 00 00 00 7F 00 00 00	x...0...
004043CC	95 00 00 00 96 00 00 00	0...0...
004043D4	00 00 00 00 00 00 00 00	.....
004043DC	00 00 00 00 00 00 00 00	.....

正如你说看到的,这就是 VB 程序的头部,这里我们将 40436C 加上 4C。

也就是

40436C + 4C

0040E050	00 00 00 00 00 00 00 00	.....
0040E060	00 00 00 00 00 00 00 00	.....
0040E070	00 00 00 00 00 00 00 00	.....
Command	40436c + 4c	HEX: 4043B8 -
Program entry point		

也就是 4043B8。

Address	Hex dump	ASCII
004043B8	0C 44 40 00 64 43 40 00	.0@.dC@.
004043C0	18 12 40 00 78 00 00 00	00.00.00.00
004043C8	7F 00 00 00 95 00 00 00	0...0...
004043D0	96 00 00 00 00 00 00 00	0...0...
004043D8	00 00 00 00 00 00 00 00	.....
004043E0	00 00 00 00 68 69 6C 6C	...kill
004043E8	6D 65 00 4B 69 6C 6C 4D	me.KillM
004043F0	45 20 62 79 20 44 65 6D	E by Dem
004043F8	69 61 6E 2F 54 4E 54 21	ian/TNT!
00404400	00 00 50 72 6F 79 65 63	..Proyec
00404408	74 6F 31 00 50 00 00 00	tol.P...
00404410	C3 68 59 BB 17 DE D4 11	hVh0i0
00404418	A6 C4 C7 10 4B BB C9 3B	0-0K0F;
00404420	00 00 00 00 00 00 00 00	.....
00404428	00 00 00 00 00 00 00 00	.....

这里我们可以看到 4043B8 指向的内存单元中保存的是 40440C。(PS:这里作者写成了 4044C0,我已经更正为 40440C)

我们在数据窗口中定位到 40440C 这个地址。

Address	Hex dump	ASCII
0040440C	50 00 00 00 C3 68 59 BB	P...hVh
00404414	17 DE D4 11 A6 C4 C7 10	0i00-0-0
0040441C	4B BB C9 3B 00 00 00 00	K0F;...
00404424	00 00 00 00 00 00 00 00	.....
0040442C	00 00 00 00 00 00 00 00	.....
00404434	10 02 00 00 00 00 00 00	00.....
0040443C	00 00 00 00 00 00 00 00	.....
00404444	00 00 00 00 00 00 00 00	.....
0040444C	05 31 00 00 00 00 00 00	01.....
00404454	5C 12 40 00 4C 00 00 00	00.L...
0040445C	50 00 00 00 D5 68 59 BB	P...hVh
00404464	17 DE D4 11 A6 C4 C7 10	0i00-0-0
0040446C	4B BB C9 3B 00 00 00 00	K0F;...
00404474	00 00 00 00 00 00 00 00	.....
0040447C	00 00 00 00 01 00 00 00	...0...
00404484	00 00 00 00 00 00 00 00	.....
0040448C	00 00 00 00 00 00 00 00	.....
00404494	00 00 00 00 00 00 00 00	.....
Command	40436c + 4c	HEX: 4043B8 -

这里我们可以看到两块类似的数据,每块 50(十六进制)个字节的长度,每块数据的第 24(十六进制)个字节处都有一个标志。该标志指定了每块代码出现的顺序。我们一起来看看吧。

40440c +24=404430

Address	Hex dump	ASCII
0040440C	50 00 00 00 C3 68 59 BB	P...hYh
00404414	17 DE D4 11 A6 C4 C7 10	ti←-A
0040441C	4B BB C9 3B 00 00 00 00	Klf;...
00404424	00 00 00 00 00 00 00 00	.....
0040442C	00 00 00 00 00 00 00 00	.....
00404434	10 02 00 00 00 00 00 00	>0.....
0040443C	00 00 00 00 00 00 00 00	.....
00404444	00 00 00 00 00 00 00 00	.....
0040444C	05 31 00 00 00 00 00 00	#1.....
00404454	5C 12 40 00 4C 00 00 00	\@.L...
0040445C	50 00 00 00 D5 68 59 BB	P...hYh
00404464	17 DE D4 11 A6 C4 C7 10	ti←-A
0040446C	4B BB C9 3B 00 00 00 00	Klf;...
00404474	00 00 00 00 00 00 00 00	.....
0040447C	00 00 00 00 01 00 00 00	.....0.....
00404484	00 00 00 00 00 00 00 00	.....
0040448C	00 00 00 00 00 00 00 00	.....
00404494	00 00 00 00 00 00 00 00	.....
0040449C	45 80 00 00 00 00 00 00	E0.....
004044A4	44 4F 40 00 9C 00 00 00	D00.f...
004044AC	01 00 01 00 8C 4B 40 00	0.0.iK0.
004044B4	00 00 00 00 E8 CF 40 00	....p00.
004044BC	FF FF FF FF 00 00 00 00	.....
004044C4	10 4C 40 00 1C E0 40 00	L0.L00.
004044CC	00 00 00 00 C0 CB 60 00	....T'.
004044D4	00 00 00 00 00 00 00 00	.....
004044DC	00 00 00 00 24 4F 40 00	....f...

我们可以看到第一块的中标志是 00,表示该部分代码将首先执行,而第二块中的 01 表示随后才会执行,所以这里我们将各两个标志的值颠倒一下。

Address	Hex dump	ASCII
0040440C	50 00 00 00 C3 68 59 BB	P...hYh
00404414	17 DE D4 11 A6 C4 C7 10	ti←-A
0040441C	4B BB C9 3B 00 00 00 00	Klf;...
00404424	00 00 00 00 00 00 00 00	.....
0040442C	00 00 00 00 01 00 00 00	.....0.....
00404434	10 02 00 00 00 00 00 00	>0.....
0040443C	00 00 00 00 00 00 00 00	.....
00404444	00 00 00 00 00 00 00 00	.....
0040444C	05 31 00 00 00 00 00 00	#1.....
00404454	5C 12 40 00 4C 00 00 00	\@.L...
0040445C	50 00 00 00 D5 68 59 BB	P...hYh
00404464	17 DE D4 11 A6 C4 C7 10	ti←-A
0040446C	4B BB C9 3B 00 00 00 00	Klf;...
00404474	00 00 00 00 00 00 00 00	.....
0040447C	00 00 00 00 00 00 00 00	.....
00404484	00 00 00 00 00 00 00 00	.....
0040448C	00 00 00 00 00 00 00 00	.....
00404494	00 00 00 00 00 00 00 00	.....
0040449C	45 80 00 00 00 00 00 00	E0.....
004044A4	44 4F 40 00 9C 00 00 00	D00.f...
004044AC	01 00 01 00 8C 4B 40 00	0.0.iK0.
004044B4	00 00 00 00 E8 CF 40 00	....p00.
004044BC	FF FF FF FF 00 00 00 00	.....
004044C4	10 4C 40 00 1C E0 40 00	L0.L00.
004044CC	00 00 00 00 C0 CB 60 00	....T'.
004044D4	00 00 00 00 00 00 00 00	.....
004044DC	00 00 00 00 24 4F 40 00	....f...

这样首先弹出的就是主窗口了,嘿嘿。然后才是 NAG 窗口,其实 NAG 窗口根本不会弹出来,因为主窗口关闭后,应用程序就退出了,NAG 窗口压根就没有机会弹出来。

Address	Hex dump	ASCII
0040440C	50 00 00 00 C3 68 59 BB	P...hYh
00404414	17 DE D4 11 A6 C4 C7 10	ti←-A
0040441C	4B BB C9 3B 00 00 00 00	Klf;...
00404424	00 00 00 00 00 00 00 00	.....
0040442C	00 00 00 00 01 00 00 00	.....0.....
00404434	10 02 00 00 00 00 00 00	>0.....
0040443C	00 00 00 00 00 00 00 00	.....
00404444	00 00 00 00 00 00 00 00	.....
0040444C	05 31 00 00 00 00 00 00	#1.....
00404454	5C 12 40 00 4C 00 00 00	\@.L...
0040445C	50 00 00 00 D5 68 59 BB	P...hYh
00404464	17 DE D4 11 A6 C4 C7 10	ti←-A
0040446C	4B BB C9 3B 00 00 00 00	Klf;...
00404474	00 00 00 00 00 00 00 00	.....
0040447C	00 00 00 00 00 00 00 00	.....
00404484	00 00 00 00 00 00 00 00	.....
0040448C	00 00 00 00 00 00 00 00	.....
00404494	00 00 00 00 00 00 00 00	.....
0040449C	45 80 00 00 00 00 00 00	E0.....
004044A4	44 4F 40 00 9C 00 00 00	D00.f...
004044AC	01 00 01 00 8C 4B 40 00	0.0.iK0.
004044B4	00 00 00 00 E8 CF 40 00	....p00.

这样我们就用 4C 法方便快捷的搞定了这个 NAG 程序,但是这样方式并不适用于所有的程序,所以为什么一开始我要给大家介绍分析 VB 程序的常规手法,了解原理对我们非常有帮助。

这里留两个 CrackMe 给大家练习,一易一难,名称分别为 CrackMe(这个程序的运行需要 MSVBVM50.DLL 的支持),CrackMe2。大

家尝试一下看看能不能找出序列号以及剔除 NAG 窗口,下一章我们再来讲解这两个例子。