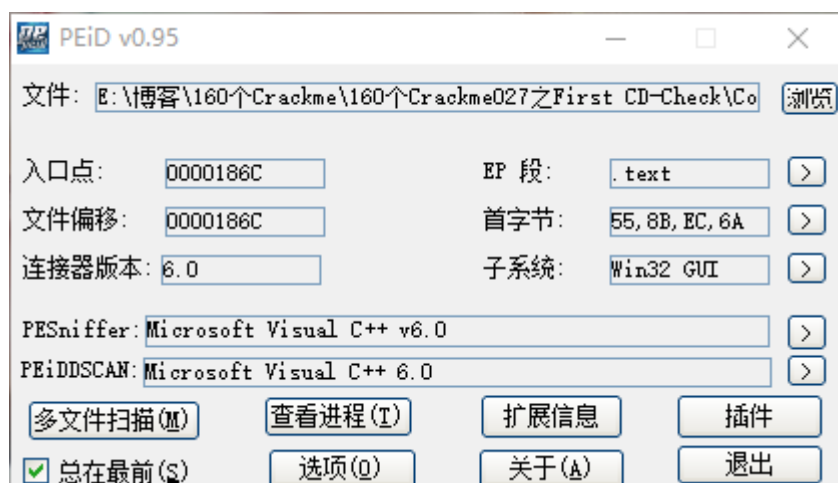


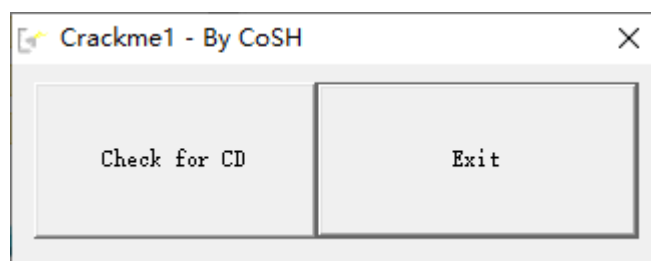
[查壳](#)
[分析程序](#)
[前置知识](#)
 [什么是光盘检测](#)
 [检测原理](#)
 [拆解光盘保护](#)
[CD-Check分析](#)
[作者低级的错误](#)
[暴力破解CD-Check](#)
[校验结果](#)

查壳

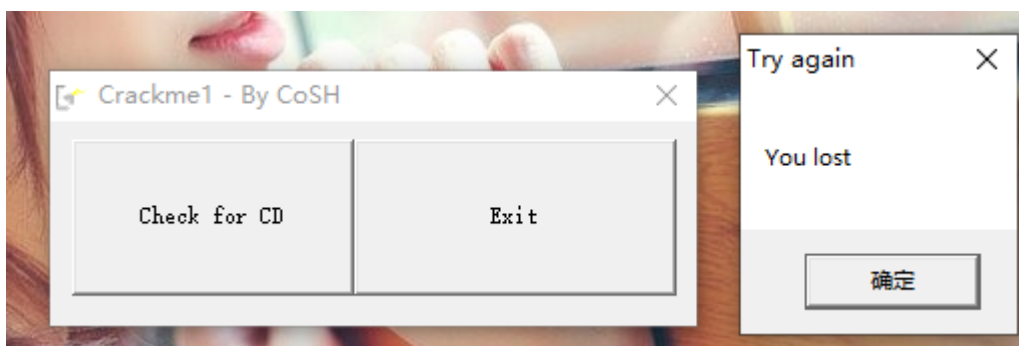


程序是使用VC6写的 没有壳

分析程序



这个程序跟之前的Crackme不一样，采取的保护方式是光盘检测，如果检测通过，点击Check for CD，会提示检测通过，但是现在是失败的



我也是第一次分析这种程序，所以需要一点相关的前置知识

前置知识

以下内容出自《加密与解密4》

什么是光盘检测

一些采用光盘形式发行的应用软件和游戏，在使用时需要检查光盘是否插在光驱中，如果没有则拒绝运行。这是为了防止用户将软件或游戏的一份正版拷贝安装在多台机器上且同时使用，其思路与DOS时代的钥匙盘保护类似，虽然能在一定程度上防止非法拷贝，但也给正版用户带来了一些麻烦——一旦光盘被划伤，用户就无法使用软件了

检测原理

最简单也最常见的光盘检测就是程序在启动时判断光驱中的光盘里是否存在特定的文件。如果不存在，则认为用户没有使用正版光盘，拒绝运行。在程序运行过程中，一般不再检查光盘是否在光驱中。在Windows下的具体实现一般是：先用GetLogicalDriveStrings0或GetLogicalDrives0函数得到系统中安装的所有驱动器的列表，然后用GetDriveType0函数检查每个驱动器，如果是光驱，则用CreateFile0或FindFirstFile0函数检查特定的文件是否存在，甚至可能进一步检查文件的属性、大小、内容等。

上述保护的一种增强类型就是把程序运行时需要的关键数据放在光盘中。这样，即使解密者能够强行跳过程序启动时的检查，但由于没有使用正版光盘，也就没有程序运行时所需要的关键数据，程序自然会崩溃，从而在一定程度上起到了防破解的作用

拆解光盘保护

第一种光盘检测方式是比较容易被破解的。解密者只要利用上述函数设置断点，找到程序启动时检查光驱的地方，然后修改判断指令，就可以跳过光盘检测

第二种增强型光盘保护还是有办法的，可以简单地利用刻录和复制工具将光盘复制多份，也可以采用虚拟光驱程序来模拟正版光盘。常用的虚拟光驱程序有Virtual CD、Virtual Drive、DaemonTools等。值得一提的是Daemon Tools，它不仅是免费的，而且能够模拟一些加密光盘。这些光盘加密工具一般都会在光轨上做文章，例如做暗记等。有的加密光盘可用工作在原始模式（Raw mode）的光

盘拷贝程序原样复制，例如Padus公司的DiscJuggler和Elaborate Bytes公司的CloneCD等。对光盘加密感兴趣的读者可以查阅ISO9660标准协议

CD-Check分析

00401219	. 57	push edi	
0040121A	. 68 9C304000	push Cosh_1.0040309C	C:\
0040121F	. 8D4D A4	lea ecx,dword ptr ss:[ebp-0x5C]	
00401222	. E8 79040000	call <jmp.&MFC42.#CString::CString_537>	
00401227	. 33DB	xor ebx,ebx	
00401229	. 68 98304000	push Cosh_1.00403098	D:\
0040122E	. 8D4D A8	lea ecx,dword ptr ss:[ebp-0x58]	
00401231	. 895D FC	mov dword ptr ss:[ebp-0x4],ebx	
00401234	. E8 67040000	call <jmp.&MFC42.#CString::CString_537>	
00401239	. 68 94304000	push Cosh_1.00403094	E:\
0040123E	. 8D4D AC	lea ecx,dword ptr ss:[ebp-0x54]	
00401241	. C645 FC 01	mov byte ptr ss:[ebp-0x4],0x1	
00401245	. E8 56040000	call <jmp.&MFC42.#CString::CString_537>	
0040124A	. 68 90304000	push Cosh_1.00403090	F:\
0040124F	. 8D4D B0	lea ecx,dword ptr ss:[ebp-0x50]	
00401252	. C645 FC 02	mov byte ptr ss:[ebp-0x4],0x2	
00401256	. E8 45040000	call <jmp.&MFC42.#CString::CString_537>	
0040125B	. 68 8C304000	push Cosh_1.0040308C	G:\
00401260	. 8D4D B4	lea ecx,dword ptr ss:[ebp-0x4C]	
00401263	. C645 FC 03	mov byte ptr ss:[ebp-0x4],0x3	

地址	数值	注释
0019F4B0	00AA2478	ASCII "C:\"
0019F4B4	00AA24C8	ASCII "D:\"
0019F4B8	00AA2518	ASCII "E:\"
0019F4BC	00AA2568	ASCII "F:\"
0019F4C0	00AA25B8	ASCII "G:\"
0019F4C4	00AA2608	ASCII "H:\"
0019F4C8	00AA2658	ASCII "I:\"
0019F4CC	00AA26A8	ASCII "J:\"
0019F4D0	00AA26F8	ASCII "K:\"
0019F4D4	00AA2748	ASCII "L:\"
0019F4D8	00AA2798	ASCII "M:\"
0019F4DC	00AA27E8	ASCII "N:\"
0019F4E0	00AA2838	ASCII "O:\"
0019F4E4	00AA2888	ASCII "P:\"
0019F4E8	5653FB24	mfc42.5653FB24

首先程序初始化了C-P的字符串

00401337	. 895D EC	mov dword ptr ss:[ebp-0x14],ebx	
0040133A	. 8D7D A4	lea edi,dword ptr ss:[ebp-0x5C]	
0040133D	> 57	push edi	
0040133E	. 8D4D E8	lea ecx,dword ptr ss:[ebp-0x18]	
00401341	. E8 48030000	call <jmp.&MFC42.#CString::operator=_858>	
00401346	. FF75 E8	push dword ptr ss:[ebp-0x18]	
00401349	. FF15 04204000	call dword ptr ds:[<&KERNEL32.GetDriveTypeA>]	RootPathName = "C:\"
0040134F	. 83F8 03	cmp eax,0x3	检测是否为本地硬盘
00401352	. 74 3E	je short Cosh_1.00401392	如果为本地硬盘则继续检测下一个磁盘
00401354	. 8D45 E8	lea eax,dword ptr ss:[ebp-0x18]	
00401357	. 68 58304000	push Cosh_1.00403058	
0040135C	. 50	push eax	CD_CHECK.DAT
0040135D	. 8D45 E0	lea eax,dword ptr ss:[ebp-0x20]	
00401360	. 50	push eax	

然后检测这些盘符的类型是否为本地硬盘，如果是本地硬盘就继续检测下一个磁盘

地址	HEX 数据	反汇编	注释	寄存器 (FPU)	
00401368	. 53	push ebx	hTemplateFile = NULL	EAX 00A128D8 ASCII "D:\CD_CHECK.D	
00401369	. 53	push ebx		ECX 5648A8C0 mfc42.5648A8C0	
0040136A	. 53	push ebx		EDX 00000003	
0040136B	. 53	push ebx		EBX 00000000	
0040136C	. 6A 01	push 0x1		ESP 0019F488	
0040136E	. 68 00000080	push 0x80000000	Access = GENERIC_READ	EBP 0019F50C	
00401373	. 50	push eax	FileName = "D:\CD_CHECK.DAT"	ESI 0040169A <jmp.&MFC42.#CString	
00401374	. FF15 00204000	call dword ptr ds:[&KERNEL32.CreateFileA	CreateFileA	EDI 0019F4B4	
00401377	. 83F8 FF	cmp eax,-0x1	读取根目录下的CD_CHECK.DAT文件	EIP 00401374 Cosh_1.00401374	
0040137D	. 8D4D E0	lea ecx,dword ptr ss:[ebp-0x20]		C 0 ES 002B 32位 0(FFFFFFFF)	
00401380	. 0F9445 f3	sete byte ptr ss:[ebp-0xd]	当文件不存在时设置[ebp-0xd]	P 0 CS 0023 32位 0(FFFFFFFF)	
00401384	. E8 11030000	call <jmp.&MFC42.#CString::~CString_800		A 0 SS 002B 32位 0(FFFFFFFF)	
00401389	. 385D F3	cmp byte ptr ss:[ebp-0xd],bl		Z 0 DS 002B 32位 0(FFFFFFFF)	
0040138C	. 0F84 F3000000	je Cosh_1.00401485	关键跳转	S 0 FS 0053 32位 2EC000(FFF)	
00401392	. FF45 EC	inc dword ptr ss:[ebp-0x14]		T 0 GS 002B 32位 0(FFFFFFFF)	
00401395	. 83C7 04	add edi,0x4		D 0	
00401398	. 837D EC 07	cmp dword ptr ss:[ebp-0x14],0x7		0 0 LastErr ERROR_SUCCESS (00000000)	
ds:[00402000]-6C5D0140 (apphelp.6C5D0140)					
地址	HEX 数据	ASCII	地址	数值	注释
00402000	40 01 5D 6C	C0 3A 8E 76	0019F488	00A128D8	FileName = "D:\CD_CHECK.DAT"
00402010	00 00 00 00	F0 21 47 56	0019F48C	80000000	Access = GENERIC_READ
00402020	C0 AC 4B 56	70 FA 46 56	0019F490	00000001	ShareMode = FILE_SHARE_READ
00402030	50 A1 4B 56	00 AA 4B 56	0019F494	00000000	pSecurity = NULL
00402040	B0 A6 4B 56	A0 45 47 56	0019F498	00000000	Mode = 0x0
00402050	D0 5A 46 56	D0 5A 46 56	0019F49C	00000000	Attributes = 0
00402060	A0 5A 46 56	C0 59 46 56	0019F4A0	00000000	hTemplateFile = NULL

检测完磁盘类型之后会读取根目录下是否存在 CD_CHECK.DAT 文件，如果文件存在则提示正确

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00401485	. > 53	push ebx		EAX 00A128D8 ASCII "D:\CD_CHECK.D
00401486	. 68 34304000	push Cosh_1.00403034	You did it	ECX 004D0000
0040148B	. 68 20304000	push Cosh_1.00403020	Well done, Cracker	EDX 004D0000
00401490	. ^ E9 14FFFFFF	jmp Cosh_1.004013A9		EBX 00000000
00401495	. 6A 00	push 0x0	Enable = FALSE	ESP 0019F4A4
00401497	. FF71 20	push dword ptr ds:[ecx+0x20]	hWnd = 087D8BF0	EBP 0019F50C
0040149A	. FF15 D0214000	call dword ptr ds:[&USER32.EnableWindow	EnableWindow	ESI 0040169A <jmp.&MFC42.#CString::~CString_800>
004014A0	. C3	retm		EDI 0019F4B4

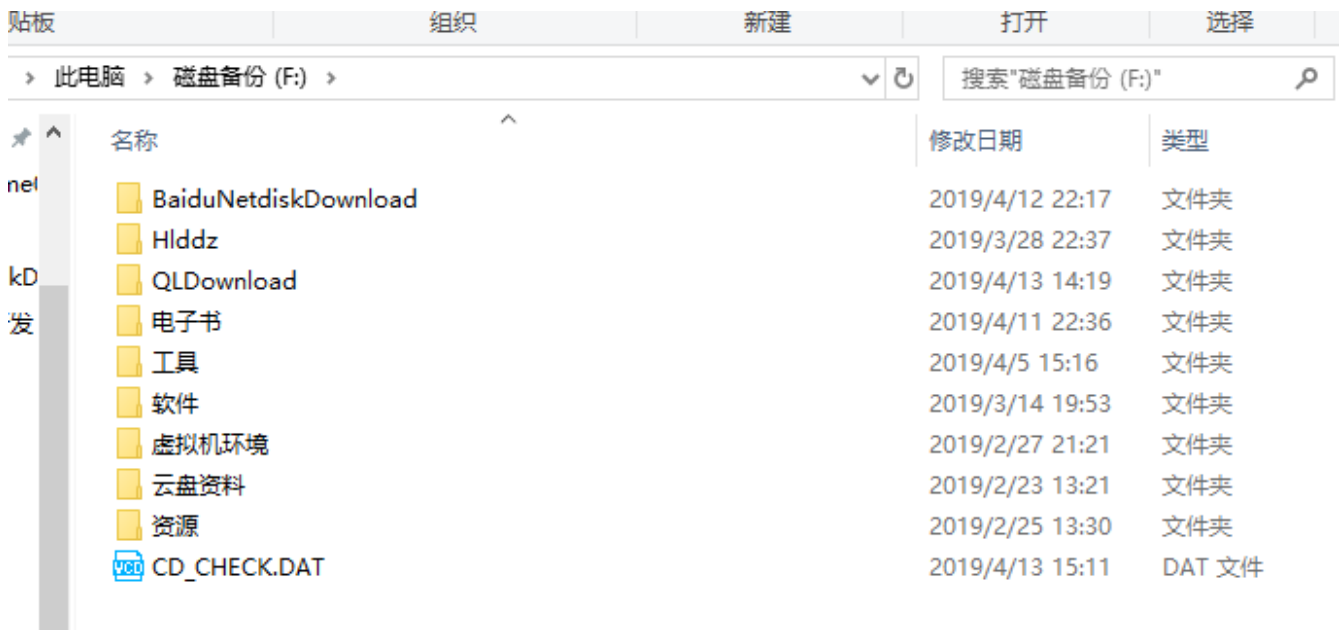
所以，正确的不修改程序的Crack方式应该是插一个U盘，然后再根目录下新建一个文件名为 CD_CHECK.DAT 的文件。

作者低级的错误

但是！当我在U盘下新建了文件之后居然发现还是通过不了检测，原因居然是这个CreateFileA的参数写错了！！！

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->									
暂停									
地址 HEX 数据 反汇编 注释 寄存器 (FPU)									
00401366	.	8B00	mov eax,dword ptr ds:[eax]		EAX FFFFFFFF				
00401368	.	53	push ebx	hTemplateFile	ECX 004D0000				
00401369	.	53	push ebx		EDX 004D0000				
0040136A	.	53	push ebx		EBX 00000000				
0040136B	.	53	push ebx		ESP 0019F4A4				
0040136C	.	6A 01	push 0x1		EBP 0019F50C				
0040136E	.	68 00000080	push 0x80000000	Access = GENER	ESI 0040169A <jmp.&MFC42.#CString::~CString_800>				
00401373	.	50	push eax	FileName = FFF	EDI 0019F4B4				
00401374	.	FF15 00204000	call dword ptr ds:[<&KERNEL32.CreateFileA	CreateFileA	EIP 0040137A Cosh_1.00401374				
00401377	.	83F8 FF	cmp eax,-0x1	读取根目录下的	C 0 ES 002B 32位 0(FFFFFFFF)				
0040137D	.	8D4D E0	lea ecx,dword ptr ss:[ebp-0x20]		P 0 CS 0023 32位 0(FFFFFFFF)				
00401380	.	0F9445 f3	sete byte ptr ss:[ebp-0xd]	当文件不存在时	A 0 SS 002B 32位 0(FFFFFFFF)				
00401384	.	E8 11030000	call <jmp.&MFC42.#CString::~CString_800		Z 0 DS 002B 32位 0(FFFFFFFF)				
00401389	.	385D F3	cmp byte ptr ss:[ebp-0xd],bl		S 0 FS 0053 32位 2EC000(FFF)				
0040138C	.	0F84 F3000000	je Cosh_1.00401485	关键跳转	T 0 GS 002B 32位 0(FFFFFFFF)				
00401392	.	FF45 EC	inc dword ptr ss:[ebp-0x14]		D 0				
00401395	.	83C7 04	add edi,0x4		0 0 LastErr ERROR_INVALID_PARAMETER (00000057)				
eax=FFFFFFFF									
地址 HEX 数据 ASCII 地址 数值 注释									

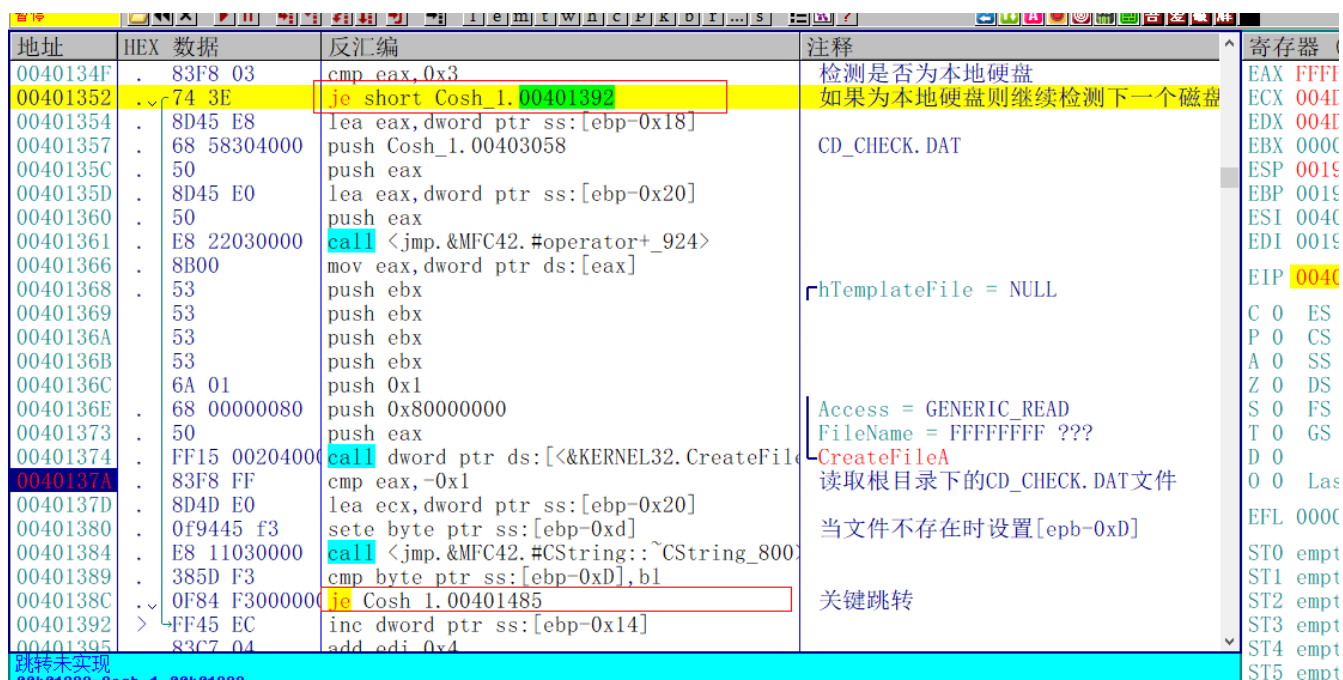
所以就算根目录下存在这样一个文件也通过不了检测，一开始我还以为自己看错了，于是又验证了一番



好吧 那就不能怪我了，是你自己写错的，只能暴力破解了

暴力破解CD-Check

修改如下两处指令，保存文件



修改后如下

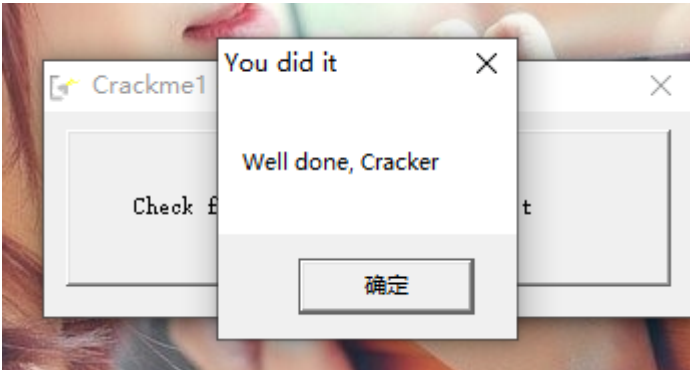
吾爱破解 - Cosh.1.exe - [LCG - 主线程 模块 - Cosh.1]

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->

暂停

地址	HEX 数据	反汇编	注释	寄存器
00401349	. FF15 04204000	call dword ptr ds:[<&KERNEL32.GetDriveTypeA	GetDriveTypeA	EAX FFFI
0040134F	. 83F8 03	cmp eax,0x3	检测是否为本地硬盘	ECX 0048
00401352	. 90	nop	如果为本地硬盘则继续检测下一个磁盘	EDX 0048
00401353	. 90	nop		EBX 0000
00401354	. 8D45 E8	lea eax,dword ptr ss:[ebp-0x18]		ESP 0018
00401357	. 68 58304000	push Cosh_1.00403058	CD_CHECK.DAT	EBP 0018
0040135C	. 50	push eax		ESI 0048
0040135D	. 8D45 E0	lea eax,dword ptr ss:[ebp-0x20]		EDI 0018
00401360	. 50	push eax		EIP 0048
00401361	. E8 22030000	call <jmp.&MFC42.#operator+_924>		C 0 ES
00401366	. 8B00	mov eax,dword ptr ds:[eax]	hTemplateFile = NULL	P 0 CS
00401368	. 53	push ebx		A 0 SS
00401369	. 53	push ebx		Z 0 DS
0040136A	. 53	push ebx		S 0 FS
0040136B	. 53	push ebx		T 0 GS
0040136C	. 6A 01	push 0x1		D 0
0040136E	. 68 00000080	push 0x80000000	Access = GENERIC_READ	
00401373	. 50	push eax	FileName = FFFFFFFF ???	
00401374	. FF15 00204000	call dword ptr ds:[<&KERNEL32.CreateFileA	CreateFileA	O 0 La:
0040137A	. 83F8 FF	cmp eax,-0x1	读取根目录下的CD_CHECK.DAT文件	EFL 0000
0040137D	. 8D4D E0	lea ecx,dword ptr ss:[ebp-0x20]		ST0 emp
00401380	. 0f9445 f3	sete byte ptr ss:[ebp-0xd]	当文件不存在时设置[ebp-0xD]	ST1 emp
00401384	. E8 11030000	call <jmp.&MFC42.#CString::~CString_800		ST2 emp
00401389	. 385D F3	cmp byte ptr ss:[ebp-0xD],bl		ST3 emp
0040138C	. E9 F4000000	jmp Cosh_1.00401485	关键跳转	ST4 emp
00401391	. 90	nop		ST5 emp
00401392	. FF45 EC	inc dword ptr ss:[ebp-0x14]		ST6 emp
				ST7 emp

校验结果



修改完成之后，点击Check，破解成功

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