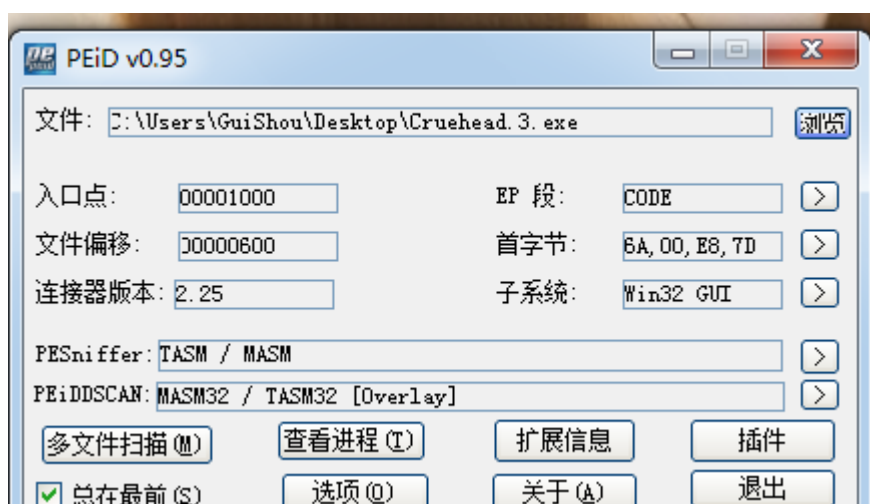


- 查壳
- 分析程序
- 用户名算法分析(前14位)
- 序列号算法分析(后4位)
- 算法总结
- 注册机探索
- 验证结果

这个Crackme和上一个是同一个作者，保护方式是KeyFile。难度两颗星

查壳



同样是汇编写的，作者伪造了一个Delphi的OEP。

分析程序

这个软件的KeyFile保护没有需要用监控工具了。验证直接就在入口处，直接单步往下跟就可以了

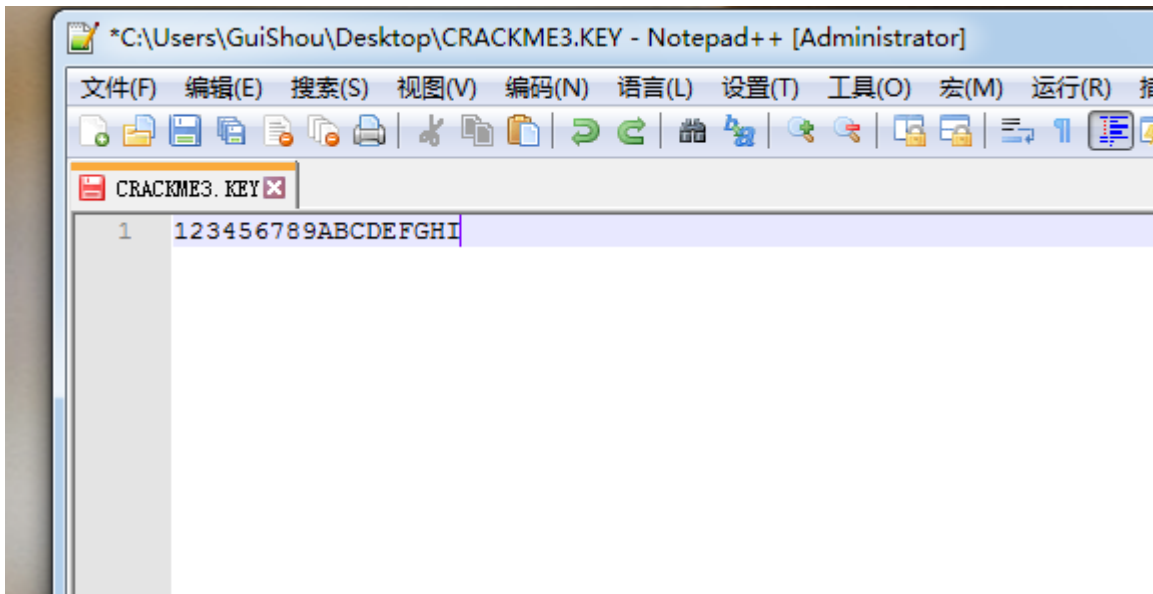
地址	HEX	数据	反汇编	注释	寄存器 (FPU)
00401000	6A 00		push 0x0	pModule = NULL	EAX 00400000 ASCII "MZP"
00401002	ES 7D040000		call < jmp. &KERNEL32.GetModuleHandleA>	GetModuleHandleA	ECX 00000000
00401007	A3 E9204000		mov dword ptr ds:[0x4020E9], eax	Cruehead.00400000	EDX 00401000 Cruehead.<ModuleEntryPoint>
0040100C	C705 F9204000		mov dword ptr ds:[0x4020F9], 0x0		EBX 7FFDF000
00401016	6A 00		push 0x0	hTemplateFile = NULL	ESP 0012FF70
00401018	68 80000000		push 0x80	Attributes = NORMAL	EBP 0012FF94
0040101D	6A 03		push 0x3	Mode = OPEN_EXISTING	ESI 00000000
0040101F	6A 00		push 0x0	pSecurity = NULL	EDI 00000000
00401021	6A 03		push 0x3	ShareMode = FILE_SHARE_READ FILE_SHARE_WRITE	EIP 0040102D Cruehead.0040102D
00401023	68 000000C0		push 0xC0000000	Access = GENERIC_READ GENERIC_WRITE	C 0 ES 0023 32位 0 (FFFFFFFF)
00401028	68 D7204000		push Cruehead.004020D7	FileName = "CRACKME3.KEY"	P 1 CS 001B 32位 0 (FFFFFFFF)
0040102D	ES 76040000		call < jmp. &KERNEL32.CreateFileA>	CreateFileA	A 0 SS 0023 32位 0 (FFFFFFFF)
00401032	83F8 FF		cmp eax, -0x1	检查CRACKME3.KEY文件是否存在	Z 1 DS 0023 32位 0 (FFFFFFFF)
00401035	75 0C		jnz short Cruehead.00401043		S 0 FS 003B 32位 7FFDE000 (FFF)
00401037	68 0E214000		push Cruehead.0040210E	ASCII "CrackMe v3.0"	T 0 GS 0000 NULL
0040103C	ES B4020000		call Cruehead.004012F5		D 0
00401041	EB 6B		jmp short Cruehead.004010AE		0 0 LastErr ERROR_SUCCESS (00000000)
00401043	A3 F5204000		mov dword ptr ds:[0x4020F5], eax	Cruehead.00400000	EFL 00000246 (NO, NB, E, BE, NS, PE, GE, LE)
00401048	B8 12000000		mov eax, 0x12		ST0 empty 0.0
0040104D	BB 08204000		mov ebx, Cruehead.00402008		ST1 empty 0.0
00401052	6A 00		push 0x0	pOverlapped = NULL	ST2 empty 0.0
00401058 < jmp. &KERNEL32.CreateFileA>					
地址	HEX	数据	反汇编	注释	寄存器 (FPU)
00402000	00 00 00 00	00 00 00 00			
00402010	00 00 00 00	00 00 00 00			
00402020	00 00 00 00	00 00 00 00			
00402030	00 00 00 00	00 00 00 00			
00402040	00 00 00 00	00 00 00 00			
00402050	00 00 00 00	00 00 00 00			
00402060	00 00 00 00	00 00 00 00			
00402070	00 00 00 00	00 00 00 00			
00402080	00 00 00 00	00 00 00 00			
00402090	00 00 00 00	00 00 00 00			
地址	HEX	数据	反汇编	注释	寄存器 (FPU)
0012FF70	004020D7			FileName = "CRACKME3.KEY"	
0012FF74	C0000000			Access = GENERIC_READ GENERIC_WRITE	
0012FF78	00000003			ShareMode = FILE_SHARE_READ FILE_SHARE_WRITE	
0012FF7C	00000000			pSecurity = NULL	
0012FF80	00000003			Mode = OPEN_EXISTING	
0012FF84	00000080			Attributes = NORMAL	
0012FF88	00000000			hTemplateFile = NULL	
0012FF8C	75E13C45			返回到 kernel32.75E13C45	
0012FF90	7FFDF000				
0012FF94	0012FF94				

首先检测CRACKME3.KEY这个文件是否存在

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00401028	68 D7204000	push Cruehead.004020D7	FileName = "CRACKME3.KEY"	EAX 00000012
0040102D	E8 76040000	call <jmp.&KERNEL32.CreateFileA>	CreateFileA	ECX 775A6570 ntdll.775A6570
00401032	83F8 FF	cmp eax,-0x1	检查CRACKME3.KEY文件是否存在	EDX 00600180
00401035	75 0C	jnz short Cruehead.00401043		EBX 00402008 Cruehead.00402008
00401037	68 0E214000	push Cruehead.0040210E	ASCII "CrackMe v3.0"	ESP 0012FF78 UNICODE "1"
0040103C	E8 B4020000	call Cruehead.004012F5		EBP 0012FF94
00401041	EB 6B	jmp short Cruehead.004010AE		ESI 00000000
00401043	A3 F5204000	mov dword ptr ds:[0x4020F5],eax	hFile = 0000006C (window)	EDI 00000000
00401048	B8 12000000	mov eax,0x12		EIP 00401061 Cruehead.00401061
0040104D	BB 08204000	mov ebx,Cruehead.00402008	pOverlapped = NULL	C 1 ES 0023 32位 0 (FFFFFFFF)
00401052	6A 00	push 0x0	pBytesRead = Cruehead.004021A0	P 0 CS 001B 32位 0 (FFFFFFFF)
00401054	68 A0214000	push Cruehead.004021A0	BytesToRead = 12 (18.)	A 1 SS 0023 32位 0 (FFFFFFFF)
00401059	50	push eax	Buffer = Cruehead.00402008	Z 0 DS 0023 32位 0 (FFFFFFFF)
0040105A	53	push ebx	hFile = 0000006C (window)	S 0 FS 003B 32位 7FFDE000 (FFF)
0040105B	FF35 F5204000	push dword ptr ds:[0x4020F5]	ReadFile	T 0 GS 0000 NULL
00401061	E8 30040000	call <jmp.&KERNEL32.ReadFile>	检查文件内容是否有0x12个字节	D 0
00401066	833D A0214000	cmp dword ptr ds:[0x4021A0],0x12		O 0 LastErr ERROR_SUCCESS (00000000)
0040106D	75 C8	jnz short Cruehead.00401037	[0x4020F9]=[0x4020F9]^0x12345678	EFL 00000213 (NO,B,NE,BE,NS,PO,GE,G)
0040106F	68 08204000	push Cruehead.00402008		ST0 empty 0.0
00401074	E8 98020000	call Cruehead.00401311		ST1 empty 0.0
00401079	8135 F9204000	xor dword ptr ds:[0x4020F9],0x12345678		ST2 empty 0.0
00401096<< jmp.&KERNEL32.ReadFile>				
地址	HEX 数据	反汇编	注释	寄存器 (FPU)
004020D7	43 52 41 43 4B 4D 45 33 2E 4B 45 59 00 00 00 00	CRACKME3.KEY....	hFile = 0000006C (window)	
004020E7	00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00	Buffer = Cruehead.00402008	
004020F7	00 00 00 00 00 00 54 72 79 20 74 6F 20 63 72 61Try to cra	BytesToRead = 12 (18.)	
00402107	63 6B 20 6D 65 21 00 43 72 61 63 6B 4D 65 20 76	ck me!.CrackMe v	pBytesRead = Cruehead.004021A0	
00402117	33 2E 30 20 20 20 20 20 20 20 20 20 20 20 20 20	3.0	pOverlapped = NULL	
00402127	00 4E 6F 20 6E 65 65 64 20 74 6F 20 64 69 73 61	.No need to disa	返回到 kernel32.75E13C45	
00402137	73 6D 20 74 68 65 20 63 6F 64 65 21 00 4D 45 4E	sm the code!.MEN		
00402147	55 00 00 00 00 00 44 4C 47 5F 41 42 4F 55 54 00	U.....DLG.ABOUT.		
00402157	47 6F 6F 64 20 77 6F 72 6B 20 63 72 61 63 6B 65	Good work cracke		
00402167	72 21 00 43 72 61 63 6B 65 64 20 62 79 3A 20 20	r!.Cracked by:		
00402177	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	N		
00402187	6F 77 20 74 72 79 20 74 68 65 20 6F 65 78 74 20	ow try the next		

然后从文件中读取0x12个字节，并检测文件内容是否少于0x12个字节，

那么我们就伪造一个CRACKME3.KEY文件，并且再里面随便输18个字母



0040104D	BB 08204000	mov ebx,Cruehead.00402008	ASCII "123456789ABCDEFGH1"	EBP 0012FF94
00401052	6A 00	push 0x0	pOverlapped = NULL	ESI 00000000
00401054	68 A0214000	push Cruehead.004021A0	pBytesRead = Cruehead.004021A0	EDI 00000000
00401059	50	push eax	BytesToRead = 0x1	EIP 00401074 Cruehead.00401074
0040105A	53	push ebx	Buffer = Cruehead.00402008	C 0 ES 0023 32位 0 (FFFFFFFF)
0040105B	FF35 F5204000	push dword ptr ds:[0x4020F5]	hFile = 0000006C (window)	P 1 CS 001B 32位 0 (FFFFFFFF)
00401061	E8 30040000	call <jmp.&KERNEL32.ReadFile>	ReadFile	A 0 SS 0023 32位 0 (FFFFFFFF)
00401066	833D A0214000	cmp dword ptr ds:[0x4021A0],0x12	检查文件内容是否有0x12个字节	Z 1 DS 0023 32位 0 (FFFFFFFF)
0040106D	75 C8	jnz short Cruehead.00401037	ASCII "123456789ABCDEFGH1"	S 0 FS 003B 32位 7FFDE000 (FFF)
0040106F	68 08204000	push Cruehead.00402008		T 0 GS 0000 NULL
00401074	E8 98020000	call Cruehead.00401311	[0x4020F9]=[0x4020F9]^0x12345678	D 0
00401079	8135 F9204000	xor dword ptr ds:[0x4020F9],0x12345678		O 0 LastErr ERROR_SUCCESS (00000000)
00401083	83C4 04	add esp,0x4	FileContent	EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
00401086	68 08204000	push Cruehead.00402008		ST0 empty 0.0
0040108B	E8 AC020000	call Cruehead.0040133C		ST1 empty 0.0
00401090	83C4 04	add esp,0x4		ST2 empty 0.0
00401311=Cruehead.00401311				
地址	HEX 数据	反汇编	注释	寄存器 (FPU)
004020D7	43 52 41 43 4B 4D 45 33 2E 4B 45 59 00 00 00 00	CRACKME3.KEY....	ASCII "123456789ABCDEFGH1"	
004020E7	00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00		
004020F7	00 00 00 00 00 00 54 72 79 20 74 6F 20 63 72 61Try to cra		

然后将文件内容放入堆栈，开始计算前14个字节。

这个文件总共需要18个字节，前14个字节为用户名，后4个字节为序列号。每一部分都有一个算法，下面分析用户名算法部分

用户名算法分析(前14位)

地址	HEX 数据	反汇编	注释	寄存器 (FPU)	
00401311	\$ 33C9	xor ecx,ecx		EAX 00000031	
00401313	33C0	xor eax,eax		ECX 00000000	
00401315	8B7424 04	mov esi,dword ptr ss:[ebp+0x4]	esi=username	EDX 775970B4 ntdll.KiFastSystemCallRet	
00401319	B3 41	mov bl,0x41	bl=0x41	EBX 00402041 Cruehead.00402041	
0040131B	8A06	mov al,byte ptr ds:[esi]	al=username[i]	ESP 0012FF84	
0040131D	32C3	xor al,bl	al=username[i]^0x41	EBP 0012FF94	
0040131F	8806	mov byte ptr ds:[esi],al	username[i]=0x41	ESI 00402008 ASCII "123456789ABCDEFGH"	
00401321	46	inc esi	username>>1	EDI 00000000	
00401322	FEC3	inc bl	bl++	EIP 0040131D Cruehead.0040131D	
00401324	0105 F9204000	add dword ptr ds:[0x4020F9],eax	[0x4020F9]=sum(username[i]^0x41)	C 0 ES 0023 32位 0(FFFFFFFF)	
0040132A	3C 00	cmp al,0x0	if(username[i]^0x41==0)	P 1 CS 001B 32位 0(FFFFFFFF)	
0040132C	74 07	js short Cruehead.00401335		A 0 SS 0023 32位 0(FFFFFFFF)	
0040132E	FEC1	inc cl	i++	Z 1 DS 0023 32位 0(FFFFFFFF)	
00401330	80FB 4F	cmp bl,0x4F	if(bl==0x4F)	S 0 FS 003B 32位 0(FFFFFFFF)	
00401333	75 E6	jnz short Cruehead.0040131B		T 0 GS 0000 NULL	
00401335	890D 49214000	mov dword ptr ds:[0x402149],ecx		D 0	
00401337	C3	ret		O 0 LastErr ERROR_SUCCESS (00000000)	
00401338	\$ 8B7424 04	mov esi,dword ptr ss:[ebp+0x4]	Cruehead.00402008	EPL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)	
00401340	83C6 0E	add esi,0xE	FileContent>>0x14	ST0 empty 0.0	
00401343	8B06	mov eax,dword ptr ds:[esi]	eax=FileContent[15-18]	ST1 empty 0.0	
00401345	C3	ret		ST2 empty 0.0	
追踪 SS:[00401315]=00402008 (Cruehead.00402008), ASCII "123456789ABCDEFGH"					
esi=00402008 (Cruehead.00402008), ASCII "123456789ABCDEFGH"					
地址	HEX 数据	ASCII	地址	数值	注释
00402008	31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 47	123456789ABCDEFGH	0012FF84	00401079	返回到 Cruehead.<ModuleEntryPoint>+79 来自 Cruehead.00401311
00402018	48 49 00 00 00 00 00 00 00 00 00 00 00 00 00 00	HI.....	0012FF88	00402008	ASCII "123456789ABCDEFGH"
00402028	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012FF8C	75E13C45	返回到 kernel32.75E13C45
00402038	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012FF90	7FFDF000	
00402048	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012FF94	0012FFD4	

用户名算法的校验过程如下：

1. bl初始值为0x41
2. 然后取出用户名第i位的ASCII值，
3. 将用户名第i位的ASCII值和bl进行异或
4. 用户名右移一位
5. bl++
6. 然后将每一轮用户名第i位和bl异或的结果相加保存到[0x4020F9]这个地址，这个地址保存的值就是用户名计算的结果
7. i++
8. 整个过程循环0xE次
9. 将用户名计算的结果和0x12345678进行异或，如下图

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
0040105B	FF35 F5204000	push dword ptr ds:[0x4020F5]	hFile = 0000006C (window)	EAX 0000000B
00401061	E8 30040000	call <jmp.&KERNEL32.ReadFile>	ReadFile	ECX 0000000E
00401066	833D A0214000	cmp dword ptr ds:[0x4021A0],0x12	检查文件内容是否有0x12个字节	EDX 775970B4 ntdll.KiFastSystemCallRet
0040106D	75 C8	jnz short Cruehead.00401037		EBX 0040204F Cruehead.0040204F
0040106F	68 08204000	push Cruehead.00402008		ESP 0012FF88
00401074	E8 98020000	call Cruehead.00401311		EBP 0012FF94
00401079	8135 F9204000	xor dword ptr ds:[0x4020F9],0x12345678	[0x4020F9]=[0x4020F9]^0x12345678	ESI 00402016 ASCII "FGH"
00401083	83C4 04	add esp,0x4		EDI 00000000
00401086	68 08204000	push Cruehead.00402008	FileContent	EIP 00401079 Cruehead.00401079
0040108B	E8 AC020000	call Cruehead.0040133C		C 0 ES 0023 32位 0(FFFFFFFF)
00401090	83C4 04	add esp,0x4		

序列号算法分析(后4位)

00401086	68 08204000	push Cruehead.00402008	FileContent	EIP 0040108B Cruehead.0040108B
0040108B	E8 AC020000	call Cruehead.0040133C		C 0 ES 0023 32位 0 (FFFFFFFF)
00401090	83C4 04	add esp,0x4		P 0 CS 001B 32位 0 (FFFFFFFF)
00401093	3B05 F9204000	cmp eax,dword ptr ds:[0x4020F9]	比较后四位是否等于前12位计算的结果	A 0 SS 0023 32位 0 (FFFFFFFF)
00401099	0f94c0	sete al		Z 0 DS 0023 32位 0 (FFFFFFFF)
0040109C	50	push eax		S 0 FS 003B 32位 7FFDE000 (FFF)
0040109D	84C0	test al,al		T 0 GS 0000 NULL
0040109F	74 96	js short Cruehead.00401037	关键跳转	D 0
004010A1	68 0E214000	push Cruehead.0040210E	ASCII "CrackMe v3.0"	O 0 LastErr ERROR_SUCCESS (00000000)
004010A6	E8 9B020000	call Cruehead.00401346		EFL 00000202 (NO,NB,NE,A,NS,PO,
004010AB	83C4 04	add esp,0x4		ST0 empty 0.0
004010AE	6A 00	push 0x0	Title = NULL	ST1 empty 0.0
004010B0	68 28214000	push Cruehead.00402128	Class = "No need to disasm the code!"	ST2 empty 0.0
00402008-Cruehead.00402008				
地址	HEX 数据	反汇编	注释	
00402008	70 70 70 70 70 70 70 70 09 0B 46 47	ppppppppppp.0.0.FG	0012FF88 00402008	Cruehead.00402008
00402018	48 49 00 00 00 00 00 00 00 00 00 00	HI.....	0012FF8C 75E13C45	返回到 kernel32.75E13C45
00402028	00 00 00 00 00 00 00 00 00 00 00 00	0012FF90 7FFDF000	
00402038	00 00 00 00 00 00 00 00 00 00 00 00	0012FF94 0012FFD4	
00402048	00 00 00 00 00 00 00 00 00 00 00 00	0012FF98 775B37F5	返回到 ntdll.775B37F5
00402058	00 00 00 00 00 00 00 00 00 00 00 00	0012FF9C 7FFDF000	
00402068	00 00 00 00 00 00 00 00 00 00 00 00	0012FFA0 775CA3F8	ntdll.775CA3F8

接下来将0x14个字节的文件内容压入堆栈，然后执行0040133C这个函数

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
0040133C	8B7424 04	mov esi,dword ptr ss:[esp+0x4]	kernel32.75E13C45	EAX 0000
00401340	83C6 0E	add esi,0xE	FileContent>>0x14	ECX 0000
00401343	8B06	mov eax,dword ptr ds:[esi]	eax=FileContent[15-18]	EDX 7759
00401345	C3	ret		EBX 0040
00401346	8B7424 04	mov esi,dword ptr ss:[esp+0x4]	kernel32.75E13C45	ESP 0012

这个函数的作用就算将文件内容右移14位，也就是去掉前14位用户名的部分，然后将最后四位赋值给eax

Windows 7 32 - VMware Workstation				
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)				
Windows 7 32				
解密 - Cruehead.3.exe - [LCG - 主线程 - 模块 - Cruehead]				
文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+]				
地址	HEX 数据	反汇编	注释	寄存器 (FPU)
0040105B	FF35 F5204000	push dword ptr ds:[0x4020F5]	hFile = 0000006C (window)	EAX 49484746
00401061	E8 30040000	call <imp.&KERNEL32.ReadFile>	ReadFile	ECX 0000000E
00401066	833D A0214000	cmp dword ptr ds:[0x4021A0],0x12	检查文件内容是否有0x12个字节	EDX 775970B4 ntdll.KiFastSystemCallRet
0040106D	75 C8	jnz short Cruehead.00401037		EBX 0040204F Cruehead.0040204F
0040106F	68 08204000	push Cruehead.00402008		ESP 0012FF8C
00401074	E8 98020000	call Cruehead.00401311		EBP 0012FF94
00401079	8135 F9204000	xor dword ptr ds:[0x4020F9],0x12345678	[0x4020F9]=[0x4020F9]^0x12345678	ESI 00402016 ASCII "FGHI"
00401083	83C4 04	add esp,0x4		EDI 00000000
00401086	68 08204000	push Cruehead.00402008	FileContent	EIP 00401093 Cruehead.00401093
0040108B	E8 AC020000	call Cruehead.0040133C		C 0 ES 0023 32位 0 (FFFFFFFF)
00401090	83C4 04	add esp,0x4		P 0 CS 001B 32位 0 (FFFFFFFF)
00401093	3B05 F9204000	cmp eax,dword ptr ds:[0x4020F9]	比较后四位是否等于前12位计算的结果	A 0 SS 0023 32位 0 (FFFFFFFF)
00401099	0f94c0	sete al		Z 0 DS 0023 32位 0 (FFFFFFFF)
0040109C	50	push eax		S 0 FS 003B 32位 7FFDE000 (FFF)
0040109D	84C0	test al,al		T 0 GS 0000 NULL
0040109F	74 96	js short Cruehead.00401037	关键跳转	D 0
004010A1	68 0E214000	push Cruehead.0040210E	ASCII "CrackMe v3.0"	O 0 LastErr ERROR_SUCCESS (00000000)
004010A6	E8 9B020000	call Cruehead.00401346		EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)
004010AB	83C4 04	add esp,0x4		ST0 empty 0.0
004010AE	6A 00	push 0x0	Title = NULL	ST1 empty 0.0
004010B0	68 28214000	push Cruehead.00402128	Class = "No need to disasm the code!"	ST2 empty 0.0
ds:[004020F9]=1234525F eax=49484746				
地址	HEX 数据	ASCII	地址	数值
004020F9	5F 52 34 12	[R4]Try to crack	0012FF8C	75E13C45
00402109	20 6D 65 21	me!.CrackMe v3.	0012FF90	7FFDF000
00402119	30 20 20 20	0 .N	0012FF94	0012FFD4
00402129	6F 20 6E 65	o need to disasm	0012FF98	775B37F5
00402139	20 74 68 65	the code!.MENU.	0012FF9C	7FFDF000
00402149	0E 00 00 00	...DLG_ABOUT.Go	0012FFA0	775CA3F8
00402159	AF 64 70 77	ad ...cracker!	0012FFA4	00000000

最后比较后四位是否等于[0x4020F9]，也就是前12位用户名计算的结果，根据比较的结果提示是否破解成功

算法总结

总结一下前面的分析，文件内容必须是18个字符，前14位是用户名，后4位是序列号。序列号必须等于用户名计算的结果。

注册机探索

接下来写出注册机，代码如下：

```
#include <iostream>
#include <windows.h>
using namespace std;
```

```

int main()
{

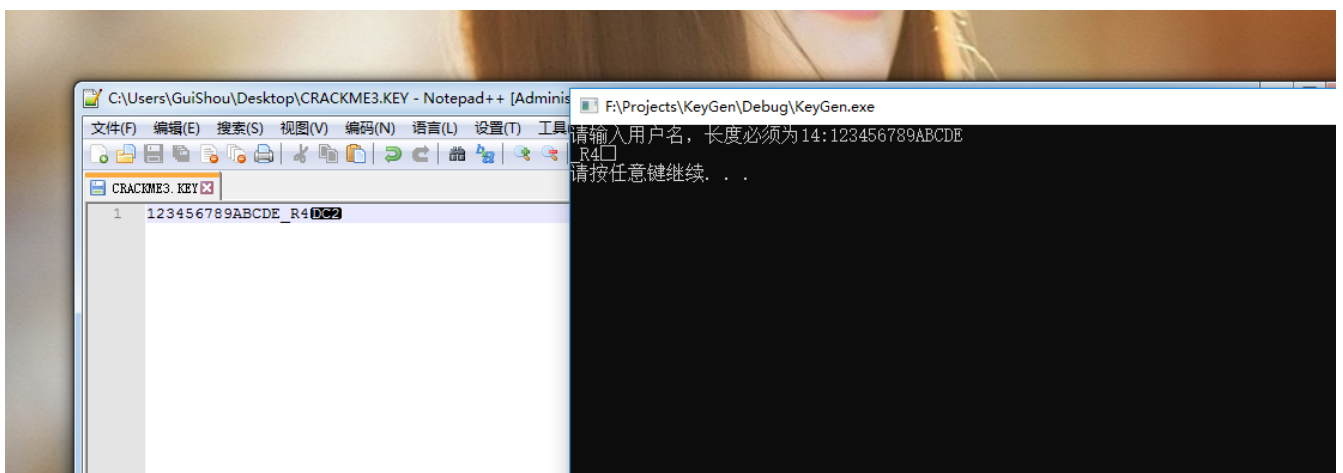
    char username[15] = { 0 };
    char serial[5] = { 0 };
    int temp = 0x41;
    int result = 0;
    printf("请输入用户名, 长度必须为14:");
    scanf_s("%s", username, 15);

    for (int i = 0; i < 14; i++)
    {
        username[i] ^= temp;
        result += username[i];
        temp++;
    }
    result ^= 0x12345678;

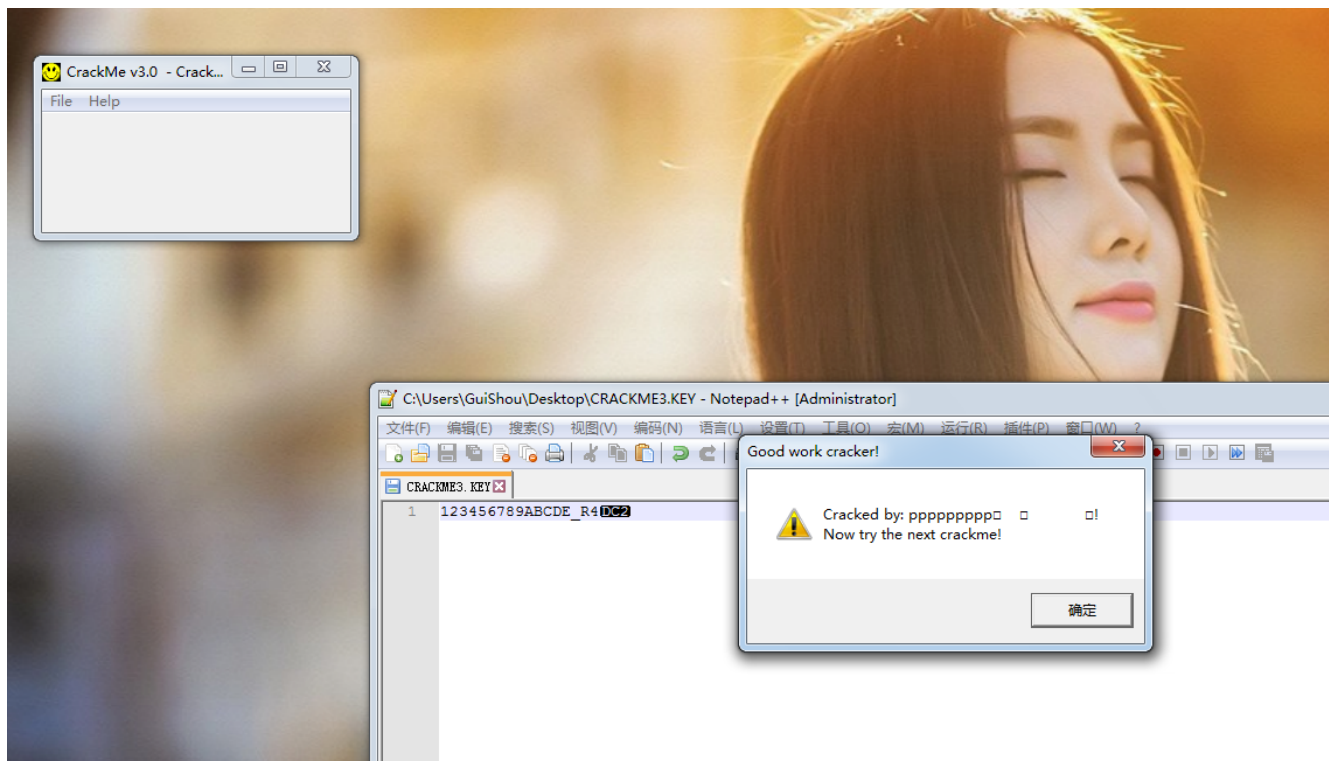
    char* p = (char*)&result;
    for (int i = 0; i < 4; i++)
    {
        printf("%c", p[i]);
    }
    printf("\n");
    system("pause");
    return 0;
}

```

验证结果



随便输入一个12位的用户名，然后把计算出来的序列号复制过去替换后四位，乱码不需要管，再打开目标程序



破解完成，最后需要相关文件可以到我的Github下载:

<https://github.com/TonyChen56/160-Crackme>