

查壳

一样的014和017?

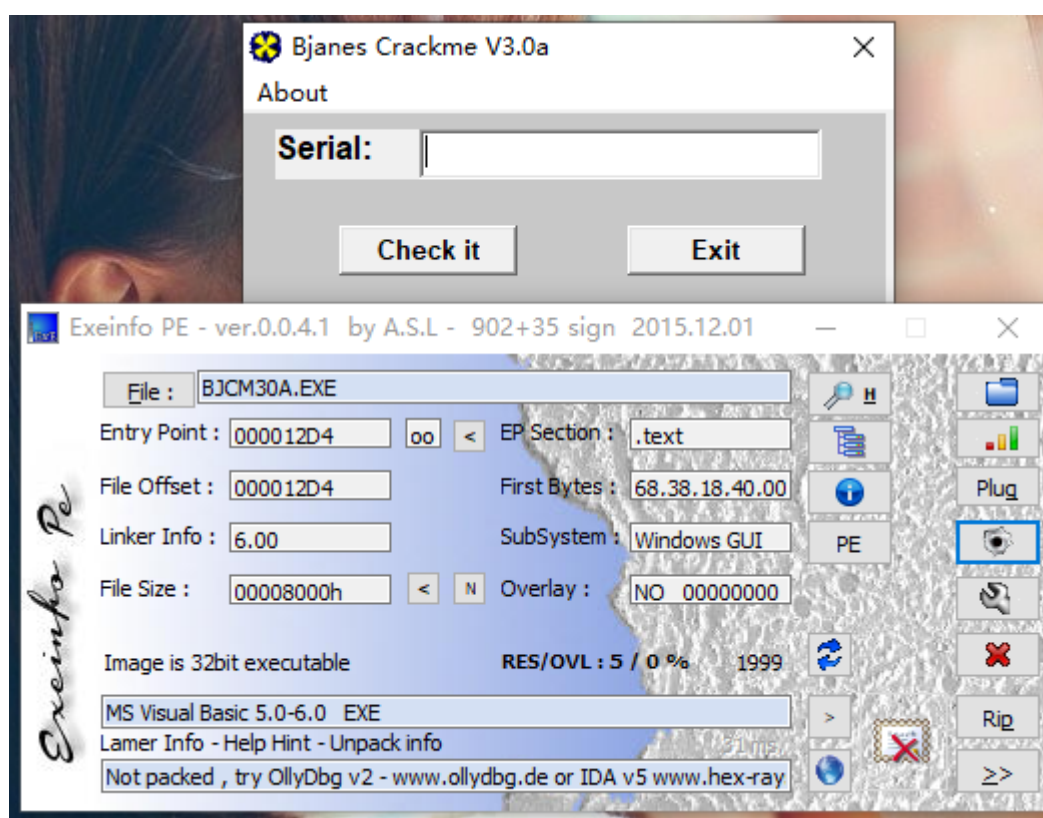
破解反调试

分析算法

014和017区别

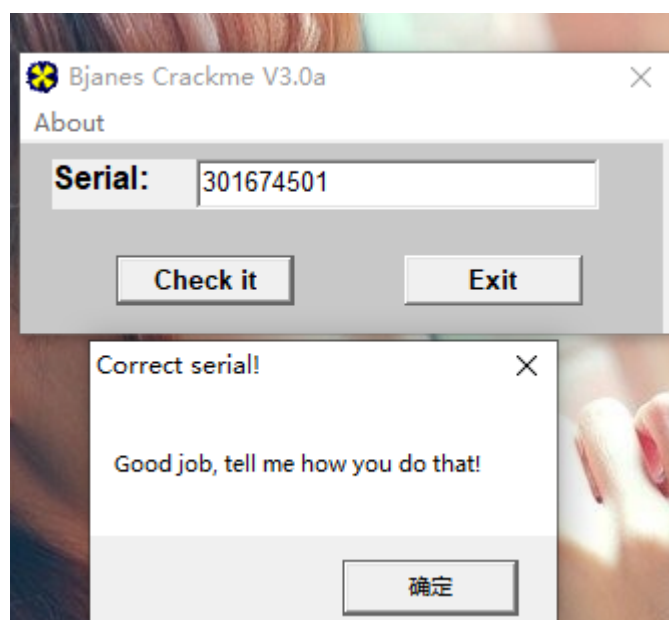
写出注册机

查壳



一样的014和017?

这个Crackme和014和016是同一个作者，不会又是重复的吧？输入014的序列号试试



好吧，虽然是能蒙对，但其实里面的算法是完全不一样的，这到底是为什么？这个问题留到最后面直接根据字符串的错误提示，来到函数头的位置，也就是按钮事件开头，来分析整个算法

破解反调试

这个Crackme跟其他的不一样的地方是有一个时间戳检查，

00404314	. 89BD 54FEFFFF	mov dword ptr ss:[ebp-0x1AC],edi		
0040431A	. 89BD 44FEFFFF	mov dword ptr ss:[ebp-0x1BC],edi		
00404320	. FF15 94104000	call dword ptr ds:[<&MSVBVM60.#rttcGetTimer	第一次获取时间戳	
00404326	. FF15 D0104000	call dword ptr ds:[<&MSVBVM60.__vbaFpI4>]	msvbvm60.__vbaFpI4	
0040432C	. 8945 A4	mov dword ptr ss:[ebp-0x5C],eax	保存结果	
0040432F	. 8D95 08FFFFFF	lea edx,dword ptr ss:[ebp-0xF8]		
00404335	. 8D85 F8FEFFFF	lea eax,dword ptr ss:[ebp-0x108]		
0040433B	. 52	push edx		
0040433C	. 8D8D F8FEFFFF	lea ecx,dword ptr ss:[ebp-0x118]		
			Step8 = 00000006	

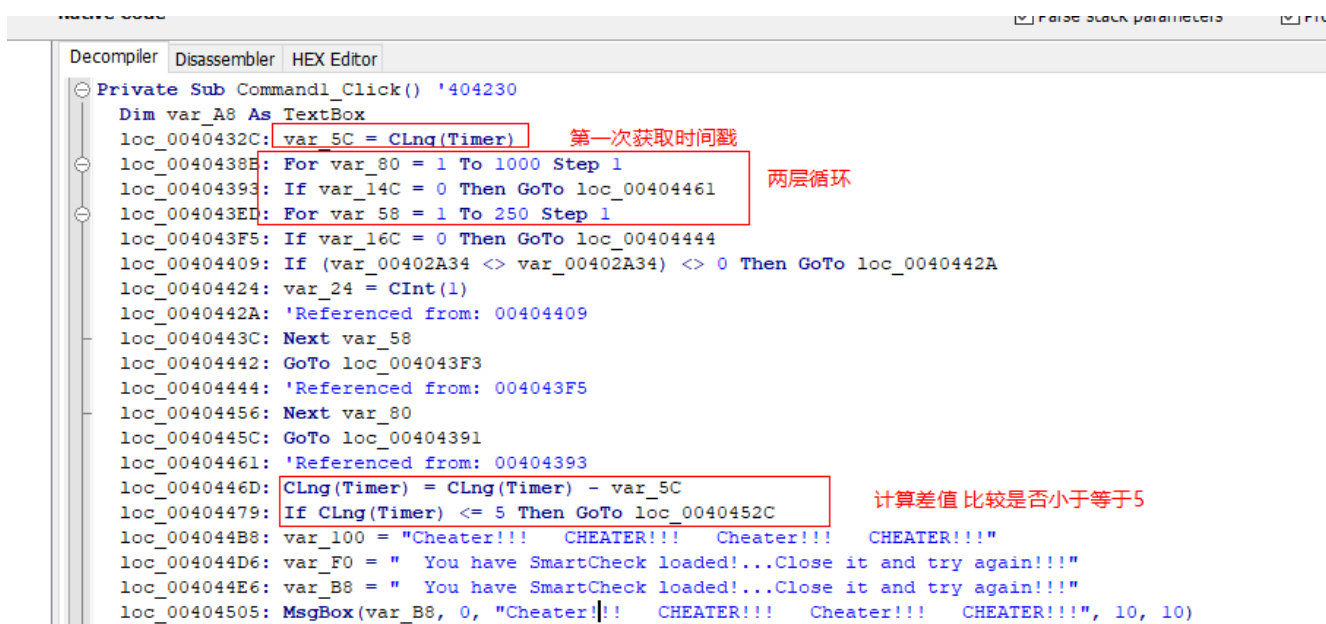
首先，获取当前事件，保存结果

地址	HEX	数据	反汇编	注释
00404430	. 8D85 94FEFFFF		lea eax,dword ptr ss:[ebp-0x16C]	
00404436	. 52		push edx	
00404437	. 8D4D A8		lea ecx,dword ptr ss:[ebp-0x58]	
0040443A	. 50		push eax	
0040443B	. 51		push ecx	
0040443C	. FF15 E8104000		call dword ptr ds:[<&MSVBVM60.__vbaVarForNext>]	
00404442	. EB AF		jmp short BJCM30A.004043F3	
00404444	> 8D95 A4FEFFFF		lea edx,dword ptr ss:[ebp-0x15C]	
0040444A	. 8D85 B4FEFFFF		lea eax,dword ptr ss:[ebp-0x14C]	
00404450	. 52		push edx	
00404451	. 8D4D 80		lea ecx,dword ptr ss:[ebp-0x80]	
00404454	. 50		push eax	
00404455	. 51		push ecx	
00404456	. FF15 E8104000		call dword ptr ds:[<&MSVBVM60.__vbaVarForNext>]	
0040445C	. E9 30FFFFFF		jmp BJCM30A.00404391	
00404391=BJCM30A.00404391				

接着是一个两层的嵌套循环，具体什么作用不知道，大概是为了拖延调试者的时间，

0040445C	E9 30FFFFFF	jmp BJCM30A.00404391	第二次获取时间戳
00404461	FF15 94104000	call dword ptr ds:[<&MSVBVM60.#rtcGetTimer_535>]	msvbvm60.__vbaFpI4
00404467	FF15 D0104000	call dword ptr ds:[<&MSVBVM60.__vbaFpI4>]	计算两次时间的差值
0040446D	2B45 A4	sub eax,dword ptr ss:[ebp-0x5C]	
00404470	0F80 340C0000	ja BJCM30A.004050AA	
00404476	83F8 05	cmp eax,0x5	比较是否小于等于5
00404479	0F8E AD000000	jle BJCM30A.0040452C	
0040447F	8B1D CC104000	mov ebx,dword ptr ds:[<&MSVBVM60.__vbaVarDup>]	msvbvm60.__vbaVarDup
00404485	B9 04000280	mov ecx,0x80020004	
0040448A	898D 20FFFFFF	mov dword ptr ss:[ebp-0xF0],ecx	

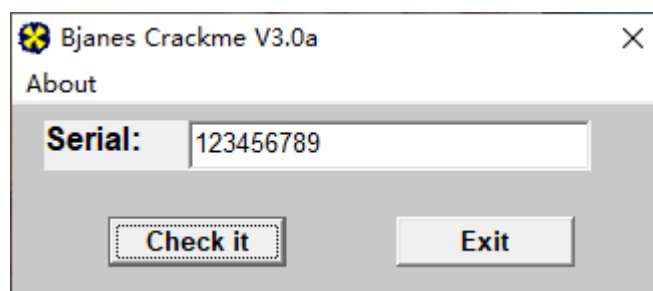
两次循环过后，再次获取时间戳，计算两次获取的时间的差值，比较是否小于等于5，如果小于继续往下走，否则报错



从VB Decompiler反汇编出的N-Code也可以直观的看出结果

知道反调试原理，破解其实就很简单了，只要在跳转的地址处下断点F9运行就可以了，或者修改ZF标志位

分析算法



先随便输入一个序列号，开始分析算法

地址	HEX 数据	反汇编	注释	寄存器 (MMX)
00404567	FF15 30104000	call dword ptr ds:[<&MSVBVM60. __vbaLenBstr>]	求序列号长度	EAX 00000009
0040456D	8B95 7CFFFFFF	mov edx,dword ptr ss:[ebp-0x84]		ECX 02200000
00404573	52	push edx		EDX 0058C98C UNICODE "123456789"
00404574	FF15 14104000	call dword ptr ds:[<&MSVBVM60. __vbaLenBstr>]	求序列号长度	EBX 00000000
0040457A	33DB	xor ebx,ebx		ESP 0019F108
0040457C	83F8 05	cmp eax,0x5	将序列号长度和5进行比较	EBP 0019F300
0040457F	0f9cc3	setl bl		ESI 005635D0
00404582	8D8D 7CFFFFFF	lea ecx,dword ptr ss:[ebp-0x84]		EDI 00000000
00404588	F7DB	neg ebx		EIP 0040457C BJCM30A.0040457C
0040458A	FF15 F0104000	call dword ptr ds:[<&MSVBVM60. __vbaFreeStr>]	msvbvm60. __vbaFreeStr	C 0 ES 002B 32位 0 (FFFFFFFF)
00404590	8D8D 5CFFFFFF	lea ecx,dword ptr ss:[ebp-0xA4]		P 1 CS 0023 32位 0 (FFFFFFFF)
00404596	FF15 F4104000	call dword ptr ds:[<&MSVBVM60. __vbaFreeObj>]	msvbvm60. __vbaFreeObj	A 0 SS 002B 32位 0 (FFFFFFFF)
0040459C	66:3BDF	cmp bx,di	比较长度	Z 1 DS 002B 32位 0 (FFFFFFFF)
0040459F	0F85 39090000	jnz BJCM30A.00404EDE	小于5则跳转	S 0 FS 0053 32位 3BE000 (FFF)
004045A5	BB 02000000	mov ebx,0x2	ebx=2	T 0 GS 002B 32位 0 (FFFFFFFF)
004045A7	8B95 7CFFFFFF	mov ebx,dword ptr ss:[ebp-0x84]		D 0
eax=00000009				0 0 LastErr ERROR_SUCCESS (00000000)

首先求序列号长度，将序列号长度和5进行比较，小于则跳转，即长度必须大于5

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
0040461C	51	push ecx		EAX 00000009
0040461D	FF15 14104000	call dword ptr ds:[<&MSVBVM60. __vbaLenBstr>]	求序列号长度	ECX 0058C98C UNICODE "123456789"
00404623	8985 00FFFFFF	mov dword ptr ss:[ebp-0x100],eax		EDX 02200000
00404629	8D95 08FFFFFF	lea edx,dword ptr ss:[ebp-0xF8]	edx=1	EBX 02363B1C
0040462F	8D85 F8FFFFFF	lea eax,dword ptr ss:[ebp-0x108]	eax=序列号长度	ESP 0019F108
00404635	52	push edx	Step8 = 02200000	EBP 0019F300
00404636	8D8D E8FFFFFF	lea ecx,dword ptr ss:[ebp-0x118]		ESI 005635D0
0040463C	50	push eax	End8 = 00000009	EDI 00000000
0040463D	8D95 64FFFFFF	lea edx,dword ptr ss:[ebp-0x196]		EIP 00404623 BJCM30A.00404623
00404643	51	push ecx	Start8 = 0058C98C	C 0 ES 002B 32位 0 (FFFFFFFF)
00404644	8D85 74FFFFFF	lea eax,dword ptr ss:[ebp-0x18C]		P 1 CS 0023 32位 0 (FFFFFFFF)
0040464A	52	push edx	TMPend8 = 02200000	A 0 SS 002B 32位 0 (FFFFFFFF)
0040464B	8D4D 94	lea ecx,dword ptr ss:[ebp-0x6C]		Z 0 DS 002B 32位 0 (FFFFFFFF)
0040464E	50	push eax	TMPstep8 = 00000009	S 0 FS 0053 32位 3BE000 (FFF)
0040464F	51	push ecx	Counter8 = 0058C98C	T 0 GS 002B 32位 0 (FFFFFFFF)
00404650	6795 F8FFFFFF	mov dword ptr ss:[ebp-0x108],0x2		D 0
eax=00000009				0 0 LastErr ERROR_SUCCESS (00000000)
堆栈 ss:[0019F200]=000000FA				

接着 再次求序列号长度，将长度作为循环的次数

004047C1	. 50	push eax	eax=序列号	ES 002B 32位 0 (FFFFFFFF)
004047C2	. FFD7	call edi	取出Key[1]	CS 0023 32位 0 (FFFFFFFF)
004047C4	. 8BD0	mov edx, eax		SS 002B 32位 0 (FFFFFFFF)
004047C6	. 8D8D 70FFFFFF	lea ecx, dword ptr ss:[ebp-0x90]	ecx=Key	DS 002B 32位 0 (FFFFFFFF)
004047CC	. FFD3	call ebx	msvbvm60. __vbaStrMove	FS 0053 32位 0 (FFFFFFFF)
004047CE	. 50	push eax		GS 002B 32位 0 (FFFFFFFF)
004047CF	. FF15 68104000	call dword ptr ds:[<&MSVBVM60. __vbaStrCmp>]	if(Key[0]==Key[1])	
004047D5	. 8BF8	mov edi, eax		LastErr ERROR_SUCCESS (00000000)
004047D7	. 8D8D 70FFFFFF	lea ecx, dword ptr ss:[ebp-0x90]	ecx=Key	00000246 (NO)
004047DD	. F7DF	neg edi	msvbvm60. rtcMidCharBstr	empty 0.0
004047DF	. 8D95 74FFFFFF	lea edx, dword ptr ss:[ebp-0x8C]		empty 0.0
004047E5	. 51	push ecx	ecx=Key	empty 0.0
004047E6	. 8D85 78FFFFFF	lea eax, dword ptr ss:[ebp-0x88]		empty 0.0
004047EC	. 52	push edx		empty 0.0
004047ED	. 1BFF	sbb edi, edi	msvbvm60. rtcMidCharBstr	empty 1.0000
004047EF	. 8D8D 70FFFFFF	lea ecx, dword ptr ss:[ebp-0x90]		empty 0.5000
ds:[00401068]=660E9596 (msvbvm60. __vbaStrCmp)				

循环比较序列号两两之间是否相等

地址	HEX 数据	反汇编	注释
0040482C	. 50	push eax	
0040482D	. 6A 03	push 0x3	
0040482F	. FF15 18104000	call dword ptr ds:[<&MSVBVM60. __vbaFreeVarList	msvbvm60. __vbaFreeVarList
00404835	. 83C4 30	add esp,0x30	
00404838	. 66:85FF	test di,di	比较序列号的前两位是否相等
0040483B	. 74 37	je short BJCM30A.00404874	不相等则跳转
0040483D	. 8D4D B8	lea ecx,dword ptr ss:[ebp-0x48]	相等则记录循环次数->[ebp-0x48]
00404840	. 8D95 08FFFFFF	lea edx,dword ptr ss:[ebp-0xF8]	
00404846	. 51	push ecx	var18 = 00190002
00404847	. 8D85 48FFFFFF	lea eax,dword ptr ss:[ebp-0xB8]	
0040484D	. 52	push edx	var28 = 0019F238
0040484E	. 50	push eax	saveto8 = 00000003
0040484F	. C785 10FFFFFF	mov dword ptr ss:[ebp-0xF0],0x1	
00404859	. C785 08FFFFFF	mov dword ptr ss:[ebp-0xF8],0x2	
00404863	. FF15 C8104000	call dword ptr ds:[<&MSVBVM60. __vbaVarAdd>]	__vbaVarAdd

如果相等 则记录循环次数到[ebp-0x48]处，不相等则不记录

地址	HEX 数据	反汇编	注释
00404874	> 8D8D 64FEFFFF	lea ecx,dword ptr ss:[ebp-0x19C]	ecx=序列号长度
0040487A	. 8D95 74FEFFFF	lea edx,dword ptr ss:[ebp-0x18C]	edx=1
00404880	. 51	push ecx	TMPend8 = 00000001
00404881	. 8D45 94	lea eax,dword ptr ss:[ebp-0x6C]	
00404884	. 52	push edx	TMPstep8 = 80000000
00404885	. 50	push eax	Counter8 = 00000001
00404886	. FF15 E8104000	call dword ptr ds:[<&MSVBVM60. __vbaVarForNext	__vbaVarForNext
0040488C	. 8985 30FEFFFF	mov dword ptr ss:[ebp-0x1D0],eax	
00404892	. 33FF	xor edi,edi	
00404894	. E9 FFFDFFFF	jmp BJCM30A.00404698	跳转到循环开始处
00404899	> 8B0E	mov ecx,dword ptr ds:[esi]	BJCM30A.00406A74
0040489B	. 56	nush esi	

然后开始新一轮循环

地址	HEX 数据	反汇编	注释
004048DE	. FF15 30104000	call dword ptr ds:[<&MSVBVM60. __vballresultChe	msvbvm60. __vballresultCheckObj
004048E4	> 8B95 7CFEFFFF	mov edx,dword ptr ss:[ebp-0x84]	
004048EA	. 52	push edx	String = "123456789"
004048EB	. FF15 14104000	call dword ptr ds:[<&MSVBVM60. __vbaLenBstr>]	求序列号长度
004048F1	. 83E8 01	sub eax,0x1	序列号长度-1
004048F4	. 8D8D 08FEFFFF	lea ecx,dword ptr ss:[ebp-0xF8]	ecx=序列号长度-1
004048FA	. 0F80 AA070000	jo BJCM30A.004050AA	
00404900	. 8985 10FEFFFF	mov dword ptr ss:[ebp-0xF0],eax	
00404906	. 8D45 B8	lea eax,dword ptr ss:[ebp-0x48]	[ebp-0x48]=记录的循环次数
00404909	. 50	push eax	var18 = 0019F2B8
0040490A	. 51	push ecx	var28 = 0019F208
0040490B	. C785 08FEFFFF	mov dword ptr ss:[ebp-0xF8],0x8003	
00404915	. FF15 6C104000	call dword ptr ds:[<&MSVBVM60. __vbaVarIntEq>]	比较记录的循环次数和序列号长度-1
0040491B	. 8D8D 7CFEFFFF	lea ecx,dword ptr ss:[ebp-0x84]	
00404921	. 66:8985 CCFE	mov word ptr ss:[ebp-0x134],ax	

循环结束之后，将记录的循环次数和序列号长度-1进行比较，比较成功则报错，也就是说序列号的每一位不能为同一个数字

接下来才是真正的序列号算法，这个算法也是很有意思

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00404A7F	. 50	push eax		EAX 0058C7D4 UNICODE "9"
00404A80	. FF15 14104000	call dword ptr ds:[<&MSVBVM60. __vbaLenBstr>]	求序列号长度	ECX 00560000
00404A86	. 8D8D 48FEFFFF	lea ecx,dword ptr ss:[ebp-0xB8]	ecx=1	EDX 00000002
00404A8C	. 8985 50FEFFFF	mov dword ptr ss:[ebp-0xB0],ecx	[ebp-0xB0]=序列号长度	EBX 660E6C30 msvbvm60. __v
00404A92	. 51	push ecx		ESP 0019F108
00404A93	. C785 48FEFFFF	mov dword ptr ss:[ebp-0xB8],0x3		EBP 0019F300
00404A9D	. FF15 A8104000	call dword ptr ds:[<&MSVBVM60. #rtcllexBstrFromVar_57	将序列号长度转为字符串	ESI 005635D0
00404AA3	. 8BD0	mov edx,eax		EDI 00000000
00404AA5	. 8D8D 64FEFFFF	lea ecx,dword ptr ss:[ebp-0x9C]		EIP 00404AA3 BJCM30A.004C
00404AAB	. FFD3	call ebx	msvbvm60. __vbaStrMove	C 1 ES 002B 32位 0 (FFFFF
00404AAD	. 8B16	mov edx,dword ptr ds:[esi]	BJCM30A.00406A74	P 0 CS 0023 32位 0 (FFFFF
00404AAF	. 56	push esi		A 1 SS 002B 32位 0 (FFFFF
00404AB0	. FF92 08030000	call dword ptr ds:[edx+0x308]		Z 0 DS 002B 32位 0 (FFFFF
00404AB6	. 50	push eax		S 1 FS 0053 32位 3BE0000
00404AB7	. 8D85 58FEFFFF	lea eax,dword ptr ss:[ebp-0xA8]		T 0 GS 002B 32位 0 (FFFFF
00404ABD	. 50	push eax		D 0

首先计算出序列号的长度，将长度转为字符串

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00404AE6	. C785 38FFFFFF	mov dword ptr ss:[ebp-0xC8],0x9		EAX 0058C98C UNICODE
00404AF0	. FF15 D4104000	call dword ptr ds:[<&MSVBVM60.#rtcLeftCharVar_617>]	截取序列号的第一个字符串	ECX 00560000
00404AF6	. 8D85 28FFFFFF	lea eax,dword ptr ss:[ebp-0xD8]		EDX 00310033
00404AFC	. 8D8D 78FFFFFF	lea ecx,dword ptr ss:[ebp-0x88]		EBX 660E6C30 msbvm6
00404B02	. 50	push eax	String = 0058C98C	ESP 0019F108
00404B03	. 51	push ecx	ARG2 = 00560000	EBP 0019F300
00404B04	. FF15 90104000	call dword ptr ds:[<&MSVBVM60.__vbaStrVarVal>]	__vbaStrVarVal	ESI 005635D0
00404B0A	. 50	push eax	String = "3"	EDI 00000000
00404B0B	. FF15 28104000	call dword ptr ds:[<&MSVBVM60.#rtcAnsiValueBstr_516>]	取出Key[0]的ASCII值	EIP 00404B2F BJCM30A
00404B11	. 8D95 18FFFFFF	lea edx,dword ptr ss:[ebp-0xE8]		C 1 ES 002B 32位 0(
00404B17	. 66:8985 20FF	mov word ptr ss:[ebp-0xE0],ax	[ebp-0xE0]=Key[0]的ASCII	P 0 CS 0023 32位 0(
00404B1E	. 52	push edx		A 1 SS 002B 32位 0(
00404B1F	. C785 18FFFFFF	mov dword ptr ss:[ebp-0xE8],0x2		Z 0 DS 002B 32位 0(
00404B29	. FF15 A8104000	call dword ptr ds:[<&MSVBVM60.#rtcHexBstrFromVar_57>]	Key[0]的ASCII值转为字符串	S 1 FS 0053 32位 3B
00404B2F	. 8BD0	mov edx,eax		

接着，再取出序列号的第一个字符，将ASCII值转成字符串。

00404B82	. 8D8D 70FFFFFF	lea ecx,dword ptr ss:[ebp-0x90]		
00404B88	. 8D95 74FFFFFF	lea edx,dword ptr ss:[ebp-0x8C]		
00404B8E	. 51	push ecx	ecx="31"	
00404B8F	. 52	push edx	edx="9"	
00404B90	. 56	push esi		
00404B91	. FF90 F8060000	call dword ptr ds:[eax+0x6F8]	计算乘积	
00404B97	. 3BC7	cmp eax,edi		
00404B99	. 7D 12	jge short BJCM30A.00404BAD		
00404B9B	. 68 F8060000	push 0x6F8		
00404BA0	. 68 B4274000	push BJCM30A.004027B4		
00404BA5	. 56	push esi		
00404BA6	. 50	push eax		
00404BA7	. FF15 30104000	call dword ptr ds:[<&MSVBVM60.__vballresultChec	BJCM30A.00406A74	
00404BAD	. 8B95 68FFFFFF	mov edx,dword ptr ss:[ebp-0x98]	msbvm60.__vballresultCheckObj	
00404BB3	. 8D4D C8	lea ecx,dword ptr ss:[ebp-0x38]	edx="1B9">Key[0]ASCII*strlen(Ke	
00404BB6	. 8BDB 68FFFFFF	mov ebx,dword ptr ss:[ebp-0x98]		
ds:[0040716C]=00401FE8 (BJCM30A.00401FE8)				

接着计算这两个的乘积，结果为0x1B9

00404CBA . FFD3 call ebx					msbvm60.__vbaStrMove					EAX 0019F108
00404CBC . 50 push eax					String = ""					ECX 76C00000
00404CBD . FF15 28104000 call dword ptr ds:[<&MSVBVM60.#rtcAnsiValueBst					取出Key[0]的ASCII值					EDX 00000000
00404CC3 . 66:8985 00FF mov word ptr ss:[ebp-0x100],ax										EBX 660E6C30
00404CCA . 8D55 CC lea edx,dword ptr ss:[ebp-0x34]										ESP 0019F108
00404CCD . 8D85 F8FFFFFF lea eax,dword ptr ss:[ebp-0x108]					eax=Key[0].ASCII					EBP 0019F300
00404CD3 . 52 push edx					var18 = NULL					ESI 005635D0
00404CD4 . 8D8D 38FFFFFF lea ecx,dword ptr ss:[ebp-0xC8]										EDI 00000000
00404CDA . 50 push eax					var28 = 0019F2CC					EIP 00404B2F
00404CDB . 51 push ecx					saveto8 = user32.76CCA5F5					C 1 ES 002B 32位 0(
00404CDC . C785 F8FFFFFF mov dword ptr ss:[ebp-0x108],0x2										P 1 CS 0023 32位 0(
00404CE6 . FF15 C8104000 call dword ptr ds:[<&MSVBVM60.__vbaVarAdd>]					每一位的ASCII值相加					A 1 SS 002B 32位 0(
00404CEC . 8BD0 mov edx,eax										Z 0 DS 002B 32位 0(
00404CEE . 8D4D CC lea ecx,dword ptr ss:[ebp-0x34]										S 1 FS 0053 32位 3B
00404CF1 . FF15 08104000 call dword ptr ds:[<&MSVBVM60.__vbaVarMove>]					将结果保存到[ebp-0x34]					T 0 GS 00000000
ds:[004010C8]=661877C1 (msbvm60.__vbaVarAdd)										O 0 La

地址	HEX 数据	UNICODE	地址	数值	注释
0019F2CC	02 00 00 00	00 00 00 00	31 00 00 00	F5 AE CC 76	...1. 76
0019F2DC	02 00 00 00	49 00 5A 00	01 00 00 00	04 00 00 00	.1Z□. □
0019F2EC	A0 F7 19 00	66 11 40 00	08 F1 19 00	30 11 40 00	□ #@

然后将计算序列号每一个为ASCII值相加的结果，将结果保存到[ebp-0x34]

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00404D39	. 50	push eax	Counter8 = 0058C7D4	EAX 0058C7D4 UNICODE "1DD"
00404D3A	. FF15 E8104000	call dword ptr ds:[<&MSVBVM60. __vbaVarForNext>]	__vbaVarForNext	ECX 00560000
00404D40	. 8985 2CFEFFFF	mov dword ptr ss:[ebp-0x1D4], eax		EDX 00440031
00404D46	. E9 D7FCFFFF	jmp BJC30A.00404A22		EBX 660E6C30 msvbvm60. __vbaSt
00404D4B	. 8D4D CC	lea ecx, dword ptr ss:[ebp-0x34]		ESP 0019F108
00404D4E	. 51	push ecx		EBP 0019F300
00404D51	. FF15 A8104000	call dword ptr ds:[<&MSVBVM60. #rtcllexBstrFromV	将ASCII值相加的结果转为字符串	ESI 005635D0
00404D55	. 8BD0	mov edx, eax		EDI 00000000
00404D57	. 8D8D 70FFFFFF	lea ecx, dword ptr ss:[ebp-0x90]		EIP 00404D55 BJC30A.00404D55
00404D5D	. FFD3	call ebx	msvbvm60. __vbaStrMove	C 1 ES 002B 32位 0 (FFFFFFFF)
00404D5F	. BA 0C294000	mov edx, BJC30A.0040290C	=	P 0 CS 0023 32位 0 (FFFFFFFF)
00404D64	. 8D8D 78FFFFFF	lea ecx, dword ptr ss:[ebp-0x88]		A 1 SS 002B 32位 0 (FFFFFFFF)
00404D6A	. FF15 B0104000	call dword ptr ds:[<&MSVBVM60. __vbaStrCopy>]	msvbvm60. __vbaStrCopy	Z 0 DS 002B 32位 0 (FFFFFFFF)
00404D70	. 8B95 70FFFFFF	mov edx, dword ptr ss:[ebp-0x90]		S 1 FS 0053 32位 3BE000 (FFF)
00404D76	. 8D8D 7CFFFFFF	lea ecx, dword ptr ss:[ebp-0x84]		T 0 GS 002B 32位 0 (FFFFFFFF)
00404D7C	. 8BD0	mov edx, eax		D 0
eax=0058C7D4, (UNICODE "1DD")				0 0 LastErr ERROR_SUCCESS (0)
edx=00440031				

接着将序列号ASCII相加的结果转为字符串->"1DD"

地址	HEX 数据	反汇编	注释	寄存器
00404D94	. 8D85 7CFFFFFF	lea eax, dword ptr ss:[ebp-0x84]		EAX 00
00404D9A	. 8D4D C8	lea ecx, dword ptr ss:[ebp-0x38]		ECX 00
00404D9D	. 50	push eax	eax="1DD"	EDX 00
00404D9E	. 51	push ecx	ecx="1B9"	EBX 60
00404D9F	. 56	push esi		ESP 00
00404DA0	. FF92 F8060000	call dword ptr ds:[edx+0x6F8]	校验结果是否相等	EBP 00
00404DA6	. 3BC7	cmp eax, edi		ESI 00
00404DA8	. 7D 12	jge short BJC30A.00404DBC		EDI 00
00404DAA	. 68 F8060000	push 0x6F8		EIP 00
00404DAF	. 68 B4274000	push BJC30A.004027B4		C 0 1
00404DB4	. 56	push esi		P 1 0
00404DB5	. 50	push eax		A 0 0
00404DB6	. FF15 30104000	call dword ptr ds:[<&MSVBVM60. __vbaHresultC	msvbvm60. __vbaHresultCheckObj	Z 1 1
00404DBE	. 8B85 74FFFFFF	mov eax, dword ptr ss:[ebp-0x8C]	eax的值相等为"FFFF", 不相等为"0"	S 0 1
00404DC2	. BE 08000000	mov esi, 0x8		T 0 0
00404DC7	. 8D95 70FFFFFF	lea ecx, dword ptr ss:[ebp-0x90]		
ds:[0040716C]=00401FE8 (BJC30A.00401FE8)				

然后比较"1DD"和"1B9"是否相等, eax的值相等为"FFFF",不相等为"0"

00404DFC	. 51	push ecx		EDX 0019F2CC
00404DFD	. 6A 03	push 0x3		EBX 660E6C30 msv
00404DFF	. FF15 B4104000	call dword ptr ds:[<&MSVBVM60. __vbaFreeStrL	msvbvm60. __vbaFreeStrList	ESP 0019F100
00404E05	. 83C4 10	add esp, 0x10		EBP 0019F300
00404E08	. 8D55 CC	lea edx, dword ptr ss:[ebp-0x34]	"0"	ESI 00000008
00404E0B	. 8D85 08FFFFFF	lea eax, dword ptr ss:[ebp-0xF8]		EDI 00000000
00404E11	. C785 10FFFFFF	mov dword ptr ss:[ebp-0xF0], BJC30A.00402B5	FFFF	EIP 00404E27 BJC
00404E1B	. 52	push edx	var18 = 0019F2CC	C 0 ES 002B 32位
00404E1C	. 50	push eax	var28 = 0019F208	P 0 CS 0023 32位
00404E1D	. C785 08FFFFFF	mov dword ptr ss:[ebp-0xF8], 0x8008		A 0 SS 002B 32位
00404E27	. FF15 6C104000	call dword ptr ds:[<&MSVBVM60. __vbaVarTstEq	校验结果	Z 0 DS 002B 32位
00404E2D	. 66:85C0	test ax, ax		S 0 FS 0053 32位
00404E30	. 0F84 AD000000	je BJC30A.00404EE3		T 0 GS 002B 32位
00404E36	. 8B1D 66104000	mov ebx, dword ptr ds:[<&MSVBVM60. __vbaVarD		D 0
堆栈地址=0019F2CC edx=0019F2CC				0 0 LastErr ERF

地址	HEX 数据	UNICODE	地址	数值	注释
0019F208	08 80 00 00 49 00 5A 00	寿. IZ O @ L	0019F100	0019F208	var28 = 0019F208
0019F218	00 00 00 00 F0 67 D7 00	.. 奈 x1. 窗	0019F104	0019F2CC	var18 = 0019F2CC
0019F228	00 00 00 00 3C 78 61 00	.. 砵 a 襪 a	0019F108	0019F30C	
0019F238	00 00 00 00 00 00 00 00 et 砵 砵	0019F10C	0019F3DC	

接着校验刚才的比较结果, 根据结果提示正确或者失败

也就是说这个Crackme的序列号并不是只有唯一解, 必须满足以下三个条件

1. 序列号每一位不能为同一个数字
2. 序列号长度必须大于5
3. 序列号第一个字符的ASCII值乘以序列号长度必须等于序列号每一位的ASCII和

014和017区别

那么再来回到一开始的问题，为什么算法不一样但是用014的序列号301674501依然可以通过校验

1. 301674501满足前两个条件 长度大于5 每一位不相同
2. 301674501的ASCII的和为0x1CB
3. 301674501第一位的ASCII值33*9(序列号长度)也是等于0x1CB

所以这个序列号才能通过校验，估计作者是先写的017这个Crackme，然后将017其中的一个解直接作为014的唯一解

写出注册机

接下来根据算法写出注册机，为了减少计算量，我把长度固定为5

```
int CalcKey()
{
    srand(time(NULL));
    byte key[6] = { 0 };

    while (true)
    {
        byte k0 = rand() % 123;
        byte k1 = rand() % 123;
        byte k2 = rand() % 123;
        byte k3 = rand() % 123;
        byte k4 = rand() % 123;

        //限制随机数不出现无意义字符
        if ((k0>=65&&K0>=90)|| (k0 >= 97 && k0 >= 122)|| (k0 >= 49 && k0 >=
57))
        {
            if ((k1 >= 65 && k1 >= 90) || (k1 >= 97 && k1 >= 122 )|| (k1 >= 49
&& k1 >= 57))
            {
                if ((k2 >= 65 && k2 >= 90 )|| (k2 >= 97 && k2 >= 122) || (k2 >=
49 && k2 >= 57))
                {
                    if ((k3 >= 65 && k3 >= 90) || (k3 >= 97 && k3 >= 122) ||
(k3 >= 49 && k3 >= 57))
                    {
                        if ((k4 >= 65 && k4 >= 90) || (k4 >= 97 && k4 >= 122)
|| (k4 >= 49 && k4 >= 57))
                        {
                            //满足限制条件
                            if (k0 + k1 + k2 + k3 + k4 == k0 * 5)
                            {
```



```

        key[0] = K0;
        key[1] = K1;
        key[2] = K2;
        key[3] = K3;
        key[4] = K4;
        key[5] = 0;
        break;
    }
}

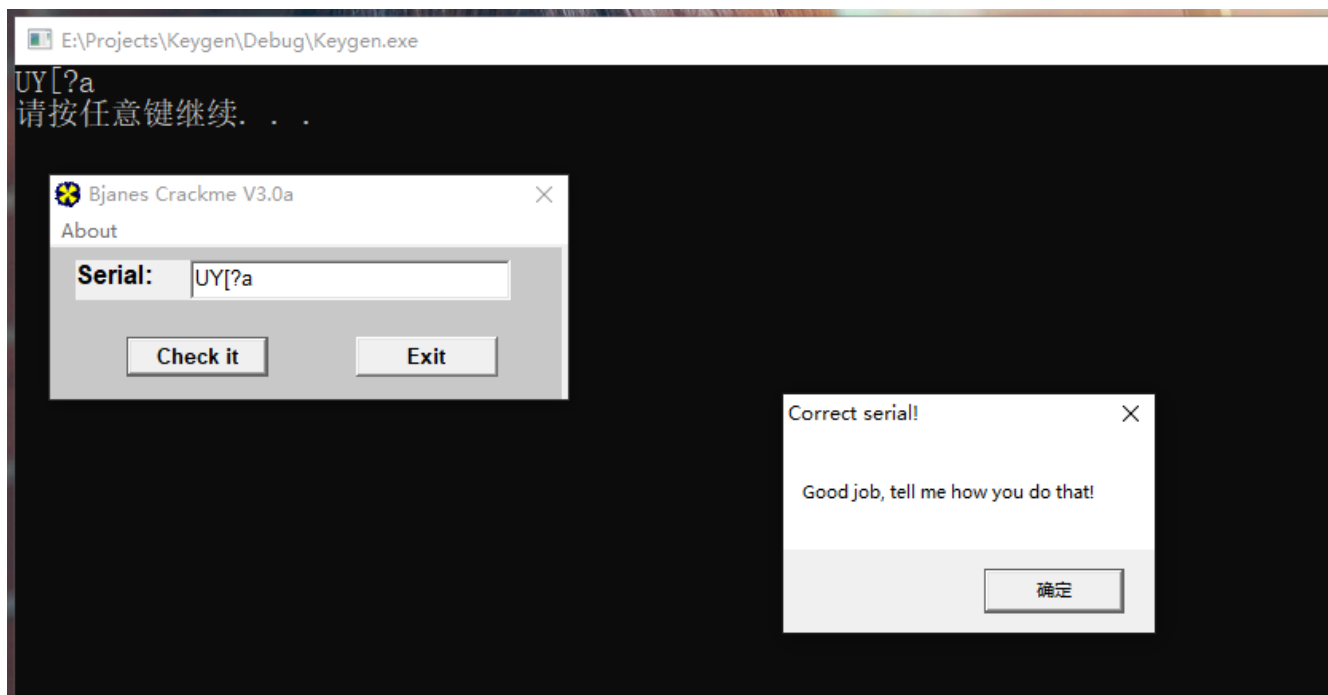
}

printf("%s\n", key);

return 0;
}

```

输入注册机的结果，提示成功，这个Crackme就完成了



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

