

查壳

VB Decompiler分析程序

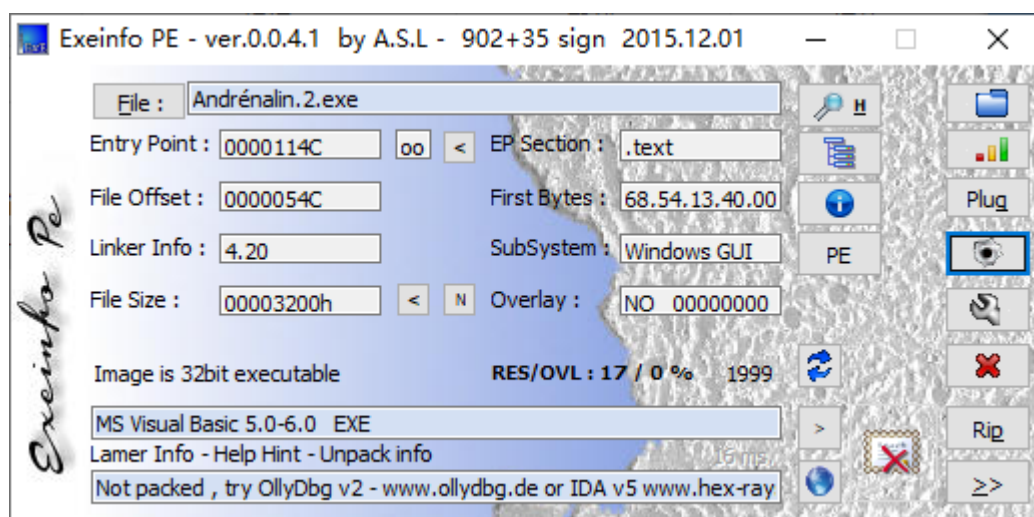
前置知识

OD分析算法

写出注册机

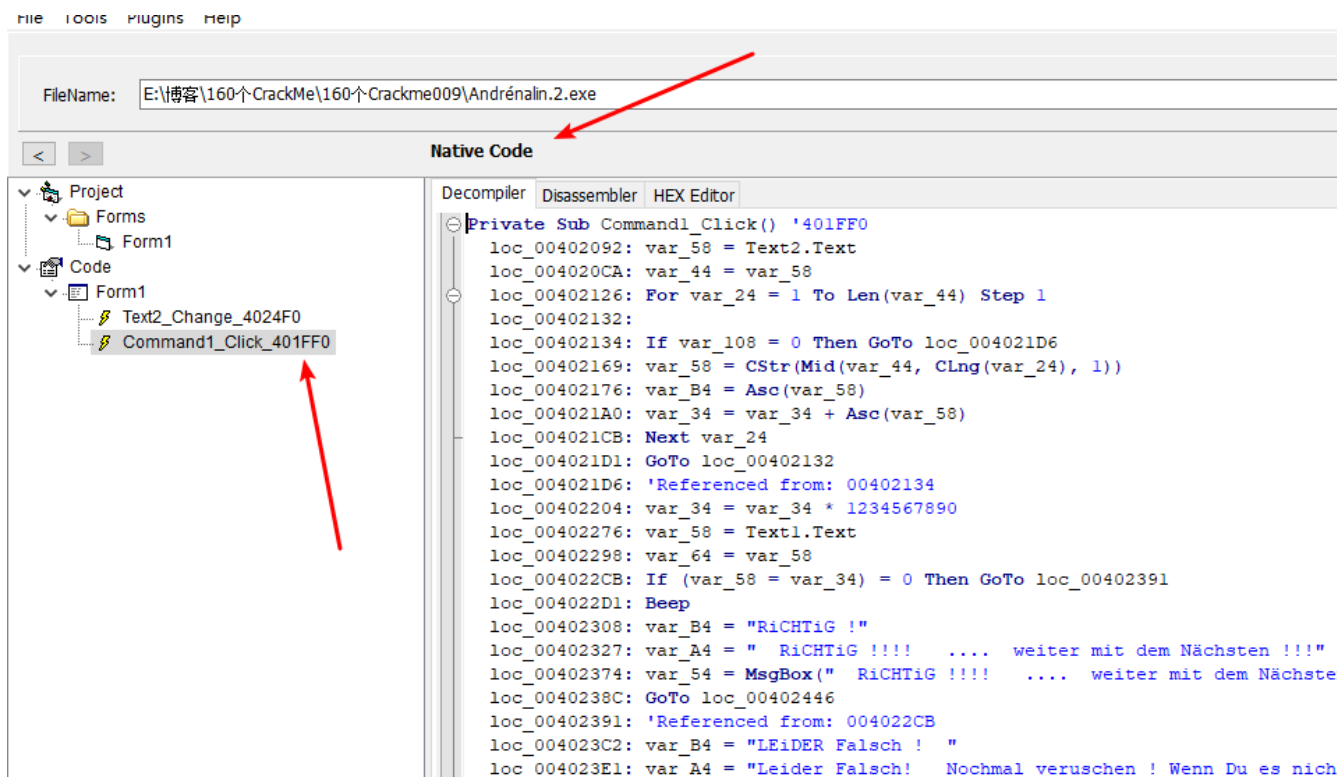
验证结果

## 查壳



先来查一下壳，这个程序是用VB写的，和008是同一个作者，用的是用户名和序列号的保护方式，难度可能会有所上升，所以得分析整个按钮点击事件的算法。

## VB Decompiler分析程序



把程序拖到VB Decompiler里，这个可是VB的逆向神器啊。

重要的信息有两个，一个是Native Code，说明是自然编译的，程序在编译的时候能够被编译为机器码，以Native Code方式编译的VB程序可以直接用OD调试。

另外一个就是Click的按钮点击事件了，因为整个程序只有一个OK按钮，所以直接就能判定这个就ok按钮的点击事件。

记下偏移，来到0x401FF0处。

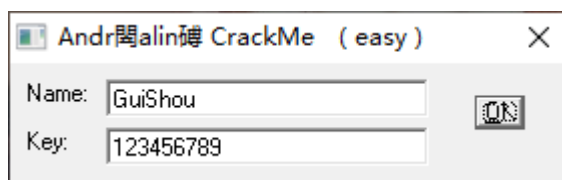
## 前置知识

想要分析出来这个Crackme，必须具备两个VB的前置知识

- 第一个就是VB的变量特征，VB是采用寄存器接收变量地址，地址是指向一个结构体，首地址存放的是类型编号，首地址+0x8。如果是字符串则首地址+0x8的位置是字符串的地址
- 第二个VB反汇编中很多时候会在[ebp-0x34]这个堆栈地址中看到最终的计算的结果，至于为什么 待会就知道了

如果你不知道上面两个规律 这个Crackme绝对会逆的一脸蒙蔽

## OD分析算法



随便输入一个账号密码，直接来到按钮事件开头，开始分析整个算法。

地址	HEX 数据	反汇编	注释
00401FEA	CC	int3	
00401FEB	CC	int3	
00401FEC	CC	int3	
00401FED	CC	int3	
00401FEE	CC	int3	
00401FEF	CC	int3	
00401FF0	> 55	push ebp	按钮事件开头
00401FF1	. 8BEC	mov ebp,esp	
00401FF3	. 83EC 0C	sub esp,0xC	
00401FF6	. 68 26104000	push <jmp.&MSVBVM50. __vbaExceptionHandler>	SE 处理程序安装
00401FFB	. 64:A1 000000	mov eax,dword ptr fs:[0]	
00402001	. 50	push eax	Andrena.00401A24
00402002	. 64:8925 0000	mov dword ptr fs:[0],esp	
00402009	. 81EC 18010000	sub esp,0x118	
ebp=0019F1DC			

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
004020F0	. 8D45 94	lea eax,dword ptr ss:[ebp-0x6C]		EDX 0019F18C
004020F3	. BB 02000000	mov ebx,0x2		EBX 00000000
004020F8	. 52	push edx		ESP 0019F080
004020F9	. 50	push eax		EBP 0019F1DC
004020FA	. 899D 54FFFFFF	mov dword ptr ss:[ebp-0xAC],ebx		ESI 741C0E22
00402100	. 899D 44FFFFFF	mov dword ptr ss:[ebp-0xBC],ebx		EDI 004032F4
00402106	. FF15 18414000	call dword ptr ds:[<&MSVBVM50. __vbaLenVar>]	求用户名长度	EIP 00402106
0040210C	. 8D8D 44FFFFFF	lea ecx,dword ptr ss:[ebp-0xBC]		C 0 ES 0021
00402112	. 50	push eax	eax=用户名长度	P 0 CS 0022
00402113	. 8D95 E8FFFFFF	lea edx,dword ptr ss:[ebp-0x118]		A 0 SS 0021
00402119	. 51	push ecx	Start8 = 0019F124	Z 0 DS 0021
0040211A	. 8D85 F8FFFFFF	lea eax,dword ptr ss:[ebp-0x108]		S 0 FS 0055
ds:[00404118]=7419EBF0 (MSVBVM50. __vbaLenVar)				T 0 GS 0021

地址	HEX 数据	UNICODE	地址	数值	注释
0019F18C	08 00 00 00 00 00 00 00	EC B3 5D 00 0E 00 00 00	0019F08C	0019F164	retBuffer8 = 0019F164
0019F19C	00 00 00 00 72 00 10 00	14 00 00 00 F4 40 29 FF	0019F090	0019F18C	var18 = 0019F18C
0019F1AC	00 00 00 00 7C 13 00 00	01 29 01 00 C0 02 49 00	0019F094	0019F124	
0019F1BC	88 F7 19 00 26 10 40 00	98 F0 19 00 00 10 40 00	0019F098	0019F1DC	
0019F1CC	01 00 00 00 DC F1 19 00	A9 E5 0D 74 98 65 58 00	0019F09C	0019F2B8	
0019F1DC	EC F1 19 00 24 1A 40 00	E0 65 58 00 00 00 00 00	0019F0A0	0287D2C4	
0019F1EC	08 F2 19 00 83 E5 0D 74	24 1A 40 00 B4 F2 19 00	0019F0A4	FFFFFFFF	
0019F1FC	02 00 00 00 01 00 00 00	E8 F2 19 00 44 F2 19 00	0019F0A8	0000137C	
0019F20C	81 B7 0E 74 E0 65 58 00	E8 F2 19 00 B4 F2 19 00	0019F0AC	0004A404	

首先函数会求出用户名的长度，怎么知道是求的用户名的长度呢？首先看堆栈找到参数的地址 数据窗口跟随，然后首地址+0x8的地方 也就是我内存窗口选中的地址再次数据窗口跟随。

ds:[00404118]=7419EBF0 (MSVBVM50. __vbaLenVar)							
地址	HEX 数据						UNICODE
005DB3EC	47	00	75	00	69	00	GuiShou.
005DB3FC	39	00	00	00	32	00	9.20. 起
005DB40C	00	05	00	88	60	1C	螭b璜m璜
005DB41C	B0	60	5A	00	5A	DD	棕Z 璜璜
005DB42C	59	5C	77	8D	2B	E6	厨起 蜈
005DB43C	00	00	00	00	02	00	... [...
005DB44C	00	00	00	00	03	00	...□...
005DB45C	00	07	00	8C	3C	13	*误&测..
005DB46C	F8	DA	5B	00	00	00	[....□.
005DB47C	00	00	00	00	3D	E6	.. 蜺螭
005DB48C	04	00	00	00	53	00	□.S.....
005DB49C	00	00	00	00	D8	BF	..豈[. 蜺
005DB4AC	00	09	00	88	60	1C	螭b璜m璜
005DB4BC	B0	60	5A	00	5A	DD	棕Z 璜璜

就找到了真正的参数，这个就是VB变量的存放规则，几乎无时无刻不在用。

地址	HEX 数据	反汇编	注释	寄存器 (FPU)	
004020F0	. 8D45 94	lea eax,dword ptr ss:[ebp-0x6C]		EAX 0019F164	
004020F3	. BB 02000000	mov ebx,0x2		ECX 00000008	
004020F8	. 52	push edx		EAX 0019F18C	
004020F9	. 50	push eax		EBX 00000002	
004020FA	. 899D 54FFFFFF	mov dword ptr ss:[ebp-0xAC],ebx		ESP 0019F094	
00402100	. 899D 44FFFFFF	mov dword ptr ss:[ebp-0x8C],ebx		EBP 0019F1D0	
00402106	. FF15 18414000	call dword ptr ds:[&MSVBVM50. _vbaLenVar]	求用户名长度	ESI 741C0E23 MSV	
0040210C	. 8D8D 44FFFFFF	lea ecx,dword ptr ss:[ebp-0xBC]		EDI 004032F0 And	
00402112	. 50	push eax		EIP 0040210C And	
00402113	. 8D95 E8FEFFFF	lea edx,dword ptr ss:[ebp-0x118]		C 1 ES 002B 32位	
00402119	. 51	push ecx		P 0 CS 0023 32位	
0040211A	. 8D85 F8FEFFFF	lea eax,dword ptr ss:[ebp-0x108]		A 1 SS 002B 32位	
堆栈地址=0019F114 ecx=00000008					
地址	HEX 数据	反汇编	注释	寄存器 (FPU)	
0019F164	03 00 00 00	00 00 00 00	07 00 00 00	0E 00 00 00	0019F094 0019F124
0019F174	00 00 00 00	00 00 00 00	00 00 00 00	FC 31 50 75	0019F098 0019F1DC
0019F184	D9 A1 CC 76	F4 40 01 29	08 00 00 00	00 00 00 00	0019F09C 0019F2B8
0019F194	EC B3 5D 00	0E 00 00 00	00 00 00 00	72 00 10 00	0019F0A0 0287D2C4
0019F1A4	14 00 00 00	F4 40 29 FF	00 00 00 00	7C 13 00 00	0019F0A4 FFFFFFFF
0019F1B4	01 29 01 00	C0 02 49 00	88 F7 19 00	26 10 40 00	0019F0A8 0000137C
0019F1C4	98 F0 19 00	00 10 40 00	01 00 00 00	DC F1 19 00	0019F0AC 0004A404
0019F1D4	A9 E5 0D 74	98 65 58 00	EC F1 19 00	24 1A 40 00	0019F0B0 004E0FC0
0019F1E4	E0 65 58 00	00 00 00 00	08 F2 19 00	83 E5 0D 74	0019F0B4 00000000
0019F1F4	24 1A 40 00	B4 F2 19 00	02 00 00 00	01 00 00 00	0019F0B8 00000000
0019F204	E8 F2 19 00	44 F2 19 00	81 B7 0E 74	E0 65 58 00	0019F0BC FFFFFFFF

接着步过这个函数，eax返回地址+0x8的位置就是用户名的长度。

地址	HEX 数据	反汇编	注释
00402100	. 899D 44FFFFFF	mov dword ptr ss:[ebp-0xBC],ebx	
00402106	. FF15 18414000	call dword ptr ds:[<MSVBVM50. __vbaLenVar>]	求用户名长度
0040210C	. 8D8D 44FFFFFF	lea ecx,dword ptr ss:[ebp-0xBC]	
00402112	. 50	push eax	eax=用户名长度
00402113	. 8D95 E8FEFFFF	lea edx,dword ptr ss:[ebp-0x118]	
00402119	. 51	push ecx	Start8 = 0019F1AC
0040211A	. 8D85 F8FEFFFF	lea eax,dword ptr ss:[ebp-0x108]	
00402120	. 52	push edx	TMPend8 = 0019F0B8
00402121	. 8D4D DC	lea ecx,dword ptr ss:[ebp-0x24]	
00402124	. 50	push eax	TMPstep8 = 0019F0C8
00402125	. 51	push ecx	Counter8 = 0019F1AC
00402126	. FF15 20414000	call dword ptr ds:[<MSVBVM50. __vbaVarForInit>]	初始化
ds:[00404120]=741C3395 (MSVBVM50. __vbaVarForInit)			

接着以字符串的长度为循环次开始循环。

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00402164	. 8D4D A8	lea ecx,dword ptr ss:[ebp-0x58]		EAX 0057E314 UNICODE "G"
00402167	. 50	push eax		ECX 0019F178
00402168	. 51	push ecx		EDX 0019F178
00402169	. FF15 70414000	call dword ptr ds:[&MSVBVM50. _vbaStrVarVal]	取出用户名的每一位	EBX 00000002
0040216F	. 50	push eax	String = "G"	ESP 0019F098
00402170	. FF15 0C414000	call dword ptr ds:[&MSVBVM50. #rtcAnsiValueBstr_516]	计算用户名每一位的ASCII	EBP 0019F1D0
00402176	. 66:8985 4CFF	mov word ptr ss:[ebp-0xB4],ax	[ebp-0xB4]=username[i]	ESI 741C0E23 MSVBVM50. _v
0040217D	. 8D55 CC	lea edx,dword ptr ss:[ebp-0x34]		EDI 741C182B MSVBVM50. _v
00402180	. 8D85 44FFFFFF	lea eax,dword ptr ss:[ebp-0xBC]		EIP 0040216F Andrena. 0040
00402186	. 52	push edx		C 0 ES 002B 32位 0(FFFFF)
00402187	. 8D8D 74FFFFFF	lea ecx,dword ptr ss:[ebp-0x8C]		P 1 CS 0023 32位 0(FFFFF)
0040218D	. 50	push eax		A 0 SS 002B 32位 0(FFFFF)
ds:[0040410C]=740FC89B (MSVBVM50. rtcAnsiValueBstr)				
地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00402164	. 8D4D A8	lea ecx,dword ptr ss:[ebp-0x58]		EAX 0057E314
00402167	. 50	push eax		ECX 0019F178
00402168	. 51	push ecx		EDX 0019F178
00402169	. FF15 70414000	call dword ptr ds:[&MSVBVM50. _vbaStrVarVal]	取出用户名的每一位	EBX 00000002
0040216F	. 50	push eax	String = "G"	ESP 0019F098
00402170	. FF15 0C414000	call dword ptr ds:[&MSVBVM50. #rtcAnsiValueBstr_516]	计算用户名每一位的ASCII	EBP 0019F1D0
00402176	. 66:8985 4CFF	mov word ptr ss:[ebp-0xB4],ax	[ebp-0xB4]=username[i]	ESI 741C0E23 MSVBVM50. _v
0040217D	. 8D55 CC	lea edx,dword ptr ss:[ebp-0x34]		EDI 741C182B MSVBVM50. _v
00402180	. 8D85 44FFFFFF	lea eax,dword ptr ss:[ebp-0xBC]		EIP 0040216F Andrena. 0040
00402186	. 52	push edx		C 0 ES 002B 32位 0(FFFFF)
00402187	. 8D8D 74FFFFFF	lea ecx,dword ptr ss:[ebp-0x8C]		P 1 CS 0023 32位 0(FFFFF)
0040218D	. 50	push eax		A 0 SS 002B 32位 0(FFFFF)
ds:[0040410C]=740FC89B (MSVBVM50. rtcAnsiValueBstr)				

首先取出用户名的每一位，然后计算每一位的ASCII值，



地址	HEX	数据	反汇编	注释
00402186	.	52	push edx	var18 = 0019F19C
00402187	.	8D8D 74FFFFFF	lea ecx,dword ptr ss:[ebp-0x8C]	
0040218D	.	50	push eax	var28 = 0019F114
0040218E	.	51	push ecx	saveto8 = 0019F144
0040218F	.	899D 44FFFFFF	mov dword ptr ss:[ebp-0xBC],ebx	
00402195	.	FF15 94414000	call dword ptr ds:[<&MSVBVM50. __vbaVarAdd>]	username[i]+username[i-1]
0040219B	.	8BD0	mov edx,eax	结果保存到edx
0040219D	.	8D4D CC	lea ecx,dword ptr ss:[ebp-0x34]	
004021A0	.	FFD6	call esi	通过vbaVarMove将结果保存到[ebp-0x34]
004021A2	.	8D4D A8	lea ecx,dword ptr ss:[ebp-0x58]	
004021A5	.	FF15 B8414000	call dword ptr ds:[<&MSVBVM50. __vbaFreeStr>]	释放字符串
004021AB	.	8D55 84	lea edx,dword ptr ss:[ebp-0x7C]	

ds:[00404194]=741C0ECC (MSVBVM50. \_\_vbaVarAdd)

然后将用户名的ASCII值和前一位的ASCII值相加得出结果，再用vbaVarMove这个函数将结果保存到[ebp-0x34]这个地址。想要动态的看到所有的过程可以用上面的方法观察堆栈参数，找到首地址+0x8的位置详细分析。

地址	HEX	数据	反汇编	注释
004021CA	.	50	push eax	Counter8 = NULL
004021CB	.	FF15 AC414000	call dword ptr ds:[<&MSVBVM50. __vbaVarForNext	__vbaVarForNext
004021D1	^	E9 5CFFFFFF	jmp Andrena.00402132	
004021D6	>	8D4D CC	lea ecx,dword ptr ss:[ebp-0x34]	ecx=用户名的ASCII之和=0x2C4
004021D9	.	8D95 54FFFFFF	lea edx,dword ptr ss:[ebp-0xAC]	edx=1
004021DF	.	51	push ecx	var18 = 0019F19C
004021E0	.	8D45 94	lea eax,dword ptr ss:[ebp-0x6C]	
004021E3	.	52	push edx	var28 = 0000000F
004021E4	.	50	push eax	SaveTo8 = NULL
004021E5	.	C785 5CFFFFFF	mov dword ptr ss:[ebp-0xA4],0x499602D2	
004021EF	.	C785 54FFFFFF	mov dword ptr ss:[ebp-0xAC],0x3	
004021F9	.	FF15 5C414000	call dword ptr ds:[<&MSVBVM50. __vbaVarMul>]	1234567890*用户名ASCII之和

堆栈地址=0019F124  
edx=0000000F

地址	HEX	数据	UNICODE	地址	数值	注释
0019F19C	02 00 00 00	66 1D F0 00	C4 02 00 00	00 00 00 00	0019F098	0019F1DC
0019F1AC	03 00 87 02	01 00 00 00	08 00 00 00	01 00 00 00	0019F09C	0019F2B8
0019F1BC	88 F7 19 00	26 10 40 00	98 F0 19 00	00 10 40 00	0019F0A0	0287D2C4
0019F1CC	01 00 00 00	DC F1 19 00	A9 E5 0D 74	98 65 58 00	0019F0A4	FFFFFFFF
0019F1DC	EC F1 19 00	24 1A 40 00	E0 65 58 00	00 00 00 00	0019F0A8	0000137C

直接跳过这轮循环在[ebp-0x34]的地址处查看最后的结果为0x2C4。

地址	HEX	数据	反汇编	注释	寄存器
004021E0	.	8D45 94	lea eax,dword ptr ss:[ebp-0x6C]		EAX 00
004021E3	.	52	push edx	var28 = 0019F124	ECX 00
004021E4	.	50	push eax	SaveTo8 = 0019F164	EDX 00
004021E5	.	C785 5CFFFFFF	mov dword ptr ss:[ebp-0xA4],0x499602D2		EBX 00
004021EF	.	C785 54FFFFFF	mov dword ptr ss:[ebp-0xAC],0x3		ESP 00
004021F9	.	FF15 5C414000	call dword ptr ds:[<&MSVBVM50. __vbaVarMul>]	1234567890*用户名ASCII之和	EBP 00
004021FF	.	8BD0	mov edx,eax		ESI 74
00402201	.	8D4D CC	lea ecx,dword ptr ss:[ebp-0x34]		EDI 74
00402204	.	FFD6	call esi	通过vbaVarMove将结果保存到[ebp-0x34]	EIP 00
00402206	.	8B1D A0414000	mov ebx,dword ptr ds:[<&MSVBVM50. __vbaMidStmntVar>]	MSVBVM50. __vbaMidStmntVar	
0040220C	.	8D4D CC	lea ecx,dword ptr ss:[ebp-0x34]		C 0 F
0040220F	.	51	push ecx		P 1 C

堆栈地址=0019F19C  
ecx=0019F19C

地址	HEX	数据	UNICODE	地址	数值	注释
0019F124	03 00 00 00	01 00 00 00	D2 02 96 49	66 1D F0 00	0019F08C	0019F164
0019F134	00 00 00 00	00 00 00 00	01 00 00 00	5F 11 04 A4	0019F090	0019F124
0019F144	00 00 00 00	66 1D F0 00	C4 02 00 00	00 00 00 00	0019F094	0019F19C
0019F154	00 00 00 00	E2 34 1C 74	14 E3 57 00	B8 F0 19 00	0019F098	0019F1DC
0019F164	00 00 00 00	00 00 00 00	01 00 00 00	0E 00 00 00	0019F09C	0019F2B8
0019F174	00 00 00 00	00 00 00 00	00 00 00 00	FC 31 50 75	0019F0A0	0287D2C4

接着用vbaVarMul将用户名ASCII的结果和0x499602D2相乘，也就是十进制的1234567890，然后再次通过vbaVarMove将计算结果保存到[ebp-0x34]。

吾爱破解 - Andr  na.2.exe - [LCG - 主线程, 模块 - Andr  na]

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
00402210	. 6A 04	push 0x4	第四位
00402212	. 8D95 54FFFFFF	lea edx,dword ptr ss:[ebp-0xAC]	
00402218	. 6A 01	push 0x1	
0040221A	. 52	push edx	
0040221B	. C785 5CFFFFFF	mov dword ptr ss:[ebp-0xA4],Andr��na.00401C34	-
00402225	. C785 54FFFFFF	mov dword ptr ss:[ebp-0xAC],0x8	
0040222F	. FFD3	call ebx	用-替换第四位; <&MSVBVM50. __vbaFreeObj>
00402231	. 8D45 CC	lea eax,dword ptr ss:[ebp-0x34]	
00402234	. 8D8D 54FFFFFF	lea ecx,dword ptr ss:[ebp-0xAC]	
0040223A	. 50	push eax	
0040223B	. 6A 09	push 0x9	
0040223D	. 6A 01	push 0x1	

堆栈地址=0019F19C  
eax=00401C00 (Andr  na.00401C00), ASCII "ck"

地址	HEX 数据	反汇编	注释
0057E314	38 00 37 00 34 00 2D 00 37 00 34 00 30 00 36 00	874-7406	
0057E324	36 00 31 00 32 00 30 00 00 00 57 00 CB AC 8A 87	6120.W 螭	
0057E334	00 1F 00 8E 4D 00 53 00 56 00 43 00 52 00 54 00	螭MSVCRT	
0057E344	34 00 30 00 2E 00 64 00 6C 00 6C 00 00 00 00 00	40.dll..	
0057E354	00 00 00 00 C6 AC 8F 87 00 20 00 8E 4F 00 4C 00	..螭 螭	
0057E364	45 00 41 00 55 00 54 00 33 00 32 00 2E 00 64 00	EAUT32.d	
0057E374	6C 00 6C 00 00 00 00 00 00 00 00 00 DD AC 80 87	11....螭	
0057E384	00 21 00 88 B8 E5 57 00 28 D0 57 00 B0 E3 57 00	螭 W 螭	

计算器

程序员

874,074,066,120

HEX CB 82DF CCC8

DEC 874,074,066,120

OCT 14 560 267 746 310

BIN 1100 1011 1000 0010 1101 1111 1100 1100

QWORD MS M

Lsh Rsh Or Xor Not And

然后用-替换掉计算结果的第4位

吾爱破解 - Andr  na.2.exe - [LCG - 主线程, 模块 - Andr  na]

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
00402234	. 8D8D 54FFFFFF	lea ecx,dword ptr ss:[ebp-0xAC]	
0040223A	. 50	push eax	Andr��na.00401C00
0040223B	. 6A 09	push 0x9	第九位
0040223D	. 6A 01	push 0x1	
0040223F	. 51	push ecx	
00402240	. C785 5CFFFFFF	mov dword ptr ss:[ebp-0xA4],Andr��na.00401C34	-
0040224A	. C785 54FFFFFF	mov dword ptr ss:[ebp-0xAC],0x8	
00402254	. FFD3	call ebx	用-替换第九位
00402256	. 8B45 08	mov eax,dword ptr ss:[ebp+0x8]	
00402259	. 50	push eax	Andr��na.00401C00
0040225A	. 8B10	mov edx,dword ptr ds:[eax]	
0040225C	. FF92 04030000	call dword ptr ds:[edx+0x304]	

堆栈 ss:[0019F1D8]=00586598  
eax=00401C00 (Andr  na.00401C00), ASCII "ck"

地址	HEX 数据	反汇编	注释
0057E314	38 00 37 00 34 00 2D 00 37 00 34 00 30 00 36 00	874-7406	
0057E324	2D 00 31 00 32 00 30 00 00 00 57 00 CB AC 8A 87	-120.W 螭	
0057E334	00 1F 00 8E 4D 00 53 00 56 00 43 00 52 00 54 00	螭MSVCRT	

地址	数值	注释
0019F098	0019F1DC	
0019F09C	0019F2B8	
0019F0A0	0287D2C4	

和结果的第9位

吾爱破解 - Andr  na.2.exe - [LCG - 主线程, 模块 - Andr  na]

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
00402298	. 8945 9C	mov dword ptr ss:[ebp-0x64],eax	
0040229B	. 8D45 94	lea eax,dword ptr ss:[ebp-0x6C]	
0040229E	. 50	push eax	
0040229F	. 51	push ecx	
004022A0	. C745 A8 0000	mov dword ptr ss:[ebp-0x58],0x0	
004022A7	. C745 94 0880	mov dword ptr ss:[ebp-0x6C],0x8008	
004022AE	. FF15 48414000	call dword ptr ds:[<&MSVBVM50. __vbaVarTstEq>]	将最后计算出的密码和原先的密码进行比较
004022B4	. 8D4D A4	lea ecx,dword ptr ss:[ebp-0x5C]	
004022B7	. 8BD8	mov ebx,ecx	
004022B9	. FF15 B4414000	call dword ptr ds:[<&MSVBVM50. __vbaFreeObj>]	MSVBVM50. __vbaFreeObj

ds:[00404148]=741CB99E (MSVBVM50. \_\_vbaVarTstEq)

地址	HEX 数据	反汇编	注释
0057E314	38 00 37 00 34 00 2D 00 37 00 34 00 30 00 36 00	874-7406	
0057E324	2D 00 31 00 32 00 30 00 00 00 57 00 CB AC 8A 87	-120.W 螭	
0057E334	00 1F 00 8E 4D 00 53 00 56 00 43 00 52 00 54 00	螭MSVCRT	
0057E344	34 00 30 00 2E 00 64 00 6C 00 6C 00 00 00 00 00	40.dll..	
0057E354	00 00 00 00 C6 AC 8F 87 00 20 00 8E 4F 00 4C 00	..螭 螭	
0057E364	45 00 41 00 55 00 54 00 33 00 32 00 2E 00 64 00	EAUT32.d	

地址	数值	注释
0019F090	0019F19C	var28 = 0019F19C
0019F094	0019F164	var18 = 0019F164
0019F098	0019F1DC	
0019F09C	0019F2B8	
0019F0A0	0287D2C4	
0019F0A4	FFFFFFFF	

接着将最后计算的结果和原先输入的密码做比较，用堆栈的参数和VB的数据结构特点可以看到正确的注册码和原先输入的密码。

到这里，这个注册算法就分析完成了

## 写出注册机

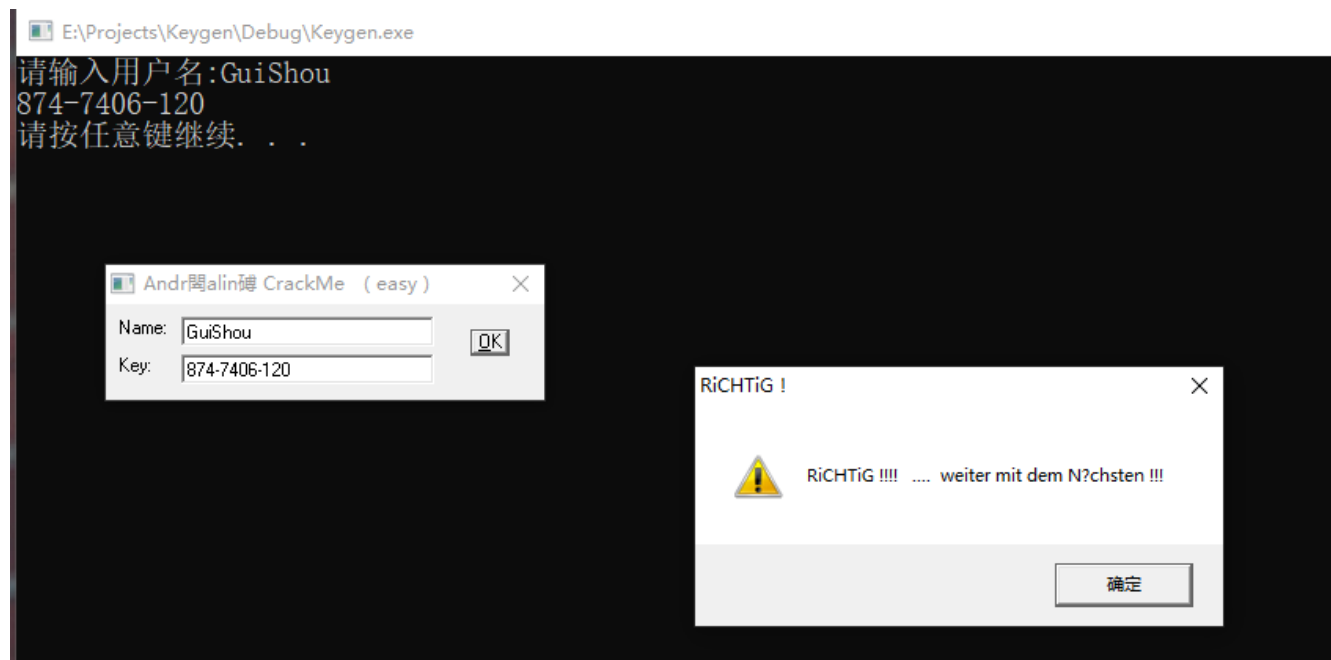
```
int CalcKey()
{
    char szUserName[20] = { 0 };
    __int64 result = 0;
    char key[50] = { 0 };
    printf("请输入用户名:");
    scanf_s("%s", szUserName, 20);

    int iUserNameLen = strlen(szUserName);
    //求出每一位用户名的和
    for (int i = 0; i < iUserNameLen; i++)
    {
        result += szUserName[i];
    }

    result *= 1234567890;
    //转为字符串
    sprintf(key, "%I64d", result);
    //替换第4位和第9位
    key[3] = '-';
    key[8] = '-';

    printf("%s\n", key);
    return 0;
}
```

## 验证结果



输入用户名，将计算的序列号填入。OK显示正常，这个Crackme也就完成了。

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