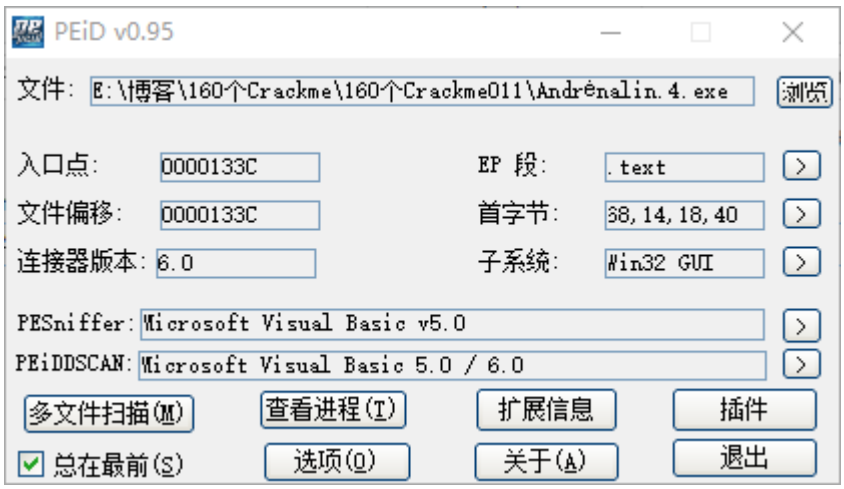
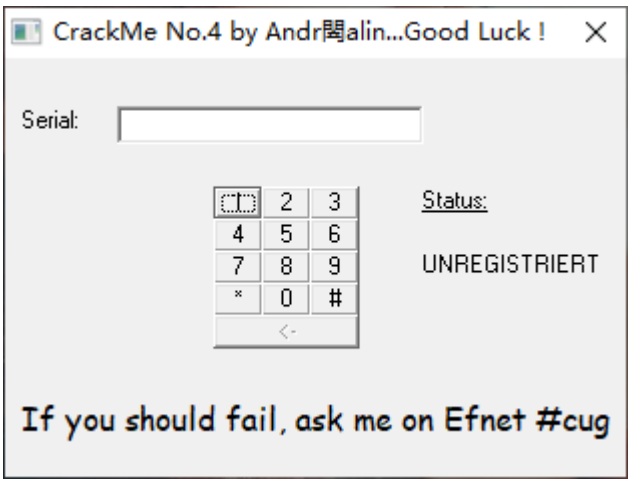
查壳



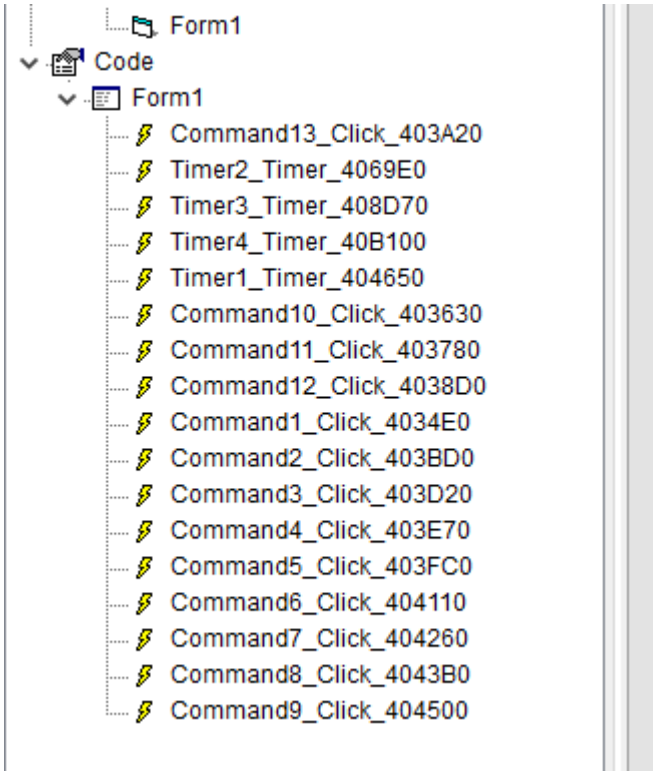
目标程序是使用VB写的，和前面三个crackme是同一个作者

分析思路寻找突破口



这个就是这次的目标程序，只提供了一排输入按键和右边的提示，没有确定按钮，那么猜测这个程序校验序列号的方式应该有两种，一种是通过Serial编辑框的变化事件来判断，一种是通过定时器来判断。

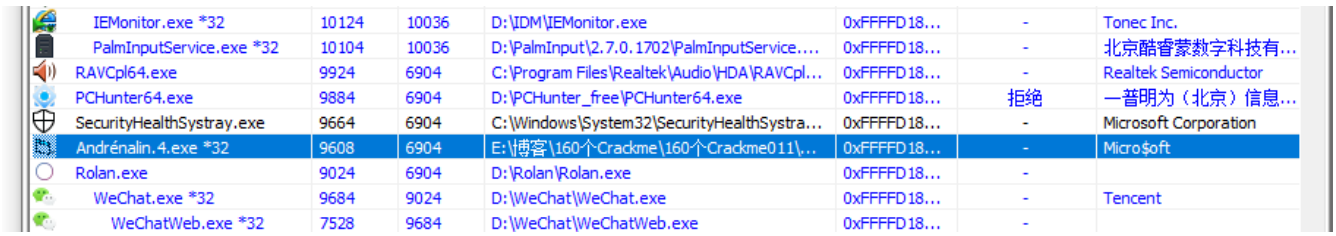
然后再用VB Decompiler来看一下有哪些事件。



从事件的部分可以得出这个程序有四个定时器和12个按钮的点击事件，按钮的点击事件应该就是输入相应的字符没什么其他作用，重点就在定时器上面。

既然有定时器，那么不妨再确认一下

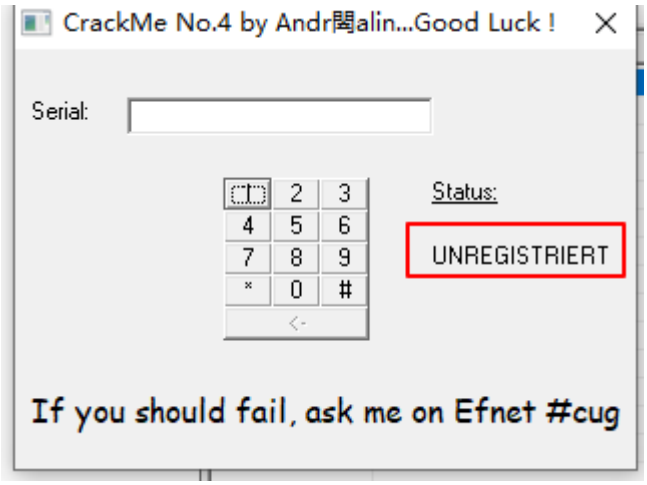
打开PC Hnuter，选中目标进程



右键查看->查看进程定时器

有五个定时器，比之前的多了一个。

那么突破口就出来了，



首先查找这个字符串，然后根据字符串找到函数头的位置，对比定时器的RVA，再进行分析

OD分析程序

接着载入OD，F9运行，搜索字符串

地址	反汇编	文本字符串
00404A6B	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747D7AFF7C7F82836D74RR7A7F7E7B7C7D826D81KE7B7C
00404AA3	push Andréna.00401EBC	REGISTRIERT
00404C88	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747D7A7D7C7F82836D74747A7F7E7G7C7D826D817E7B7C
00404CC0	push Andréna.00401EBC	REGISTRIERT
00404EA5	mov dword ptr ss:[ebp-0xAC],Andréna.004(	Q817E747D7AFF7C7F82836D74RR7A7F7E7B7C7D826D81KEZB7C
00404EDD	push Andréna.00401EBC	REGISTRIERT
004050C2	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747D7AFP7C7F82836D74RR7A7F7E7B7C7D826D81KE7B7C
004050FA	push Andréna.00401EBC	REGISTRIERT
004052DF	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747G7AFF7C7F82836D74RR7A7F7E7B7C7D826D81KE7B7C
00405317	push Andréna.00401EBC	REGISTRIERT
004054FC	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747DüAFF7C7F82836D74RR7A7F7E7B7C7D826D81KE7B7C
00405534	push Andréna.00401EBC	REGISTRIERT
00405719	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E7WOD7AFF7C7F82836D74RR7A7F7E7B7C7D826D81KE7B7C
00405751	push Andréna.00401EBC	REGISTRIERT
00405936	mov dword ptr ss:[ebp-0xAC],Andréna.004(	http://beam.to/cugABCDEFGHIJKLMNOPQRSTUVWXYZ1234567
0040596E	push Andréna.00401EBC	REGISTRIERT
00405B53	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747D7AFF7C7F82836D74RR7A7F7E7B7C7D826D8KKE7B7C
00405B8B	push Andréna.00401EBC	REGISTRIERT
00405D70	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747$7AFF7C7F82836D74RR7A7F7E7B7C7D826D81KE7B7C
00405DA8	push Andréna.00401EBC	REGISTRIERT
00405F8D	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747#7AFF7C7F82836D74RR7A7F7E7B7C7D826D81KE7B7C
00405FC5	push Andréna.00401EBC	REGISTRIERT
004061AA	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747D7AFFF7C7F82836D74RR7A7F7E7B7C7D826D81KE7B7C
004061E2	push Andréna.00401EBC	REGISTRIERT
004063C7	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747G7AFF7C7F82836D74RR7A7F7E7B7C7D826D81KE7B7C
004063FF	push Andréna.00401EBC	REGISTRIERT
004065E4	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747D7A7D7C7F82836D74747A7F7E7B7C7D826D817E7B7C
0040661C	push Andréna.00401EBC	REGISTRIERT
00406801	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747D7A7D7C7F82836D74747A7F7E7B7C7D826D8H7E7B7C
00406839	push Andréna.00401EBC	REGISTRIERT
00406DFB	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747D7AFF7C7F82836D74RR7A7F7E7B7C7D826D81KE7B7C
00406E33	push Andréna.00401EBC	REGISTRIERT
00407018	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747D7A7D7C7F82836D74747A7F7E7G7C7D826D817E7B7C
00407050	push Andréna.00401EBC	REGISTRIERT
00407235	mov dword ptr ss:[ebp-0xAC],Andréna.004(	Q817E747D7AFF7C7F82836D74RR7A7F7E7B7C7D826D81KEZB7C
0040726D	push Andréna.00401EBC	REGISTRIERT
00407452	mov dword ptr ss:[ebp-0xAC],Andréna.004(	0817E747D7AFP7C7F82836D74RR7A7F7E7B7C7D826D81KE7B7C

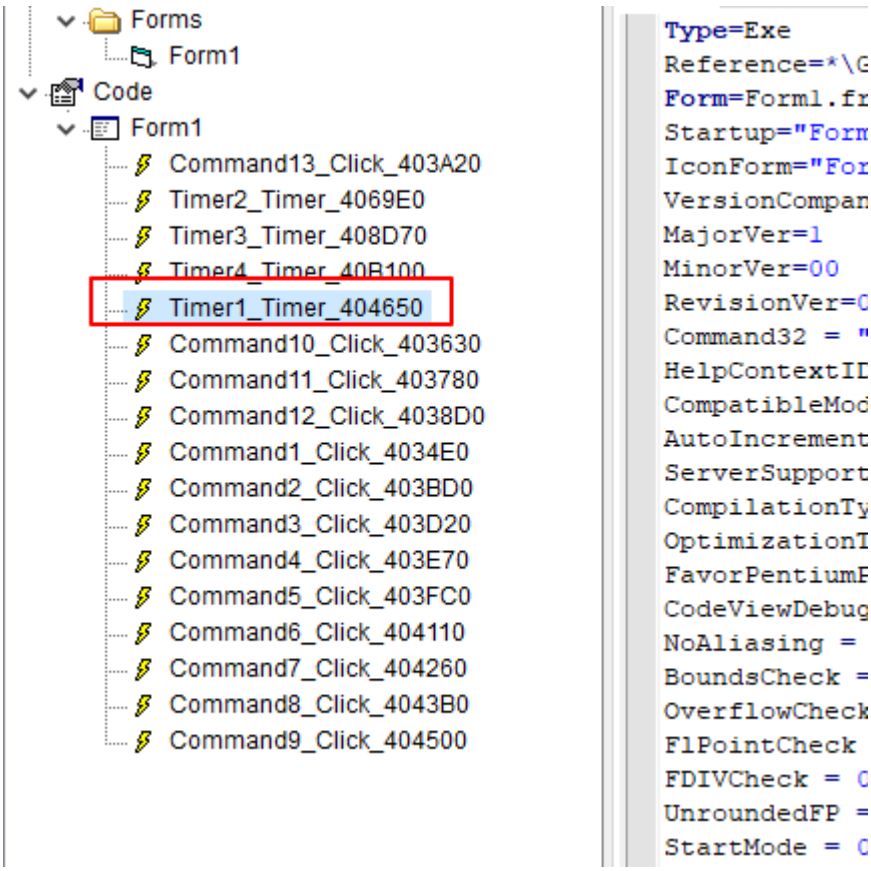
虽然没找到预期的 UNREGISTRIERT，但是找到了对应的REGISTRIERT，这个应该是注册成功的提示信息，看这样子这个源代码好像是复制粘贴了好几份，用来混淆的，随便点一个进去，拉到函数头的位置，

暂停

地址	HEX 数据	反汇编	注释
0040464E	90	nop	
0040464F	90	nop	
00404650	55	push ebp	定时器事件
00404651	8BEC	mov ebp,esp	
00404653	83EC 0C	sub esp,0xC	
00404656	68 F6114000	push <jmp.&MSVBVM60.__vbaExceptHandler>	SE 处理程序安装
0040465B	64:A1 0000000(	mov eax,dword ptr fs:[0]	
00404661	50	push eax	Andréna.004033A5
00404662	64:8925 0000(	mov dword ptr fs:[0],esp	
00404669	81EC A0030000(	sub esp,0x3A0	
0040466F	53	push ebx	
00404670	56	push esi	
00404671	57	push edi	
00404672	8965 F4	mov dword ptr ss:[ebp-0xC],esp	
00404675	C745 F8 B011(	mov dword ptr ss:[ebp-0x8],Andréna.0040	
0040467C	8B7D 08	mov edi,dword ptr ss:[ebp+0x8]	
0040467F	8BC7	mov eax,edi	
00404681	83E0 01	and eax,0x1	
00404684	8945 FC	mov dword ptr ss:[ebp-0x4],eax	Andréna.004033A5

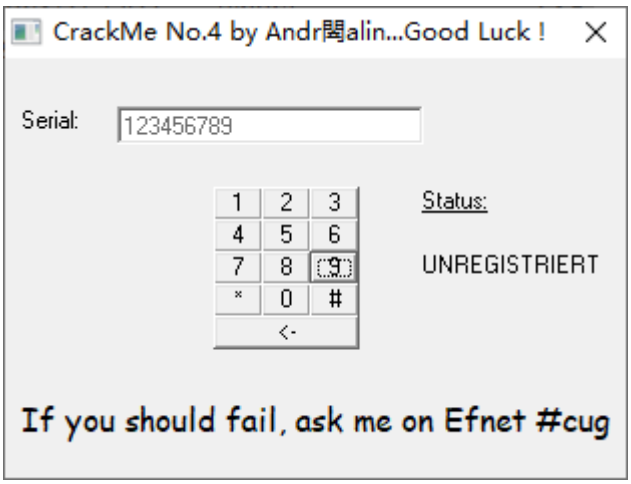
ebp=0019FAAC

下断点之后马上就断下来了，然后对比一下VB Decomper的定时器事件

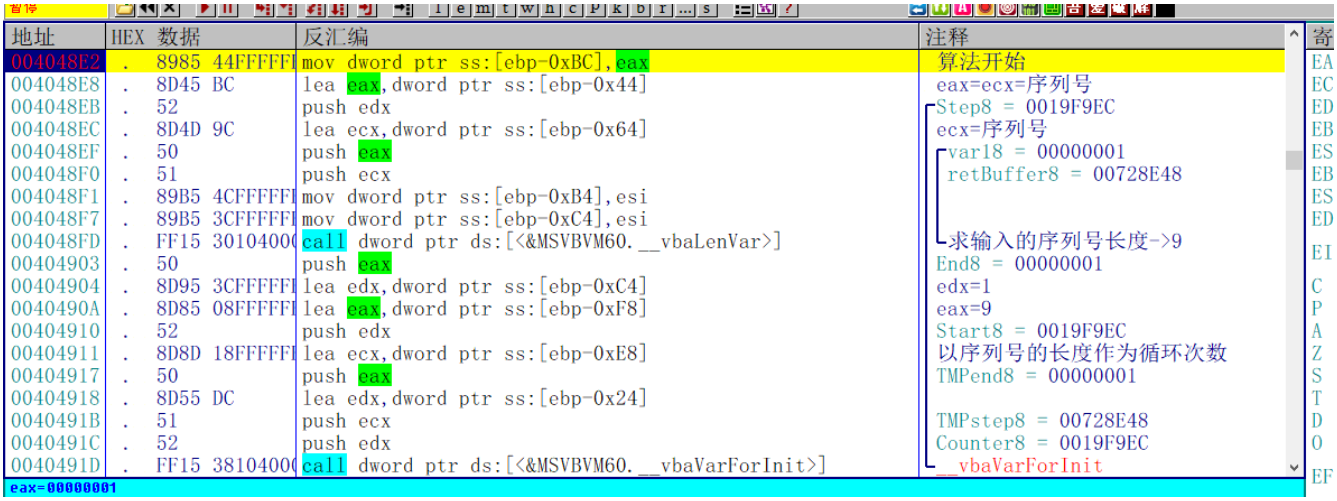


果然就是第一个定时器的回调函数，

分析核心算法

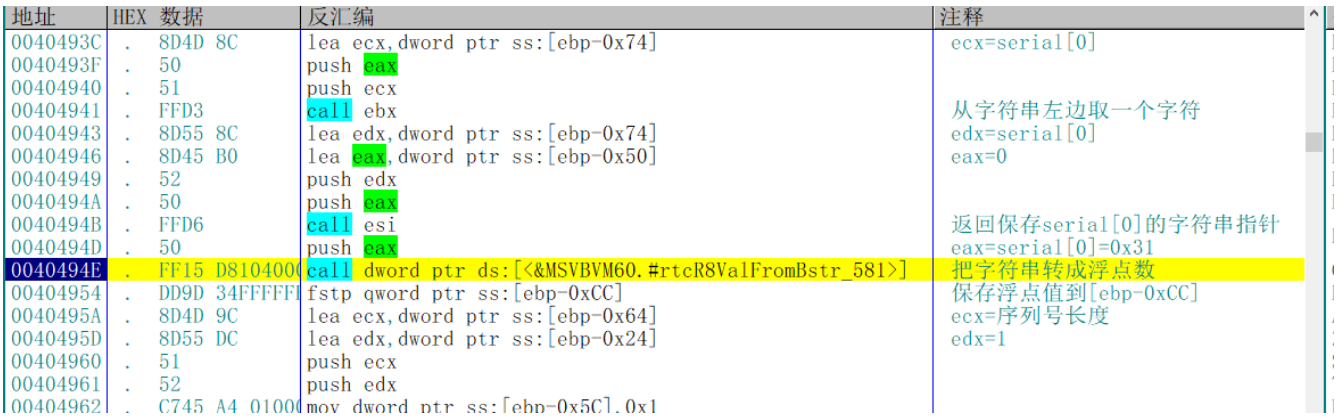


输入123456789，开始分析整个函数的核心算法



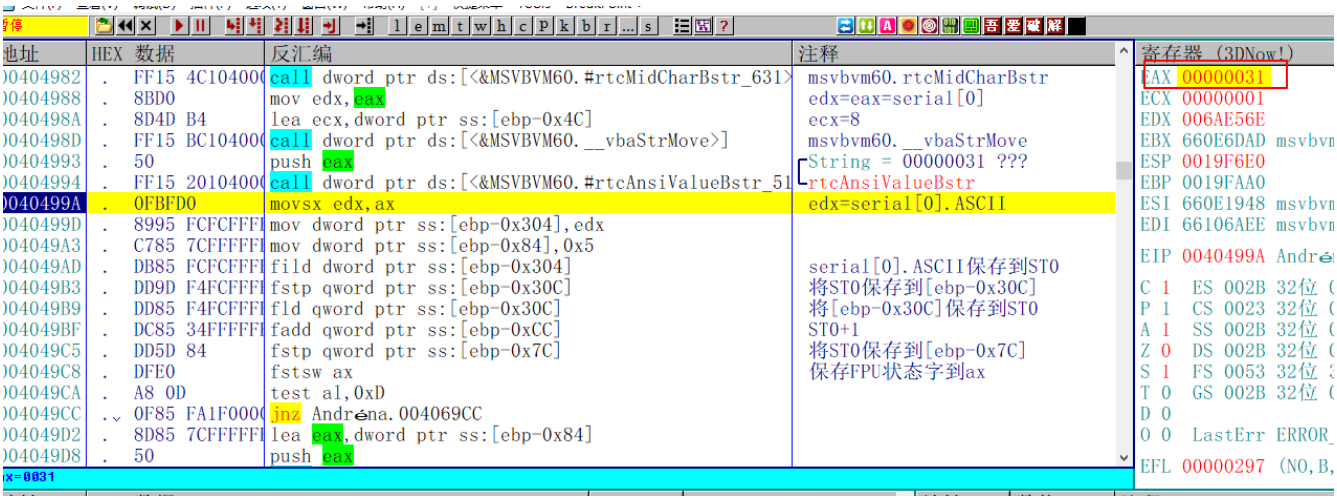
```
地址      HEX 数据      反汇编                                                  注释
004048E2  .   8985 44FFFFFF mov dword ptr ss:[ebp-0xBC],eax                    算法开始
004048E8  .   8D45 BC       lea eax,dword ptr ss:[ebp-0x44]                   eax=ecx=序列号
004048EB  .   52            push edx                                          Step8 = 0019F9EC
004048EC  .   8D4D 9C       lea ecx,dword ptr ss:[ebp-0x64]                   ecx=序列号
004048EF  .   50            push eax                                          var18 = 00000001
004048F0  .   51            push ecx                                          retBuffer8 = 00728E48
004048F1  .   89B5 4CFFFFFF mov dword ptr ss:[ebp-0xB4],esi
004048F7  .   89B5 3CFFFFFF mov dword ptr ss:[ebp-0xC4],esi
004048FD  .   FF15 30104000 call dword ptr ds:[<&MSVBVM60.__vbaLenVar>]       求输入的序列号长度->9
00404903  .   50            push eax                                          End8 = 00000001
00404904  .   8D95 3CFFFFFF lea edx,dword ptr ss:[ebp-0xC4]                   edx=1
0040490A  .   8D85 08FFFFFF lea eax,dword ptr ss:[ebp-0xF8]                   eax=9
00404910  .   52            push edx                                          Start8 = 0019F9EC
00404911  .   8D8D 18FFFFFF lea ecx,dword ptr ss:[ebp-0xE8]                   以序列号的长度作为循环次数
00404917  .   50            push eax                                          TMPend8 = 00000001
00404918  .   8D55 DC       lea edx,dword ptr ss:[ebp-0x24]
0040491B  .   51            push ecx                                          TMPstep8 = 00728E48
0040491C  .   52            push edx                                          Counter8 = 0019F9EC
0040491D  .   FF15 38104000 call dword ptr ds:[<&MSVBVM60.__vbaVarForInit>]   __vbaVarForInit
eax=00000001
```

首先是获取序列号的长度，然后将序列号的长度作为循环的次数，开始算法的循环



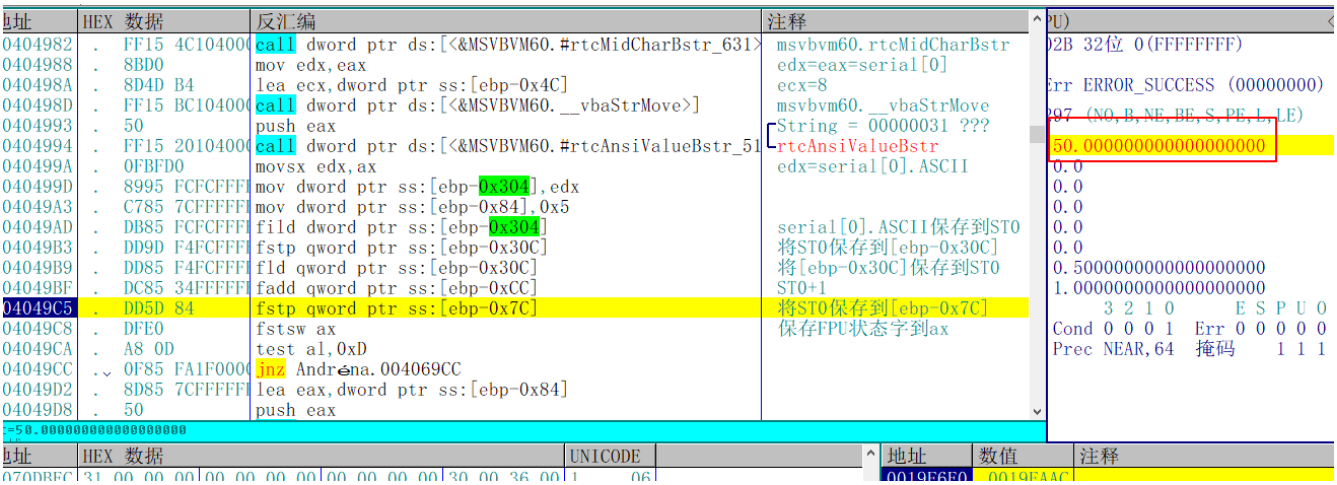
```
地址      HEX 数据      反汇编                                                  注释
0040493C  .   8D4D 8C       lea ecx,dword ptr ss:[ebp-0x74]                   ecx=serial[0]
0040493F  .   50            push eax
00404940  .   51            push ecx
00404941  .   FFD3          call ebx                                          从字符串左边取一个字符
00404943  .   8D55 8C       lea edx,dword ptr ss:[ebp-0x74]                   edx=serial[0]
00404946  .   8D45 B0       lea eax,dword ptr ss:[ebp-0x50]                   eax=0
00404949  .   52            push edx
0040494A  .   50            push eax
0040494B  .   FFD6          call esi                                          返回保存serial[0]的字符串指针
0040494D  .   50            push eax                                          eax=serial[0]=0x31
0040494E  .   FF15 D8104000 call dword ptr ds:[<&MSVBVM60.#rtcR8ValFromBstr_581>] 把字符串转成浮点数
00404954  .   DD9D 34FFFFFF fstp qword ptr ss:[ebp-0xCC]                      保存浮点值到[ebp-0xCC]
0040495A  .   8D4D 9C       lea ecx,dword ptr ss:[ebp-0x64]                   ecx=序列号长度
0040495D  .   8D55 DC       lea edx,dword ptr ss:[ebp-0x24]                   edx=1
00404960  .   51            push ecx
00404961  .   52            push edx
00404962  .   C745 A4 01000 mov dword ptr ss:[ebp-0x5C],0x1
```

然后从序列号左边取一个字符，然后转成浮点数，保存到[ebp-0xCC]。

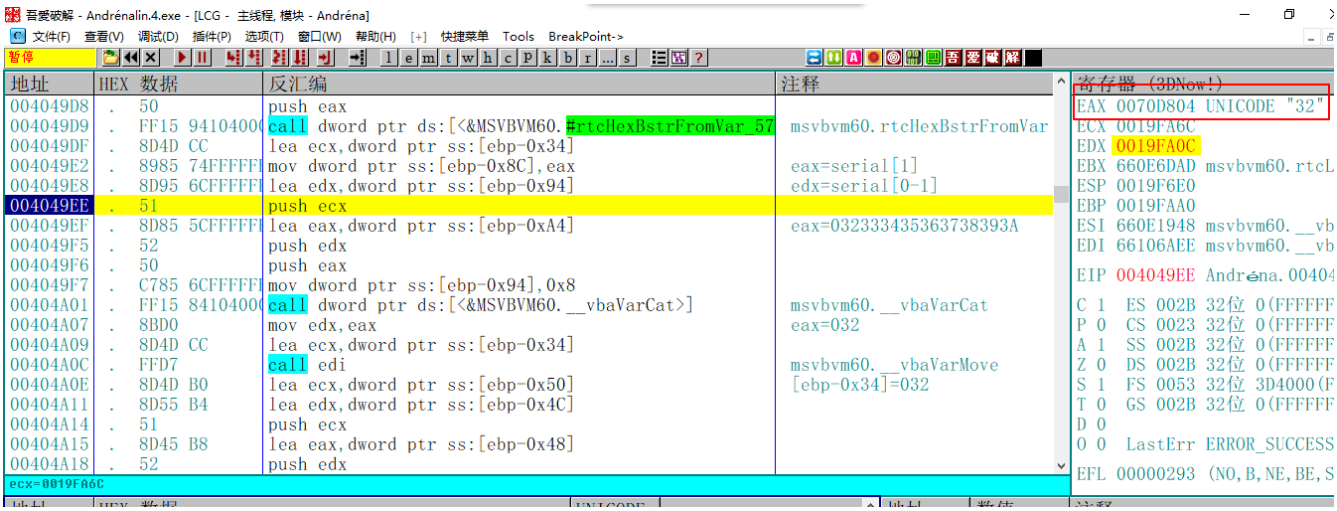


```
地址      HEX 数据      反汇编                                                  注释                          寄存器 (3DNow!)
00404982  .   FF15 4C104000 call dword ptr ds:[<&MSVBVM60.#rtcMidCharBstr_631> msvbvm60.rtcMidCharBstr        EAX 00000031
00404988  .   8BD0          mov edx,eax                                       edx=eax=serial[0]             ECX 00000001
0040498A  .   8D4D B4       lea ecx,dword ptr ss:[ebp-0x4C]                   ecx=8                         EDX 006AE56E
0040498D  .   FF15 BC104000 call dword ptr ds:[<&MSVBVM60.__vbaStrMove>]      msvbvm60.__vbaStrMove         EBX 660E6DAD msvbvm
00404993  .   50            push eax                                          String = 00000031 ???         ESP 0019F6E0
00404994  .   FF15 20104000 call dword ptr ds:[<&MSVBVM60.#rtcAnsiValueBstr_51 rtcAnsiValueBstr            EBP 0019FAA0
0040499A  .   0FBFD0        movsx edx,ax                                      edx=serial[0].ASCII           ESI 660E1948 msvbvm
0040499D  .   8995 FCFCFFFF mov dword ptr ss:[ebp-0x304],edx                                               EDI 66106AEE msvbvm
004049A3  .   C785 7CFFFFFF mov dword ptr ss:[ebp-0x84],0x5                                                EIP 0040499A Andr
004049AD  .   DB85 FCFCFFFF fild dword ptr ss:[ebp-0x304]                     serial[0].ASCII保存到ST0      C 1  ES 002B 32位
004049B3  .   DD9D F4FCFFFF fstp qword ptr ss:[ebp-0x30C]                     将ST0保存到[ebp-0x30C]        P 1  CS 0023 32位
004049B9  .   DD85 F4FCFFFF fld qword ptr ss:[ebp-0x30C]                      将[ebp-0x30C]保存到ST0        A 1  SS 002B 32位
004049BF  .   DC85 34FFFFFF fadd qword ptr ss:[ebp-0xCC]                      ST0+1                         Z 0  DS 002B 32位
004049C5  .   DD5D 84       fstp qword ptr ss:[ebp-0x7C]                      将ST0保存到[ebp-0x7C]         S 1  FS 0053 32位
004049C8  .   DFE0          fstsw ax                                          保存FPU状态字到ax             T 0  GS 002B 32位
004049CA  .   A8 0D         test al,0xD                                                                     D 0
004049CC  .v  0F85 FA1F0000 jnz Andréna.004069CC                                                            O 0  LastErr ERROR_
004049D2  .   8D85 7CFFFFFF lea eax,dword ptr ss:[ebp-0x84]                                                 EFL 00000297 (NO,B,
004049D8  .   50            push eax
x=0031
```

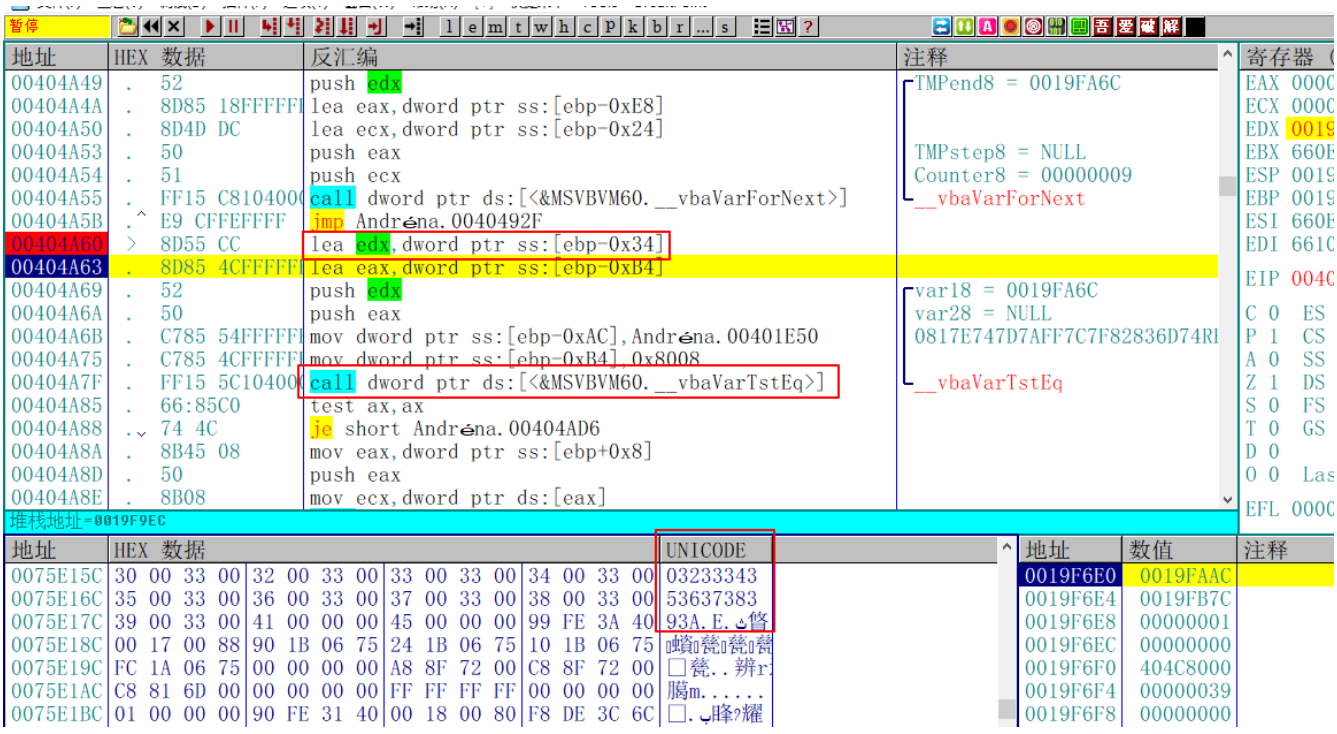
然后取出序列号第一位的ASCII值，转成浮点数，保存到ST0



然后将ST0+1之后，保存到[ebp-0x7C]的位置，50=49(1的ASCII值)+1

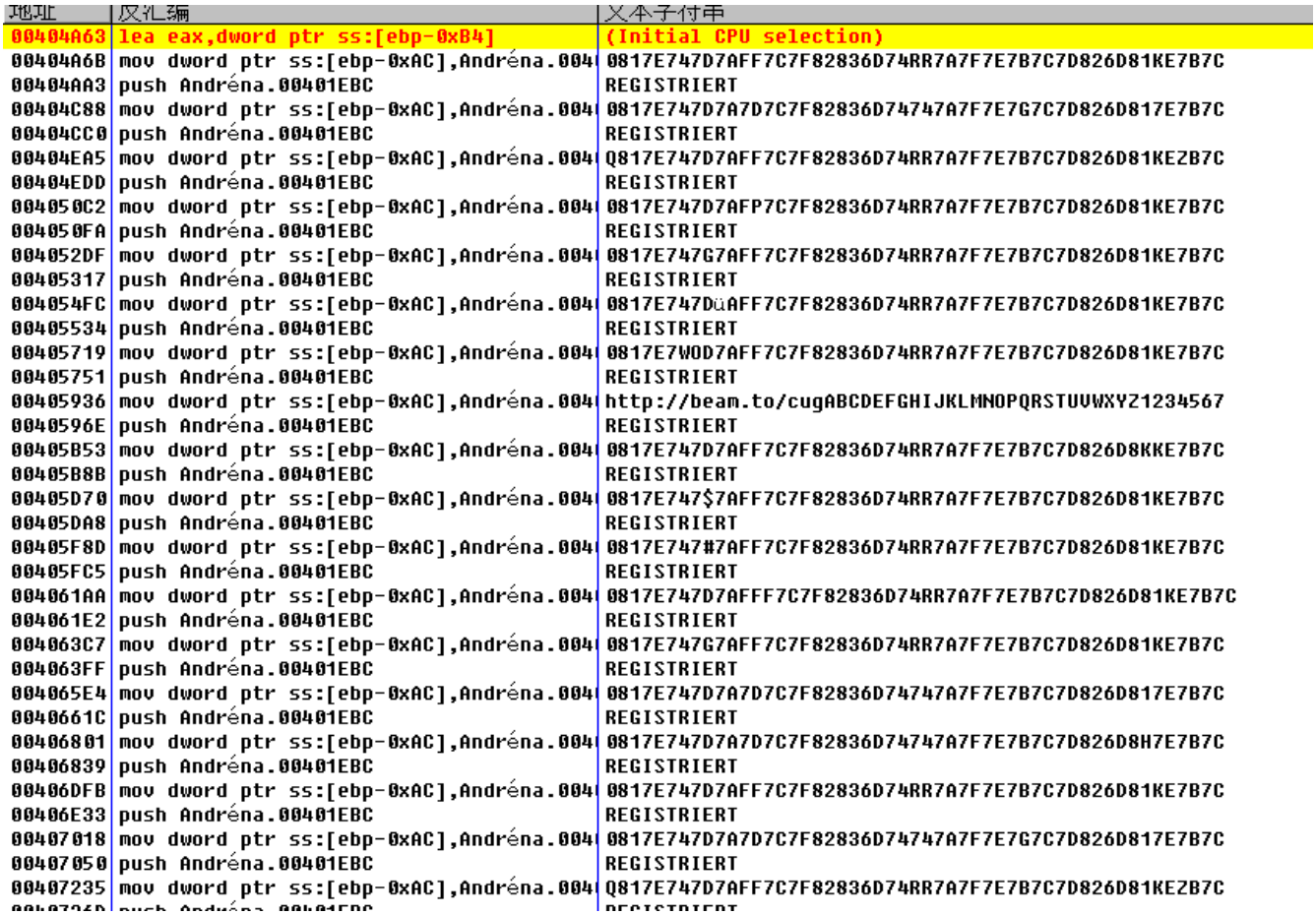


然后将结果的十六进制转成字符串——50(十进制)=32(十六进制)

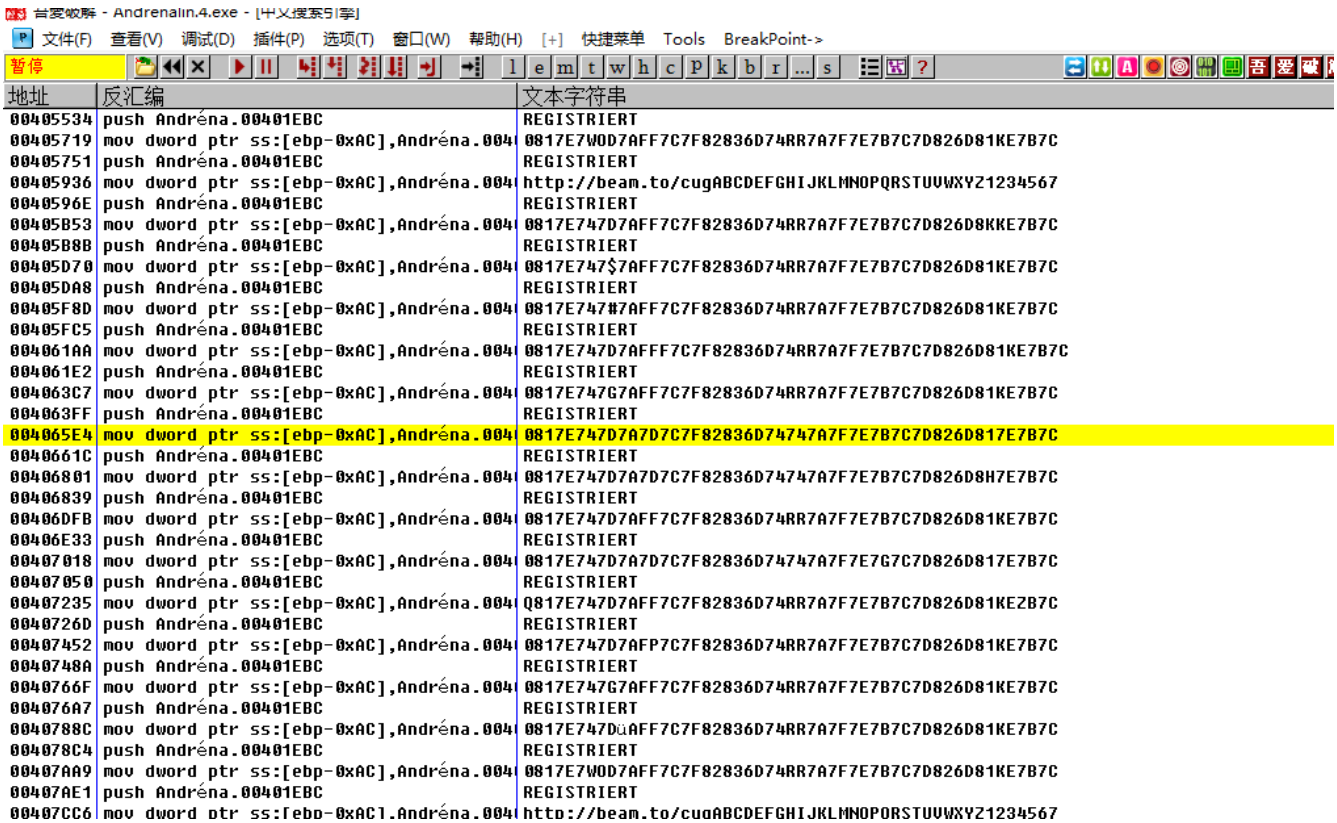


接着开始下一轮循环，循环结束之后，将最后的结果保存到[ebp-0x34]，然后用vbaVarTstEq把最终计算的字符串和硬编码的作比较。这个算法倒是异常的简单，很容易就能根据字符串逆推出序列号

再回到刚才的字符串的问题，这里为什么会有这么多相同的序列号



因为只有一个是真的，只有通过了那唯一一个真的字符串的验证才能验证成功，根据刚才的分析可以得出结论，序列号必须的0-9 A-F之间的十六进制数字，我们可以根据这个结论来找出正确的唯一解。



满足要求的就只有一个，

```
0 81 7E 74 7D 7A 7D 7C 7F 82 83 6D 74 74 7A 7F 7E 7B 7C 7D 82 6D 81 7E 7B 7C
```

这个唯一解的算法和之前我分析的一模一样，接着根据硬编码的字符串逆推出正确的序列号

强行推序列号

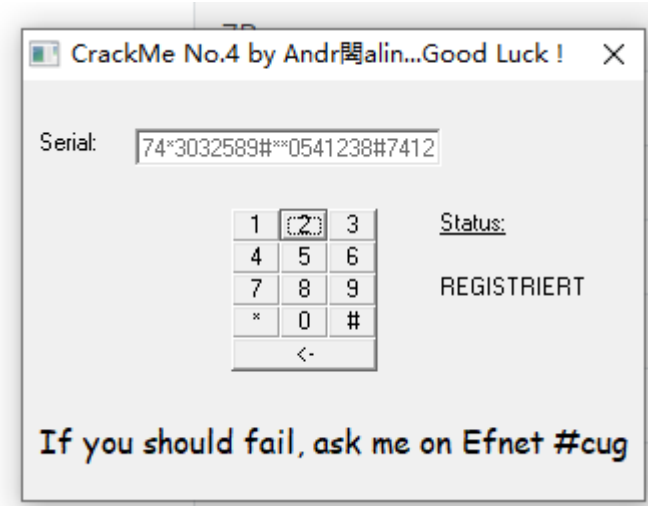
到了这里我们有意思的事情就来了，我们已经知道了正确的唯一解字符串，我还还知道一个按键对应一个字符串，所以可以我们可以根据把程序自带的12个按钮逆推出对应的Key，即使你不知道前面的算法，也可以强行逆推，只要知道一个事实

- 按键的ASCII和字符串的数值成正比，ASCII值越大，结果越大

所以我们只要把上述的正确字符串从小到大排序，然后再将程序的十二个按键的ASCII值从小到大排序，就能让字符串和按键的值一一对应。然后再根据唯一正确的字符串，输入相应的按键，就能破解程序

Key	Code
6D	*
74	#
7A	0
7B	1
7C	2
7D	3
7E	4
7F	5
80	6
81	7
82	8
83	9

所以字符串`0817E747D7A7D7C7F82836D74747A7F7E7B7C7D826D817E7B7C`对应的序列号就是`74*3032589#**0541238#7412`



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