

# HCTF 2017 bin Level1 Evr\_Q Writeup

原创

inori jam

于 2017-11-13 21:57:04 发布 605 收藏

分类专栏: [Writeup](#) 文章标签: [CTF](#) [逆向工程](#) [动态调试](#) [网络安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/atmysy/article/details/78525284>

版权



[Writeup](#) 专栏收录该内容

1 篇文章 0 订阅

订阅专栏

HCTF 2017完结撒花, 萌新领教到了赛棍们的强大, 但也有了一些自己的收获。我想将其中一些题的思路记录下来, 一是记录, 二是与有需要的人分享, 当然本篇WP和大佬们的比还是很菜。话不多说, 进入正题。

首先我们拿到一个看起来很吊的程序, 如下图:

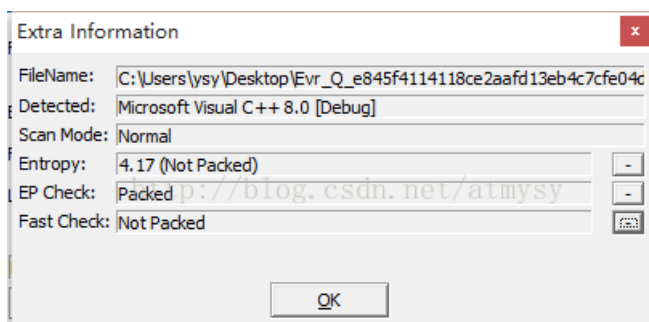
```
C:\Users\ysy\Desktop\Evr_Q_e845f4114118ce2aafd13eb4c7cfe04db6f6130cc1785d9fe5e1f35e7674faa3.exe
////////
WARNING
////////
Welcome to HCTF 2017

Mark.09 is hijacking Shinji Ikari now...

Check User:
_

http://blog.csdn.net/atmysy
```

例行公事, PEID查壳, 查到EP段有加壳, 但是不影响我们Crack



直接上神器X32-DBG, 考虑到壳的关系, 附加到程序进程, 来到主程序模块发现什么都没有

001C1000	00 00	add byte ptr ds:[eax],al
001C1002	00 00	add byte ptr ds:[eax],al
001C1004	00 00	add byte ptr ds:[eax],al
001C1006	00 00	add byte ptr ds:[eax],al
001C1008	00 00	add byte ptr ds:[eax],al
001C100A	00 00	add byte ptr ds:[eax],al
001C100C	00 00	add byte ptr ds:[eax],al
001C100E	00 00	add byte ptr ds:[eax],al
001C1010	00 00	add byte ptr ds:[eax],al
001C1012	00 00	add byte ptr ds:[eax],al
001C1014	00 00	add byte ptr ds:[eax],al
001C1016	00 00	add byte ptr ds:[eax],al
001C1018	00 00	add byte ptr ds:[eax],al
001C101A	00 00	add byte ptr ds:[eax],al
001C101C	00 00	add byte ptr ds:[eax],al
001C101E	00 00	add byte ptr ds:[eax],al
001C1020	00 00	add byte ptr ds:[eax],al
001C1022	00 00	add byte ptr ds:[eax],al
001C1024	00 00	add byte ptr ds:[eax],al
001C1026	00 00	add byte ptr ds:[eax],al
001C1028	00 00	add byte ptr ds:[eax],al
001C102A	00 00	add byte ptr ds:[eax],al
001C102C	00 00	add byte ptr ds:[eax],al
001C102E	00 00	add byte ptr ds:[eax],al
001C1030	00 00	add byte ptr ds:[eax],al
001C1032	00 00	add byte ptr ds:[eax],al
001C1034	00 00	add byte ptr ds:[eax],al
001C1036	00 00	add byte ptr ds:[eax],al
001C1038	00 00	add byte ptr ds:[eax],al
001C103A	00 00	add byte ptr ds:[eax],al
001C103C	00 00	add byte ptr ds:[eax],al
001C103E	00 00	add byte ptr ds:[eax],al
001C1040	00 00	add byte ptr ds:[eax],al
001C1042	00 00	add byte ptr ds:[eax],al
001C1044	00 00	add byte ptr ds:[eax],al
001C1046	00 00	add byte ptr ds:[eax],al
001C1048	00 00	add byte ptr ds:[eax],al
001C104A	00 00	add byte ptr ds:[eax],al
001C104C	00 00	add byte ptr ds:[eax],al
001C104E	00 00	add byte ptr ds:[eax],al
001C1050	00 00	add byte ptr ds:[eax],al
001C1052	00 00	add byte ptr ds:[eax],al
001C1054	00 00	add byte ptr ds:[eax],al
001C1056	00 00	add byte ptr ds:[eax],al
001C1058	00 00	add byte ptr ds:[eax],al
001C105A	00 00	add byte ptr ds:[eax],al
001C105C	00 00	add byte ptr ds:[eax],al
001C105E	00 00	add byte ptr ds:[eax],al
001C1060	00 00	add byte ptr ds:[eax],al
001C1062	00 00	add byte ptr ds:[eax],al
001C1064	00 00	add byte ptr ds:[eax],al

<http://blog.csdn.net/atmysy>

## 搜一下字符串发现有这么多

The image displays a Windows XP desktop environment. The primary focus is on two windows: **x32dbg** (a debugger) and **xstring** (a string search utility). The **x32dbg** window shows a list of assembly instructions with their corresponding addresses and hex values. The **xstring** window displays a search results table with columns for address, hex code, and the string content. The strings include various system-related messages, warnings, and error messages, such as 'InQueryInformationProcess', 'WARNING', 'DETONATION FUNCTION', and 'Stack pointer corruption'. The desktop also shows a taskbar with several icons and a system tray.

找到看起来比较关键的字符串点进去，我们的老朋友：汇编代码就回来了

```
000D317D 68 44 8E 1D 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D8E44
000D3182 68 88 8D 1D 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D8D88
000D3187 3B 84 00 00 00 call_dword_ptr_ds:[<messagebox>]
000D3189 FF 15 80 C0 1D 00 cmp_es1,esp
000D318F 3B F4 00 00 00 call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D117C
000D3191 68 78 8C 1D 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D8C78
000D3196 68 57 E2 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D13F7
000D31A0 83 C4 04 add_esp,4
000D31A3 83 64 04 push_eax
000D31A5 8D 45 8B lea_eax,dword_ptr_ss:[ebp-45]
000D31A8 75 50 push_eax
000D31AB 68 54 8C 1D 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D8C54
000D31AE E8 A1 0C FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D1154
000D31B3 83 C4 04 add_esp,c
000D31B6 0F BE 45 8B movsx_eax,byte_ptr_ss:[ebp-45]
000D31BA 83 F8 59 cmp_eax,59
000D31BD 74 09 jbe_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D31C8
000D31BF 0F BE 45 8B movsx_eax,byte_ptr_ss:[ebp-45]
000D31C3 75 14 jne_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D31DC
000D31C6 E8 85 D2 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D1082
000D31C8 68 58 8C 1D 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D8C58
000D31D2 E8 20 E2 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D13F7
000D31D7 83 C4 04 add_esp,4
000D31DA 0F BE 0A push_eax
000D31DC E8 07 E1 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D1398
000D31E1 E8 19 D1 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D10FF
000D31E6 0F BE 0A push_eax
000D31F1 68 58 8C 1D 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D31F7
000D31F3 FF 15 90 C1 1D 00 call_dword_ptr_ds:[<systems>]
000D31F6 83 C4 04 add_esp,4
000D3202 3B F4 00 00 00 cmp_es1,esp
000D3204 68 33 C0 1D 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D117C
000D3209 33 C0 xor_eax,eax
000D320B 52 push_edx
000D320E 0F 69 CD mov ecx,ebp
000D3212 50 push_eax
000D3215 8D 15 3C 32 1D 00 lea_eax,dword_ptr_ds:[0323C]
000D321A 58 58 8C 1D 00 call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D1302
000D321B 5A pop_eax
000D321C 5F pop_edx
000D321D 5F pop_eax
000D321E 5B pop_esi
000D321F 8B 4D FC mov ecx,dword_ptr_ss:[ebp-4]
000D3222 5D xor ecx,ebp
000D3224 E8 F2 E0 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.1D1318
<
AcDbFCF "Prevent IMPACT success!\n"
```

接下来可以查看check user后几个关键call，发现程序将用户名倒序，再对用户名中每个ASCII字符对应的十六进制作XOR，AND等一系列运算，最后与密文比较，根据这个，我们直接就能用十六进制计算器反算出正确的用户名原文为：M.KATSURAGI。实际上直接改跳转也能使验证成功，且不会对我们get flag造成任何影响，但稳妥起见，还是按流程走吧。

```
00292E73 83 C4 04 add_esp,4
00292E76 68 E0 8C 29 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.298CE0
00292E7B E8 77 E5 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.2913F7
00292E80 83 C4 04 add_esp,4
00292E83 68 01 00 00 00 push_100
00292E88 68 F0 B1 29 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.29B1F0
00292E8D 68 38 8C 29 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.298C38
00292E92 E8 BD E2 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.291154
00292E97 83 C4 04 add_esp,c
00292E9A 68 77 E4 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.291316
00292E9F 85 C0 test_eax,eax
00292EA1 74 07 jbe_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.292EAA
00292EA3 E8 51 E3 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.2911F9
00292EA8 E8 36 8C 29 00 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.292EC0
00292EA9 68 84 F0 29 00 call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.29114A
00292EAB 6A 00 xor_esi,esp
00292EAE 75 15 jnz_test_0
00292EB1 6A 00 xor_esi,esp
00292EB3 FF 15 94 C1 29 00 call_dword_ptr_ds:[<exit>]
00292EB9 3B F4 00 00 00 cmp_es1,esp
00292EBB E8 3C 8C 29 00 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.29117C
00292EC0 68 3C 8C 29 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.298C3C
00292EC5 E8 2D E5 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.2913F7
00292ECA 83 C4 04 add_esp,4
00292ECD 68 80 80 00 00 push_80
00292ED2 E8 3C 8C 29 00 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.29B4F0
00292ED7 68 38 8C 29 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.298C38
00292EDC E8 73 E2 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.291154
00292EED 83 C4 04 add_esp,c
298CE0:"Check User: \n"
298C38:"%s"=="L"猴
298C3C:"Check Start Code: \n"
298C38:"%s"=="L"猴
```

```
00292418 48 75 44 mov_eax,word_ptr_ss:[ebp+3]
00292419 0F BE 88 F0 B1 29 00 movsx ecx,byte_ptr_ds:[eax+29B1F0]
00292420 8B 55 CC mov_edx,dword_ptr_ss:[ebp-34]
00292423 0F BE 82 F0 B1 29 00 movsx eax,byte_ptr_ds:[edx+29B1F0]
0029242A 33 C1 xor_eax,ecx
0029242C 8B 4D CC mov ecx,dword_ptr_ss:[ebp-34]
0029242F 8B 81 F0 B1 29 00 mov byte_ptr_ds:[ecx+29B1F0],al
00292435 8B 45 88 mov_eax,dword_ptr_ss:[ebp-8]
00292438 83 E8 01 sub_eax,1
0029243B 2B 45 CC sub_eax,dword_ptr_ss:[ebp-34]
0029243E 8B 4D CC mov ecx,dword_ptr_ss:[ebp-34]
00292441 0F BE 91 F0 B1 29 00 movsx edx,byte_ptr_ds:[ecx+29B1F0]
00292448 0F BE 80 F0 B1 29 00 movsx eax,byte_ptr_ds:[eax+29B1F0]
0029244F 33 C2 xor_eax,edx
00292451 0B 4D F8 mov ecx,dword_ptr_ss:[ebp-8]
00292454 83 E9 01 sub ecx,1
00292457 2B 4D CC sub ecx,dword_ptr_ss:[ebp-34]
0029245A 8B 81 F0 B1 29 00 mov byte_ptr_ds:[ecx+29B1F0],al
00292460 8B 45 88 mov_eax,dword_ptr_ss:[ebp-8]
00292463 83 E8 01 sub_eax,1
00292466 2B 45 CC sub_eax,dword_ptr_ss:[ebp-34]
00292470 0F BE 88 F0 B1 29 00 movsx ecx,byte_ptr_ds:[eax+29B1F0]
00292473 8B 55 CC mov_edx,dword_ptr_ss:[ebp-34]
00292476 0F BE 82 F0 B1 29 00 movsx eax,byte_ptr_ds:[edx+29B1F0]
0029247C 8B 4D CC mov ecx,dword_ptr_ss:[ebp-34]
0029247F 8B 81 F0 B1 29 00 mov byte_ptr_ds:[ecx+29B1F0],al
00292485 E9 70 FF FF FF jmp_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.2923FA
0029248A C7 45 C0 00 00 00 mov_dword_ptr_ss:[ebp-4],0
00292491 E8 09 85 CO FF FF jmp_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.29249C
00292493 8B 45 C0 mov_eax,dword_ptr_ss:[ebp-40]
00292496 83 C0 01 add_esi,1
00292499 89 45 C0 mov_dword_ptr_ss:[ebp-40],eax
0029249C 8B 45 C0 mov_eax,dword_ptr_ss:[ebp-40]
002924A2 3B 45 F8 cmp_eax,dword_ptr_ss:[ebp-3]
002924A4 74 0A jbe_evr_q_e845f4114118ce2aaf13eb4c7cf0e04db6f6130cc1785d9fe5e1f35e7674faa3.2924D8
002924A7 8B 45 C0 mov_eax,dword_ptr_ss:[ebp-40]
002924AE 0F BE 88 F0 B1 29 00 movsx ecx,byte_ptr_ds:[eax+29B1F0]
002924B3 5D 35 8C 1D 00 mov_edx,dword_ptr_ss:[ebp-40]
002924B8 83 F2 76 xor_edx,76
002924BA 81 C2 CC 00 00 00 add_edx,CC
002924BB 81 F2 80 00 00 00 xor_edx,80
002924BC 82 C2 28 add_edx,28
002924C3 33 CA xor ecx,edx
002924C5 81 E1 FF 00 00 00 and ecx,FF
002924C8 8B 45 C0 mov_eax,dword_ptr_ss:[ebp-40]
002924CE 66 89 04 5F B2 29 00 mov_word_ptr_ds:[eax+2+29B2F0],cx
002924D6 E8 BB 85 mov_dword_ptr_ds:[eax+2+29B2F0],eax
eax+29B1F0:"IGARUSTAK.M"
eax+29B1F0:"IGARUSTAK.M"
eax+29B1F0:"IGARUSTAK.M"
eax+29B1F0:"IGARUSTAK.M"
eax+29B1F0:"IGARUSTAK.M"
eax+29B1F0:"IGARUSTAK.M"
eax+29B1F0:"IGARUSTAK.M"
eax+29B1F0:"IGARUSTAK.M"
eax+2+29B2F0:"L"R"
```

输入正确的用户名，程序弹框提示验证成功，但由于程序有反调试，所以在弹框后可能会自动退出，我们能轻易找到反调试的代码，在所有exit调用函数处下断,我们再运行

00CE2A4F	68	08	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE8BA0	CE8BA0: "Warning\n"		
00CE2A54	E8	9E	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE13F7			
00CE2A59	83	C	04		add esp,4			
00CE2A5C	8B	F4			mov esi,esp			
00CE2A5E	FF	15	94	C1	CE	00	call dword ptr ds:[<&exit>]	
00CE2A60	3B	F4			cmp esi,esp			
00CE2A68	E8	0F	E7	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C		
00CE2A6D	8B	F4			mov esi,esp			
00CE2A6F	68	08	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE8BBC	CE8BBC: "o1lydbg.exe"		
00CE2A74	8D	85	FD	FF	FF	lea eax,dword ptr ss:[ebp-210]		
00CE2A7A	50				push eax			
00CE2A7B	FF	15	A4	C1	CE	00	call dword ptr ds:[<&wscmp>]	
00CE2A81	83	C	08		add esp,8			
00CE2A84	3B	F4			cmp esi,esp			
00CE2A86	E8	F1	E6	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C		
00CE2A8B	85	C0			test eax,eax			
00CE2A90	75	F			jnz evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE2A2AD			
00CE2A9F	68	08	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE8BD8	CE8BD8: "Warning\n"		
00CE2A99	E8	5E	99	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE13F7		
00CE2A9C	83	C	04		add esp,4			
00CE2A9E	8B	F4			mov esi,esp			
00CE2AA0	50				push 0			
00CE2AA8	FF	15	94	C1	CE	00	call dword ptr ds:[<&exit>]	
00CE2AA9	75	F			test eax,eax			
00CE2AA8	E8	CF	E6	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C		
00CE2A9D	8B	F4			mov esi,esp			
00CE2A9F	68	08	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE8BF4	CE8BF4: "peid.exe"		
00CE2A9C	8D	85	FD	FF	FF	lea eax,dword ptr ss:[ebp-210]		
00CE2A8A	50				push eax			
00CE2A8B	FF	15	A4	C1	CE	00	call dword ptr ds:[<&wscmp>]	
00CE2A91	83	C	08		add esp,8			
00CE2A94	3B	F4			cmp esi,esp			
00CE2A96	E8	B1	E6	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C		
00CE2A9C	85	C0			test eax,eax			
00CE2A9D	75	F			jnz evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE2A2ED			
00CE2A9C	68	08	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE8BD8	CE8BD8: "Warning\n"		
00CE2A99	E8	1E	E9	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE13F7		
00CE2AD9	83	C	04		add esp,4			
00CE2AD0	8B	F4			mov esi,esp			
00CE2AD2	6A	00			push 0			
00CE2AD0	FF	15	94	C1	CE	00	call dword ptr ds:[<&exit>]	
00CE2A96	3B	F4			cmp esi,esp			
00CE2A8B	E8	8F	E6	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C		
00CE2A9D	68	10	8C	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE8C10	CE8C10: "ida.exe"	
00CE2AF4	8D	85	FD	FF	FF	lea eax,dword ptr ss:[ebp-210]		
00CE2AFA	50				push eax			
00CE2AFB	FF	15	A4	C1	CE	00	call dword ptr ds:[<&wscmp>]	
00CE2B01	83	C	08		add esp,8			
00CE2B04	3B	F4			cmp esi,esp			

发现罪魁祸首原来不在刚才那里，而在于此处被断下来，修改前面的je跳转，弹框后便不再退出

00CE261A	89	45	F4		mov dword ptr ss:[ebp-4],eax			
00CE261D	83	7D	F4	00	cmp dword ptr ss:[ebp-4],0			
00CE2621	74	1E			je evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE2644			
00CE2622	68	08	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE8D8	CE8D8: "Warning\n"		
00CE2628	E8	CA	ED	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE13F7		
00CE262D	83	C	04		add esp,4			
00CE2630	8B	F4			mov esi,esp			
00CE2633	6A	00			push 0			
00CE2634	FF	15	94	C1	CE	00	call dword ptr ds:[<&exit>]	
00CE263A	3B	F4			cmp esi,esp			
00CE263C	3B	F4			cmp esi,esp			
00CE2641	5F				pop esi			
00CE2642	5E				pop esi			
00CE2643	5B				pop ebx			
00CE2644	8B	4D	FC		mov ecx,dword ptr ss:[ebp-4]	ebx: "(5)"		
00CE2647	33	CD			xor ecx,ebp			
00CE2649	E8	C9	ED	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE1318		
00CE264E	83	C	04		add esp,4			
00CE2654	3B	EC			cmp ebx,esp			
00CE2656	E8	21	E8	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C		
00CE265B	8B	E5			mov esi,ebp			
00CE265D	5D				pop ebp			
00CE265E	C3				ret			

接下来进入重头戏，程序要求我们输入Start Code，按照常识，这应该就是我们要找的flag无疑，所以我们在check start code字符串后的getchar函数处下断点，然后随意输入一个start code，单步调试一遍后，知道了各个call的功能并做好注释

00CE2E8B	E8	BC	E2	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C		
00CE2E8D	68	08	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE83C	CE83C: "Check Start Code: \n"		
00CE2E93	E8	2D	E5	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE13F7		
00CE2E9A	83	C	04		add esp,4			
00CE2E9D	8B	F4			mov esi,esp			
00CE2E9E	6A	00			push 0			
00CE2E9E	FF	15	94	C1	CE	00	call dword ptr ds:[<&getchar>]	
00CE2E9E	3B	F4			cmp esi,esp			
00CE2E9F	E8	89	E2	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C		
00CE2E9F	3B	F8	0A		cmp eax,A	A: '\n'		
00CE2E9F	74	04			je evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE2EFC			
00CE2E9F	A	E8			jmp evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE2E4			
00CE2E9F	A	E8			jmp evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE2E4			
00CE2E9F	68	08	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE84F0	CE84F0: "12345678"		
00CE2F01	E8	08	E2	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE110E		
00CE2F06	83	C	04		add esp,4			
00CE2F09	83	F8	23		cmp eax,23	23: '8'		
00CE2F0C	74	18			je evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE2F29			
00CE2F0E	E8	85	E4	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE1398		
00CE2F13	E8	E7	E1	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE10FF		
00CE2F18	8B	F4			mov esi,esp			
00CE2F1A	6A	00			push 0			
00CE2F1C	FF	15	94	C1	CE	00	call dword ptr ds:[<&exit>]	blog.csdn.net/atmysy
00CE2F22	3B	F4			cmp esi,esp			
00CE2F24	E8	53	E2	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C		
00CE2F29	68	08	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE84F0	CE84F0: "12345678"		
00CE2F2E	68	70	85	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE8570		
00CE2F33	E8	17	E2	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE114F		
00CE2F38	83	C	08		add esp,8			
00CE2F38	A1	84	87	CE	00	mov eax,dword ptr ds:[CEB784]	XOR运算	
00CE2F40	50				push eax			
00CE2F41	8B	0D	80	B7	CE	00	mov ecx,dword ptr ds:[CEB78D]	
00CE2F47	51				push ecx			
00CE2F48	68	1A	11	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE11EA		
00CE2F4D	68	5A	1D	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE105A		
00CE2F52	E8	94	E1	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE10E8		
00CE2F57	83	C	10		add esp,10			
00CE2F5A	6A	01			push 1			
00CE2F5C	E8	00	E4	FF	FF	call evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE1361	反调试	
00CE2F61	83	C	04		add esp,4			
00CE2F64	85	C0			test eax,eax			
00CE2F66	75	6A			jnz evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE2FD2			
00CE2F6B	A1	74	87	CE	00	mov eax,dword ptr ds:[CEB774]		
00CE2F6D	50				push eax			
00CE2F6E	8B	0D	80	B7	CE	00	mov ecx,dword ptr ds:[CEB77D]	
00CE2F74	51				push ecx			
00CE2F75	68	9D	13	CE	00	push evr_q_e845f4114118ce2aaf13eb4c7fe04db6f6130cc1785d9fe5e1f35e7674faa3.CE139D		



<pre> 00CE2F7A 68 23 10 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1023 00CE2F7F 68 47 E1 FF FF 00 call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1028 00CE2F84 83 C4 00 test_eax_eax 00CE2F89 74 09 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE2F94 00CE2F92 66 89 F3 FE FF FF 01 mov_byte_ptr_ss:[ebp-100],1 00CE2F97 66 07 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE2F98 00CE2F9A 66 85 F3 FE FF FF 00 mov_byte_ptr_ss:[ebp-100],0 00CE2FA1 8A 95 F3 FE FF FF 00 mov_dword_ptr_ss:[ebp-100],d1 00CE2FA4 68 70 B5 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE8570 00CE2FA9 68 F0 B5 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE85F0 00CE2FB6 83 C4 08 call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1023 00CE2FB8 68 C0 00 00 00 push_cc 00CE2FB9 41 74 B7 CE 00 mov_eax_dword_ptr_ds:[CEB774] 00CE2FC0 50 push_eax 00CE2FC1 8B 00 70 B7 CE 00 mov_ecx_dword_ptr_ds:[CEB770] 00CE2FC7 51 push_ecx 00CE2FC8 8B 82 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1258 00CE2FCD 83 C4 0C add_esp,c 00CE2FD0 6E 18 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE2FED 00CE2FD2 68 E1 C1 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1398 00CE2FD7 68 50 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE10FF 00CE2FDC 8B FA 00 mov_es1_esp 00CE2FE0 50 push_0 00CE2FE6 3B F4 call_dword_ptr_ds:[&lt;eax+1&gt;]blog.csdn.net/atmysy 00CE2FE8 8E 8F E1 FF FF cmp_es1_esp 00CE2FED 6A 02 call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C 00CE2FF2 68 70 B5 CE 00 push_2 00CE2FF4 83 C4 03 call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1361 00CE2FF7 85 C0 test_eax_eax 00CE2FF8 6A 64 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE3065 00CE2FFB 66 07 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE302E 00CE3000 50 push_eax 00CE3001 8B 00 78 B7 CE 00 mov_ecx_dword_ptr_ds:[CEB778] 00CE3007 51 push_ecx 00CE3008 68 46 10 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1046 00CE300D 68 6E 10 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE106E 00CE3012 68 70 B5 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE10E8 00CE3017 83 C4 10 add_esp,10 00CE301A 85 C0 test_eax_eax 00CE301C 74 09 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE3027 00CE301E 66 07 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE302E 00CE3025 66 85 F3 FE FF FF 00 mov_byte_ptr_ss:[ebp-100],0 00CE3027 66 85 F3 FE FF FF 00 mov_dword_ptr_ss:[ebp-100],d1 00CE3028 8B 55 DF mov_byte_ptr_ss:[ebp-21],d1 00CE3033 68 70 B5 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE8570 00CE303C 68 70 B5 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE8670 </pre>	<p>7位加密</p> <p>反调试</p> <p>CE1046: "WILLE" CE106E: "WILLE"</p>
---	---

<pre> 00CE301E 66 85 F3 FE FF FF 01 mov_byte_ptr_ss:[ebp-100],1 00CE3025 66 07 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE302E 00CE3027 66 85 F3 FE FF FF 00 mov_byte_ptr_ss:[ebp-100],0 00CE3028 8A 95 F3 FE FF FF 00 mov_dword_ptr_ss:[ebp-100],d1 00CE3033 68 70 B5 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE8570 00CE303C 68 F0 B5 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE85F0 00CE3041 83 C4 08 call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE106E 00CE3046 68 C0 00 00 00 add_esp,8 00CE3049 68 CD 00 00 00 push_cd 00CE304E 68 70 B7 CE 00 mov_eax_dword_ptr_ds:[CEB77C] 00CE3053 50 push_eax 00CE3054 8B 00 78 B7 CE 00 mov_ecx_dword_ptr_ds:[CEB778] 00CE305A 51 push_ecx 00CE3060 68 F8 E1 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1258 00CE3063 83 C4 0C add_esp,c 00CE3065 68 2E E2 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE3080 00CE306A 68 90 E0 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1398 00CE306F 8B FA 00 mov_es1_esp 00CE3071 6A 00 push_0 00CE3073 66 15 94 C1 CE 00 call_dword_ptr_ds:[&lt;eax+1&gt;] 00CE3079 3B F4 call_dword_ptr_ds:[&lt;eax+1&gt;] 00CE307B 68 F0 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C 00CE3080 6A 03 push_3 00CE3082 68 DA E2 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1361 00CE3087 83 C4 04 add_esp,4 00CE308A 85 C0 test_eax_eax 00CE308C 75 6A jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE30F8 00CE308E 81 84 B7 CE 00 mov_eax_dword_ptr_ds:[CEB784] 00CE3093 50 push_eax 00CE3094 8B 00 80 B7 CE 00 mov_ecx_dword_ptr_ds:[CEB780] 00CE309A 51 push_ecx 00CE309B 68 45 11 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE115A 00CE30A0 68 5A 10 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE105A 00CE30A5 68 41 ED FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE10E8 00CE30AA 83 C4 10 add_esp,10 00CE30AD 85 C0 test_eax_eax 00CE30AF 74 09 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE30BA 00CE30B1 66 85 F3 FE FF FF 01 mov_byte_ptr_ss:[ebp-100],1 00CE30B8 66 07 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE30C1 00CE30BA 66 85 F3 FE FF FF 00 mov_byte_ptr_ss:[ebp-100],0 00CE30C7 8A 95 F3 FE FF FF 00 mov_dword_ptr_ss:[ebp-100],d1 00CE30CA 68 70 B5 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE8570 00CE30CF 68 81 D0 FF FF push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE86F0 00CE30D4 83 C4 08 call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE105A 00CE30D9 68 DD 00 00 00 add_esp,DD 00CE30DC 68 DD 00 00 00 push_DD 00CE30E1 68 81 84 B7 CE 00 mov_eax_dword_ptr_ds:[CEB784] 00CE30F6 50 push_eax </pre>	<p>第二个7位加密</p> <p>反调试</p> <p>第三个7位加密</p>
---	--

<pre> 00CE30CF 68 F0 B6 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE86F0 00CE30D4 68 81 D0 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE105A 00CE30D9 68 DD 00 00 00 add_esp,DD 00CE30DC 68 DD 00 00 00 push_DD 00CE30E1 50 push_eax 00CE30E6 8B 00 80 B7 CE 00 mov_ecx_dword_ptr_ds:[CEB780] 00CE30ED 51 push_ecx 00CE30EE 65 E1 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1258 00CE30F3 83 C4 0C add_esp,c 00CE30F6 6E 18 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE3113 00CE30F8 68 92 E2 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE10FF 00CE3102 8B FA 00 mov_es1_esp 00CE3104 6A 00 push_0 00CE3106 66 15 94 C1 CE 00 call_dword_ptr_ds:[&lt;eax+1&gt;] 00CE310C 3B F4 call_dword_ptr_ds:[&lt;eax+1&gt;] 00CE310E 68 69 E0 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE117C 00CE3111 68 81 84 B7 CE 00 mov_dword_ptr_ss:[ebp-3C],d1 00CE311A 66 09 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE3125 00CE311C 8B 45 C4 mov_eax_dword_ptr_ss:[ebp-3C] 00CE311F 83 C0 01 add_eax,1 00CE3122 89 45 C4 mov_dword_ptr_ss:[ebp-3C],eax 00CE3125 83 7D C4 07 cmp_dword_ptr_ss:[ebp-3C],7 00CE3129 7D 38 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE3163 00CE312B 68 45 C4 mov_eax_dword_ptr_ss:[ebp-3C] 00CE312E 68 45 C4 mov_ecx_dword_ptr_ss:[ebp-3C] 00CE3131 8A 91 F0 B5 CE 00 mov_dword_ptr_ds:[ecx-CE85F0] 00CE3137 8B 90 77 B5 CE 00 mov_byte_ptr_ds:[eax-CE8577],d1 00CE313D 8B 45 C4 mov_eax_dword_ptr_ss:[ebp-3C] 00CE3140 8B 4D C4 mov_ecx_dword_ptr_ss:[ebp-3C] 00CE3143 8A 93 70 B6 CE 00 mov_dword_ptr_ds:[ecx-CE8670] 00CE3149 8B 90 7E B5 CE 00 mov_byte_ptr_ds:[eax-CE857E],d1 00CE314F 8B 45 C4 mov_eax_dword_ptr_ss:[ebp-3C] 00CE3152 8B 4D C4 mov_ecx_dword_ptr_ss:[ebp-3C] 00CE3155 8A 93 70 B6 CE 00 mov_dword_ptr_ds:[ecx-CE8670] 00CE3158 8B 90 7E B5 CE 00 mov_byte_ptr_ds:[eax-CE857E],d1 00CE3161 6E B9 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE311C 00CE3163 68 DC 00 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE80DC 00CE3168 68 70 B5 CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE8570 00CE316D 68 E5 D2 FF FF call_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE1447 00CE3172 83 C4 08 add_esp,8 00CE3177 85 C0 test_eax_eax 00CE3178 66 07 jmp_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE31E8 00CE3179 8B F4 mov_es1_esp 00CE317B 6A 00 push_0 00CE317D 68 44 8E CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE8E44 00CE3182 68 88 8D CE 00 push_evr_q_e845f4114118ce2aaf13eb4c7cf04db6f6130cc1785d9fe5e1f35e7674faa3.CE8D88 00CE3187 50 push_0 00CE3189 66 15 94 C1 CE 00 call_dword_ptr_ds:[&lt;MessageBox&gt;] 00CE318E 50 push_eax </pre>	<p>3个7位密文替换对应位置的原文</p> <p>CE8E44: "WILLE" CE8D88: "&gt; DETONATION FUNCTION\n READY"</p>
--	--

我们现在知道程序先检查输入长度是否为35，然后将输入的start code进行XOR运算，然后取第8位到第14位，第15位到21位，第22位到28位三组7位字符串，这三组字符串分别经过方法相同但参数不同的运算，用结果替换掉原输入对应位置的字符串，最后与密文比较。到这里，我们可以根据程序中所给参数建立方程，方程的解即为正确的原文，但方程比较复杂，这里采用穷举来解方程，我们都知道，二位十六进制数范围为00到FF，转换为十进制即0到255，于是写出脚本如下：

```

def solve(idx1,idx2,idx3,idx4):
    for x in range(0,255):
        if(((x ^ idx1 )& idx2 )>> idx3) | (((x ^ idx1) << idx3 )& idx2) == idx4:
            print hex(x^0x76)
key1 = [0x6f,0xdd,0xdd,0x48,0x64,0x63,0xd7]
for ele in key1:
    solve(0xAD,0xAA,0x1,ele)
key2 = [0x2e,0x2c,0xfe,0x6a,0x6d,0x2a,0xf2]
for ele in key2:
    solve(0xbe,0xcc,0x2,ele)
key3=[0x6f,0x9a,0x4d,0x8b,0x4b,0xfa,0x1a]
for ele in key3:
    solve(0xef,0xf0,0x4,ele)

```

穷举出3组7位原文，再把35位原文经XOR得到的前七位和后七位与之前XOR运算的参数再作一次XOR运算（我们都知道，XOR运算是可逆的），最后将所有结果拼接起来放到在线16进制转ASCII码的网站，转换为ASCII码，get flag! 为: `hctf{>>D55_CH0CK3R_B0o0M!-68271c0a}`

今年hctf对bin选手不友好是真的，ORZ。总之还得多跟大佬们学学，嗯，就这样。