

[writeup] reversing.kr - Easy Crack

原创

hellotaqini 于 2016-03-12 16:55:56 发布 2478 收藏 1

分类专栏: [ctf](#) 文章标签: [逆向](#) [reversing.kr](#)

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/hellotaqini/article/details/50866329>

版权



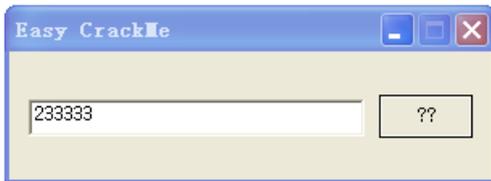
[ctf](#) 专栏收录该内容

4 篇文章 0 订阅

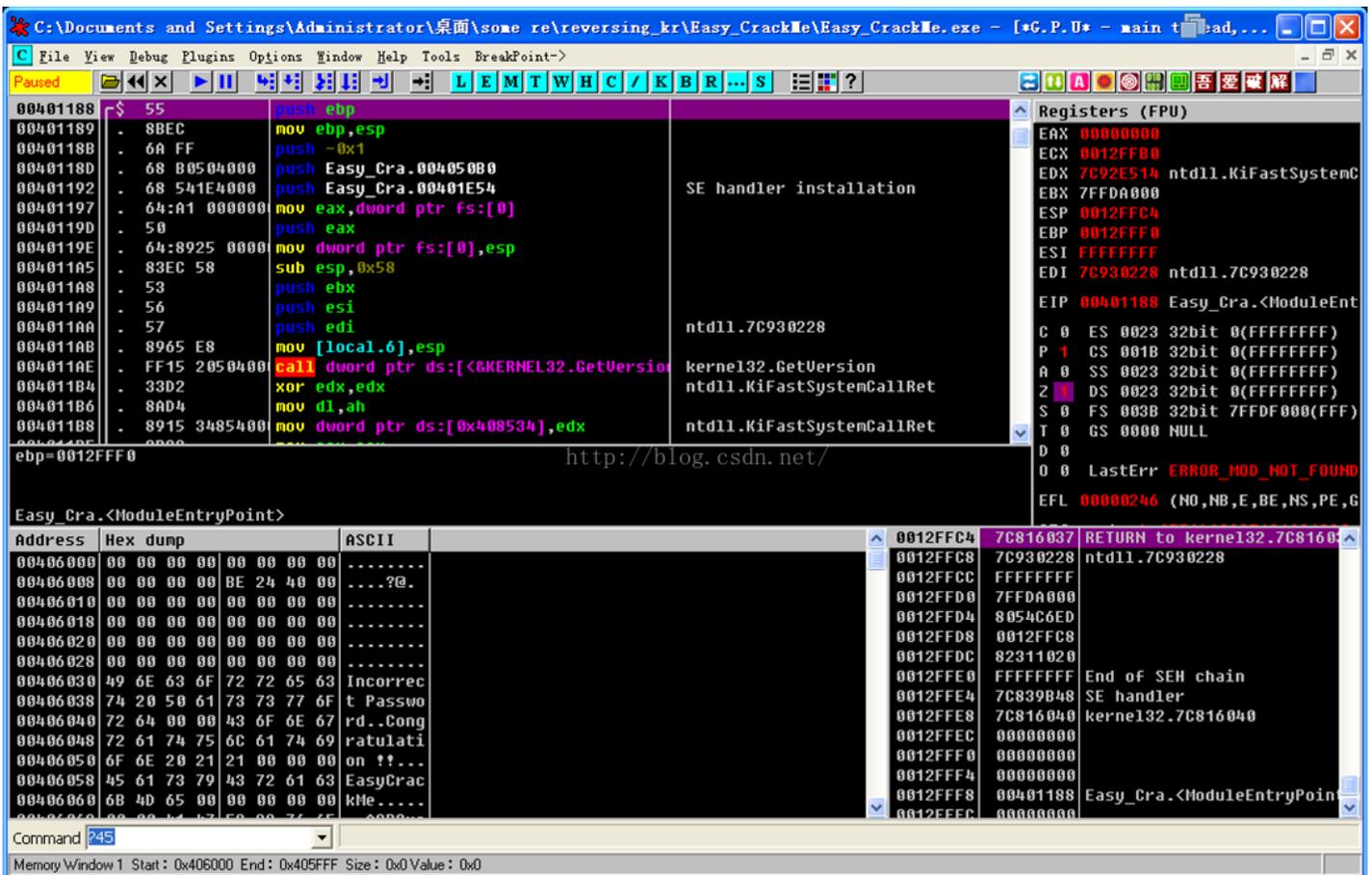
订阅专栏

最近在学习二进制, 本文的例子是reversing.kr的Easy Crack, 因为之前没写过博客, 所以先拿一个简单的题目练练手。。

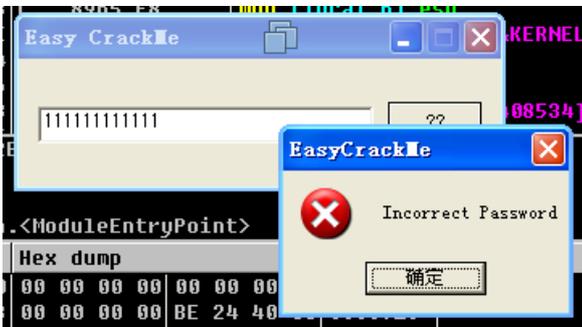
首先运行程序, 发现有一个输入框, 一个按钮, 输入的内容应该就是最后的答案。



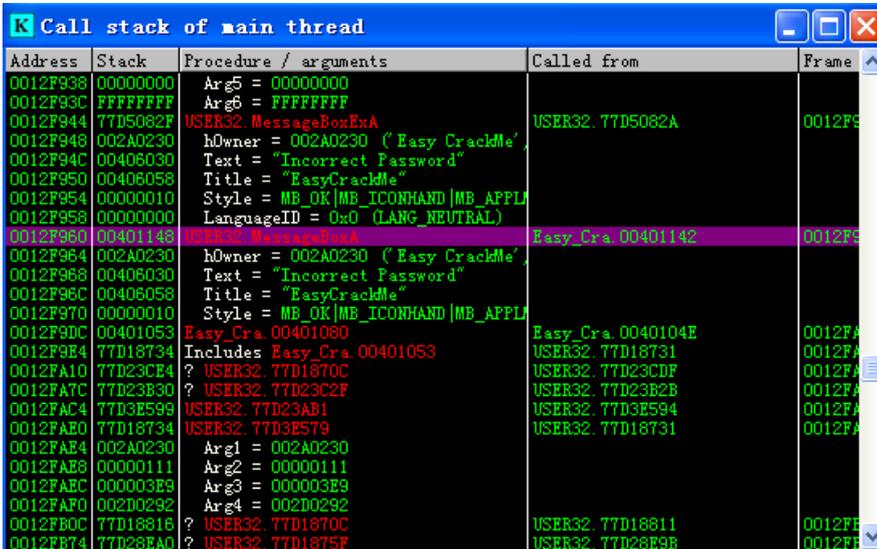
因为是easycrack, 所以应该没加壳, 直接ollydbg载入, 程序停到了入口点, 确实没有加壳



接着F9运行程序，随便输入一串字符，点击按钮，程序跳出了密码错误的提示窗



F12暂停程序，然后Alt+K查看调用栈



在MessageBoxA处右键show call，定位到了弹窗所在的程序段

发现一共有四个判断会跳转至密码错误，分别是4010B5，4010CD，40110B，401112

```

00401131 - 83C4 64      add esp,0x64
00401134 - C3          retn
00401135 > 6A 10       push 0x10
00401137 - 68 58604000 push Easy_Cra.00406058
0040113C - 68 30604000 push Easy_Cra.00406030
00401141 - 57         push edi
00401142 - FF15 A050400 call dword ptr ds:[&USER32.MessageBoxA]
00401148 - 5F         pop edi
00401149 - 83C4 64      add esp,0x64
0040114C - C3          retn
0040114D - 90         nop
0040114F - 90         nop

```

Jumps from 004010B5, 004010CD, 0040110B, 00401112

在这四处下断点，再往上看，发现读取输入文本的GetDlgItemTextA，GetDlgItemTextA和第一个跳转之间有一个字符比较的指令，也下个断点

重新载入程序，运行，

这次输入1234567890，然后点击按钮，程序停到了第一个断点处，

此时进行比较的是输入的第二位'2'和'a'（ASCII=0x61）

```

004010A4 - 68 E8030000 push 0x3E8
004010A9 - 57         push edi
004010AA - FF15 9C50400 call dword ptr ds:[&USER32.GetDlgItemTextA]
004010B0 - 807C24 05 61 cmp byte ptr ss:[esp+0x5],0x61
004010B5 - 75 7E     jnz short Easy_Cra.00401135
004010B7 - 6A 02     push 0x2
004010B9 - 8D4C24 0A lea ecx,dword ptr ss:[esp+0xA]
004010BD - 68 78604000 push Easy_Cra.00406078
004010C2 - 51         push ecx
004010C3 - E8 88000000 call Easy_Cra.00401150
004010C8 - 83C4 0C     add esp,0xC
004010CB - 85C0       test eax,eax
004010CD - 75 66     jnz short Easy_Cra.00401135
004010CF - 53         push ebx
004010D0 - 56         push esi
004010D1 - BE 6C604000 mov esi,Easy_Cra.0040606C
004010D6 - 8D4424 10 lea eax,dword ptr ss:[esp+0x10]
004010DA > 8A10     mov dl,byte ptr ds:[eax]

```

ControlID = 3E8 (1000.)
hWnd = NULL
GetDlgItemTextA
5y
ntdll.KiUserCallbackDispatcher
R3versing

F8单步执行程序，遇到跳转时，把Z标志位置1，让跳转不发生，程序进行后续的字符比较

```

0040108F - C64424 04 00 mov byte ptr ss:[esp+0x4],0x0
00401094 - 6A 64     push 0x64
00401096 - F3:AB    rep stos dword ptr es:[edi]
00401098 - 66:AB    stos word ptr es:[edi]
0040109A - AA       stos byte ptr es:[edi]
0040109B - 8B7C24 70 mov edi,dword ptr ss:[esp+0x70]
0040109F - 8D4424 08 lea eax,dword ptr ss:[esp+0x8]
004010A3 - 50       push eax
004010A4 - 68 E8030000 push 0x3E8
004010A9 - 57         push edi
004010AA - FF15 9C50400 call dword ptr ds:[&USER32.GetDlgItemTextA]
004010B0 - 807C24 05 61 cmp byte ptr ss:[esp+0x5],0x61
004010B5 - 75 7E     jnz short Easy_Cra.00401135
004010B7 - 6A 02     push 0x2
004010B9 - 8D4C24 0A lea ecx,dword ptr ss:[esp+0xA]
004010BD - 68 78604000 push Easy_Cra.00406078
004010C2 - 51         push ecx
004010C3 - E8 88000000 call Easy_Cra.00401150
004010C8 - 83C4 0C     add esp,0xC
004010CB - 85C0       test eax,eax
004010CD - 75 66     jnz short Easy_Cra.00401135
004010CF - 53         push ebx
004010D0 - 56         push esi
004010D1 - BE 6C604000 mov esi,Easy_Cra.0040606C
004010D6 - 8D4424 10 lea eax,dword ptr ss:[esp+0x10]
004010DA > 8A10     mov dl,byte ptr ds:[eax]
004010DC - 8A1E     mov bl,byte ptr ds:[esi]
004010DE - 8AC0     mov cl,dl

```

Count = 64 (100.)
USER32.77D18734
Buffer = 0000000A
ControlID = 3E8 (1000.)
hWnd = 002B0230 ('Easy CrackMe')
GetDlgItemTextA
5y
USER32.77D321CC
Easy_Cra.00401020
R3versing

cmp is NOT taken
00401135

EIP 004010B5 Easy_Cra.004010B5
C 1 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
H 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 4 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 000002C7 (NO,B,E,PE,L,
ST0 empty -UNORM FC4C 00000202
ST1 empty -UNORM FDB8 00000286
ST2 empty +UNORM 2985 00000000
ST3 empty +UNORM 1F80 00000000
ST4 empty 9.5099838587436458100
ST5 empty 5.4728616292580094010
ST6 empty 5.4728600857832026170
ST7 empty 0.0000000117122203660
3 2 1 0 E S
FST 4000 Cond 1 0 0 0 Err 0 0
FCW 027F Prec NEAR,53 Mask

观察堆栈，这时是'34'和'5y'的比较

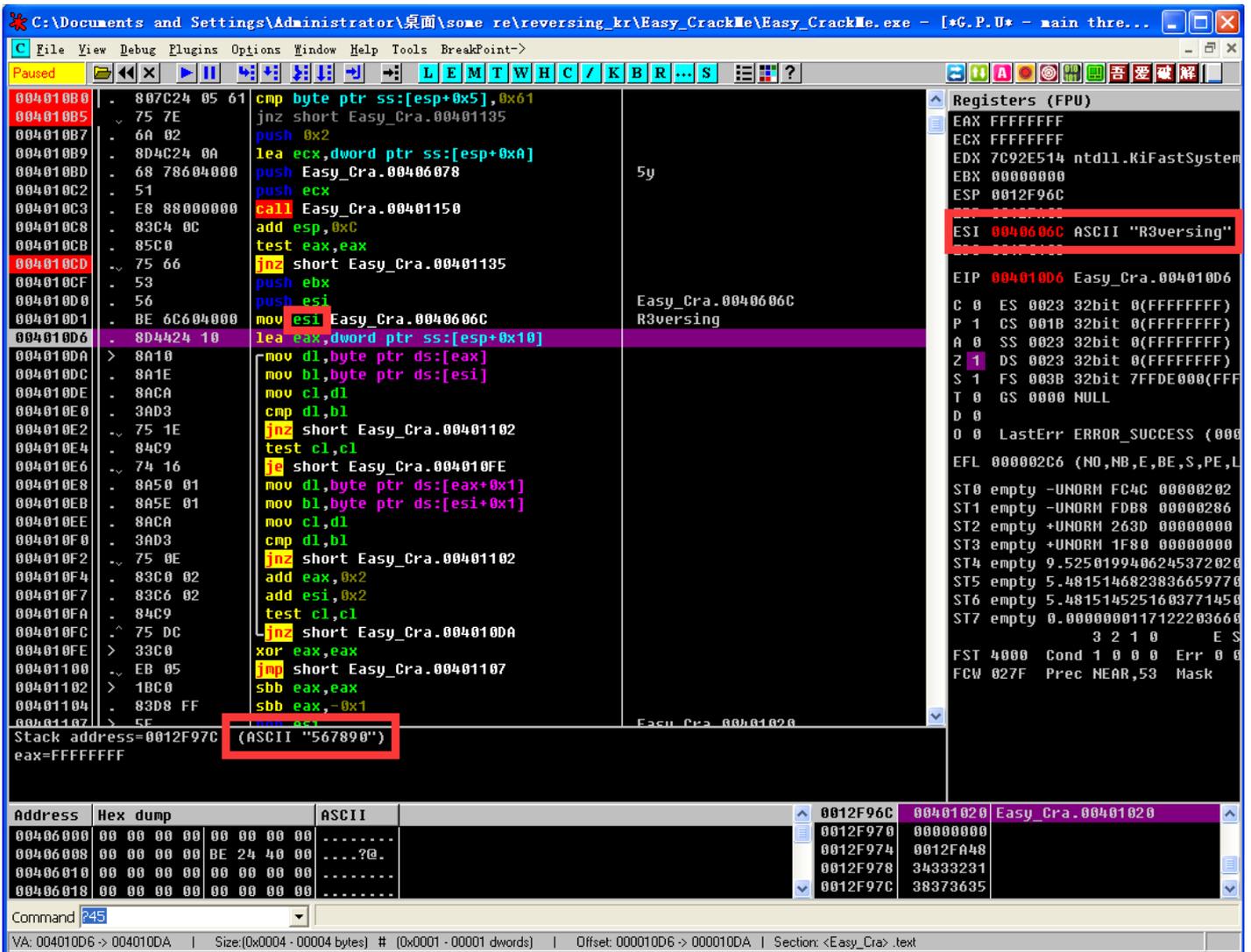
The screenshot shows a debugger window with the following components:

- Assembly Window:** Displays assembly instructions with addresses, hex values, and mnemonics. The current instruction is `call Easy_Cra.00401150` at address `004010C3`.
- Registers (FPU) Window:** Shows the state of CPU registers. EAX is 00000000, ECX is 0012F97A (ASCII "34567890"), EDX is 7C92E514 (ntdll.KiFastSystem), and EIP is 004010C3.
- Memory Dump:** Shows a table of memory addresses, hex dumps, and ASCII representations. A red box highlights the ASCII string "5y" at address 0012F970.

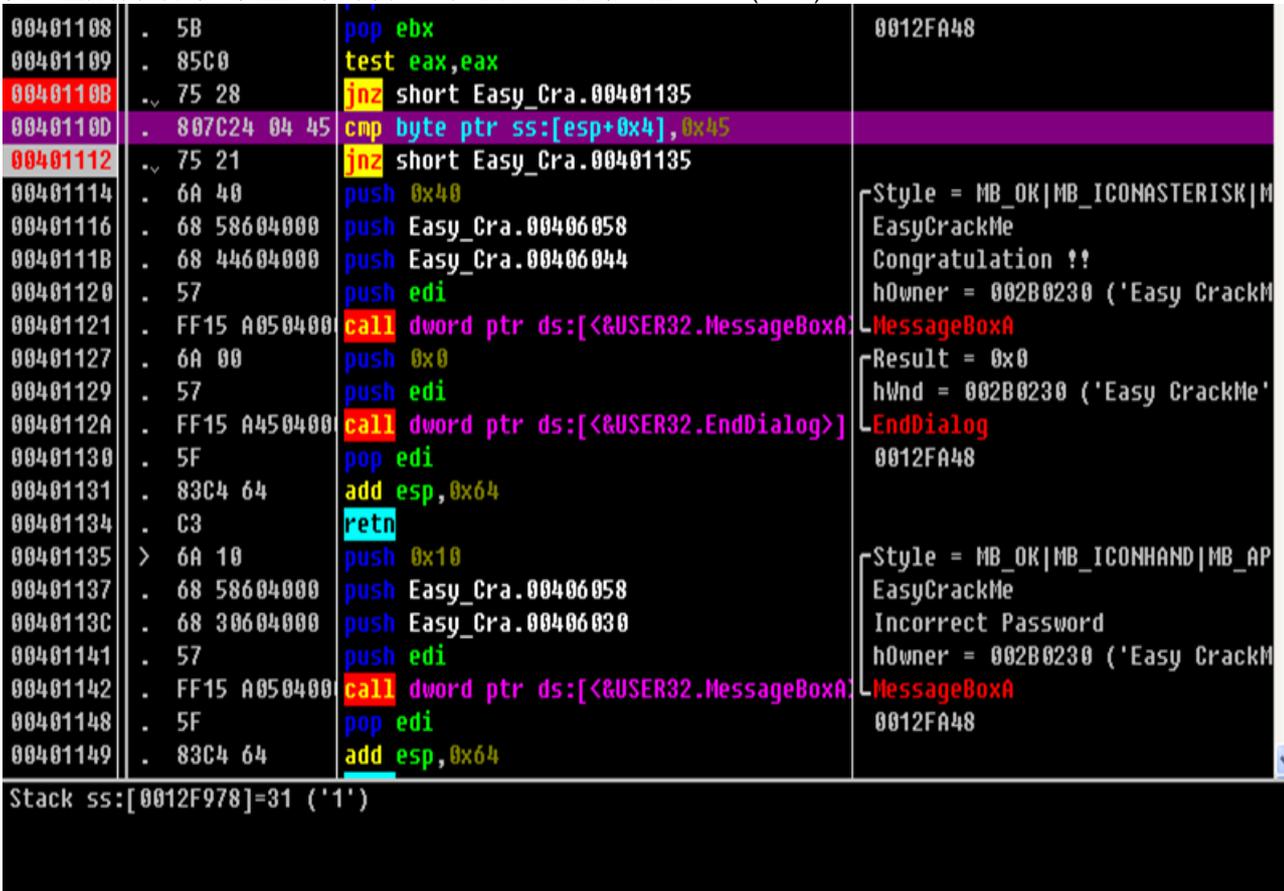
Address	Hex dump	ASCII
00406000	00 00 00 00 00 00 00 00
00406008	00 00 00 00 BE 24 40 00	...?@.
00406010	00 00 00 00 00 00 00 00
00406018	00 00 00 00 00 00 00 00

遇到跳转时，把Z置1，让跳转不发生，

接下来的是比较'567890'和'R3versing'，对应的是密码的5到13位



依然让所有跳转都不发生，来到最后一个比较，比较第一位'1'和'E'(0x45)



于是得出密码第2位是a，3到4位是5y，5到13位是R3versing，第一位是E

