آموزش های R4NDOM -فارسی

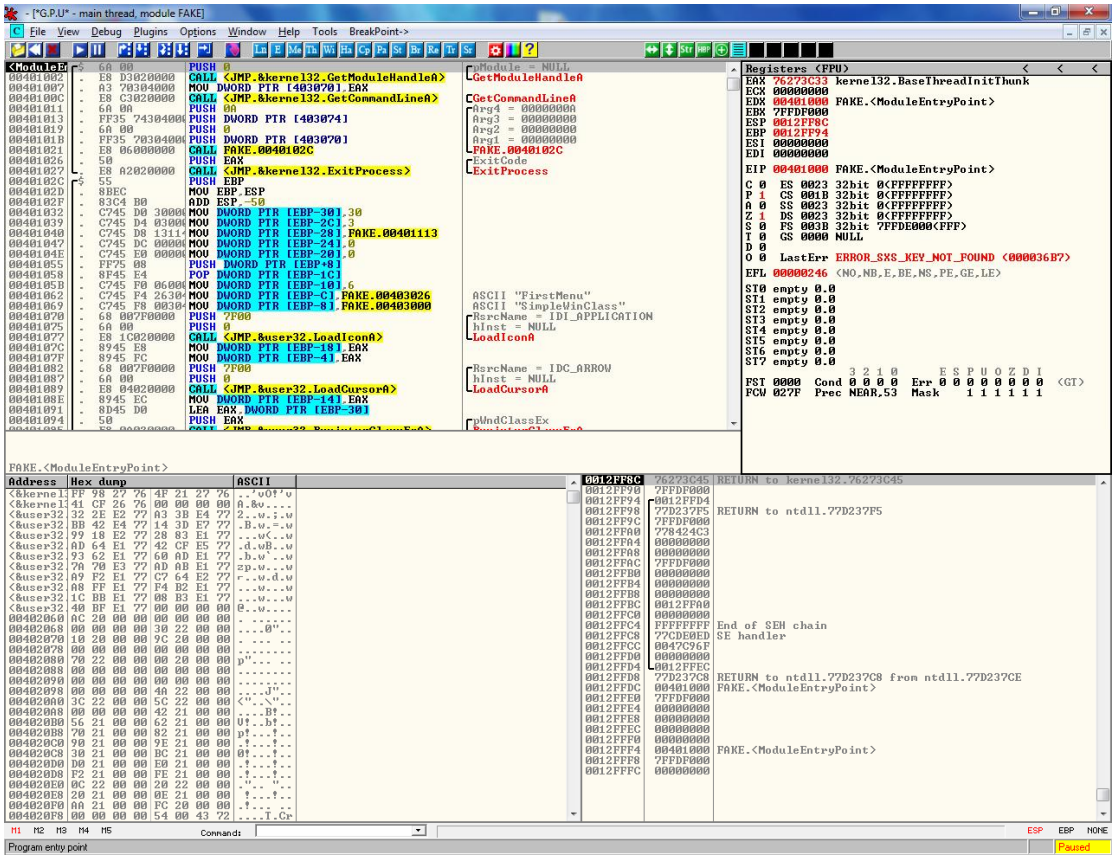
قسمت ۵ – اولین تجربه ی کرک

در این قسمت از آموزش ما میخواهیم یک فایل تحت عنوان Crackme با نام FAKE.exe را بررسی کنیم ،این برنامه شبیه به مثال های قبل است با این تفاوت که در این فایل در منوی Register ازما ریجیستر کد میخواهد که با توجه به نوع کد دریافتی ،پیامی به ما نشان میدهد ،بدیهی است که اگر رمز صحیح را وارد کنیم ،پیام صحت رمز را نشان میدهد و در غیر اینصورت پیامی مبنی بر اشتباه بودن رمز نمایش میدهد

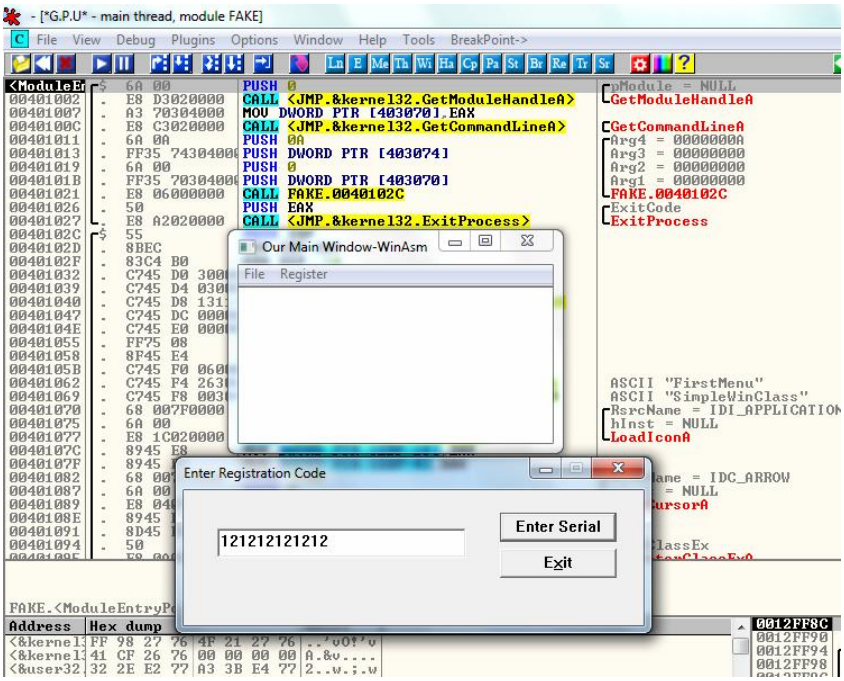
البته قول میدهم در آموزش بعدی یک مثال واقعی تر بزنم

تمام ابزاری که احتیاج دارید OllyDBG (هر ورژن و ویرایشی) و برنامه ی Fake.exe (ضمیمه شده)

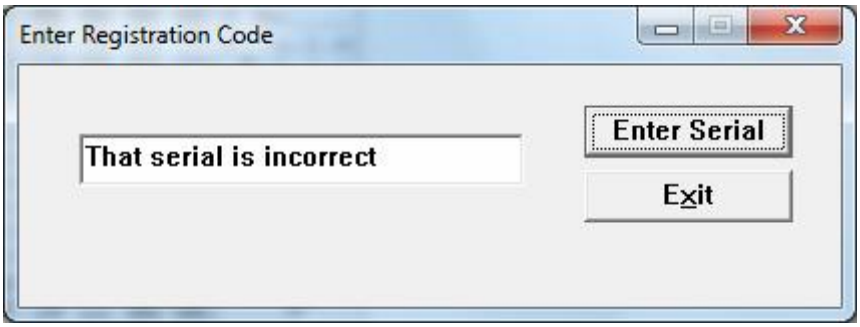
بیایید شروع کنیم ،برنامه را درون ollydbg باز کنید :



برنامه را با F9 اجرا کنید و از منوی Register یک کد دلخواه وارد کنید و کلید Enter Serial را بفشارید:



پس از فشردن کلید Enter Serial داریم :



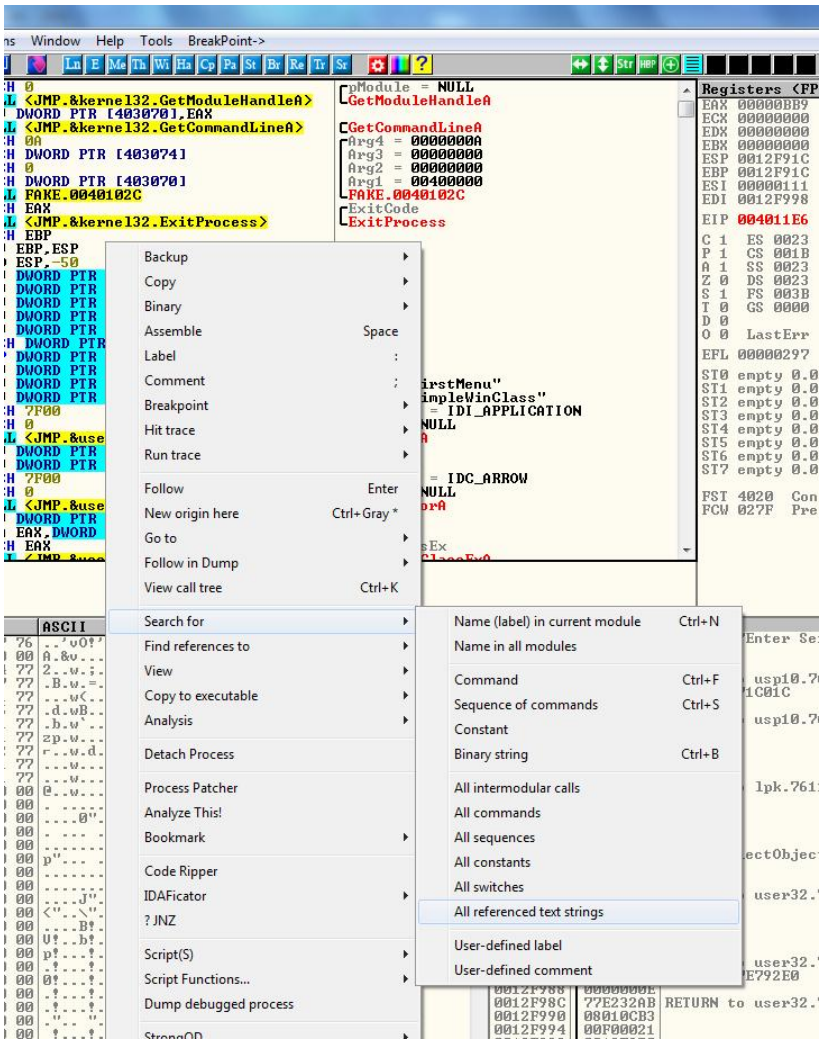
حالا میخواهم به شما یکی از اولین تکنیک های مهندسی معکوس را نشان دهیم ،ساز کار به این صورت است که ما پیام های مورد استفاده در برنامه ی Fake.exe را جستجو میکنیم و سپس روتین چک رمز را بیابیم ،چطوری؟

Searching For All Text Strings

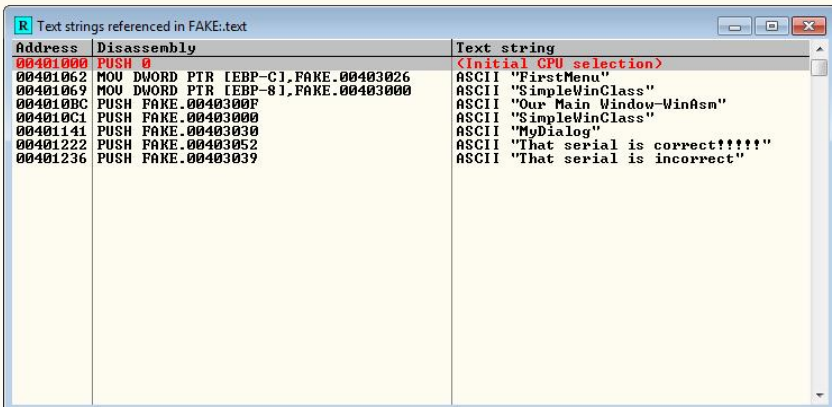
اول اجازه دهید بگویم استفاده از این تکنیک بسیار رایج است ،از طرفی سازندگان نرم افزارها برای حفاظت از محصول خود به طریق گوناگونی از این متن ها ،پیام ها را، با استفاده از پکر ها یا با رمزنگاری کردن آنها را محافظت میکنند.

اساسا در این روش Ollydbg فضای حافظه ی مربوط به برنامه را برای رشته هایی در قالب ASCII یا Unicode جستجو میکند،گاهی اوقات موفقیت آمیز هست و ممکن است با این پیام دوست داشتنی "Thank you for registering" رو برو شویم و بعضی وقت ها هم چیزی شبیه این "F@7=" برخورد میکنیم

برای شروع بروی پنجره ی assembly راست کلیک کنید و سپس: "Serach For" >"All ReferencedText Strings"

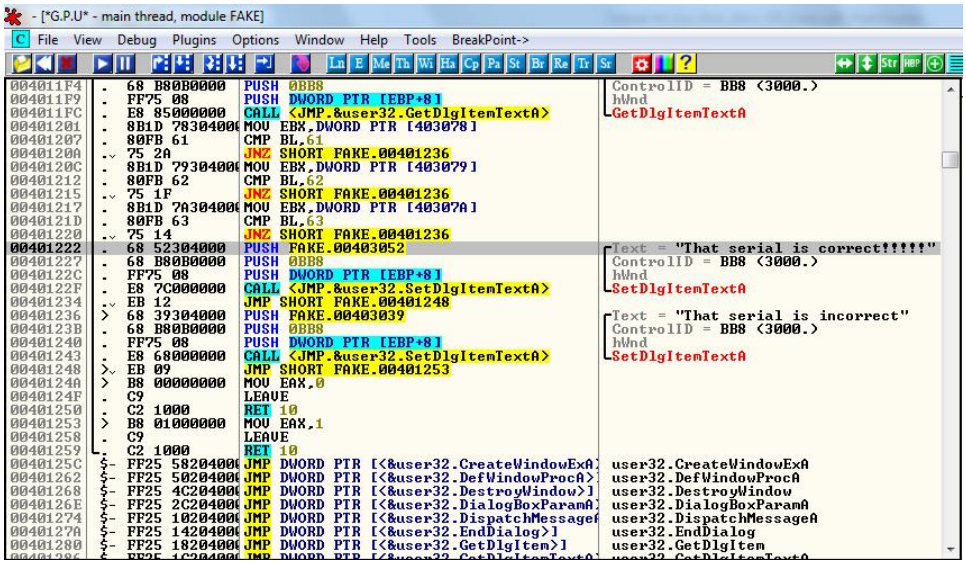


و سپس ollydbg فضای حافظه ی مربوط به برنامه را جستجو کرده و نتیجه را در پنجره زیر نشان میدهد :



به رشته های بالا نگاه کنید (البته این مثال کوچکی است در برنامه های بزرگ هزارن رشته ممکن است وجود داشته باشد) رشته ایی جلب توجه میکند:

به نظر میرسد رشته ی جالب و امیدوار کننده ایی باشد بروی آن دبل کلیک کنید تا به آدرس آن برویم :



معرفی قانون دوم

بعضی از طرح های امنیتی را میتوان با جابجا کردن دستور Jump به جای اینکه به قسمت پیام "بد" برود،پیام "خوب" را نشان دهد! این بدین معنا است قبلا از نمایش پیام "بد" نوعی چک وجود دارد ،مثلا قبل از نمایش پیام خطا ،نوعی چک با مضمون "آیا رمز وارد شده صحیح است؟" صورت میگیرد، پس از نتیجه ی مقایسه ،بسته به نوع نتیجه ،پرش به قسمت بد یا خوب صورت میگیرد

به پیام "This serial is correct!!!!" در آدرس 401222 نگاه کنید،این پیام پیامی مبنی بر موفقیت ثبت نام برنامه است و موقعی نمایش پیدا میکند که ما رمز صحیح را وارد کرده ایم ،حال به خط بالای آن نگاه کنید :

```
- [*G.P.U* - main thread, module FAKE]
File View Debug Plugins Options Window Help Tools BreakPoint->
004011F4  .  68 B80B0000    PUSH 0BB8                          ControlID = BB8 (3000.)
004011F9  .  FF75 08        PUSH DWORD PTR [EBP+8]             hWnd
004011FC  .  E8 85000000    CALL <JMP.&user32.GetDlgItemTextA> GetDlgItemTextA
00401201  .  8B1D 7830400(  MOV EBX,DWORD PTR [403078]
00401207  .  80FB 61        CMP BL,61
0040120A  .v 75 2A          JNZ SHORT FAKE.00401236
0040120C  .  8B1D 7930400(  MOV EBX,DWORD PTR [403079]
00401212  .  80FB 62        CMP BL,62
00401215  .v 75 1F          JNZ SHORT FAKE.00401236
00401217  .  8B1D 7A30400(  MOV EBX,DWORD PTR [40307A]
0040121D  .  80FB 63        CMP BL,63
00401220  .v 75 14          JNZ SHORT FAKE.00401236
00401222  .  68 52304000    PUSH FAKE.00403052                 Text = "That serial is correct!!!!!"
00401227  .  68 B80B0000    PUSH 0BB8                          ControlID = BB8 (3000.)
0040122C  .  FF75 08        PUSH DWORD PTR [EBP+8]             hWnd
0040122F  .  E8 7C000000    CALL <JMP.&user32.SetDlgItemTextA> SetDlgItemTextA
00401234  .v EB 12          JMP SHORT FAKE.00401248
00401236  >  68 39304000    PUSH FAKE.00403039                 Text = "That serial is incorrect"
0040123B  .  68 B80B0000    PUSH 0BB8                          ControlID = BB8 (3000.)
00401240  .  FF75 08        PUSH DWORD PTR [EBP+8]             hWnd
00401243  .  E8 68000000    CALL <JMP.&user32.SetDlgItemTextA> SetDlgItemTextA
00401248  >v EB 09          JMP SHORT FAKE.00401253
0040124A  >  B8 00000000    MOV EAX,0
0040124F  .  C9             LEAVE
```

همانطور که میبینید قبل از نمایش پیام "خوب" یک نوع مقایسه در خط 401220 صورت میگیرد،این پرش انجام خواهد شد ،به فلش قرمز رنگ نگاه کنید ،انتهای فلش مقصد را نشان میدهد که در واقع نمایش پیام "بد" است ،دقیقا جایی میرود که ما نمیخواهیم !!بالای دستور JNZ یک دستور با نام CMP وجود دارد ،این بدان معنی است که نتیجه ی یک مقایسه را درون خود نگه میدارد و JNZ بر اساس آن خط فعالیت میکند،همانطور که در تصویر زیر میبینید ما ۲ تا مقایسه دیگر داریم که به پیام بد ختم میشوند :

```
004011D6  .  6A 00          PUSH 0                             lParam = 0
004011D8  .  6A 00          PUSH 0                             wParam = 0
004011DA  .  6A 10          PUSH 10                            Message = WM_CLOSE
004011DC  .  FF75 08        PUSH [ARG.1]                       hWnd = 401000
004011DF  .  E8 C6000000    CALL <JMP.&user32.SendMessageA>    SendMessageA
004011E4  .v EB 62          JMP SHORT FAKE.00401248
004011E6  >  3D B90B0000    CMP EAX,0BB9
004011EB  .v 75 5B          JNZ SHORT FAKE.00401248
004011ED  .  6A 64          PUSH 64                            Count = 64 (100.)
004011EF  .  68 78304000    PUSH FAKE.00403078                 Buffer = FAKE.00403078
004011F4  .  68 B80B0000    PUSH 0BB8                          ControlID = BB8 (3000.)
004011F9  .  FF75 08        PUSH [ARG.1]                       hWnd = 00401000
004011FC  .  E8 85000000    CALL <JMP.&user32.GetDlgItemTextA> GetDlgItemTextA
00401201  .  8B1D 78304000  MOV EBX,DWORD PTR DS:[403078]
00401207  .  80FB 61        CMP BL,61
0040120A  .v 75 2A          JNZ SHORT FAKE.00401236
0040120C  .  8B1D 79304000  MOV EBX,DWORD PTR DS:[403079]
00401212  .  80FB 62        CMP BL,62
00401215  .v 75 1F          JNZ SHORT FAKE.00401236
00401217  .  8B1D 7A304000  MOV EBX,DWORD PTR DS:[40307A]
0040121D  .  80FB 63        CMP BL,63
00401220  .v 75 14          JNZ SHORT FAKE.00401236
00401222  .  68 52304000    PUSH FAKE.00403052                 Text = "That serial is correct!!!!!"
00401227  .  68 B80B0000    PUSH 0BB8                          ControlID = BB8 (3000.)
0040122C  .  FF75 08        PUSH [ARG.1]                       hWnd = 00401000
0040122F  .  E8 7C000000    CALL <JMP.&user32.SetDlgItemTextA> SetDlgItemTextA
00401234  .v EB 12          JMP SHORT FAKE.00401248
00401236  >  68 39304000    PUSH FAKE.00403039                 Text = "That serial is incorrect"
0040123B  .  68 B80B0000    PUSH 0BB8                          ControlID = BB8 (3000.)
00401240  .  FF75 08        PUSH [ARG.1]                       hWnd = 00401000
00401243  .  E8 68000000    CALL <JMP.&user32.SetDlgItemTextA> SetDlgItemTextA
00401248  >v EB 09          JMP SHORT FAKE.00401253
0040124A  >  B8 00000000    MOV EAX,0
0040124F  .  C9             LEAVE
00401250  .  C2 1000        RETN 10
00401253  >  B8 01000000    MOV EAX,1
00401258  .  C9             LEAVE
00401259  .  C2 1000        RETN 10
0040125C  $- FF25 58204000  JMP DWORD PTR DS:[<&user32.CreateWindowExA>]   user32.CreateWindowExA
00401262  $- FF25 50204000  JMP DWORD PTR DS:[<&user32.DefWindowProcA>]    user32.DefWindowProcA
00401268  $- FF25 4C204000  JMP DWORD PTR DS:[<&user32.DestroyWindow>]     user32.DestroyWindow
0040126E  $- FF25 2C204000  JMP DWORD PTR DS:[<&user32.DialogBoxParamA>]   user32.DialogBoxParamA
00401274  $- FF25 10204000  JMP DWORD PTR DS:[<&user32.DispatchMessageA>]  user32.DispatchMessageA
0040127A  $- FF25 14204000  JMP DWORD PTR DS:[<&user32.EndDialog>]         user32.EndDialog
```

If no jumps are taken, this will fall through to good message

اگر هیچکدام از این پرش ها صورت نگیرد ما مستقیم به سمت نمایش پیام خوب میرویم

اضافه کردن توضیحات

اضافه کردن توضیحات میتواند خیلی مفید باشد مخصوصا موقعی که کد ها مشابه و پیچیده هستند ،همچنین توضیحات میتواند ما را به درک کلی از رفتار یک برنامه برساند برای اضافه کردن توضیحات در هر خط بروی قسمت توضیحات دبل کلیک کرده و توضیحات خود را درج کنید،ما پرش هایی که به سمت پیام "بد" هست را نمیخواهیم پس اینطور عمل میکنیم :

```
- [*G.P.U* - main thread, module FAKE]
File View Debug Plugins Options Window Help Tools BreakPoint->

004011F4  .   68 B80B0000   PUSH 0BB8                      ControlID = BB8 (3000.)
004011F9  .   FF75 08       PUSH DWORD PTR [EBP+8]         hWnd
004011FC  .   E8 85000000   CALL <JMP.&user32.GetDlgItemTextA>   GetDlgItemTextA
00401201  .   8B1D 7830400( MOV EBX,DWORD PTR [403078]
00401207  .   80FB 61       CMP BL,61
0040120A  .v  75 2A         JNZ SHORT FAKE.00401236        NABAYAD JMP SOORAT GIRAD!!
0040120C  .   8B1D 7930400( MOV EBX,DWORD PTR [403079]
00401212  .   80FB 62       CMP BL,62
00401215  .v  75 1F         JNZ SHORT FAKE.00401236        NABAYAD JMP SOORAT GIRAD
00401217  .   8B1D 7A30400( MOV EBX,DWORD PTR [40307A]
0040121D  .   80FB 63       CMP BL,63
00401220  .v  75 14         JNZ SHORT FAKE.00401236        NABAYAD JMP SOORAT GIRAD
00401222  .   68 52304000   PUSH FAKE.00403052             Text = "That serial is correct!!!!!"
00401227  .   68 B80B0000   PUSH 0BB8                      ControlID = BB8 (3000.)
0040122C  .   FF75 08       PUSH DWORD PTR [EBP+8]         hWnd
0040122F  .   E8 7C000000   CALL <JMP.&user32.SetDlgItemTextA>   SetDlgItemTextA
00401234  .v  EB 12         JMP SHORT FAKE.00401248
00401236  >   68 39304000   PUSH FAKE.00403039             Text = "That serial is incorrect"
0040123B  .   68 B80B0000   PUSH 0BB8                      ControlID = BB8 (3000.)
00401240  .   FF75 08       PUSH DWORD PTR [EBP+8]         hWnd
00401243  .   E8 68000000   CALL <JMP.&user32.SetDlgItemTextA>   SetDlgItemTextA
00401248  >v  EB 09         JMP SHORT FAKE.00401253
0040124A  >   B8 00000000   MOV EAX,0
```

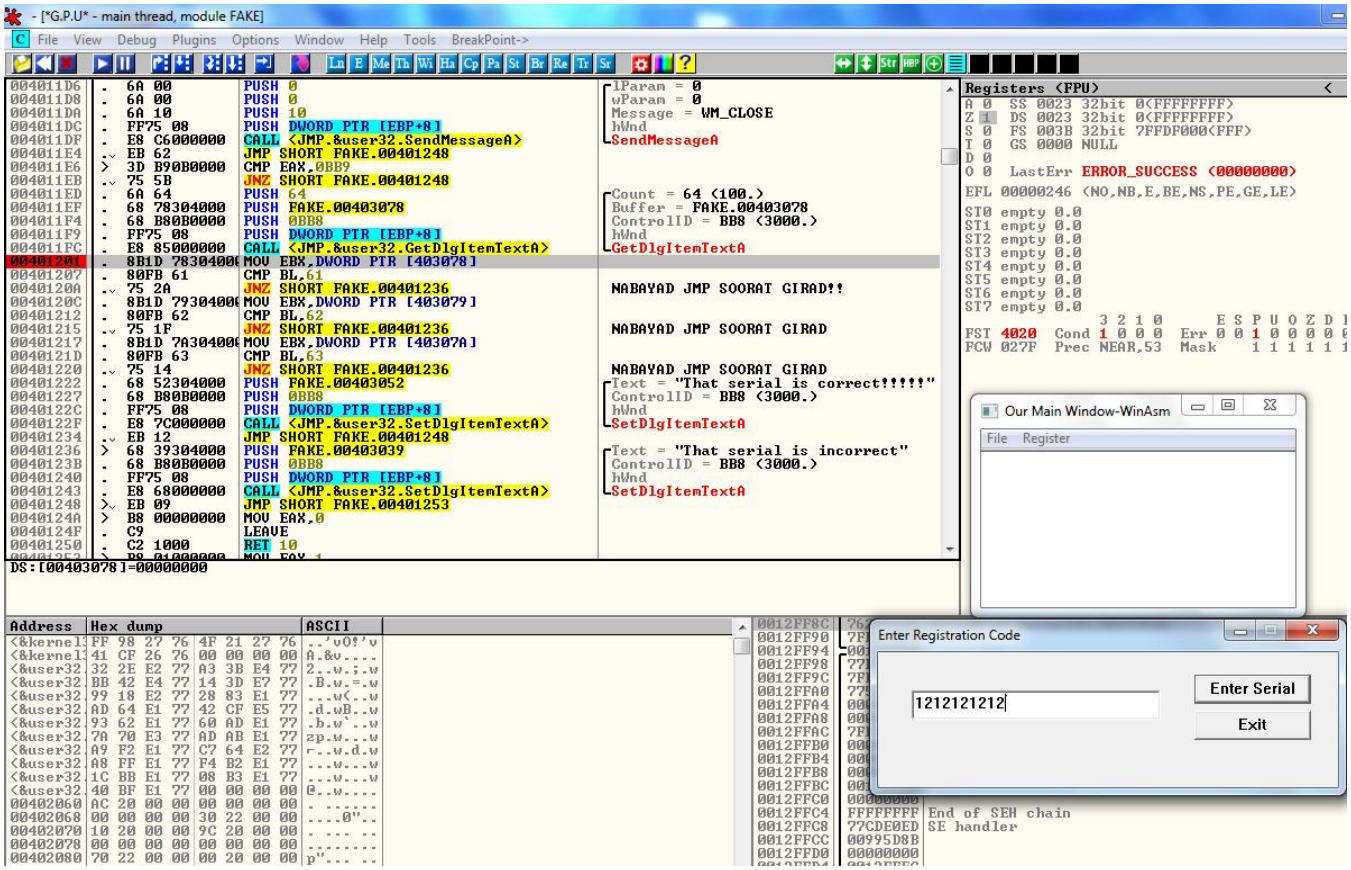
همان طور که میبینید ۳ پرش به سمت نمایش پیام خطا صورت میگیرد که ما آنها را با پیام" NABAYAD JMP SOORAT GIRAD!!" مشخص کرده ایم

حالا بیایید یک نقطه ی توقف بروی آدرس 401201 یا آدرس قبل از این پرش ها بگذاریم:

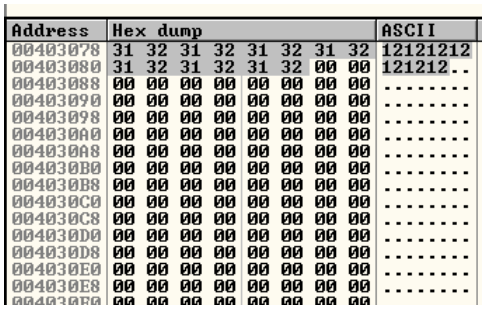
```
- [*G.P.U* - main thread, module FAKE]
File View Debug Plugins Options Window Help Tools BreakPoint->

004011E6  >   3D B90B0000   CMP EAX,0BB9
004011EB  .v  75 5B         JNZ SHORT FAKE.00401248
004011ED  .   6A 64         PUSH 64                        Count = 64 (100.)
004011EF  .   68 78304000   PUSH FAKE.00403078              Buffer = FAKE.00403078
004011F4  .   68 B80B0000   PUSH 0BB8                      ControlID = BB8 (3000.)
004011F9  .   FF75 08       PUSH DWORD PTR [EBP+8]         hWnd
004011FC  .   E8 85000000   CALL <JMP.&user32.GetDlgItemTextA>   GetDlgItemTextA
00401201  .   8B1D 7830400( MOV EBX,DWORD PTR [403078]
00401207  .   80FB 61       CMP BL,61
0040120A  .v  75 2A         JNZ SHORT FAKE.00401236        NABAYAD JMP SOORAT GIRAD!!
0040120C  .   8B1D 7930400( MOV EBX,DWORD PTR [403079]
00401212  .   80FB 62       CMP BL,62
00401215  .v  75 1F         JNZ SHORT FAKE.00401236        NABAYAD JMP SOORAT GIRAD
00401217  .   8B1D 7A30400( MOV EBX,DWORD PTR [40307A]
0040121D  .   80FB 63       CMP BL,63
00401220  .v  75 14         JNZ SHORT FAKE.00401236        NABAYAD JMP SOORAT GIRAD
00401222  .   68 52304000   PUSH FAKE.00403052             Text = "That serial is correct!!!!!"
00401227  .   68 B80B0000   PUSH 0BB8                      ControlID = BB8 (3000.)
0040122C  .   FF75 08       PUSH DWORD PTR [EBP+8]         hWnd
0040122F  .   E8 7C000000   CALL <JMP.&user32.SetDlgItemTextA>   SetDlgItemTextA
00401234  .v  EB 12         JMP SHORT FAKE.00401248
00401236  >   68 39304000   PUSH FAKE.00403039             Text = "That serial is incorrect"
0040123B  .   68 B80B0000   PUSH 0BB8                      ControlID = BB8 (3000.)
```

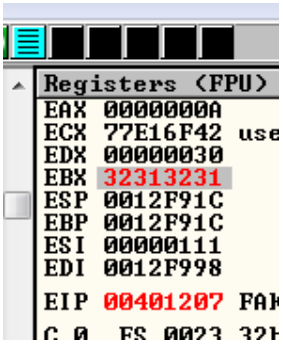
و F9 بزنیم تا برنامه اجرا شود سپس از منوی Register یک شماره دلخواه وارد میکنیم (من 1212121212 را وارد میکنم) و کلید Enter serialرا میفشاریم ،میبینید که به محیط Ollydbg منتقل شدیم این یعنی برنامه در خط حاوی نقطه ی توقف ،متوقف شده :



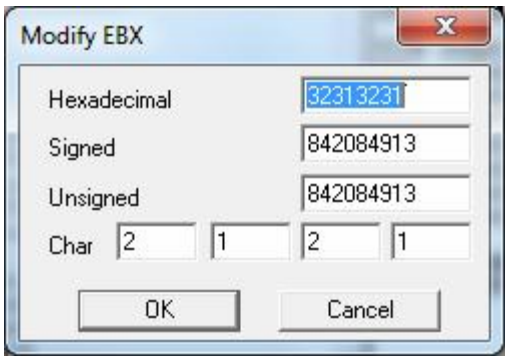
ما بروی MOV EBX, DWORD PTR DS:[403078] متوقف شده ایم ، طبق آموزش های قبلی بروی دستور العمل جاری راست کلیک کرده "Follow in Dump"->"Memory Address":



خب خب ، همانطور که میبینید در تصویر بالا میبینید، عددی که وارد کرده ایم اینجاست،خب ما میدانیم که ۴ بایت اول که در اینجا(31 32 31 32) (کد اسکی ۱۲۱۲) درون ثبات EBX ریخته خواهد شد. کلید F8 را بفشارید :



اگر میخواهید مقدار عدد را به صورت ASCII ببنید بروی ثبات EBX دبل کلیک کنید:



Little Endian

اندین یا Endian استعاره‌ای است به اختلاف دو قوم لی‌لی‌پوتی بر سر جهت بریدن سر تخم‌مرغ آبپز در ماجراهای گالیور .یکی از این دوقوم تخم‌مرغ آبپز را به روش سنتی از سر پهنتر باز میکردند که به **بیگ اندین** یا Big Endian معروف بودند و دیگری معتقد به بریدن سر کوچکتر تخم‌مرغ بودند که به ایشان **لیتل اندین** یا Little Endian گفته می شد.

پردازنده ها داده های را به اشکال گوناگونی در حافظه ذخیره میکنند،بسته به معماری پردازنده دو راه کلی برای ذخیره سازی داریم :

Big-Endian
Little-Endian

پردازنده های شرکت اینتل از Little-Endian استفاده میکنند

فرض کنید داده ایی ۴ بایتی (۳۲ بیتی) 7E04F172 داریم که میخواهیم در آدرس ۱۰۰۰ ذخیر شود:

Little-Endian
1000::72
1001::F1
1002::04
1003::7E

Big-Endian

1000::7E
1001::04
1002::F1
1003::72

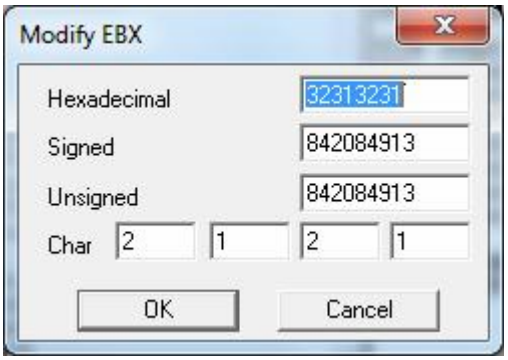
Big-Endian بایت پرارزش متغیر را در ابتدایی ترین خانه حافظه ذخیره میکند،

Little-Endian کم ارزش ترین بایت متغیر را در ابتدایی ترین خانه حافظه ذخیره مینماید.

```
- [*G.P.U* - main thread, module FAKE]
File View Debug Plugins Options Window Help Tools BreakPoint->

004011D8  .  6A 00          PUSH 0                                  wParam = 0
004011DA  .  6A 10          PUSH 10                                 Message = WM_CLOSE
004011DC  .  FF75 08        PUSH DWORD PTR [EBP+8]                  hWnd
004011DF  .  E8 C6000000    CALL <JMP.&user32.SendMessageA>         SendMessageA
004011E4  .v EB 62          JMP SHORT FAKE.00401248
004011E6  >  3D B90B0000    CMP EAX,0BB9
004011EB  .v 75 5B          JNZ SHORT FAKE.00401248
004011ED  .  6A 64          PUSH 64                                 Count = 64 (100.)
004011EF  .  68 78304000    PUSH FAKE.00403078                      Buffer = FAKE.00403078
004011F4  .  68 B80B0000    PUSH 0BB8                               ControlID = BB8 (3000.)
004011F9  .  FF75 08        PUSH DWORD PTR [EBP+8]                  hWnd
004011FC  .  E8 85000000    CALL <JMP.&user32.GetDlgItemTextA>      GetDlgItemTextA
00401201  .  8B1D 78304000  MOV EBX,DWORD PTR [403078]
00401207  .  80FB 61        CMP BL,61
0040120A  .v 75 2A          JNZ SHORT FAKE.00401236                 NABAYAD JMP SOORAT GIRAD!!
0040120C  .  8B1D 79304000  MOV EBX,DWORD PTR [403079]
00401212  .  80FB 62        CMP BL,62
00401215  .v 75 1F          JNZ SHORT FAKE.00401236                 NABAYAD JMP SOORAT GIRAD
00401217  .  8B1D 7A304000  MOV EBX,DWORD PTR [40307A]
0040121D  .  80FB 63        CMP BL,63
00401220  .v 75 14          JNZ SHORT FAKE.00401236                 NABAYAD JMP SOORAT GIRAD
00401222  .  68 52304000    PUSH FAKE.00403052                      Text = "That serial is correct!!!!
```

خب حالا پنجره ی ثبات خودمان برمیگردیم :



خب نمایش هگزا دسیمال داده های ما طبق مدل Little-Endian باید به صورت "31 32 31 32" باشد و در قسمت Char عدد ما برعکس نمایش داده شده ، در صورتی که عدد وارد شده ی ما باید 12 12 بود نه "21 21" پس ما از سیستم Little-Endian استفاده میکنیم

بیایید به دستور العمل بعدی نگاه کنیم :

CMP BL, 61

این دستور اولین بایت واقع در ثبات EBX با مقدار 61 (هگزا) مقایسه میکند ،یعنی چک میکند که اولین بایت از کد ما با 61 برابر است یا نه، که در خط بعد نتیجه ی این مقایسه با دستور ارجاعی:

JNZ SHORT FAKE.401236

به معنای نتیجه ی این مقایسه است ،کوتاه شده ی Jump Not Zero (بپر اگر صفر نیست)،پس معنی دوخط بالا به این شکل است:

مقایسه کن اولین بایت ورودی کاربر را، اگر برابر با 61 نیست به قسمت پیام بد پرش کن

خب به وضوح میتوانیم ببینیم اولین بایت ما 31h است نه 61h،بنابراین پرش به قسمتی که ما نمیخواهیم انجام خواهد شد،☹ اما صبر کنید Ollydbg یه دیباگر dynamic است ، یعنی ما میتوانیم تغییرات را در زمان اجرا اعمال و نتیجه را مستقیم ببینیم ☺ اما قبل از ادامه ما نیاز داریم درباره ی Flag با پرچم چیزهایی بدانیم :

CPU Flags

مجموعه ایی از بیت ها که تک تک بیت های آن به صورت مستقل تحت تاثیر عملیات محاسباتی یا منطقی قرار می گیرند که در ALUانجام می شود.اگر بخواهیم مثالی زده باشیم در زبان های سطح بالا چنین خواهد بود :

```
if( serialNumber == 3 )
dontShowNag();
else
showNag();
```

شبه مونتاژ در زبان اسمبلی

```
compare serialNumber with 3
jump (if they are equal) to dontShowNag();
jump to showNag();
```

و در زبان واقعی اسمبلی:

```
MOV EAX, addressOfSerialNumber
CMP EAX, 3
JE addressOfDontShowNag
JMP adressOfShowNag
```

خط اول ثبات EAX با شماره ی سریال بارگذاری میشود ،و در خط دوم مقدار 3 با مقدار EAX مقایسه میشود و در خط سوم اگر نتیجه ی مقدار EAX با مقدار 3 برابر بود به قسمت addressOfDontShowNag ارجاع خواهیم شد و درغیر این صورت در خط چهارم به قسمت adressOfShowNag ارجاع داده خواهیم شد،مهمترین ثبات برای ما در مهندسی معکوس دو پرچم Z f وCarry هستند که با "Z" و "C" نشان داده میشوند

پرچمZ

پرچم صفر نامیده می شود و به اختصار آن را با ZF نشان می دهیم و مشخص کننده صفر یا غیر صفر بودن حاصل عملیات است. پس از اتمام عملیات اگر حاصل عملیات برابر صفر باشد این بیت یک خواهد شد، در غیر این صورت این بیت صفر می شود.

پرچمC

پرچم نقلی بوده و به اختصار CF نامیده می شود. مشخص کننده بیت (نقلی انتقالی) خروجی مربوط به باارزش ترین بیت در طی عملیات محاسباتی است. بیت شیفت یافته در عملیات جابجایی نیز در این پرچم قرار می گیرد. این بیت همچنین مشخص کننده بیت قرضی در عملیات تفریق نیز هست. عملیات ممکن است به صورت ۸ بیت یا ۱۶ بیت باشد. بنابراین به طور خلاصه می توان گفت که اگر پس از عملیات محاسباتی بیت نقلی یا بیت قرضی وجود داشته باشد، این پرچم یک خواهد شد، در غیر این صورت صفر می شود.

به برنامه باز میگردیم ،و یک بار کلید F8 را میفشاریم :

همانطور که میبینید بروی دستور JNZ ایستاده ایم که نشان دهنده ی عبارت "پرش کن اگر نتیجه 0 نیست" و در قسمت پنجره ی ثبات ها میبینیم :

میبینیم که مقدار این پرچم برابر با 0 است ، که یعنی نتیجه مقایسه ی ثبات EBXبا مقدار 61 صفر یا غلط است ،پس پرش انجام خواهد شد ،همچنین در پنجره ی Info حالت این پرش گزارش شده است :

که نمایانگر انجام پرش است

حالا بروی پرچم Z دبل کلیک کنید با انجام این کار مقدار 0 به 1 تغییر خواهد کرد:

به محض انجام علامت قرمز به رنگ خاکستری تغییر پیدا کرد که به معنی صحیح بودن مقایسه است :

حالا دوبار دیگر کلید F8 را بفشارید باز در خط بعد (401217)مقایسه ایی صورت گرفته مانند مقایسه ی قبلی ،میبینید که دوباره وضعیت پرچم Z تغییر کرده ،آن را به 1 تغییر دهید ودوباه دوبار F8 بزنید و همین کار هم برای آخرین JNZ در آدرس (401220) انجام دهید یعنی وضعیت پرچم را از 0 به 1 تغییر دهید ،سپس کلید F9 را بفشارید:»