سری آموزش های R4NDOM-فارسی

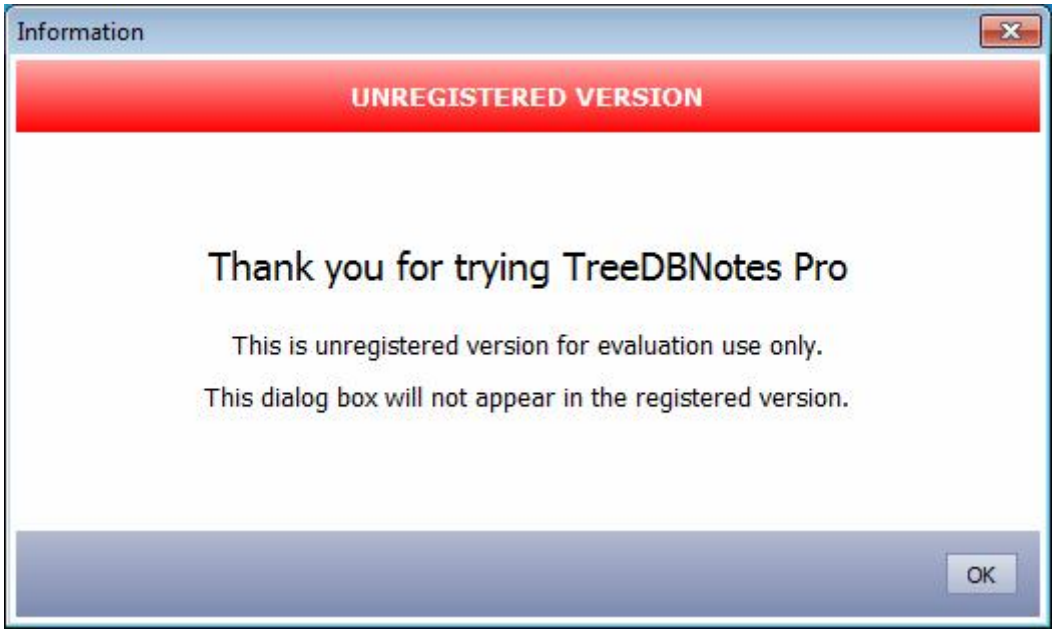
قسمت ۱۳-کرک یک برنامه ی واقعی

در این قسمت ما قصد داریم یک برنامه ی تجاری را کرک کنیم(ضمیمه شده) این برنامه محدودیت زمانی دارد و پس از به پایان رسیدن زمان دیگر کار نمیکند حال بیایید شروع کنیم :

بررسی عملکرد برنامه:

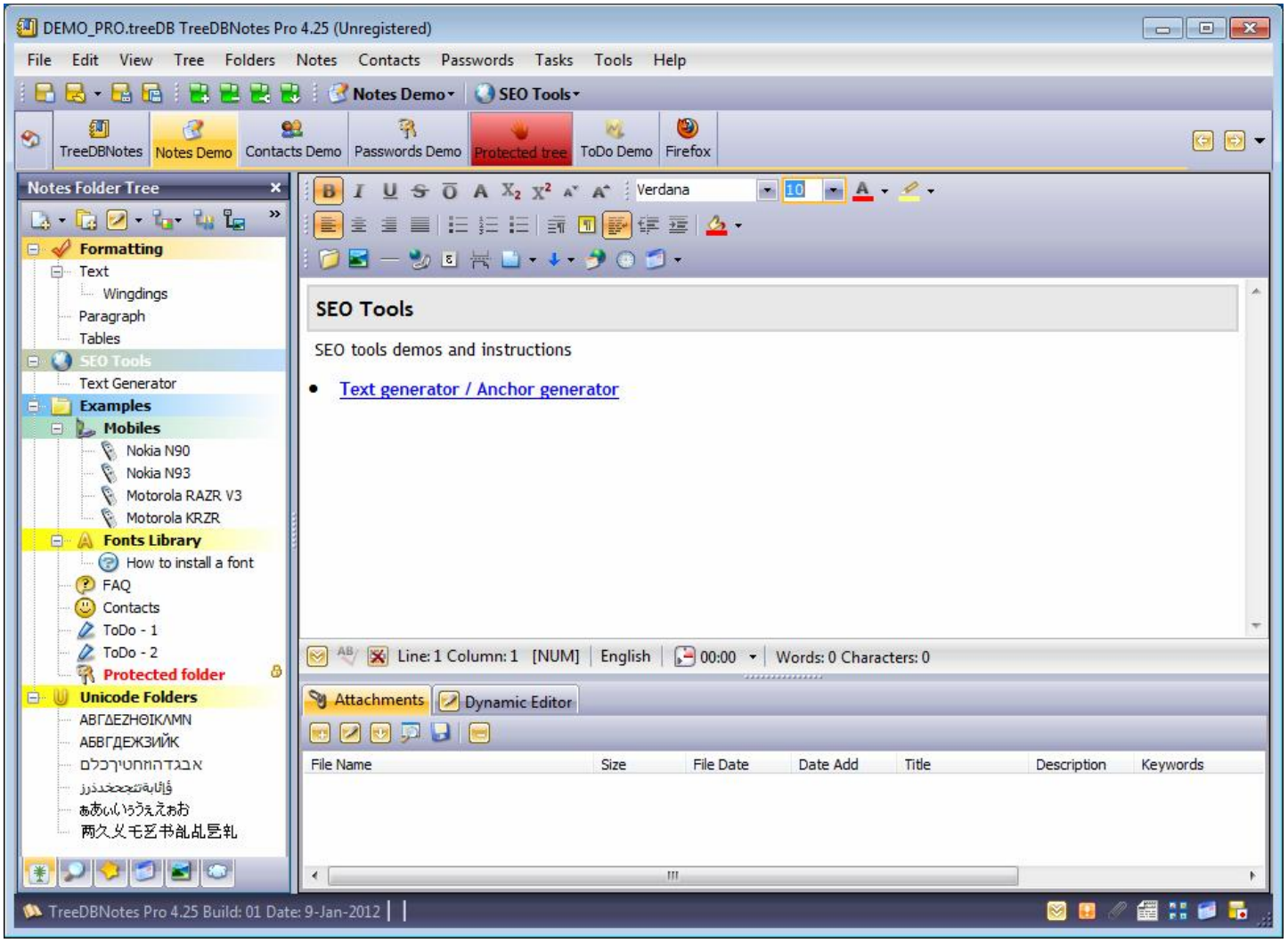
ابتدا برنامه را نصب کنید و در انتها :

بسیار خوب برنامه را اجرا کنید :



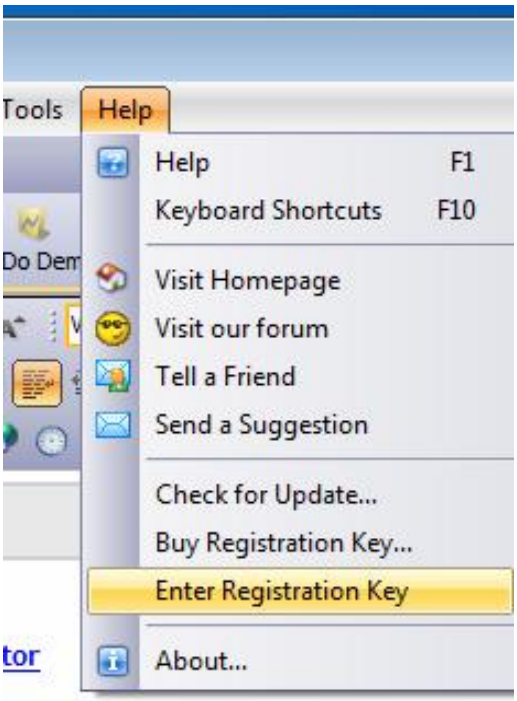
بسیار خوب،البته این خیلی خوشایند نیست ،در این صفحه رشته های مانند :evalution,registerd,unregistered را میبینید،بروی Ok کلیک کنید تا برنامه اجرا شود:

بروی (Open demo file) کلیک کنید تا وارد پنجره ی اصلی شویم :

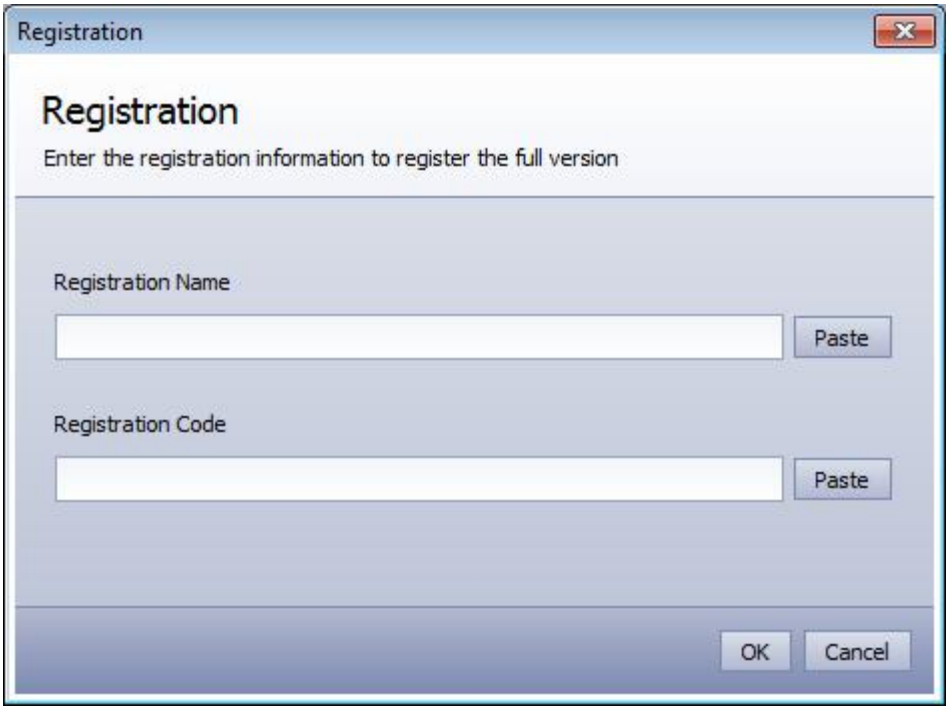


به تایتل برنامه نگاه کنید میبینید که به نوشته شده Unregisterd،محل های دیگری هم برای تست محدودیت های برنامه وجود دارد مانند پنجره ی About ،اساسا هرچیزی که به ما سرنخ بدهد مهم است ،حال به پنجره ی About از منوی Help میرویم :

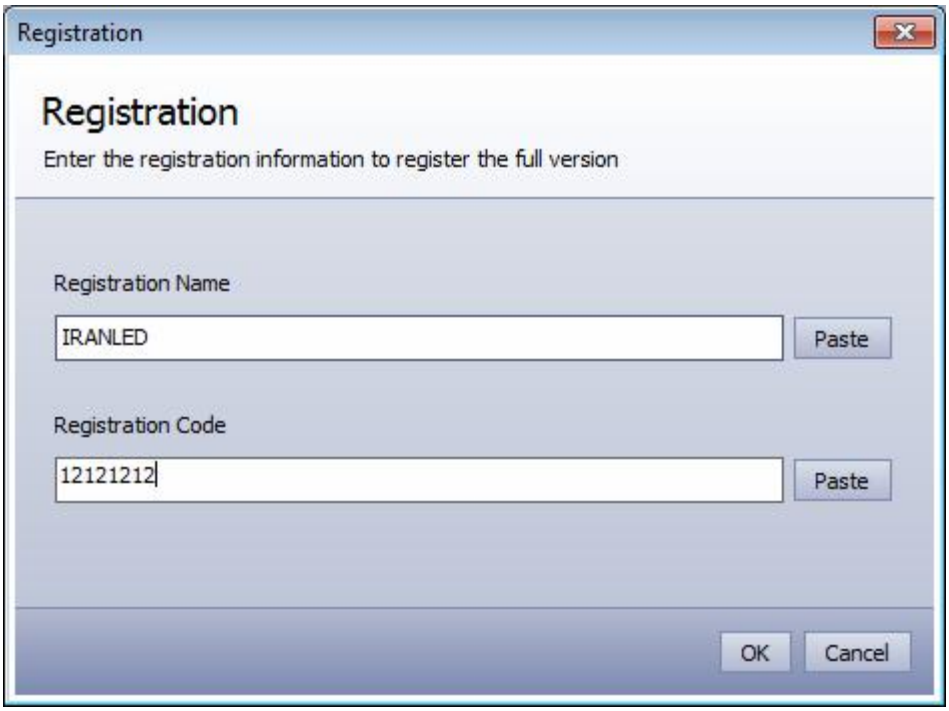
همانطور که میبینید اینجا هم دوباره رشته ی Unregistered را میبینیم ،باز به منوی Help نگاه کنید :



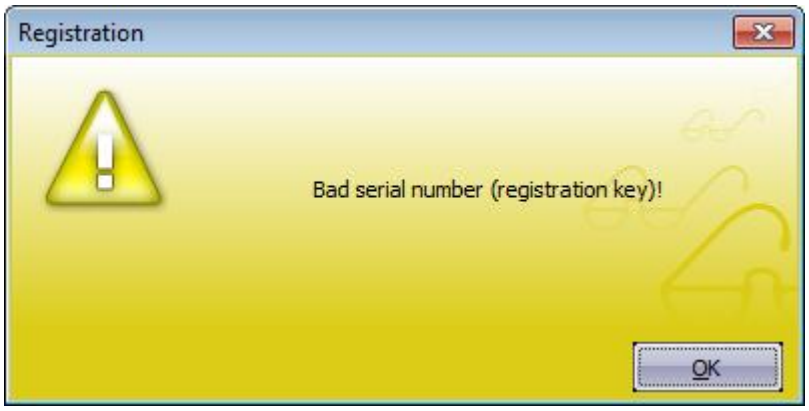
سرنخ خوبی است اینطور نیست؟این سرنخ ها موقعی بارزش تر میشوند که مثلا تکنیک "search for strings" جواب ندهد حال بروی Enter Registrayion Key کلیک کنید:



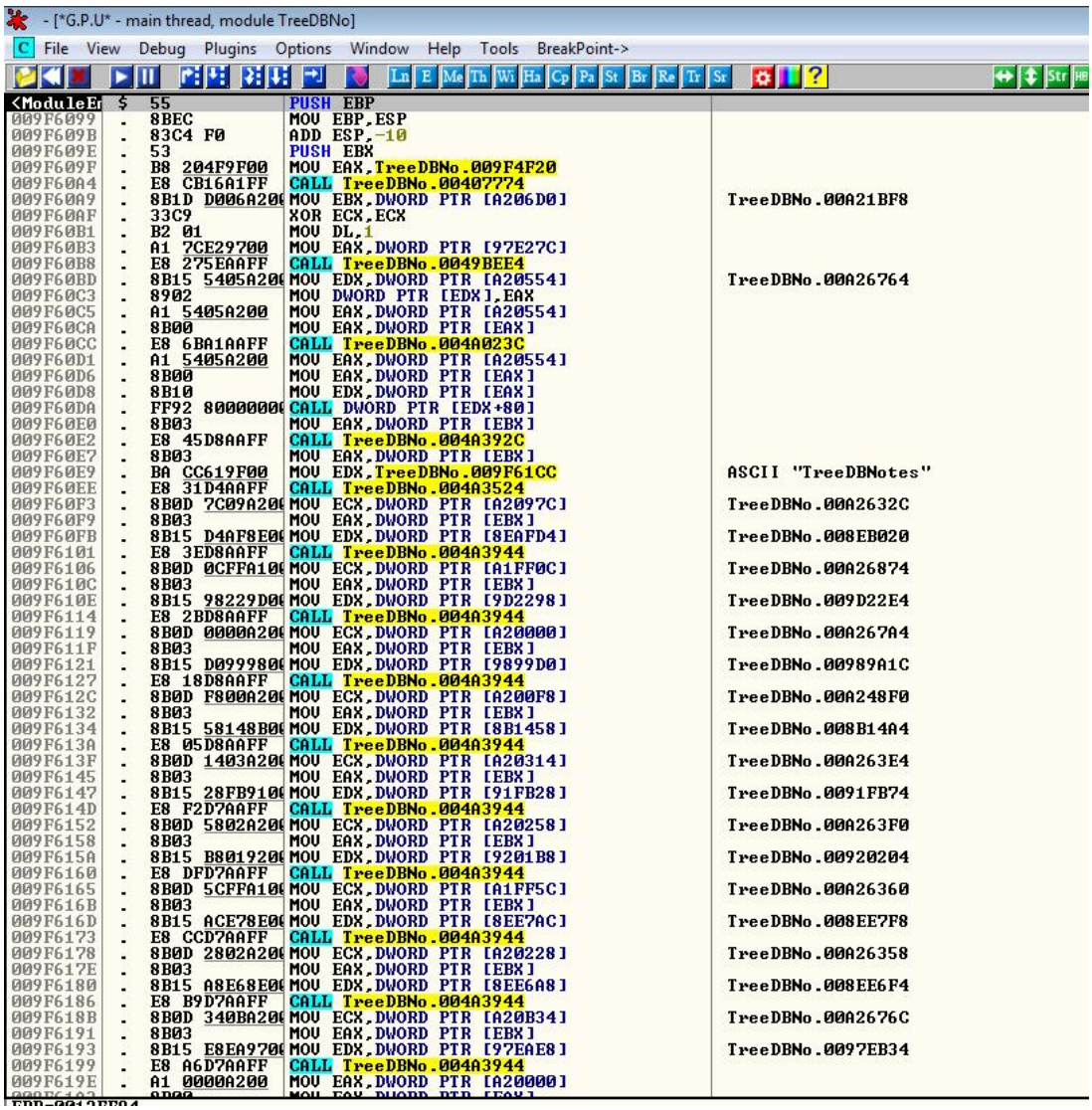
حال طبق معمول همیشه یک نام و کد را وارد کنید و کلید OK را بفشارید :



و بعد از کلیک بروی OK:



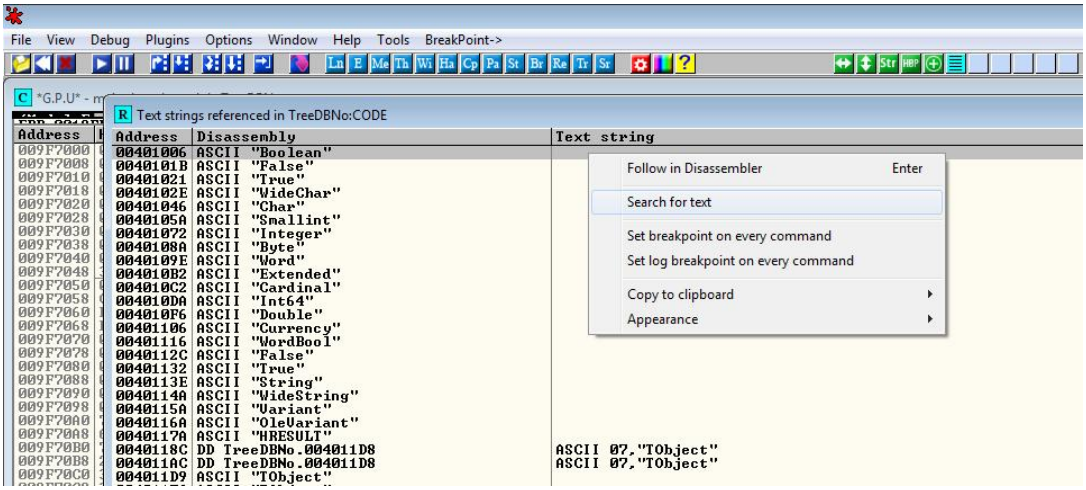
انتظارش را داشتیم ،نه؟بسیار خوب ،برنامه را درون Olly باز کنید :



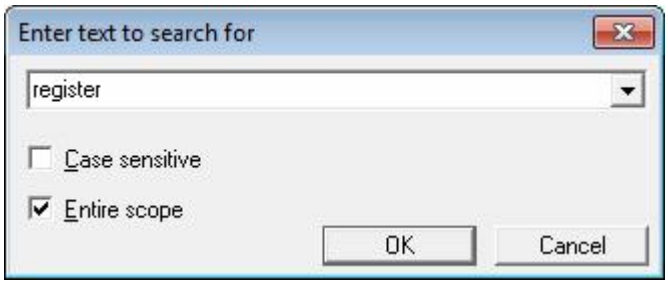
شاید در نگاه اول کمی گیج کننده بیاید چون تا به حال چنین چیزی ندیده ایم ، Call های متعدد بدون هیچ گونه توضیحی ،این نشانه حاکی از این است که این برنامه با زبان Delphi نوشته شده است ،البته در آموزش های بعدی روش تشخیص مفصلا بحث خواهد شد همچنین روش آنالیز برنامه های دلفی ،اما خوشبختانه در این مثال نیازی نیست

جستجو

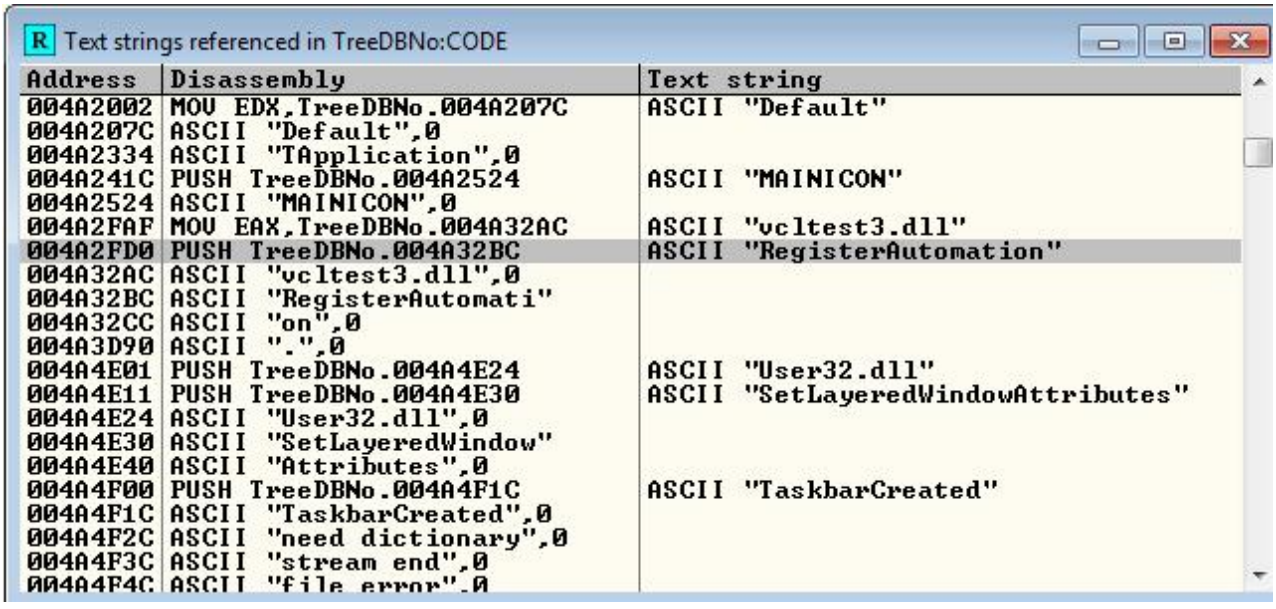
ابتدا بیایید رشته های بکار رفته در برنامه را از طریق "All referenced text strings" -> "Search for" ببینیم ،بعد از نمایش رشته ها متوجه فراوانی آن ها میشویم و برای سرعت بخشیدن به کار رشته ی مورد نظر را جستجو میکنیم :



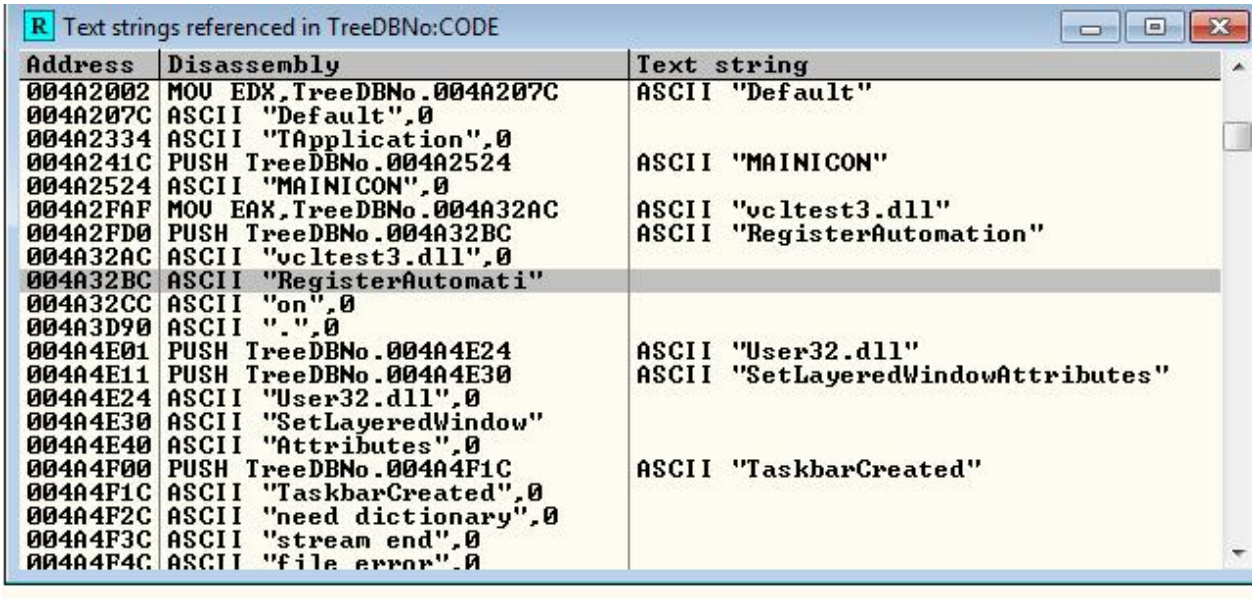
معمولا جستجو را با کلماتی مانند Registerd یا Registraion شروع میکنیم ،اما من معمولا از رشته ی Register استفاده میکنم چون در این حالت هم Registration وهم Registerd پوشش داده خواهند شد بسیار خوب :



مطمئن شوید که چک باکس Entries cope در حالت انتخاب شده و "Case sensitive" غیر فعال باشد :

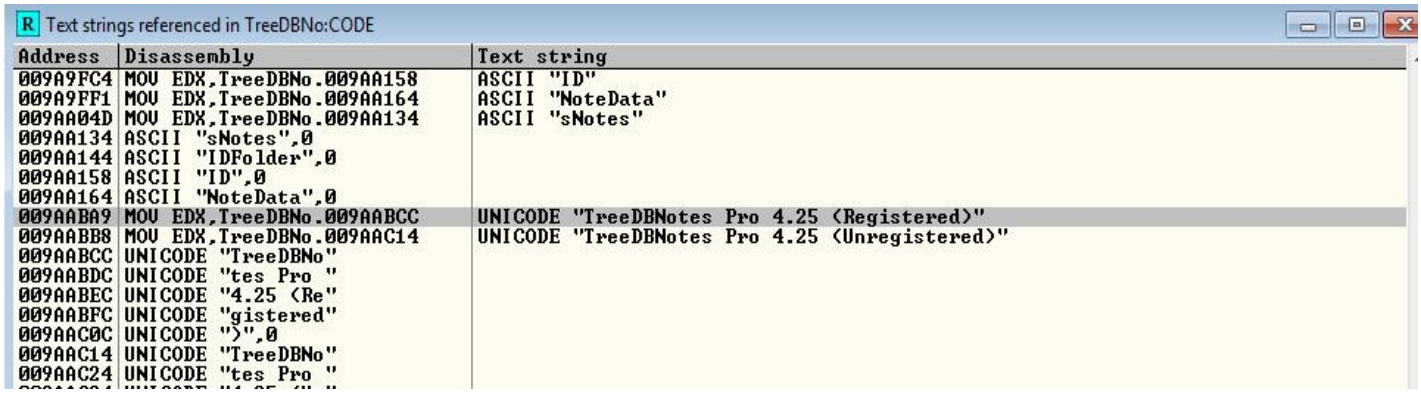


این اولین نتیجه ی جستجو است و به نظر به درد نمیخورد با فشردن کلید Ctrl +L جستجو را ادامه میدهیم:

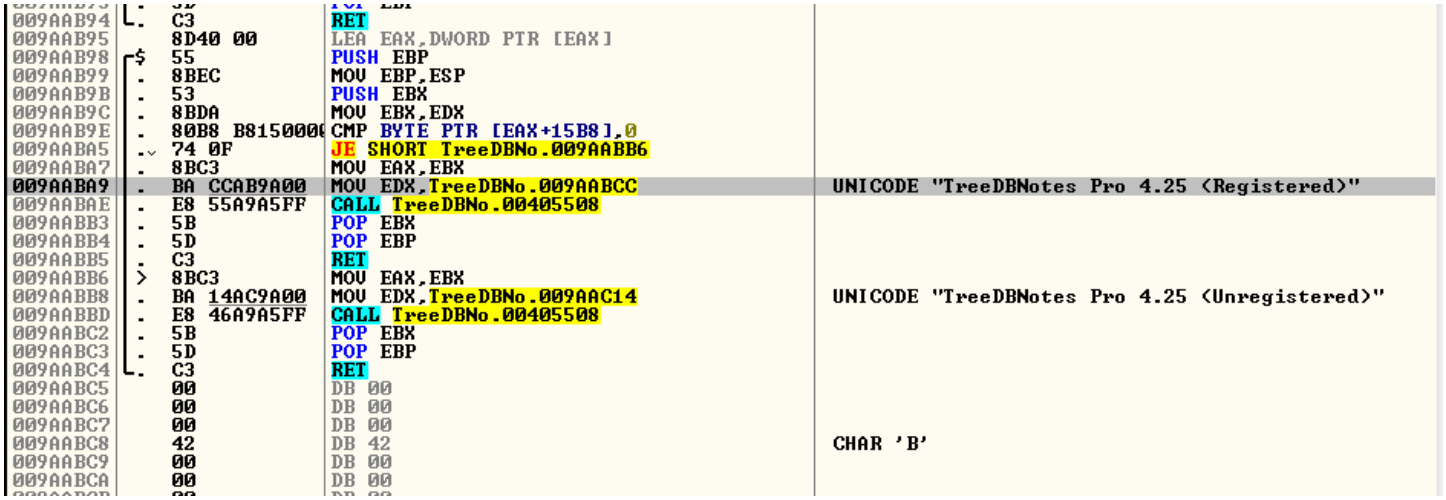


یکی دیگر☺ آیا میدانید فرق این دو چیست؟هر دو یک رشته ی یکسان هستند با این تفاوت که اولی "RegisterAutomation" درون پشته قرار گرفته و دومی "RegisterAutomation" یک داده ی واقعی در برنامه است.در سطون دوم عبارت ASCII را میبینید ،باید بدانید که اکثرا از هر رشته ۲ تا است ، اولی برای دستیابی به آن و دومی محل واقعی آن است

بسیار خوب دوباره کلید Ctrl +L را چند باری بفشارید تا به رشته زیر برسید:



بهتر شد ،به نظر میرسد که این محلی است که تاتیل برنامه از آن استفاده میکند و میتواند نقطه ی شروع خوبی باشد بروی آن دبل کلیک کنید تا به محل آن برویم :



همانطور که در تصویر بالا میبینید اولین چیزی که میبینیم این است که رشته در آدرس 9AABA9 مورد استفاده قرار گرفته و دومین چیز این است که این رشته درآدرس حافظه ی 9AABCC ذخیره شده است ،حالا به خط 9AABA5 دقت کنید :

```
009AAB9B  .  53              PUSH EBX
009AAB9C  .  8BDA            MOV EBX,EDX                          TreeDBNo.<ModuleEntryPoint>
009AAB9E  .  80B8 B8150000 00 CMP BYTE PTR DS:[EAX+15B8],0
009AABA5  .v 74 0F           JE SHORT TreeDBNo.009AABB6
009AABA7  .  8BC3            MOV EAX,EBX
009AABA9  .  BA CCAB9A00     MOV EDX,TreeDBNo.009AABCC            UNICODE "TreeDBNotes Pro 4.25 (Registered)"
009AABAE  .  E8 55A9A5FF     CALL TreeDBNo.00405508
009AABB3  .  5B              POP EBX                              kernel32.7747ED6C
009AABB4  .  5D              POP EBP                              kernel32.7747ED6C
009AABB5  .  C3              RETN
009AABB6  >  8BC3            MOV EAX,EBX
009AABB8  .  BA 14AC9A00     MOV EDX,TreeDBNo.009AAC14            UNICODE "TreeDBNotes Pro 4.25 (Unregistered)"
009AABBD  .  E8 46A9A5FF     CALL TreeDBNo.00405508
009AABC2  .  5B              POP EBX                              kernel32.7747ED6C
009AABC3  .  5D              POP EBP                              kernel32.7747ED6C
009AABC4  L. C3              RETN
009AABC5     00              DB 00
009AABC6     00              DB 00
009AABC7     00              DB 00
009AABC8     42              DB 42                                CHAR 'B'
```

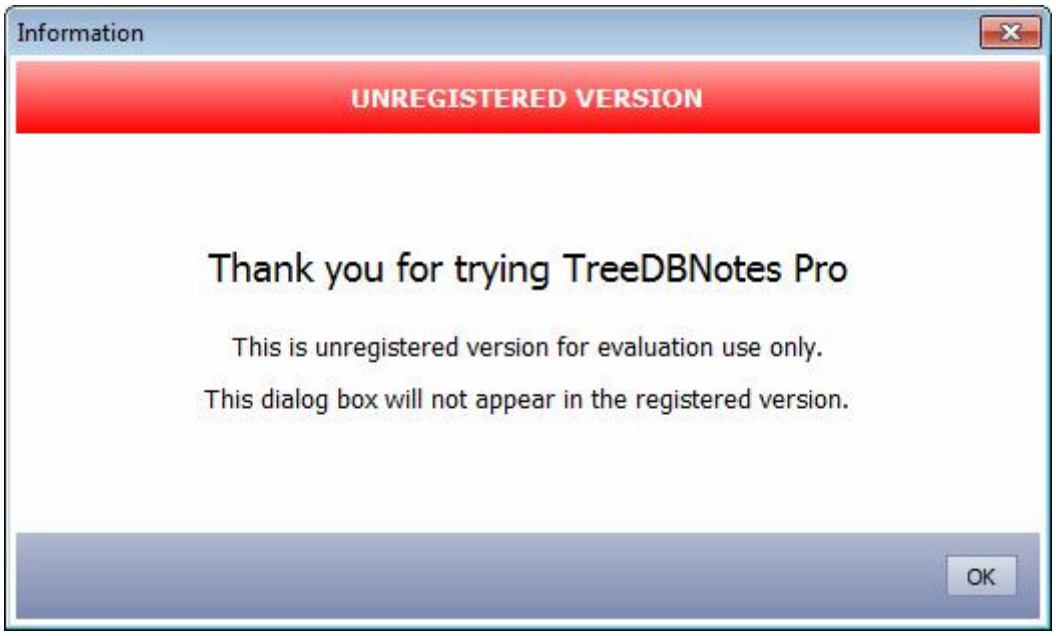
یک دستور العمل پرشی میبینیم که به این معنی است که اگر نتیجه برابر باشد ما به قسمت "unregistered" پرش خواهیم کرد از آنجایی که ما نمیخواهیم چنین اتفاقی رخ دهد بروی این دستورالعمل یک BP گذاشته و برنامه را اجرا میکنیم :

```
009AAB94  L. C3              RETN
009AAB95     8D40 00         LEA EAX,DWORD PTR DS:[EAX]
009AAB98  r$ 55              PUSH EBP
009AAB99  .  8BEC            MOV EBP,ESP
009AAB9B  .  53              PUSH EBX
009AAB9C  .  8BDA            MOV EBX,EDX                          TreeDBNo.<ModuleEntryPoint>
009AAB9E  .  80B8 B8150000 00 CMP BYTE PTR DS:[EAX+15B8],0
009AABA5  .v 74 0F           JE SHORT TreeDBNo.009AABB6
009AABA7  .  8BC3            MOV EAX,EBX
009AABA9  .  BA CCAB9A00     MOV EDX,TreeDBNo.009AABCC            UNICODE "TreeDBNotes Pro 4.25 (Registered)"
009AABAE  .  E8 55A9A5FF     CALL TreeDBNo.00405508
009AABB3  .  5B              POP EBX                              kernel32.7747ED6C
009AABB4  .  5D              POP EBP                              kernel32.7747ED6C
009AABB5  .  C3              RETN
009AABB6  >  8BC3            MOV EAX,EBX
009AABB8  .  BA 14AC9A00     MOV EDX,TreeDBNo.009AAC14            UNICODE "TreeDBNotes Pro 4.25 (Unregistered)"
009AABBD  .  E8 46A9A5FF     CALL TreeDBNo.00405508
009AABC2  .  5B              POP EBX                              kernel32.7747ED6C
009AABC3  .  5D              POP EBP                              kernel32.7747ED6C
009AABC4  L. C3              RETN
009AABC5     00              DB 00
009AABC6     00              DB 00
009AABC7     00              DB 00
```

همانطور که انتظار داشتیم Olly به روی دستور العمل JE متوقف شده ما اکنون در آستانه ی ارجاع به Bad Boy هستیم ، با تغییر پرچم Z این ارجاع با Good Boy تعویض خواهد شد :

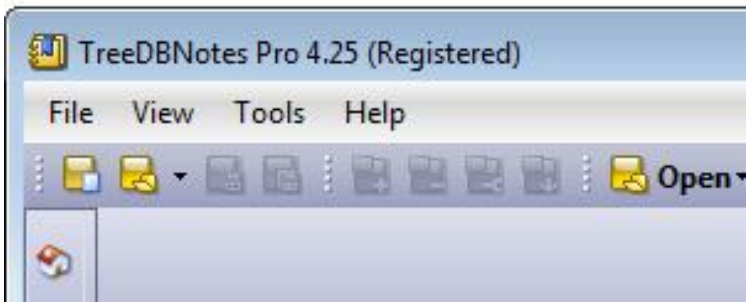
```
C 0  ES 0023 32bi
P 1  CS 001B 32bi
A 0  SS 0023 32bi
Z 0  DS 0023 32bi
S 0  FS 003B 32bi
T 0  GS 0000 NULL
D 0
O 0  LastErr ERRO
```

و برنامه را با کلید F9 اجرا کنید ،میبینید که olly مجددا بروی همان آدرس متوقف شده که میخواهد به bad Boy پرش کند،دوباره پرچم Z را تغییر دهید و برنامه را با F9 اجرا کنید،اگر olly دوباره متوقف شد ،باز مقدار پرچم Z را تغیر دهید و برنامه را با F9 اجرا کنید :

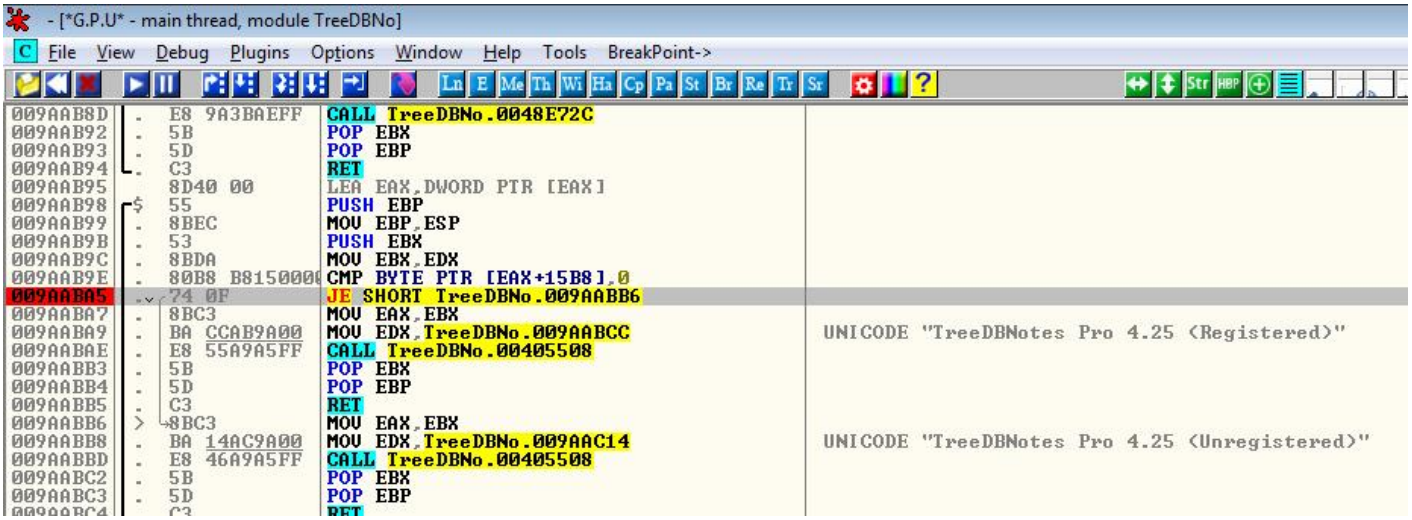


میبیند که پچ ما بروی این قسمت جواب نداده ،حال اگر کلید Ok را بفشارید ollyباز در همان آدرس چندین بار متوقف میشود و شما هم به همان تعداد پرچم Z را تغییر دهید تا به سمت Good Boyپرش کنید و برنامه را با F9 اجرا کنید :

بروی Close کلیک کنید و باز olly متوقف میشود و باز هم مقدار پرچم Z را در صورت نیاز تغییر دهید تا موقعی که برنامه بالا بیاید:



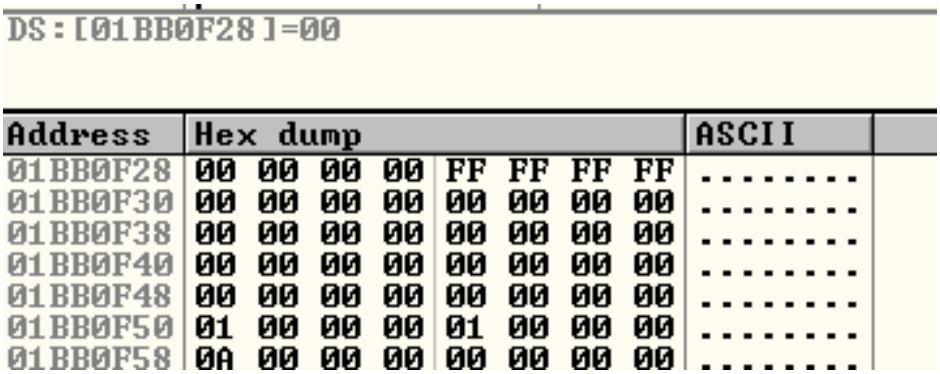
بسیار خوب ،برنامه را ری استارت کنید و به آدرس قبلی باز گردید :



اگر متوجه شده باید هیچ فراخوانی قبل از آدرس 9AABA5 وجو ندارد ولی در آدرس 9AAB9E یک مقایسه وجود دارد:

CMP BYTE PTR DS:[EAX+15B8],0

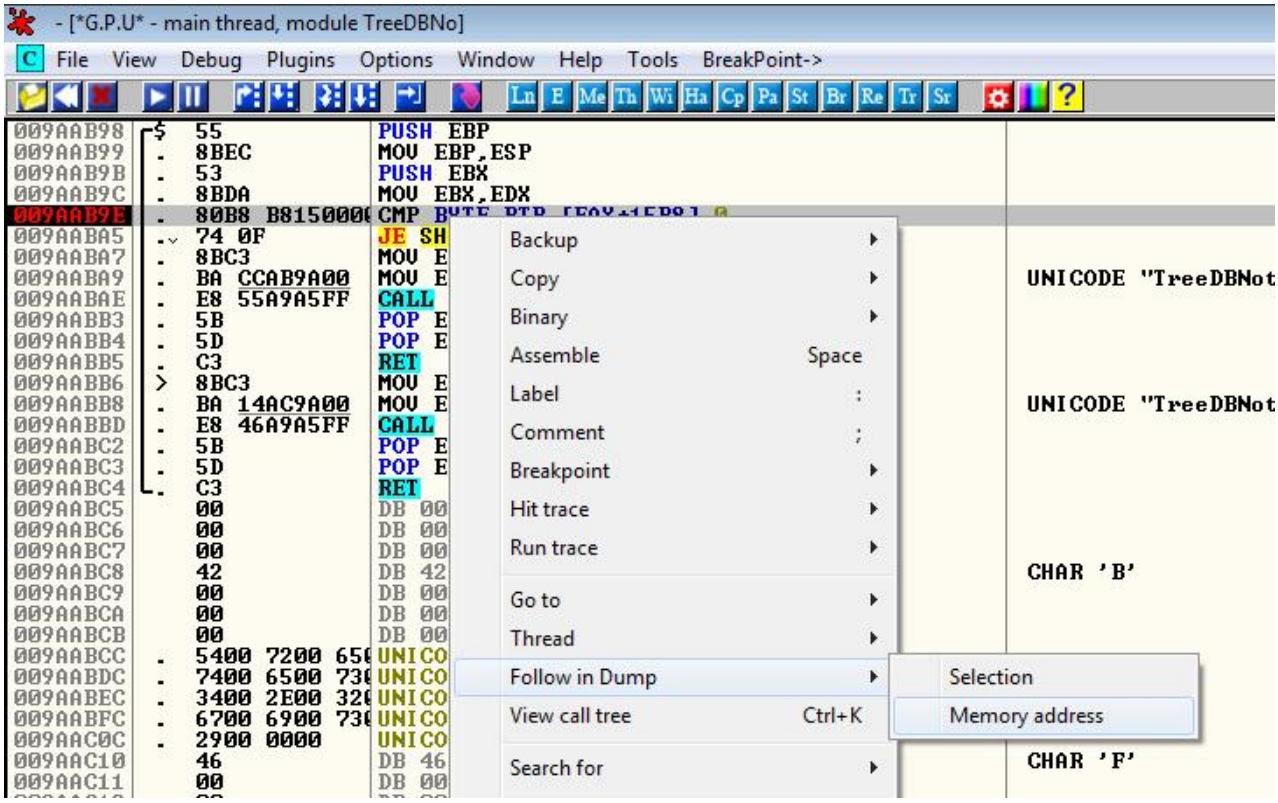
این دستور چک میکند که آیا برنامه رجیستر شده یا نه ، یک بایت از آدرس حافظه ی [EAX+15B8] واقع در DS را که متغییری است از نوع Global که وظیفه ی چک کردن وضعیت رجیستر شدن برنامه را به عهده دارد ،بیایید ببینیم در آدرس [EAX+15B8] چه مقداری وجود دارد ،با راست کلیک کردن بروی خط مذکور و Fllow in Dump:



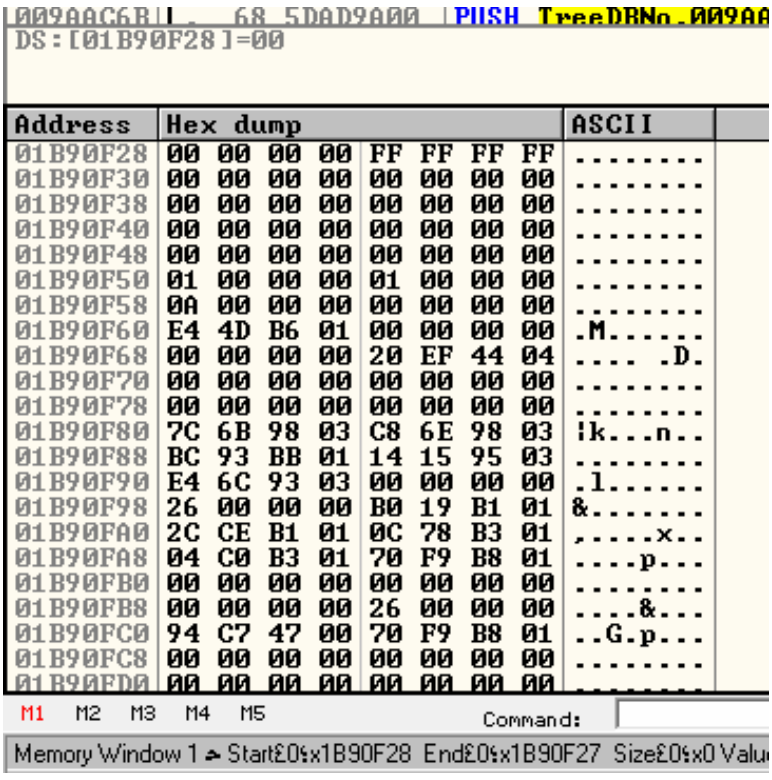
نکته:آدرس یقینا در سیستم شما متفاوت خواهد بود

همانطور که میبینید اولین بایت 00 از آدرس 01BB0F28 مسئول وضعیت ریجیستر شدن را به عهده دارد، به این معنی که اگر مقدارش هر چیزی بجز صفر باشد برنامه ریجیستر خواهد شد

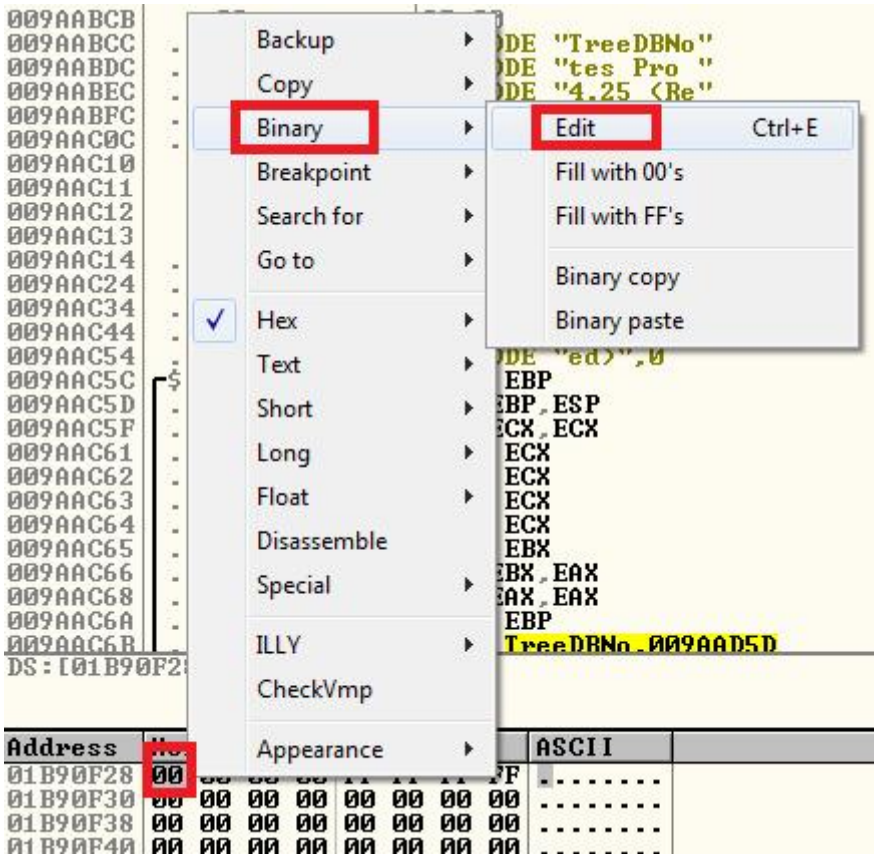
بسیار خوب بیایید بروی خط 9AAB9E یک BP بگذارید و بقیه ی BP ها را بردارید ،بعد برنامه را ری استارت کنید، همانطور که مشاهده میکنید Olly متوقف شده است حال بروی خط مذکور راست کلید کرده "Follow in dump" :



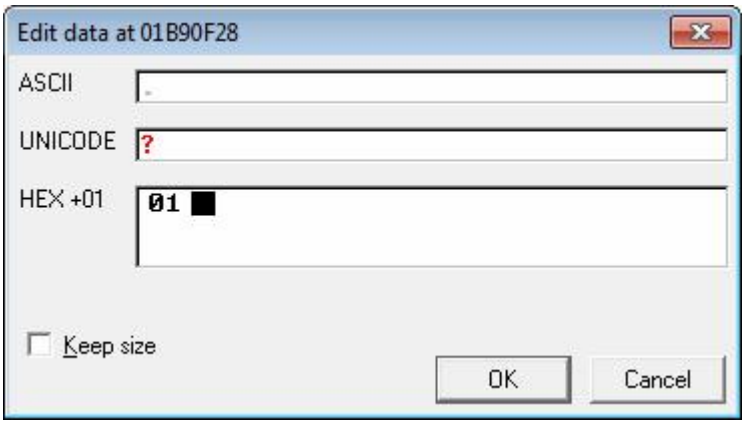
و در پنجره ی Dump:



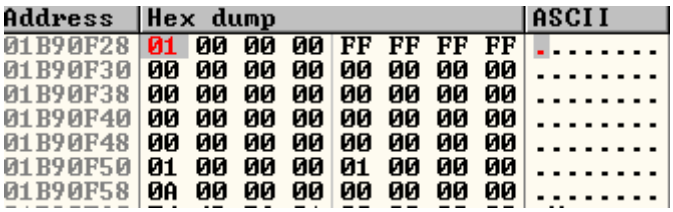
بله ☺ همانطور که میبینید این آدرس با آدرس قبلی متفاوت هست (در سیستم شما هم همینطور است) اگـر خاطرتـان باشـد آدرس اولی 01BB0F28 بود و این آدرس 01B90F28 است ، درواقع هر بار که برنامه را ری استارت کنید این آدرس تغییر خواهـد کرد،حـال بروی آدرس 01B90F28 (همون که جدیده ☺) در پنجره ی دامپ راست کلیک کنید :



و بعد مقدار 00 را به 01 تغییردهید :



سپس کلید Ok را بفشارید تا تغییرات اعمال شود :

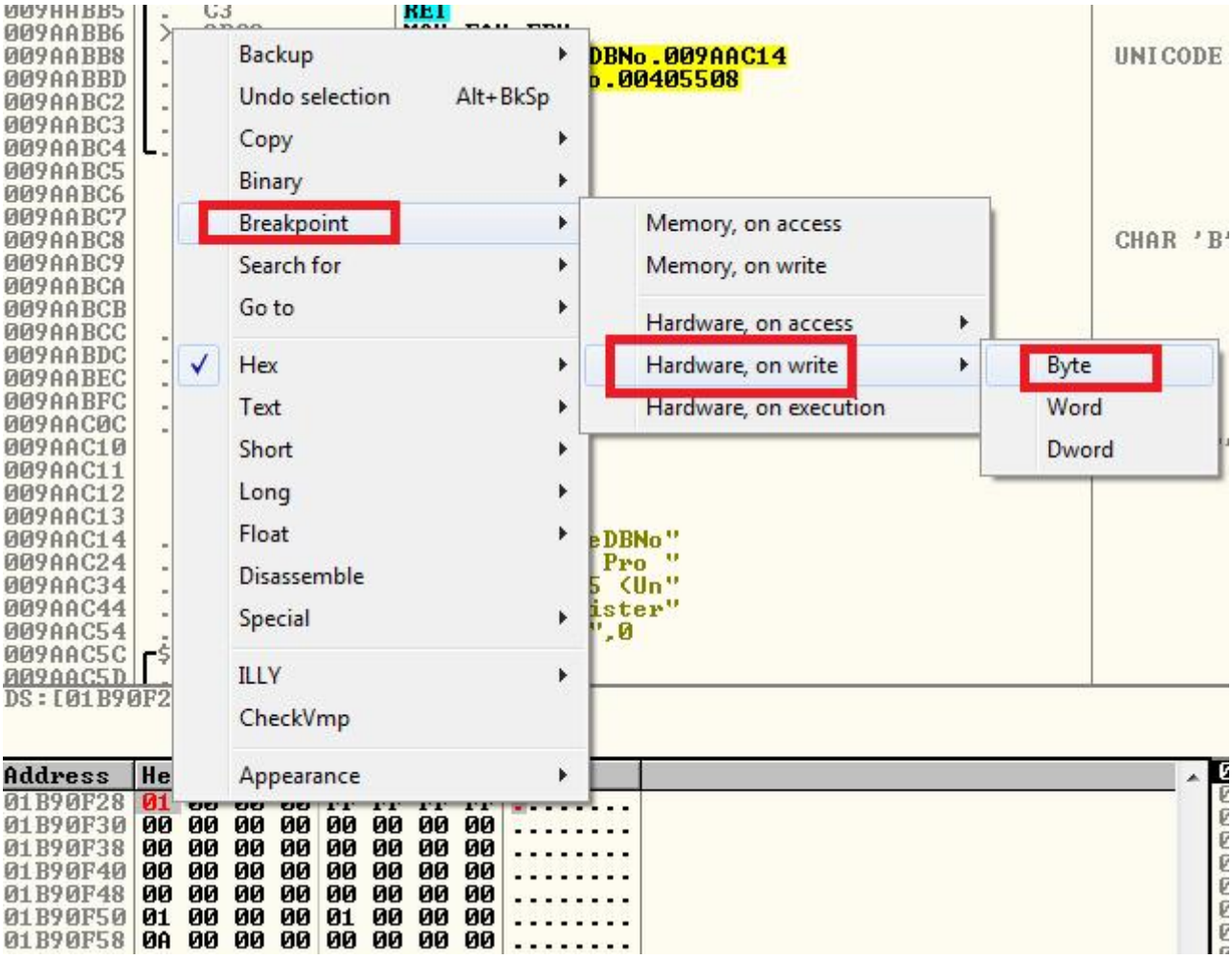


حالا یک بار کلید F8 را بفشارید میبینید که پرش به سمت Good Boy میرود چرا که مقدار [EAX+15B8] دیگر صفر نیست ،سپس با فشردن کلید F9 برنامه را اجرا کنید ،بله ، برنامه دوباره در همان آدرس قبلی 9AAB9E متوقف شده و مقدار آدرس 01B90F28 دوباره به 00 تغییر کرد ☹. حال به Bad Boy میرویم ☹ ،اما این تغییر مجدد 00 به 01 یعنی یک جای دیگر از برنامه روتین چک مجدد دارد ☺

این روتین دوباره مقدار 01 را به 00 برمیگرداند ، پس باید این قسمت پیدا شود ،اما چطوری؟سناریو اینگونه است ،اگر این چک دوم قادر است که بایت ما را پیدا کند و آن را به 00 تغییر دهد ،پس ما هم میتوانیم موقعیت آن را هنگامی که میخواهد بایت ما را صفر کند پیدا کنیم ☺ برای این کار ما نیاز به یک hardware breakpoint داریم . که در آدرس حافظه ی 01B90F28 باید بروی بایت اول که الان 00 شده قرار دهیم ،این کار به olly میگوید موقعی که هرجای برنامه میخواهد در این آدرس چیزی بنویسد ،آن را متوقف کن ☺

پس از یافتن olly به محل قسمتی میرود که قصد داشته بایت ما را به 00 تغییر دهد:

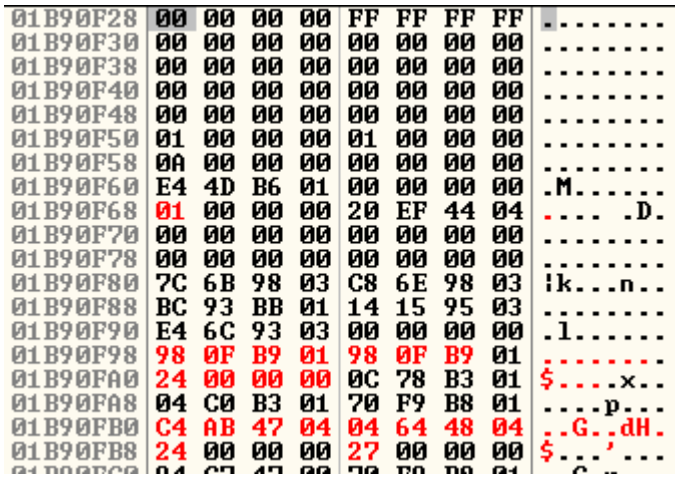
اگر کماکان در آدرس 01B90F2 (در سیستم شما متفاوت است) هستید و از آن عبور نکرده اید (در غیر اینصورت برنامه را ری استارت کنید و دوباره بروی Fllow in Dump کلیک کنید و مقدار 00 را به 01 تغییر دهید) سپس بروی آدرس مورد نظر (در بایت اول 01) راست کلیک کنید و:



طبق تصویر بالا عمل کنید ،اما باید بگویم ما از خصوصیت Byte و Write در نقطه ی توقف سخت افزاری، استفاده کرده ایم چون میخواهیم ببینیم کجای برنامه میخواهد یک بایت 01 واقع در آدرس 1B90F28 را با مقدار 00 جایگزین کند و همچینین ما از نقطه ی توقف سخت افزاری استفاده کرده ایم چون برای برنامه خیلی مشکل است که بتواند آن را شناسایی کند، حال برنامه را باکلید F9 اجرا میکنیم :

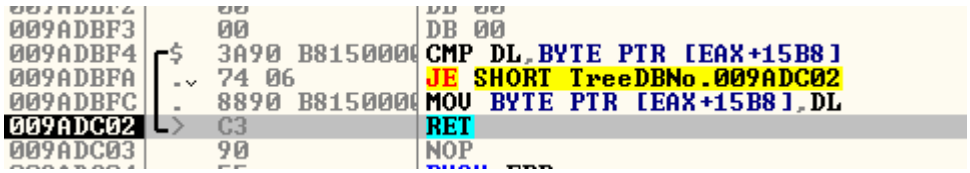
همانطور که میبینید olly در این آدرس متوقف شده و در پایین میبینید که علت این توقف نقطه ی توقف سخت افزاری ما بوده:

و همینطور در پنجره ی دامپ میبینید که مقدار بایت ما برابر با 00 شده :

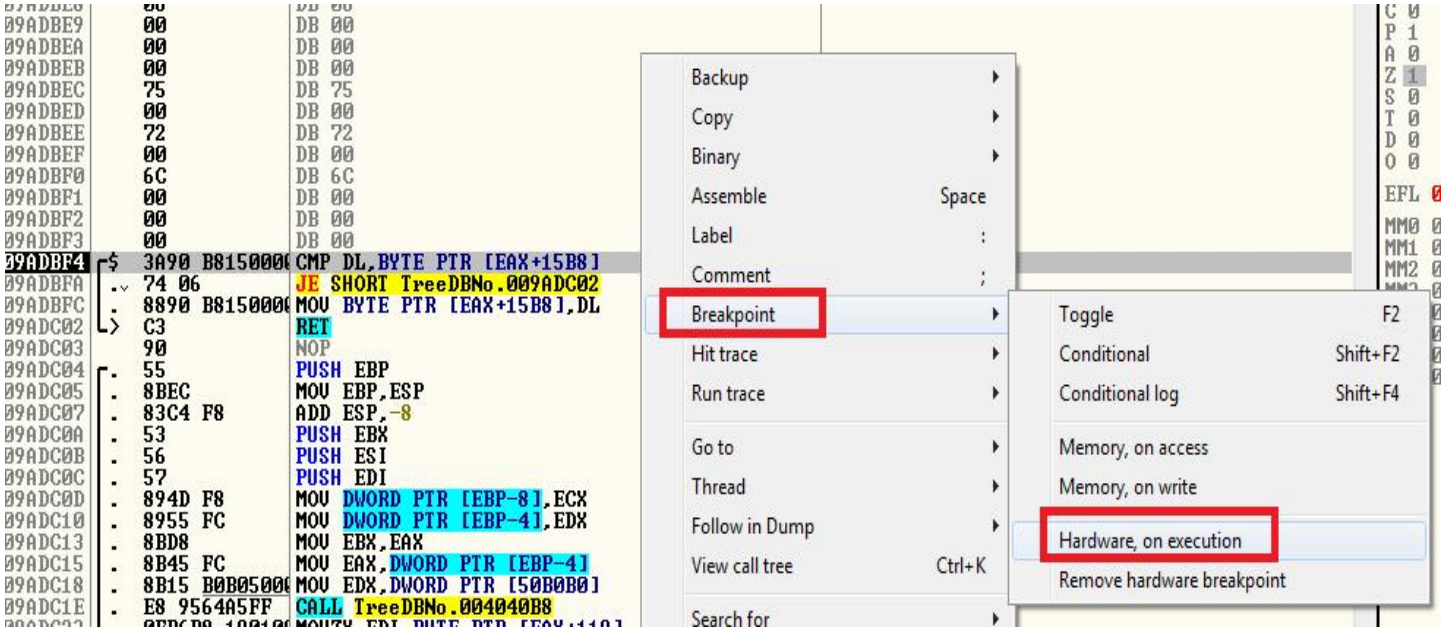


طریقه ی پچ :

اگر بخواهیم برنامه را پچ کنیم لازم است محل توقف سخت افزاری را کمی مطالعه کنیم :



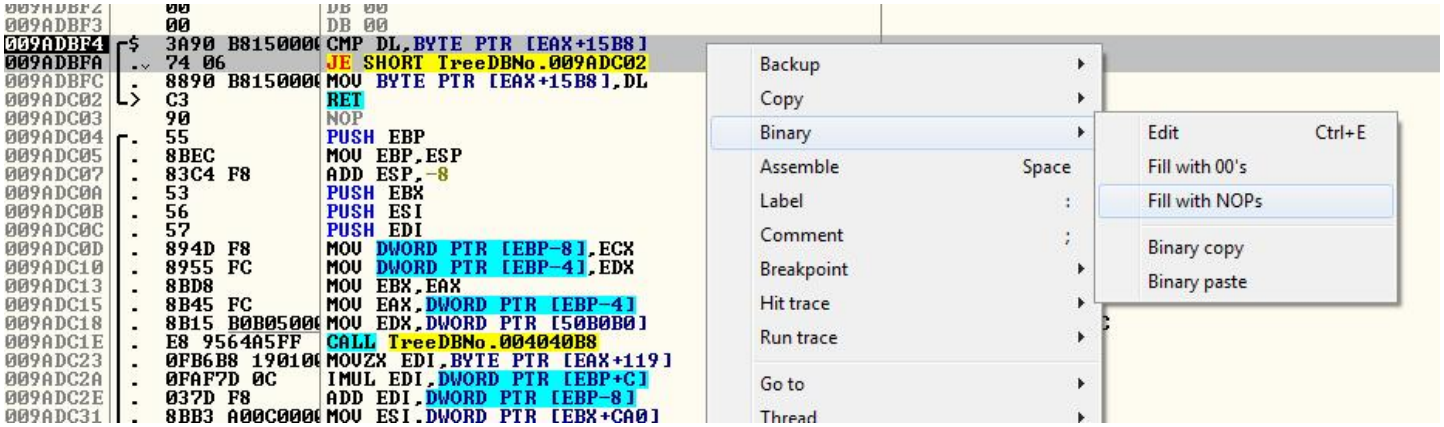
در خط 9ADBF4 مقدار یک بایت از آدرس [EAX+15B8] (یادتان هست؟) که الان 00 است را با DL مقایسه میکند و در خط بعد اگر مساوی بود به خط 9ADC02 پرش میکنیم و در خط بعد مقدار DL را به آدرس [EAX+15B8] منتقل میکند و خط بعد هم یک دستور ساده ی بازگشت است،قبل از هر چیز ابتدا نقطه ی توقف سخت افزاری را پاک کنید برای اینکاراز منوی “Debug” -> “Hardware breakpoints” را انتخاب و آن را Delete کنید ،و اما مقدار DL=0 است چون ما در آدرس حافظه ایی 1B90F28 این مقدار را دیدیم که از یک به صفر تغییر کرد ،پس پرش به سمت bad Boy صورت میگیرد،حالا دوباره نیاز به یک نقطه ی توقف سخت افزاری داریم که روی آدرس 9ADBF4 گذاشته و قصد داریم قبل از اینکه در آدرس حافظه مقدار صفر را مقایسه کند متوقف شویم:



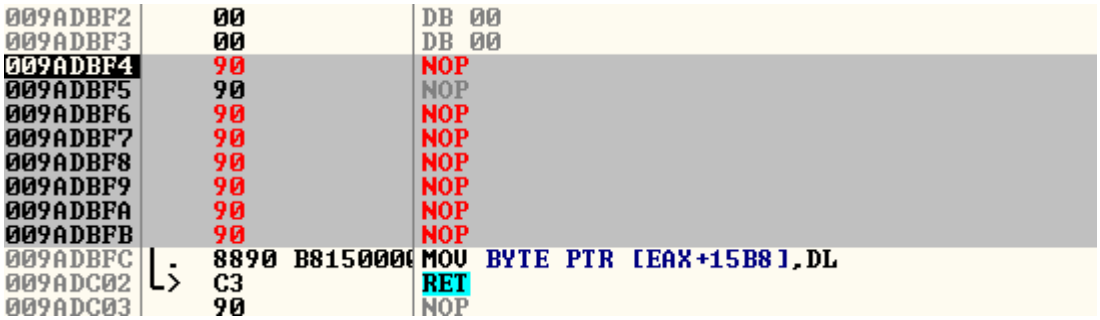
برنامه را اجرا کنید،اوه ،متوقف نشد،شاید تعجب کنید که دلیل آن چیست،باید بگویم دلایل متعددی میتواند داشته باشد یکی از آن دلایل تغییر کد به صورت خصوصیت polymorphically (عکس العمل متفاوت از دو رویداد مشترک) در آینده مبحث ترفند های آنتی دیباگ این مسئله بررسی خواهد شد، حالا برنامه را ری استارت کنید و با کلید F9 برنامه را اجرا کنید:

برنامه در آدرسی که نقطه ی توقف سخت افزاری گذاشته بودیم متوقف شد.حال ما قبل از اینکه حرکتی انجام بشود در محل حضور داریم این خط چک میکند که آیا ما ریجیستر شده ایم یا نه ،اگر نشده وبودیم مقدار صفر را در آدرس حافظه ی [EAX+15B8] میگذارد. واگر رجیستر شده بودیم مقدار 01 یا هرمقدار دیگری بجز 00 را در آن آدرس میگذارد ،پس اگر ما کاری کنیم که همیشه مقدار 01 را درون آدرس [EAX+15B8] بگذارد مشکل حل میشود ،حالا سوال بعدی این است که چه روشی بهترین گزینه است ؟ خب همانطور که میدانیم مقدار 00 در آدرس [EAX+15B8] درون DL ریخته میشود ، و در خط 9ADBFC مقدار DL در آدرس حافظه ایی [EAX+15B8] قرار میگیرد، پس بهتر است که از قبل DL دارای مقدار 01 باشد ☺برای این کار ما به دو خط اولی نیاز نداریم پس من آنها را هایلایت میکنم :

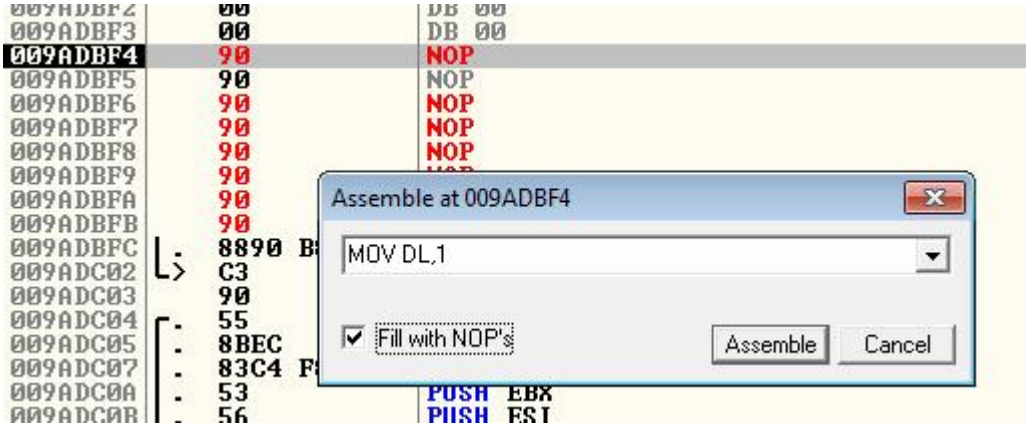
و راست کلیک میکنم سپس: "Binary" -> "Fill with NOPs"



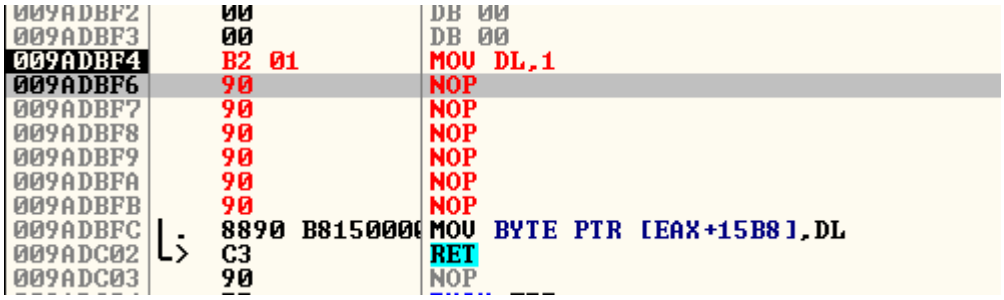
و سپس:



حالا بروی دستور العمل NOP در آدرس 9ADBF4 دبل کلیک کنید و دستور زیر را وارد کنید:

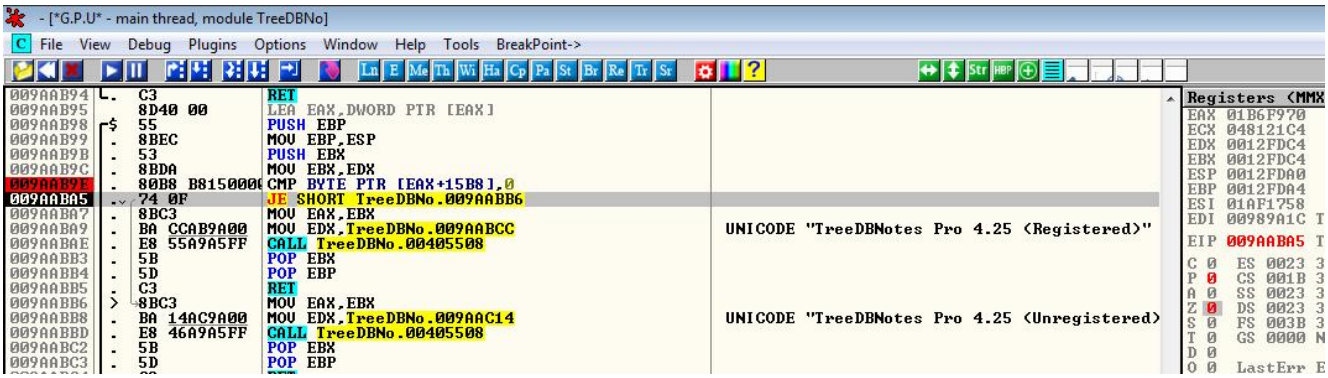


وبعد دکمه ی Assemble را بفشارید و مطمئن شوید Fill with NOP's فعال باشد و در آخر دکمه ی Cancel را فشارید:



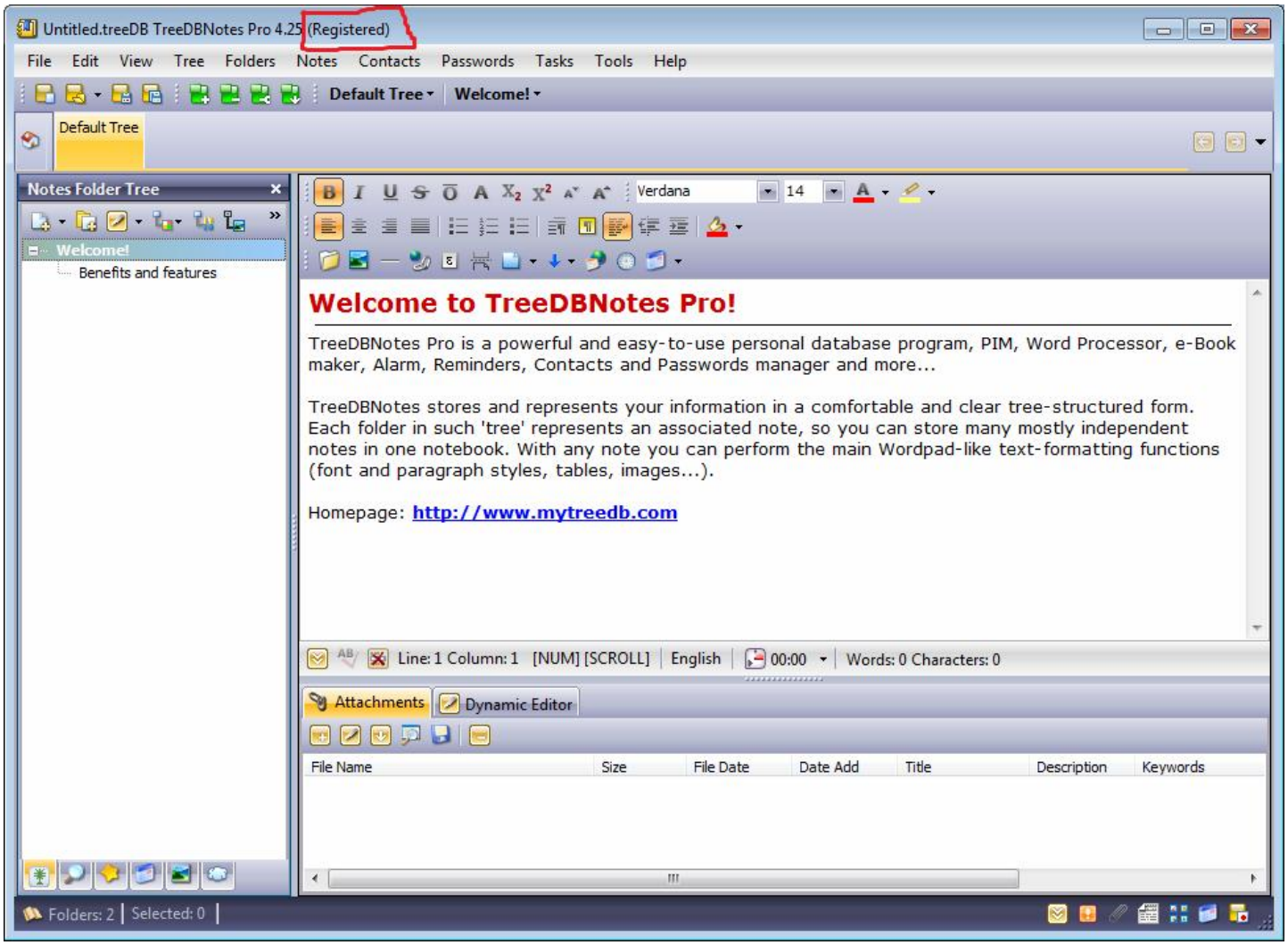
حالا هر موقع که این روتین فراخوانی بشود مقدار 1 درون DL حاضر است و در خط 9ADBFC این مقدار درون آدرس [EAX+15B8] ریخته و نتیجه این میشود که برنامه ریجیستر میشود

حالا کلید F9 را بفشارید ،میبینید که olly در روتین اصلی متوقف شده ،کلید F8را یکبار بفشارید ، بله به نظر همه چیز درست است حالا کلید F9 را بفشارید تا برنامه کاملا اجرا شود:



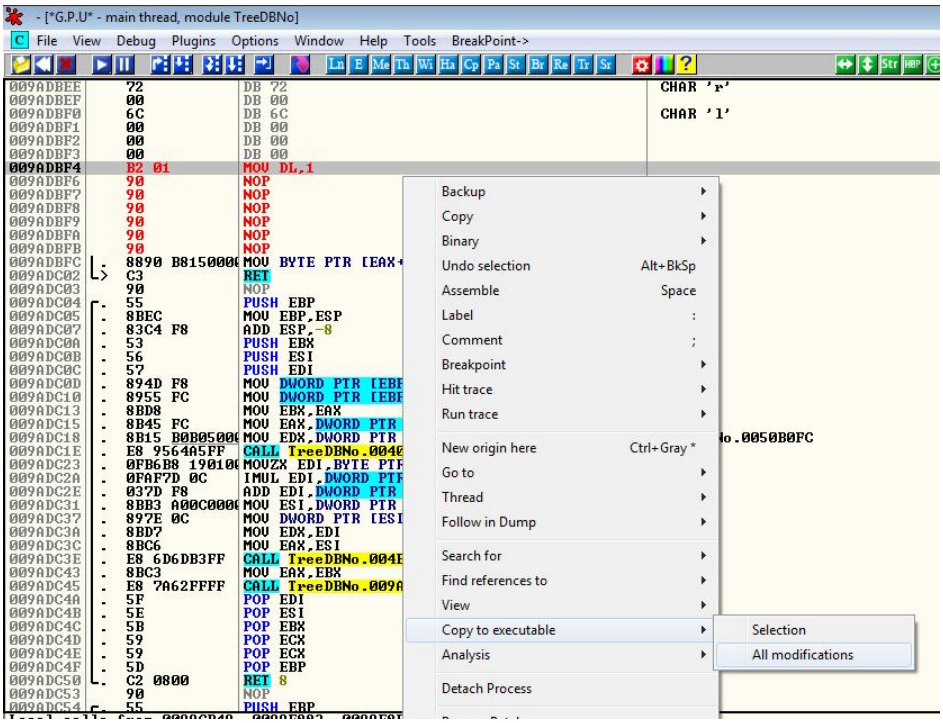
البته بخاطر نقاط توقف تا بارگذاری کامل حدود ۴ بار متوقف میشویم که پس از فشردن کلید F9 در هر بار توقف :

سپس بروی Close کلیک کنید هر موقع که olly متوقف شده (در نقاط توقف) با فشردن کلید F9 از آن عبور کنید تا برنامه کاملا اجرا شود:



و در پنجره ی About میبینیم که برنامه رجیستر شده :

تبریک ☺ شما اولین برنامه ی واقعی را کرک کردید فقط یادتان نرود که نتیجه ی کارها را ذخیره کنید:



و همینطور Hardware breakpoints رااز قسمت (“Debug” -> “Hardwarebreakpoints”) پاک کنید :