

dot.Net patching without decompiling



By
ColdFever

Tools of the trade:

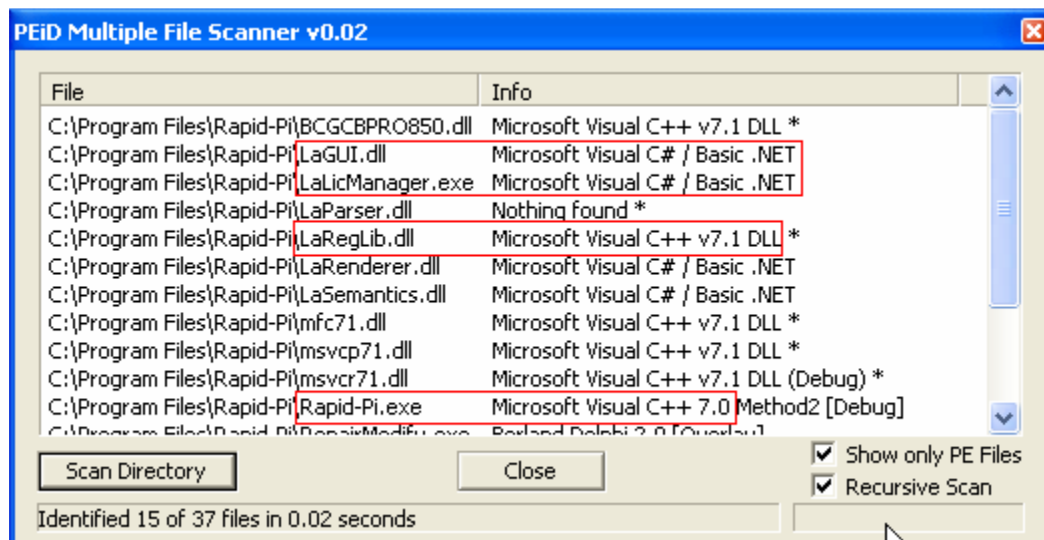
1. PEiD
2. Reflector
3. IDA
4. XVI32 or any Hex editor

Introduction:

In patching .Net, you could disassemble/decompile the executable with ildasm and when done patching you would assemble/compile it again with ilasm. In this tutorial I will show you how to patch the executable with hex editor ☺

The fun part:

It is a good practice to scan your target's directory to have a picture of what you are dealing with.

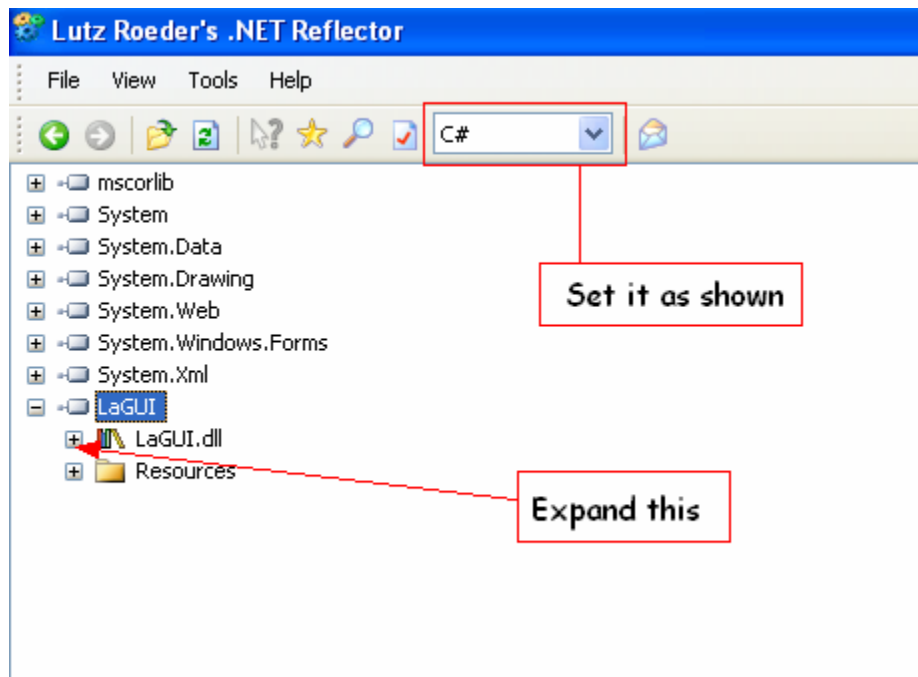


It is a mix. The main executable file and LaRegLib are in Win32 format while the license manager and LaGUI are in .Net format.

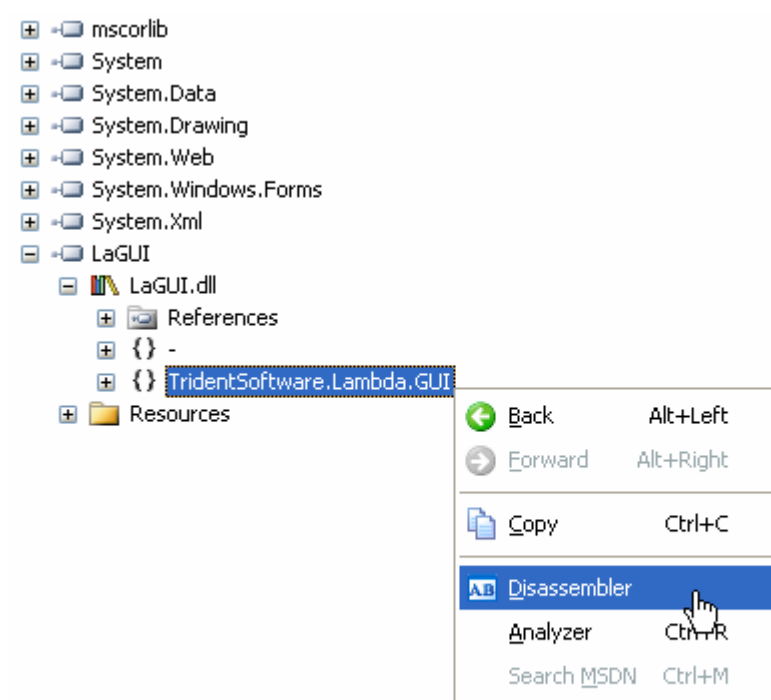
BTW, the program uses a simple algo for validating the serial and it can be easily KeyGenned ☺ dig a little deeper in the code and would see it

```
MOV EAX,DWORD PTR SS:[ESP+28]
MOV EDX,DWORD PTR SS:[ESP+7C]
MOV ECX,DWORD PTR SS:[ESP+20]
XOR EDX,EAX
XOR ECX,EAX
CMP EDX,F9AC3701
MOV EDX,DWORD PTR SS:[ESP+4]
XOR EDX,ECX
XOR EDX,F9AC3701
```

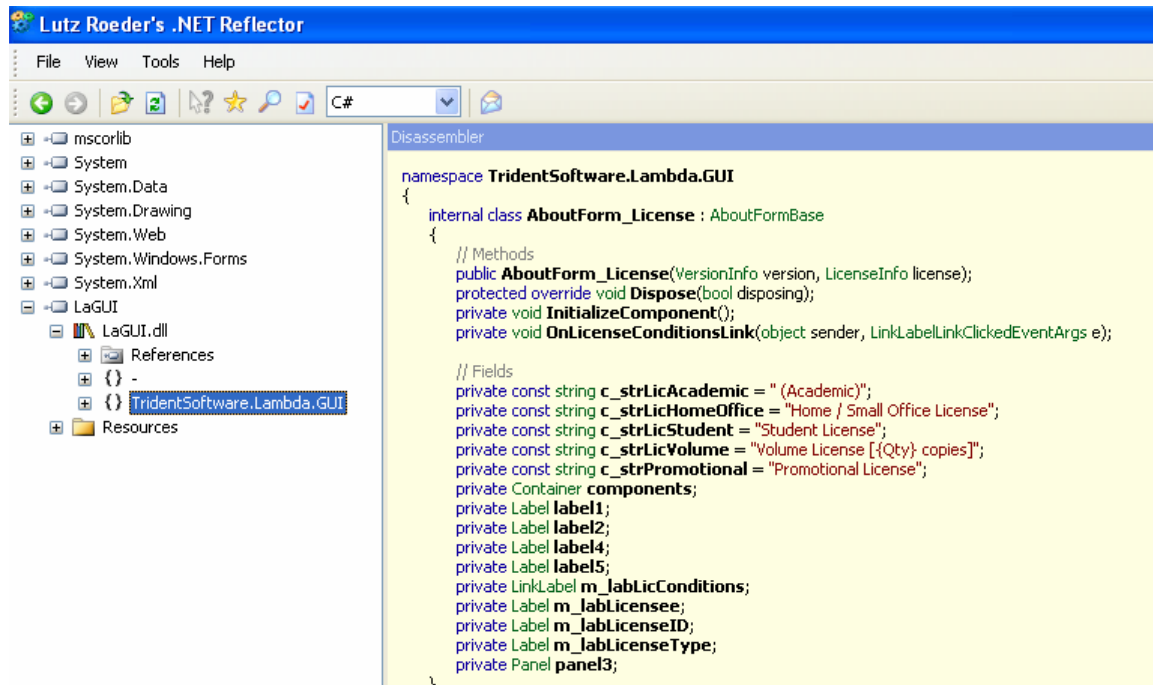
Run Reflector and open LaGUI.dll in it. Set the language to C#



To see the actual code do this



This is the code in C#. Reflector did a very nice job, indeed ☺

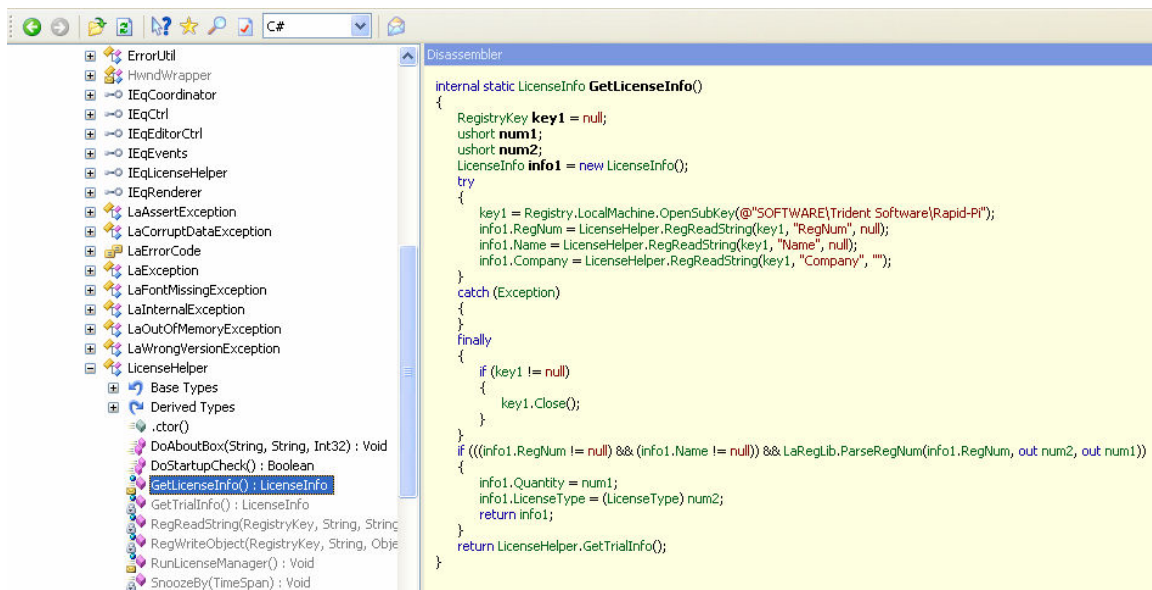


If you scroll a bit down you should be able to see

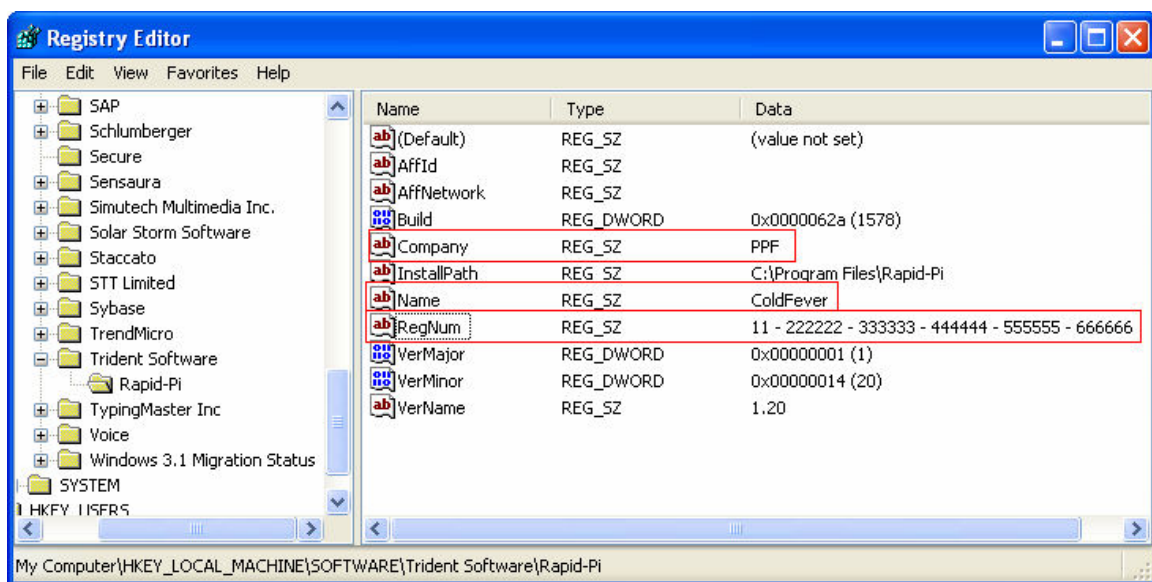
```
[Flags]
internal enum LicenseType : ushort
{
    // Fields
    AcademicBit = 8,
    HomeOffice = 2,
    Promotional = 0x10,
    Student = 1,
    Trial = 0,
    Volume = 4
}
```

Pretty obvious we don't want the License Type to be zero.

Let's dig around in the code to see what else we could find ... see what I found!



Very interesting, we need to create three-registry items. Otherwise we would get trial info. So, let's create those keys.



Now the keys are done, we just need to point the program in the right direction. (Hopefully ☺)

Fire-up IDA and open LaGUI.dll in it. Search for GetLicenseInfo routine in the "names".

IDA View-A Hex View-A Exports Imports Names Functions Strings		
Name	Address	P
Deserialize	00002210	
Dispose	000000D0	
DistanceToNode	00002030	
DoAboutBox	00003CD0	
DoStartupCheck	00003CC0	
DoUndoRedo	00002FA0	
EndInvoke	00003C40	
EqChanged	00001680	
EqElementClickedHandler	00003C50	
Focus	00002320	
FormatErrorMessage	00003D90	
FormatRegNum	000058F0	
GetKeywordModelInfo	00003690	
GetLicenseInfo	000041C0	
GetOrderUrl	00005910	
GetTrialInfo	000042B0	
HandleUnexpected	00003F90	
HideCaret	00005BA0	
InitBuffer	00005EA0	
InitializeComponent	000000F0	

Double click on it and it will take you to the code

```

?? ?? ?? ?? ?? ?? ?? ??+
    .method assembly static hidebysig value class TridentSoftware.Lambda.GUI.Lic
                                     // CODE XREF: sub_40D07p
                                     // sub_4150+87p ...
    {
        .locals init (class [mscorlib]Microsoft.Win32.RegistryKey U0,
                       value class TridentSoftware.Lambda.GUI.LicenseInfo U1,
                       unsigned int16 U2,
                       unsigned int16 U3)

14          ldnull
0A          stloc.0
12 01       ldloc.s 1
FE 15 20 00 00 02  initobj TridentSoftware.Lambda.GUI.LicenseInfo

        .try {
7E 67 01 00 0A     ldsfld class [mscorlib]Microsoft.Win32.RegistryKey [mscorlib]Microsoft.Win
72 39 0A 00 70     ldstr "SOFTWARE\\Trident Software\\Rapid-Pi"
6F 68 01 00 0A     callvirt class [mscorlib]Microsoft.Win32.RegistryKey [mscorlib]Microsoft.W
0A          stloc.0
12 01       ldloc.s 1
06          ldloc.0
72 7F 0A 00 70     ldstr "RegNum"
14          ldnull
28 C7 00 00 06     call class System.String TridentSoftware.Lambda.GUI.LicenseHelper::RegRead
7D 5D 00 00 04     stfld class System.String TridentSoftware.Lambda.GUI.LicenseInfo::RegNum

```

You might wonder, why would we need IDA since we can get the same code in the “IL” view in reflector? Well, reflect will not give you the Op-Code or the offset ☹ ... if you don’t get what I mean just wait, you will see what I mean.

```

}
if (((info1.RegNum != null) && (info1.Name != null)) && LaRegLib.ParseRegNum(info1.RegNum, out num2, out num1))
{
    info1.Quantity = num1;
    info1.LicenseType = (LicenseType) num2;
    return info1;
}

```

For our patch we need to remove the last part of the if-statement. Also, we need to fix info1 quantity and LicenseType. As you can see ☺

<pre> * 12 01 * 7B 5B 00 00 04 * 2C 24 * 12 01 * 7B 5D 00 00 04 * 12 03 * 12 02 * 28 05 00 00 06 * 2C 12 * 12 01 * 08 * 7D 5F 00 00 04 * 12 01 * 09 * 7D 5E 00 00 04 * 07 * 2A </pre>	<pre> ldloc.s 1 ldfld class System.String TridentSoftware.La brfalse.s loc_425E ldloc.s 1 ldfld class System.String TridentSoftware.La ldloc.s 3 ldloc.s 2 call bool LaRegLib::ParseRegNum(class System brfalse.s loc_425E ldloc.s 1 ldloc.2 stfld int32 TridentSoftware.Lambda.GUI.Licen ldloc.s 1 ldloc.3 stfld value class TridentSoftware.Lambda.GUI ldloc.1 ret loc_425E: // CODE // GetLi call value class TridentSoftware.Lambda.GUI. ret } </pre>
---	--

We need to NOP the Op-Codes in the large box, replace the second box with one and the third box with two. The Op-Codes are different in .Net environment. Example in Win32, the Op-Code for NOP instruction is 90 while in .Net it is 00. The Op-Code for number one is 31 while in .Net it is 17.

Let's get the offset from IDA

```

7B 5B 00 00 04
2C 24
12 01
7B 5D 00 00 04
12 03
12 02
28 D5 00 00 06
2C 12
12 01
08
7D 5F 00 00 04
12 01
09
7D 5E 00 00 04
07
2A

ldfld class System.String TridentSoftware.Lambda.
brfalse.s loc_425E
ldloca.s 1
ldfld class System.String TridentSoftware.Lambda.
ldloca.s 3
ldloca.s 2
call bool LaRegLib::ParseRegNum(class System.Stri
brfalse.s loc_425E
ldloca.s 1
ldloc.2
stfld int32 TridentSoftware.Lambda.GUI.LicenseInf
ldloca.s 1
ldloc.3
stfld value class TridentSoftware.Lambda.GUI.Lice
ldloc.1
ret

loc_425E:
// CODE XREF:
// GetLicense
call value class TridentSoftware.Lambda.GUI.Licen
ret
}

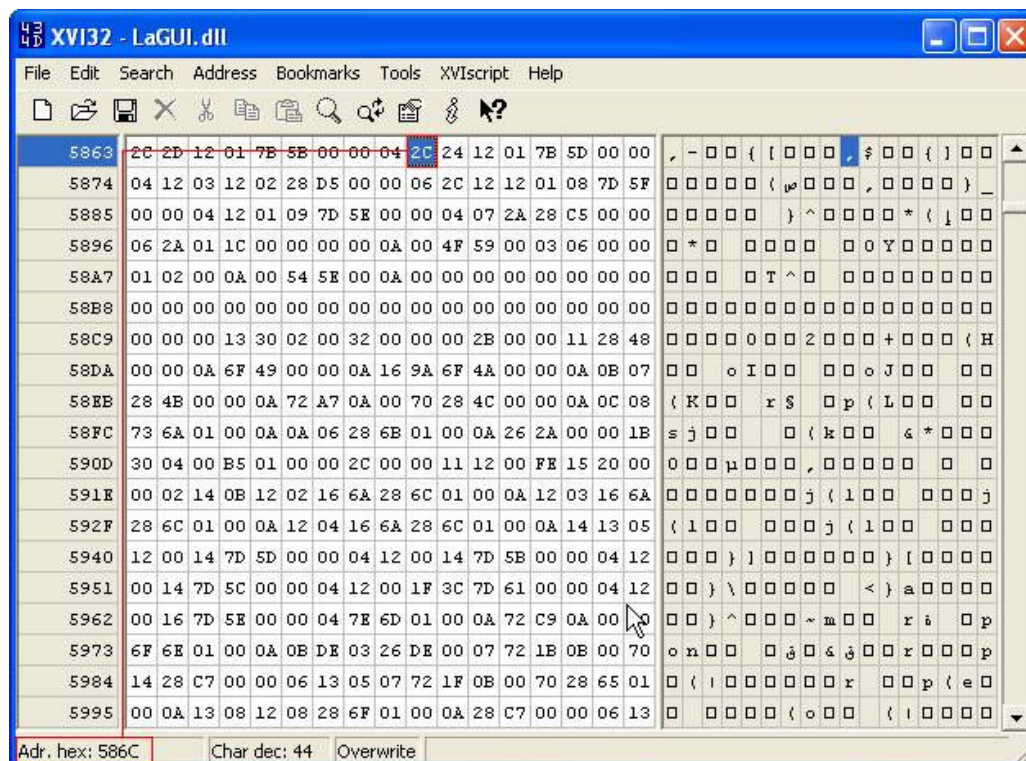
?? ?? ?? ?? ?? ?? ?? ??+

```

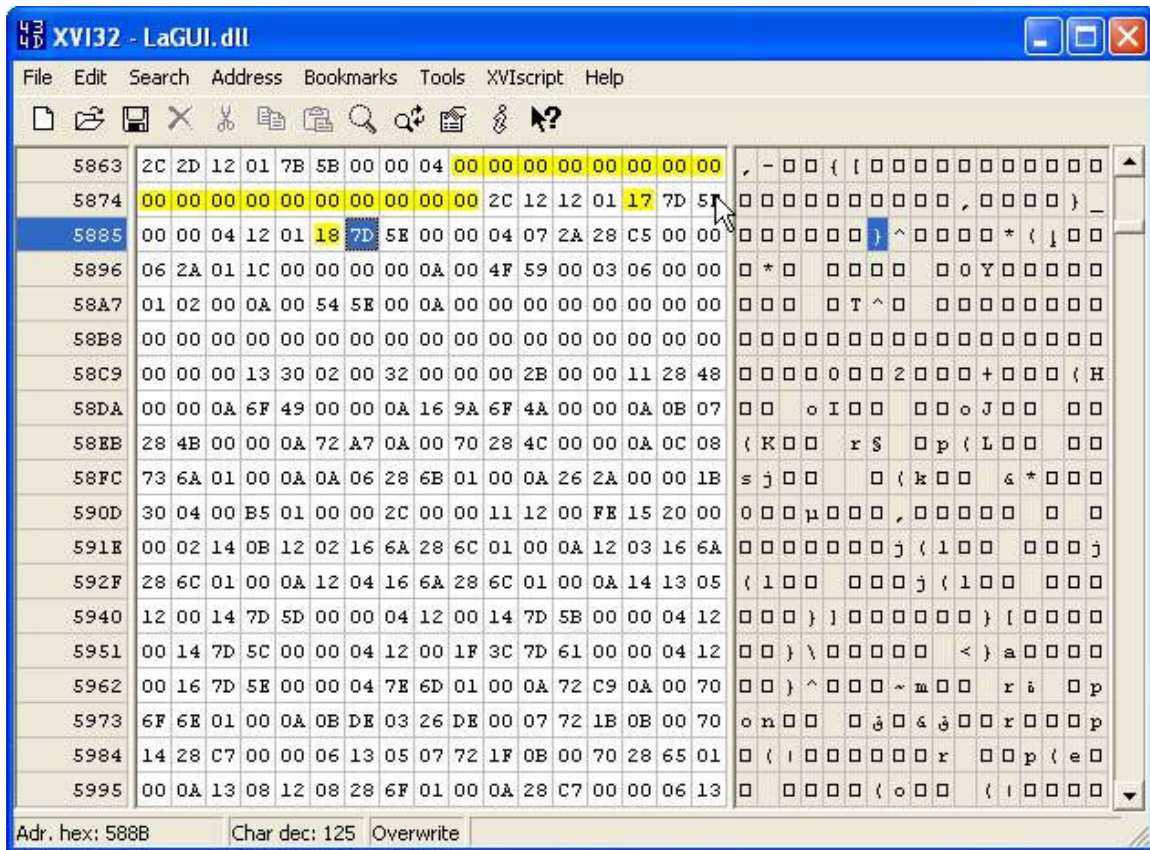
Flushing buffers, please wait...ok
file 'C:\Program Files\Rapid-Pi\LaGUI.dll' is successfully loaded into the database.
compiling file 'C:\Program Files\IDA\idc\ida.idc'...
recuting function 'main'...
compiling file 'C:\Program Files\IDA\idc\onload.idc'...
recuting function 'OnLoad'...
IDA is analysing the input file...
you may start to explore the input file right now.
the initial autoanalysis has been finished.
retrieving information from the database... ok
retrieving information from the database... ok

J: idle Down Disk: 5GB 0000586C 00004238: GetLicenseInfo+78

Open LaGUI.dll in XVI32 or any hex editor.



Do the changes



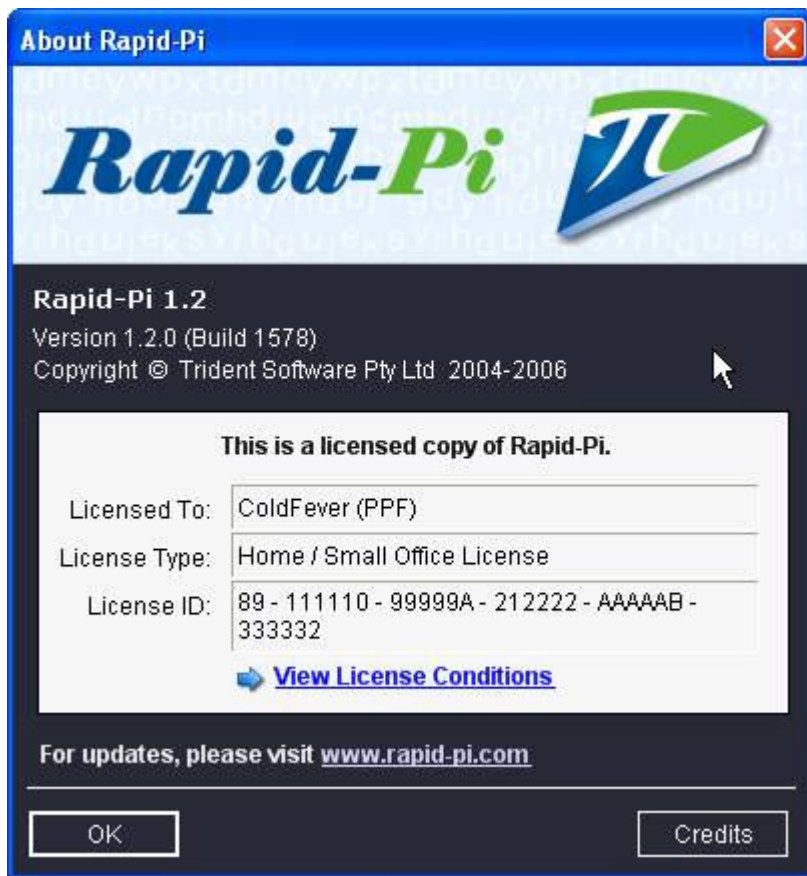
Save the file. I know that you made a backup just in case. Let's check the file in reflector

```

    }
}
if ((info1.RegNum != null) && (info1.Name != null))
{
    info1.Quantity = 1;
    info1.LicenseType = LicenseType.HomeOffice;
    return info1;
}
return LicenseHelper.GetTrialInfo();
}

```

Run the program and check the about



Nicely done ☺

Greetings:

To my friends at www.alalme.com/vb

ARteam, Snd, CORE, tuts4you and all the cracking teams

Special Thanks:

Lena151, your series is a crack must ☺ (my point of view of course)

Sub-Zero, your piece on "API Hooking" fascinated me ☺

Teddy Rogers, your site is my home ... I'm camping in ☺

NoLimit, I would be looking for your tutorials ☺