)))

إذا مات ابن آدم انقطع

عمله إلا من ثلاث :

صدقة جارية أو علم ينتفع به

أو ولد صالح يدعو له

(((

اللهم تقبلها في ميزان حسنات أستاذنا د.عبد العزيز سليمان

Reversing .NET

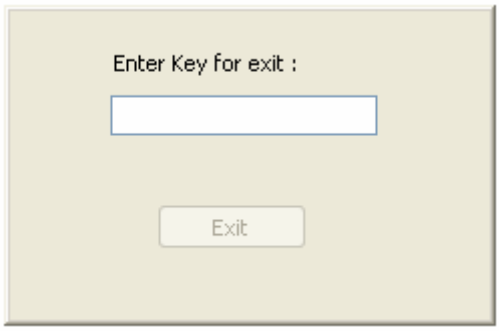
بسم الله نبدأ ..

الهندسة العكسية للبرامج المكتوبة ببيئة أل .NET. مختلفة كليا عن البرامج المكتوبة بالغات العادية ، بصورة عامة الأمر أسهل بكثير لبرامج أل .NET.

طبعا هنالك برامج خاصة لعمل هندسة عكسية لبرامج أل .NET. مثلا برنامج Reflector

و لنأخذ مثال بسيط جدا

افتح الفيجوال **C#** و كون برنامج كما في الصورة



و سيكون جزء الكود الخاص به هكذا

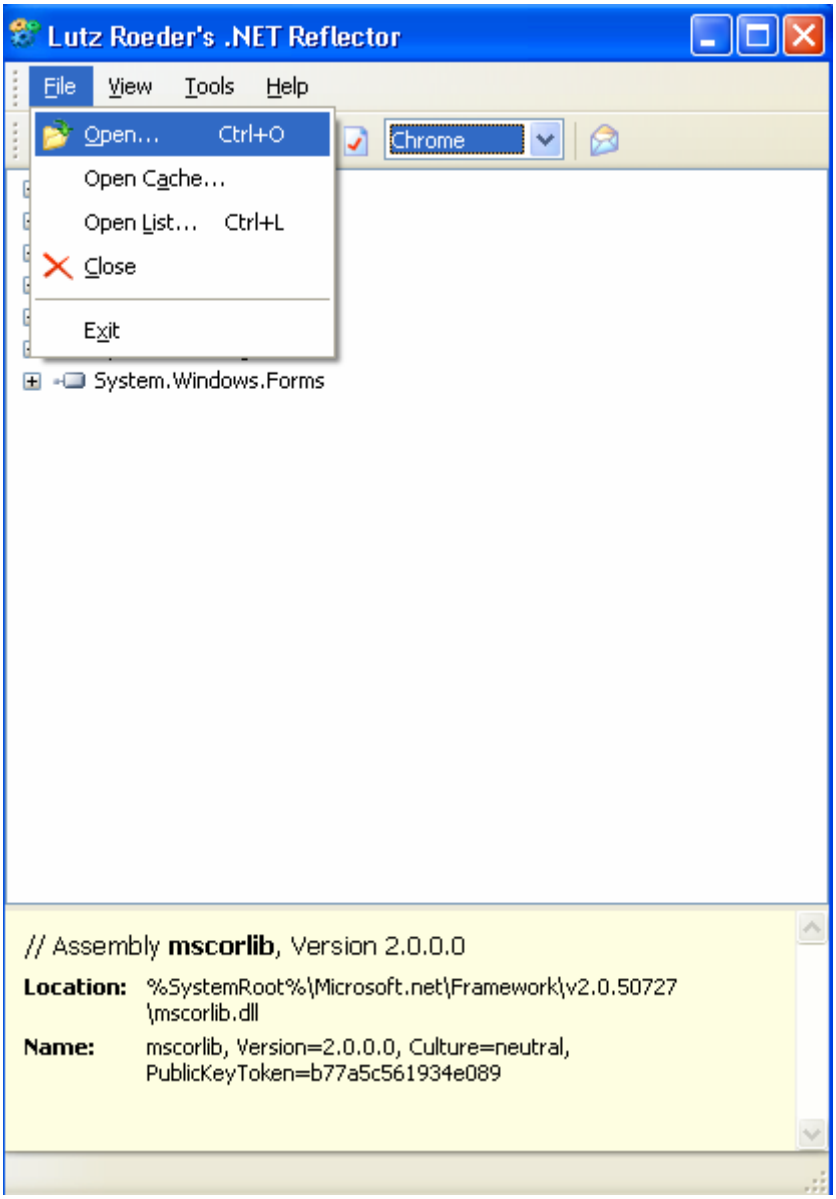
```
private void button1_Click(object sender, EventArgs e)
{
   Application.Exit();
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
   if (textBox1.Text == "done")
      button1.Enabled = true;

}
```

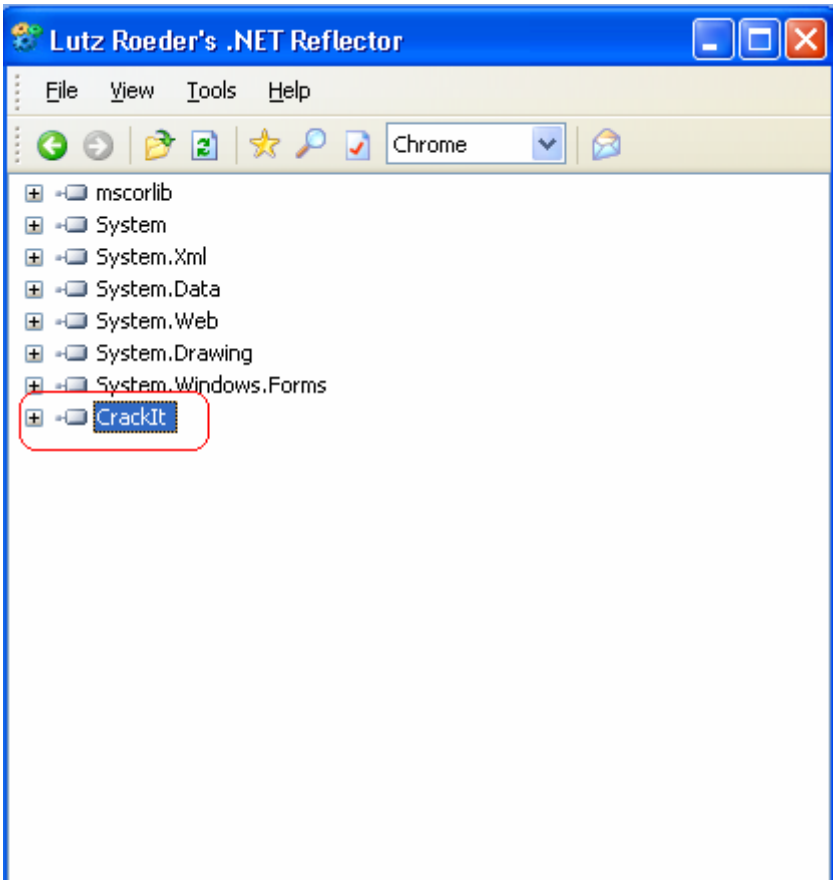
كما هو واضح عليك أن تدخل key لتفعيل زر الخروج ، و على فرض انك لا تعلم ما هو أل key .

الان شغل برنامج أل Reflector ومن قائمة file اختر open كما في الصورة

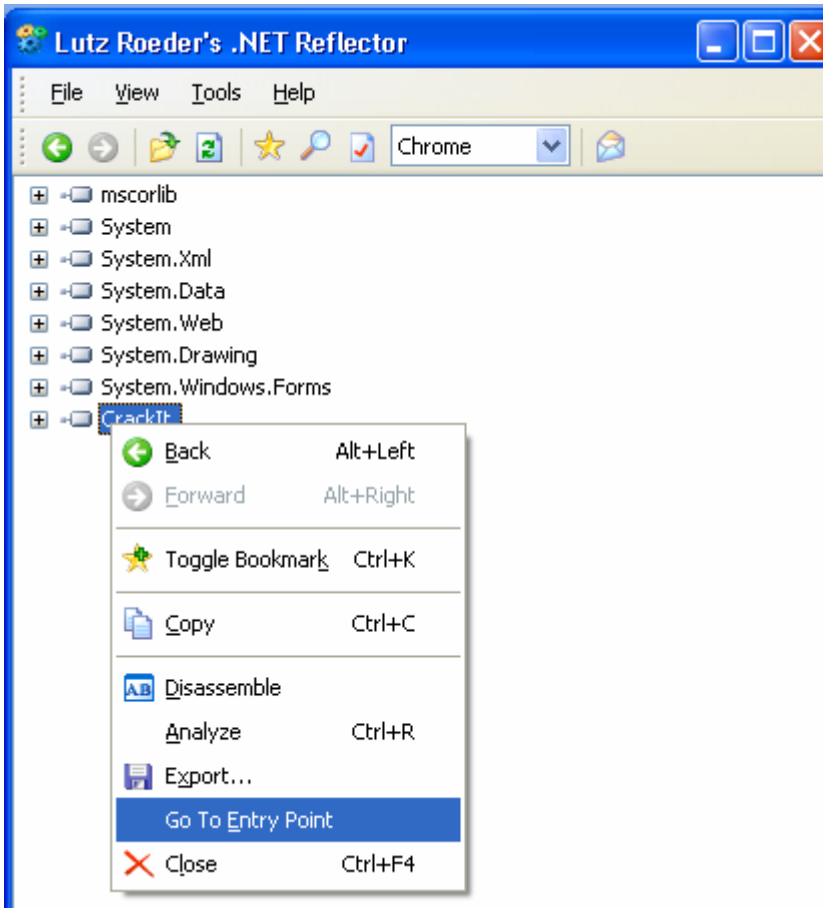


ثم حدد مسار البرنامج (CrackIt)

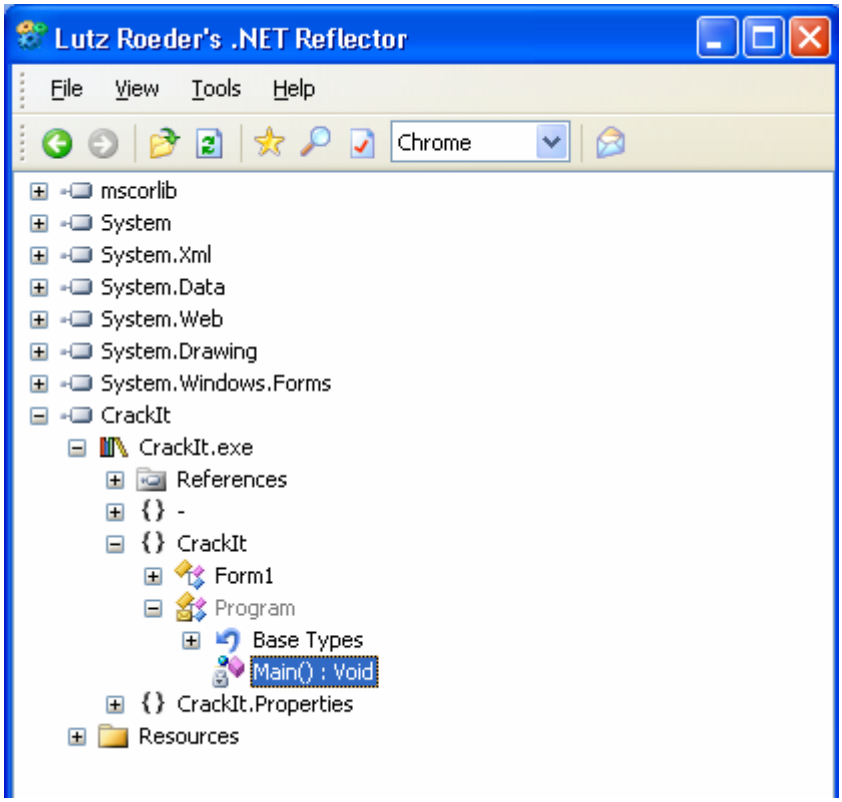
سيفتح كما في الصورة



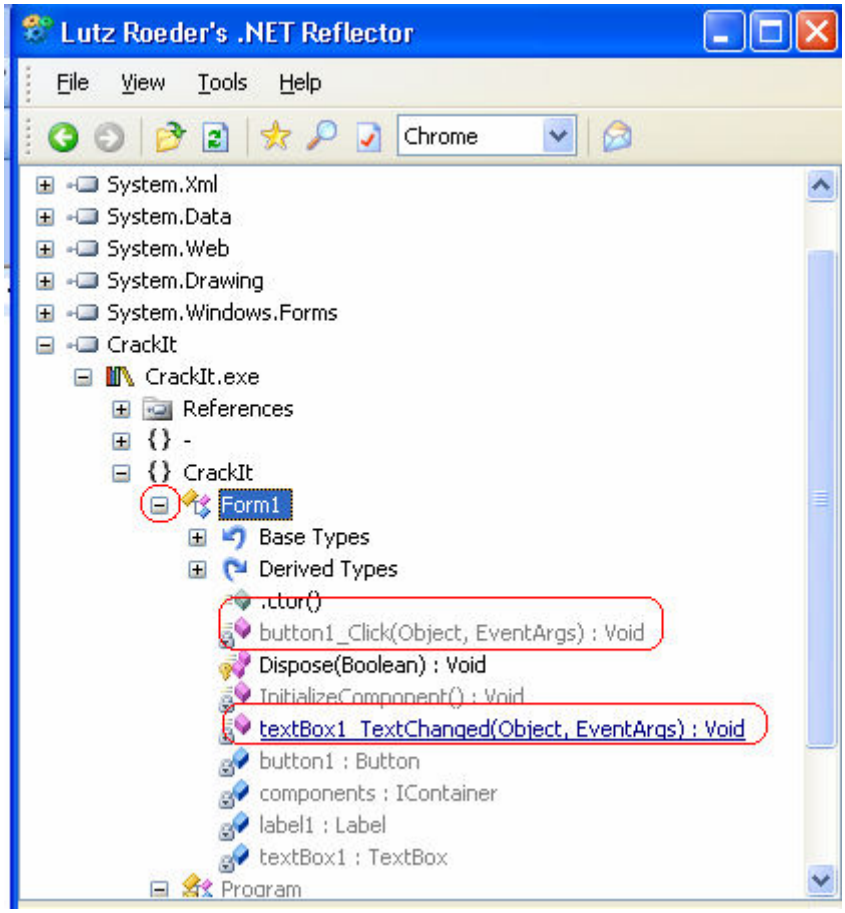
و للذهاب إلى بداية البرنامج (أي ألـ Main) ، تتبع الصورة


Lutz Roeder's .NET Reflector
File
View
Tools
Help
Chrome
mscorlib
System
System.Xml
System.Data
System.Web
System.Drawing
System.Windows.Forms
Back Alt+Left
Forward Alt+Right
Toggle Bookmark Ctrl+K
Copy Ctrl+C
Disassemble
Analyze Ctrl+R
Export...
Go To Entry Point
Close Ctrl+F4

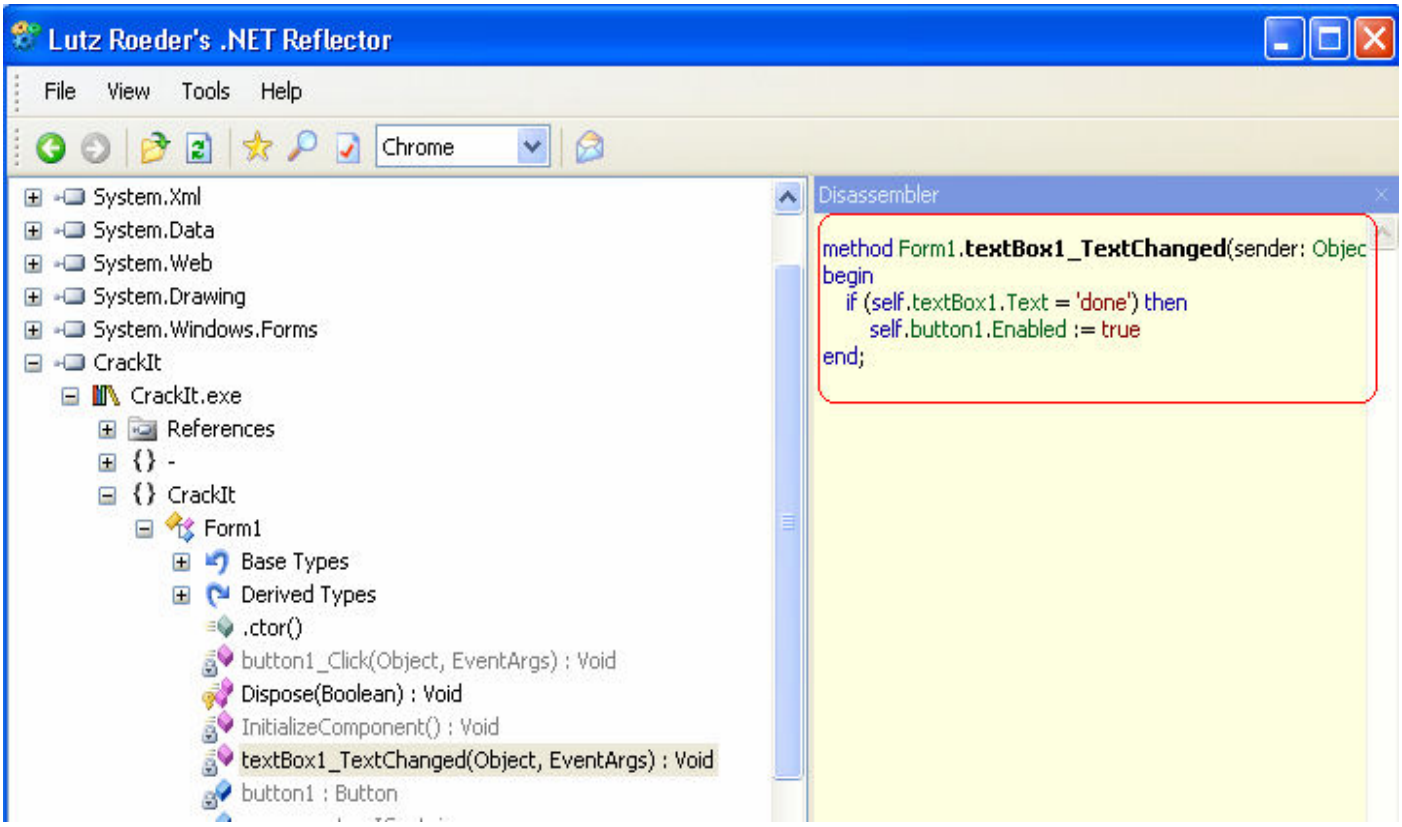
و النتيجة



الان افتح علامة أل + التي تسبق كلمة Form1 لترى أهم ما في البرنامج



هناك زر الخروج ، و الحدث textChanged وهو ما يهمنا هنا لأن المقارنة تحدث بداخله ، اضغط double click على هذا الحدث ليفتح الكود الخاص به و سترى العجب



هل رأيت وضوح الكود ،

الان بسهولة ممكن اخذ أل key وكما هو واضح هي كلمة done

نفذ برنامجنا مرة أخرى واكتب done و سترى انه يتم تفعيل رز الخروج.

هل رأيت مدى سهولة الأمر ..

الان ربما سائل يسأل : لما هذه السهولة و كيف تستطيع برامج الهندسة العكسية باسترجاع الكود بصورة واضحة من البرامج المكتوبة بلغات الدوت نت ؟

و الجواب :

يرجع السبب إلى طبيعة تنفيذ البرامج على بيئة الدوت نت ،

فعندما تنفذ برنامج في بيئة الدوت نت لا يتكون binarycode و لكن يتكون ما يعرف ب bytecode ،

وهناك ما يعرف ب intermediate language .

(هذا أل bytecode يتم قراءته بواسطة برامج خاصة تعـــرف ب virtual machine و احدها هو CLR و هو الذي يعمل مع بيئة مايكروسوفت . عرفت الان لما لا تستطيع أن تنفذ برنامج مكتوب بلغة الدوت نت إلا بوجود أل framework !!)

عموما إذا كان كلامي غير واضح أقرء الأتي .. و لا تطلب مني أن أترجم

** Some software development platforms don't produce executable machine code that directly runs on a processor. Instead, they generate some kind of intermediate representation of the program, or *bytecode*. This bytecode is then read by a special program on the user's machine, which executes the program on the local processor. This program is called a *virtual machine*.

** Two common virtual machine architectures are the Java Virtual Machine (JVM) that runs Java programs, and the Common Language Runtime (CLR) that runs Microsoft .NET applications.

** Reversing bytecode programs is often an entirely different experience compared to that of conventional, native executable programs. First and foremost, most bytecode languages are far more detailed compared to their native machine code counterparts. For example, Microsoft .NET executables contain highly detailed data type information called metadata. Metadata provides information on classes, function parameters, local variable types, and much more.

** .NET programs are compiled into an intermediate language (or bytecode) called MSIL (Microsoft Intermediate Language). MSIL is highly detailed; it contains far more high-level information regarding the original program than an IA-32 compiled program does. These details include the full definition of every data structure used in the program, along with the names of almost every symbol used in the program.

الان و بسبب السهولة هذه ، ظهرت برامج تعرف ب *obfuscators* (بـــرامج تـــشويش) و وظيفتها تعقيد و تصعيب عمل هندسة عكسية لبرامج الدوت نت

** *obfuscators*—programs that try to eliminate as much sensitive information from the executable as possible (while keeping it functional).

** Because of the inherent vulnerability of .NET executables, the concept of obfuscating .NET executables to prevent quick decompilation of the program is very common. This is very different from native executables where processor architectures such as IA-32 inherently provide a decent amount of protection because it is difficult to read the assembly language code. IL code is highly detailed and can be easily decompiled into a very readable high-level language representation.

** Common strategies for obfuscating .NET executables:

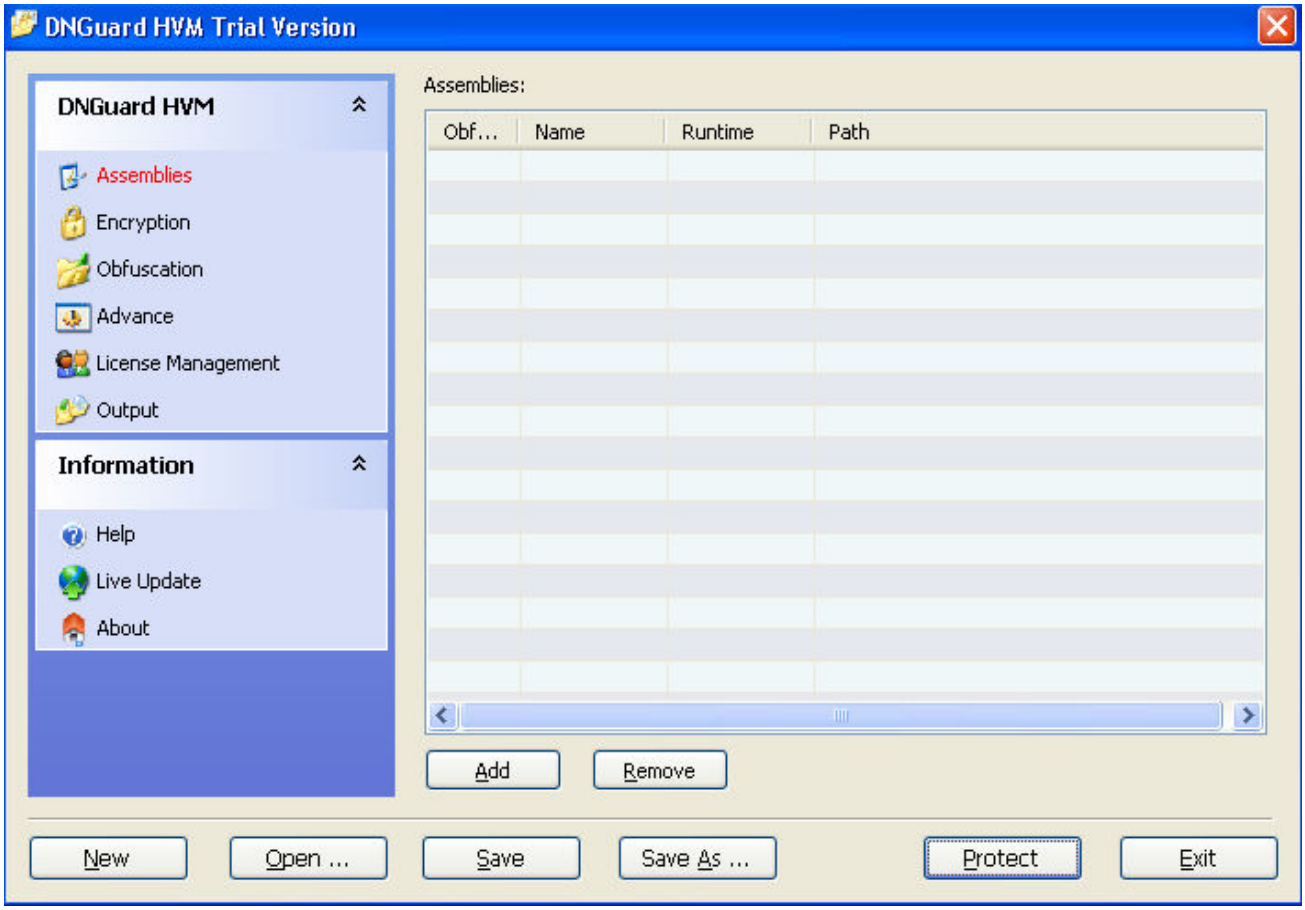
- *Renaming Symbols*
- *Control Flow Obfuscation*
- *Breaking Decompilation and Disassembly*

لتحميل مثل هذه البرامج

قام الأخ محمد مشكور بوضع الكثير من هذه البرامج في هذا الرابط

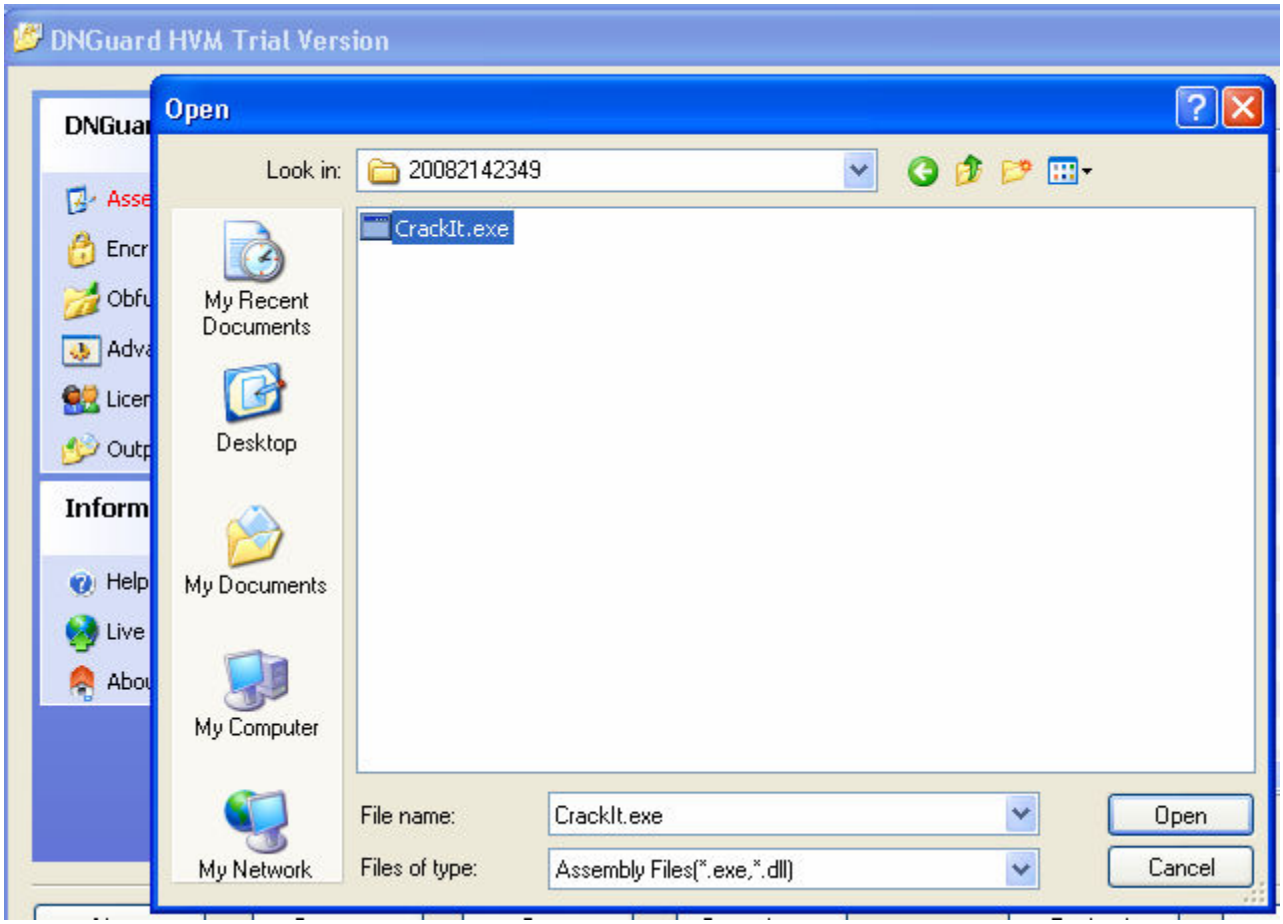
http://www.arabteam2000-forum.com/index.php?showtopic=74499

هذه واجهة برنامج HVM DNGuard **مثلا و الـــذي سنـــستخدمه في تـــشفير مثالنـــا** CrackIt

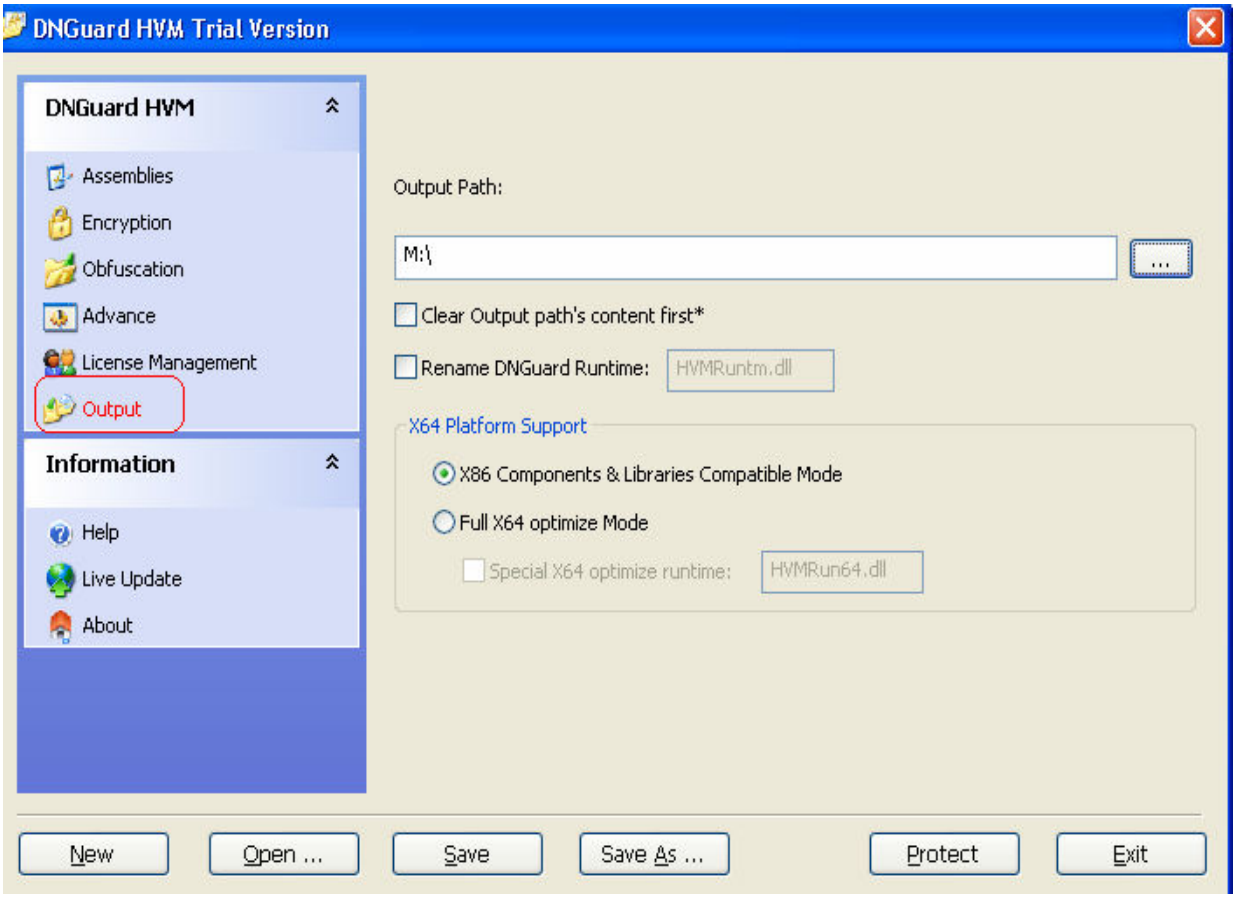


(عذرا ليس لدي معلومات لشرح البرنامج كاملا فبالأمس فقط قمت بالتعامل معه !!)

الان اضغط على زر Add وحدد برنامجنا

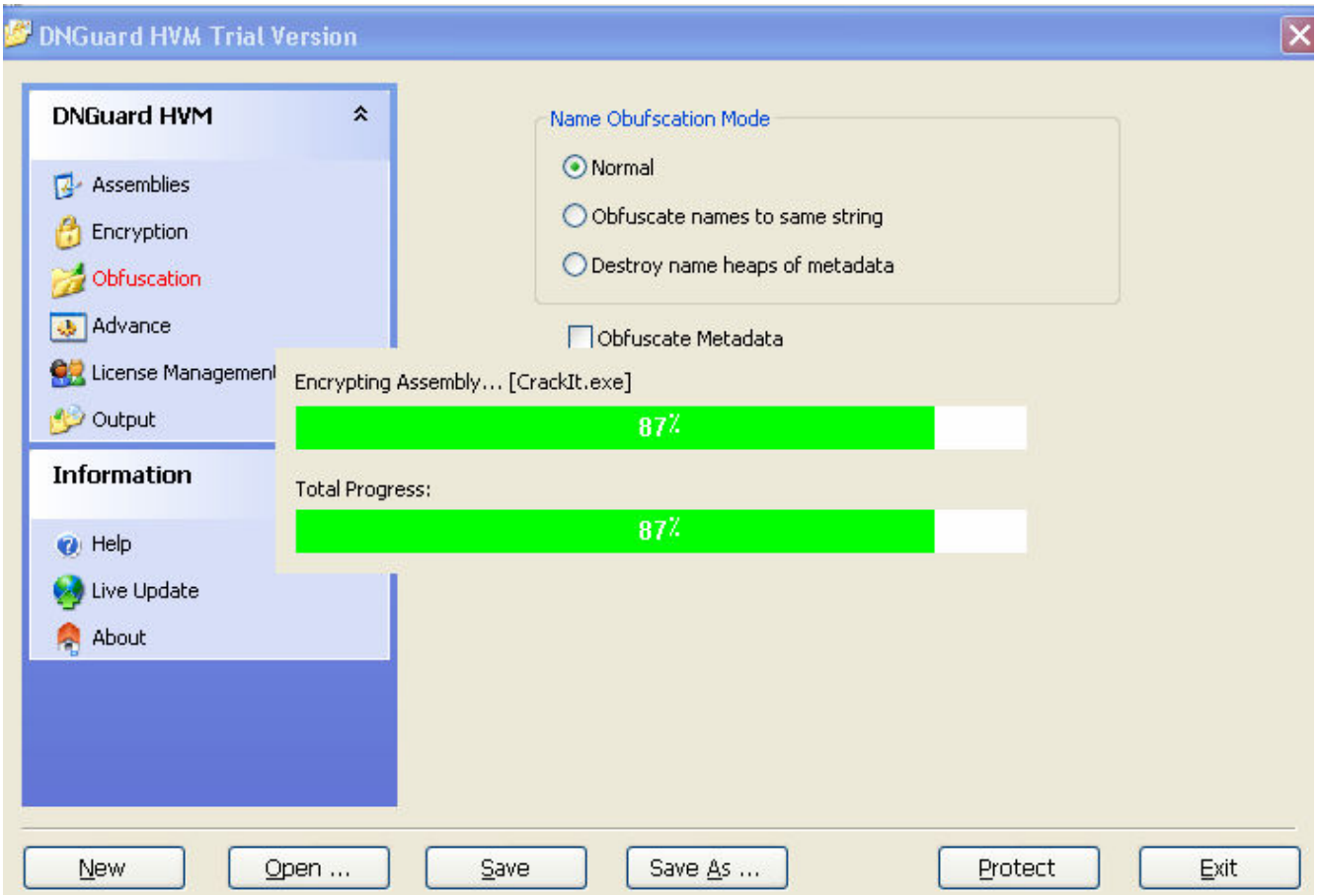


حدد مكان النسخة الجديدة من البرنامج

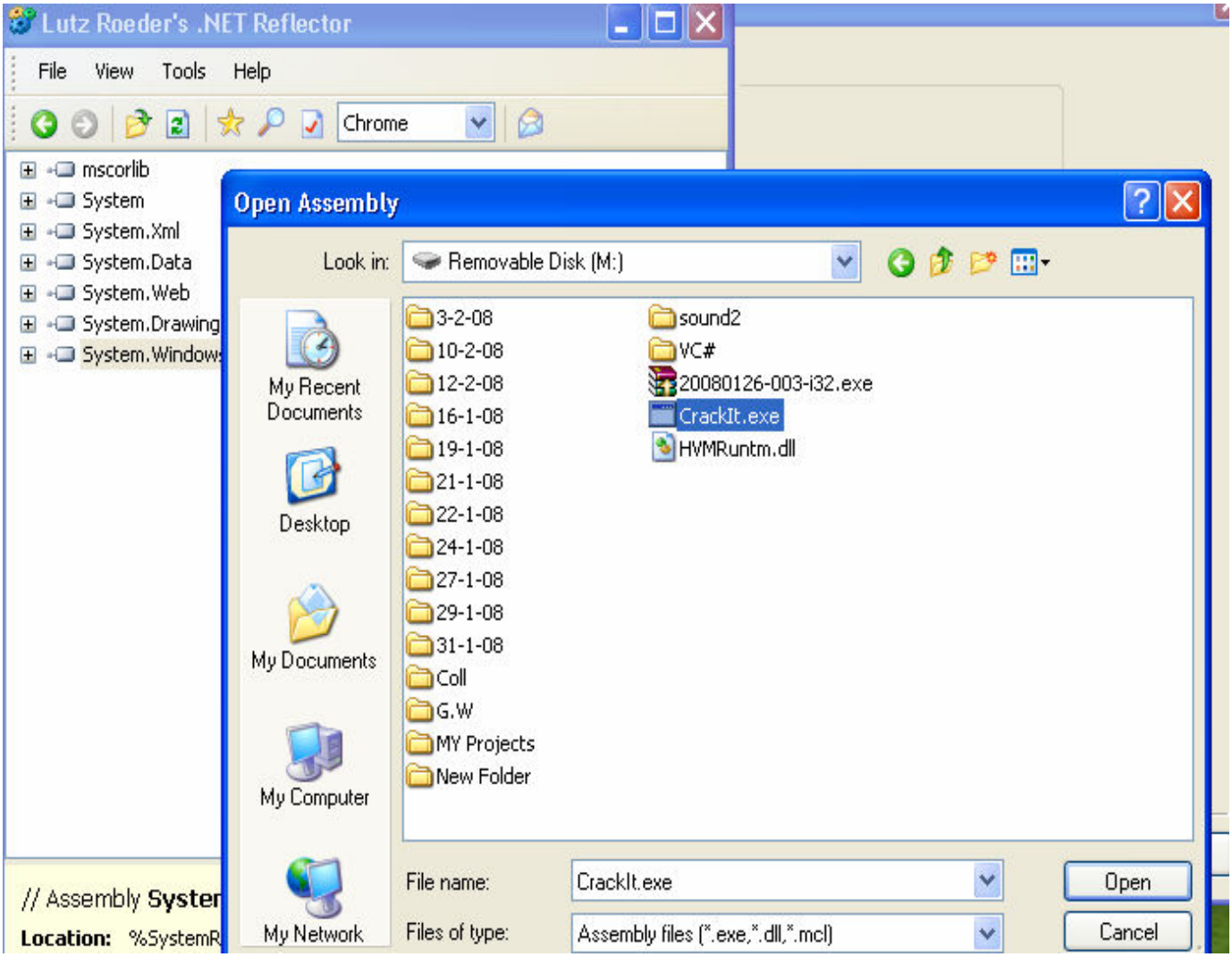


الان اضغط على زر Protect

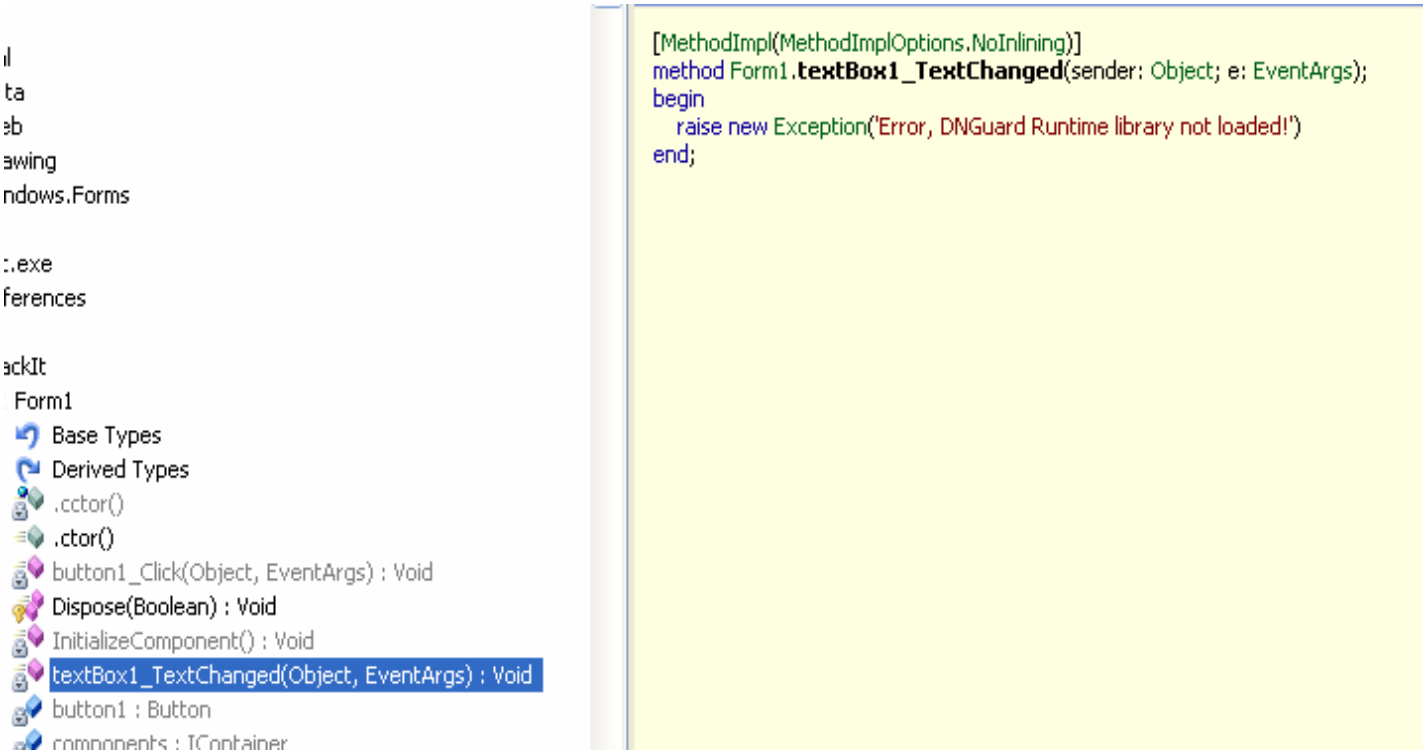
و انتظر لحين انتهاء العملية



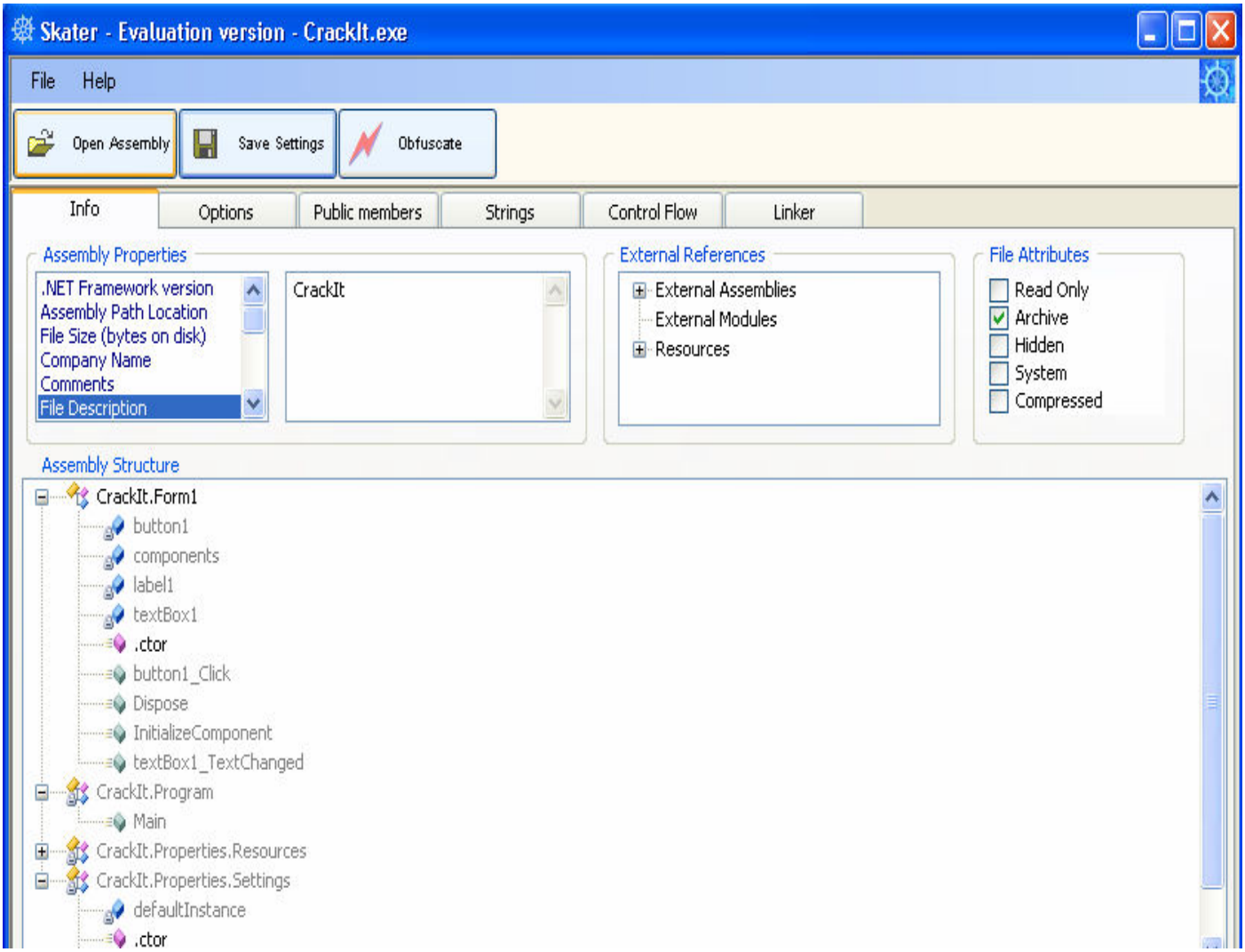
الان اعد فتح النسخة الجديدة (المشوشة) ببرنامج أل Reflector و اتبع نفـــس الخطـــوات السابقة لترى الفرق



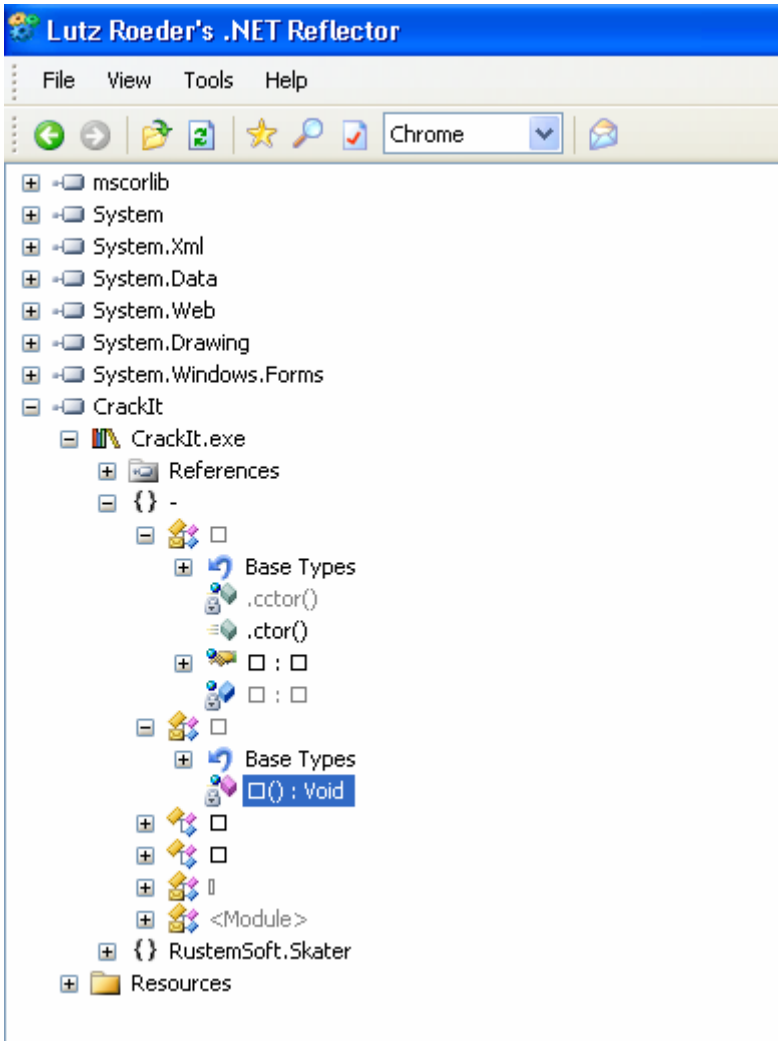
أنا يظهر لي شيء كهذا



و إذا استخدمنا برنامج Skater مثلا



سيكون التشفير



لا تسألني كيفية عمل هندسة عكسية لهذه النتائج لأني .. ببساطة.. لا أعلم !!

و لكن هذا ما يشرحه و يوضحه junkie في هذا الرابط

http://www.arabteam2000-forum.com/index.php?showtopic=144660

الان بقي أمران .

1. كيفية استخدام برامج التشويش *obfuscators* بصورة احترافية .
2. كيفية عمل هندسة عكسية لبرامج تم تشويشها و حمايتها ببرامج أل *obfuscators*

أتمنى من الأعضاء أن يدلو بدلوهم في الموضوع

هذا ما لدي الان ، وان شاء الله سأضيف ما أتوصل إليه في المستقبل .

ملاحظة:

الأجزاء في اللغة الانكليزية مأخوذة من كتاب

Reversing: Secrets of Reverse Engineering

15-2-2008