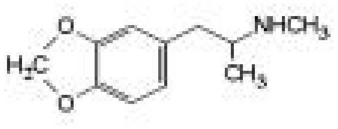
MaDMAn_H3rCuL3s

.NET Reverse Engineering Tutorial Episode 1

Version 1.0
November 2006



Editor: MaDMAn_H3rCuL3s

1. Forewords

We begin another chapter in the ongoing ARTeam series of Reverse Engineering tutorials. Today's topic will go over .NET Reverse Engineering. This topic has been hardly touched upon, thus giving me room to add some information to the reader.

BEE Seeing YA.

MDMA

Disclaimers

Verification

ARTeam.esfv can be opened in the ARTeamESFVChecker to verify all files have been released by ARTeam and are unaltered. The ARTeamESFVChecker can be obtained in the release section of the ARTeam site: http://releases.accessroot.com

Table of Contents

1. .NET Reverse Engineering Tutorial Episode 1

1.1. Abstract

This target is protected by a license check. We are greeted at the beginning with a nag asking for the license file. As a trial user you would get one. The scope of this tutorial isn't to make the program act like a full version, but rather patch around the nag at startup. So if the patch does indeed render it full, this is not my responsibility. You are encouraged to delete this program from your hard drive upon full reversing.

1.2. Targets

To keep the tutorial on a current level, you can find the target at the following place. Since programs are usually regularly updated this will keep this tutorial valid for some time to come.

- StreamPatrol v2.0
http://arteam.accessroot.com/tools/StreamPatrol-2.0.0.zip

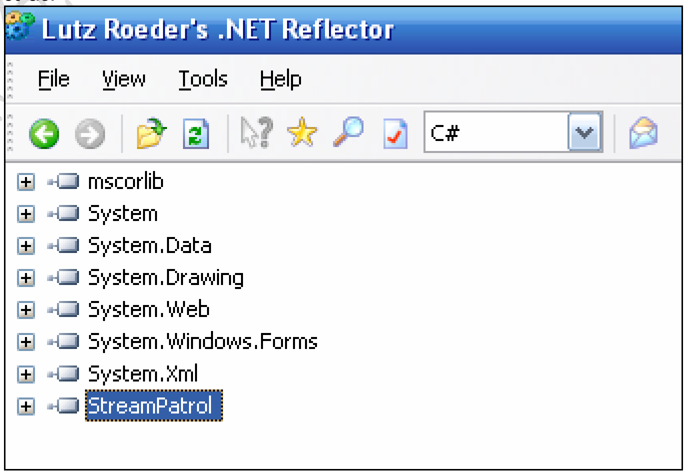
1.3. How to reverse the application

1.3.1 Preparation

We need the following tools to help us along the way.
1. Reflector
2. IDA (Google it)
3. FlexHEX (Google it)

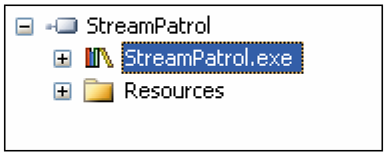
1.3.2 Disassembling the target

Today's target will be StreamPatrol. The fix we will discuss is not 100 percent, which I will talk about later on. To start we get the victim program from link we saw before. We will load up the victim into Reflector and look for anything that might interest us.

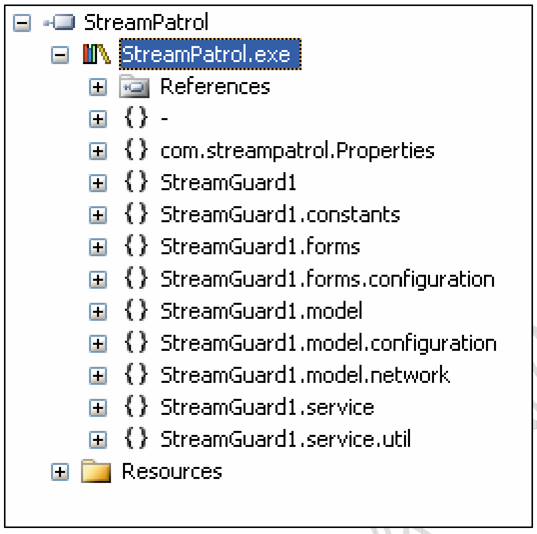


We see the target is loaded up.

We then expand the target using the "+" sign.



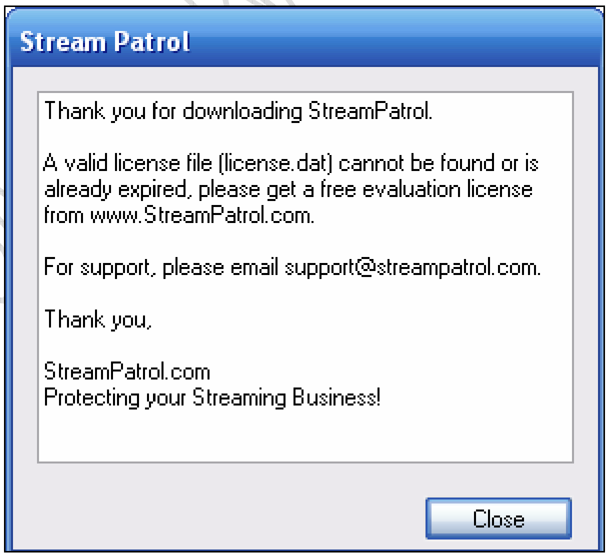
And then we expand the "StreamPatrol.exe"



```
StreamPatrol
  StreamPatrol.exe
    References
    {} -
    {} com.streampatrol.Properties
    {} StreamGuard1
    {} StreamGuard1.constants
    {} StreamGuard1.forms
    {} StreamGuard1.forms.configuration
    {} StreamGuard1.model
    {} StreamGuard1.model.configuration
    {} StreamGuard1.model.network
    {} StreamGuard1.service
    {} StreamGuard1.service.util
  Resources
```

And we see what the exe has for us to view.

If we were to run the exe outside any kind of disasm or debug engine, we would be greeted by this:



Thank you for downloading StreamPatrol.

A valid license file (license.dat) cannot be found or is already expired, please get a free evaluation license from www.StreamPatrol.com.

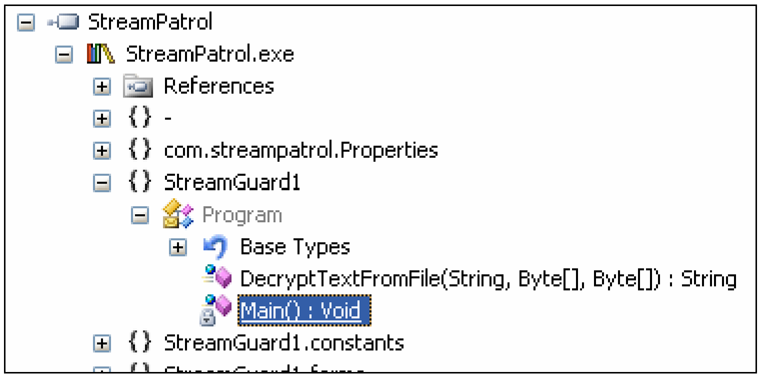
For support, please email support@streampatrol.com.

Thank you,

StreamPatrol.com
Protecting your Streaming Business!

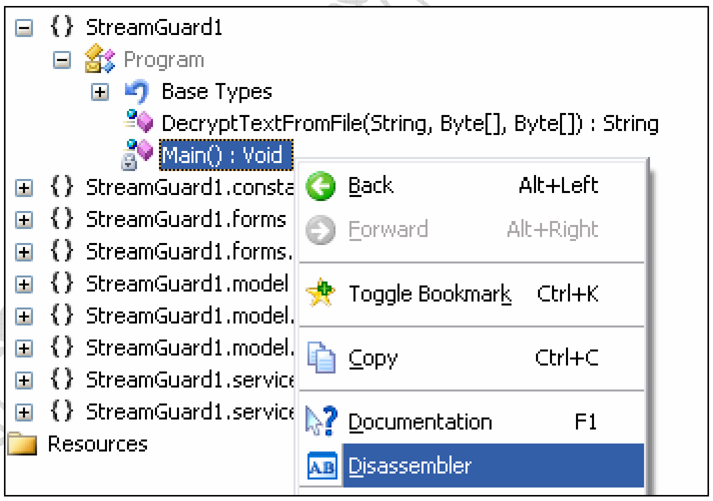
A nasty nag!!!!!

The author has led us right to the check. Look closely at the nag, its looking for "license.dat".

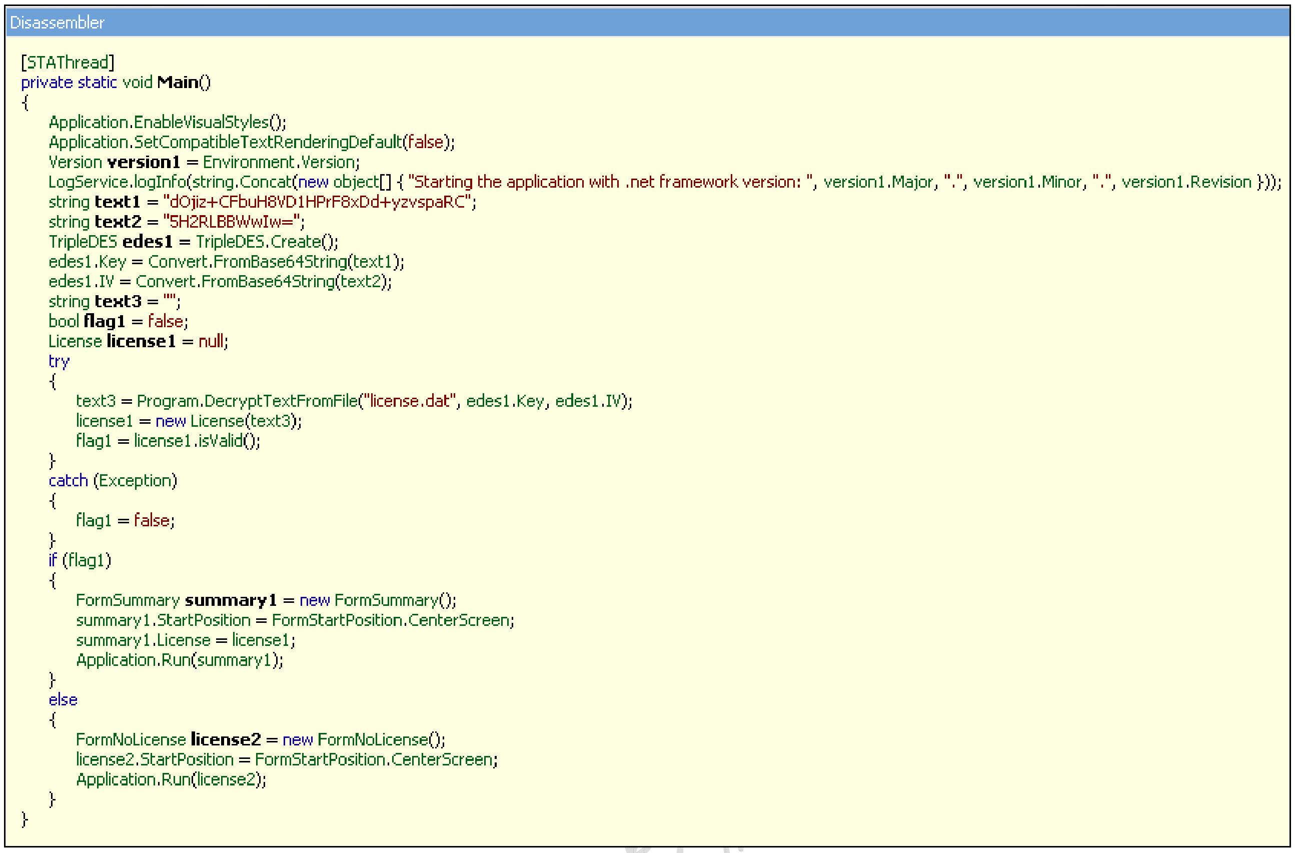

StreamPatrol
StreamPatrol.exe
References
{ } -
{ } com.streampatrol.Properties
{ } StreamGuard1
Program
Base Types
DecryptTextFromFile(String, Byte[], Byte[]) : String
Main() : Void
{ } StreamGuard1.constants

We see the Main() : Void

We then right click it then we will disassemble it.



And then look to the right side to see our disasm.


Disassembler

```
[STAThread]
private static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Version version1 = Environment.Version;
    LogService.logInfo(string.Concat(new object[] { "Starting the application with .net framework version: ", version1.Major, ".", version1.Minor, ".", version1.Revision }));
    string text1 = "dOjiz+CFbuH8VD1HPrF8xDd+yzvspaRC";
    string text2 = "5H2RLBBWwIw=";
    TripleDES edes1 = TripleDES.Create();
    edes1.Key = Convert.FromBase64String(text1);
    edes1.IV = Convert.FromBase64String(text2);
    string text3 = "";
    bool flag1 = false;
    License license1 = null;
    try
    {
        text3 = Program.DecryptTextFromFile("license.dat", edes1.Key, edes1.IV);
        license1 = new License(text3);
        flag1 = license1.isValid();
    }
    catch (Exception)
    {
        flag1 = false;
    }
    if (flag1)
    {
        FormSummary summary1 = new FormSummary();
        summary1.StartPosition = FormStartPosition.CenterScreen;
        summary1.License = license1;
        Application.Run(summary1);
    }
    else
    {
        FormNoLicense license2 = new FormNoLicense();
        license2.StartPosition = FormStartPosition.CenterScreen;
        Application.Run(license2);
    }
}
```

And a few things will pop out right away.

First off we will learn a bit about Reflector and what we should be doing here. If we look close we see

License Check

```
try
    {
        text3 = Program.DecryptTextFromFile("license.dat", edes1.Key, edes1.IV);
        license1 = new License(text3);
        flag1 = license1.isValid();
    }
    catch (Exception)
    {
        flag1 = false;
    }
```

So we see we get something set here, like "Flag1". So we could easily set the flag, but more problems could arise from this, as other members could use the same function. So let's check to see if it does. If you go to the "flag1 = License1.isValid();" and the "isValid" should be underlined. If you click it you should get here: