Lab: Code Forensics and Ransomware

The code objectives of this part of the lab are to:

- Understand the lack of protection that .NET and Java have with code protection.
- Investigate methods of obfuscation of code.
- Create Microsoft .NET code in order to investigate a host.
- Analyse a ransomware evidence bag.

Lab demo: http://www.youtube.com/watch?v=x1jhSIo-GoI

Microsoft .NET Obfuscation

A.1 Microsoft .NET does not have inherent protection against the reverse engineering of the code. To prove this, first create a C# program named **simple.cs**, with the contents of:

```
namespace simple {
    class simple {
      private static void Main(string[] args) {
         string s;
         System.Console.Write("What is your name?");
         s = System.Console.ReadLine();
         System.Console.WriteLine("Hello " + s);
      }
    }
}
```

A.2 Compile the program, and program and make sure that that it works. From the command prompt you can compile it with:

```
csc simple.cs
```

Note: To compile a .NET 2.0 program, you can access the compiler from:

c:\windows\Microsoft.NET\Framework\v2.0.50727\csc.exe

A.3 Next download the reverse engineering package from:

http://asecuritysite.com/exemplar.zip

and prove that you can reverse the code using:

```
    exemplar simple.exe > mycode.cs
```

A.4 Next run the obfuscator (from 9Rays) with:

```
    ob.exe FTBSNM4ALPERC9# /src=simple.exe
```

The obfuscator is downloaded from:

- http://asecuritysite.com/ob.zip

A.5 Go into the /obfuscated folder, and copy the obfuscated EXE into the home folder. Show that the EXE is now obfuscated.

What has changed in the obfuscated EXE?

Is it still possible to compile the reverse engineered code? Yes/No

Using Google, which packages can be used to obfuscate .NET assemblies?

Which options in the obfuscator changes the names of the variables to non-printing characters?

Create the following C# file and compile it to an EXE:

```
using System;
namespace simple {
class simple {

public static int calc(int a, int b)
{
        return(a+b);
}
private static void Main(string[] args) {
string s;
        s="What is the capital of England";
        int val1=5;
        int val2=6;

        System.Console.Write(s);
        s = System.Console.ReadLine();
        if (s=="London")
        {
                System.Console.WriteLine("Correct");
        }
        else
                System.Console.WriteLine("Incorrect");
        System.Console.WriteLine("Result is: "+Convert.ToString(calc(val1,val2)));

        }

 }

}
```

Now download ILSPY from:

http://ilspy.net/

Can you view your EXE in ILSPY?

Now obfuscated your EXE with the following options and observe the changes in ILSPY:

ob.exe NT /src=simple.exe

ob.exe 9 /src=simple.exe

ob.exe 8 /src=simple.exe

Java Reverse Engineering

A.6 Create a Java program (sample.java) with:

```
public class sample
{
   public static void main(String[] args)
   {
     int i;
     i=10;
     System.out.println("This is an example of the ");
     System.out.println("output from the standalone");
     System.out.println("program");
     System.out.println("The value of i is " + i);
   }
}
```

A.7 Next produce the byte code with:

```
javac sample.java
```

If your system does not find the Java compiler you can normally run from a folder on your system, such as:

C:\Program Files (x86)\Java\jdk1.7.0_71\bin\javac.exe

A.8 Finally download JAD, and try and decompile the byte code. Prove that you can reverse the code. The download for JAD is at:

http://asecuritysite.com/jad.zip

Using Google, which packages can be used to obfuscate Java class files?

Ransomware Analysis

The following page contains an evidence bag for the Cerber ransomware. Complete the tutorial:

https://asecuritysite.com/subjects/chapter87

Additional Python Lab

We normally detect a file with its magic number, which is often the first few bytes at the start of the file, or at the end. For example, a JPEG file begins with the hex sequence of 'FF' and 'D8'. The following is the Python code to determine a JPEG file:

```
f = open("1111.jpg", "rb")

try:
      byte1 = hex(ord(f.read(1)))
      byte2 = hex(ord(f.read(1)))
      if (byte1=='0xff' and byte2=='0xd8'):
            print 'JPEG'

finally:
      f.close()
```

Table 1 outlines some magic number (refer to http://asecuritysite.com/forensics/magic). Implement a Python program which detects file types for their magic numbers.

Table 1: Magic numbers

Description	Extension	Magic Number
Adobe Illustrator	.ai	25 50 44 46 [%PDF]
Bitmap graphic	.bmp	42 4D [BM]
JPEG graphic file	.jpg	FFD8
JPEG 2000 graphic file	.jp2	0000000C6A5020200D0A [....jP..]
GIF graphic file	.gif	47 49 46 38 [GIF89]
TIF graphic file	.tif	49 49 [II]
PNG graphic file	.png	89 50 4E 47 .PNG
Photoshop Graphics	.psd	38 42 50 53 [8BPS]
Windows Meta File	.wmf	D7 CD C6 9A
MIDI file	.mid	4D 54 68 64 [MThd]
Icon file	.ico	00 00 01 00
MP3 file with ID3 identity tag	.mp3	49 44 33 [ID3]
AVI video file	.avi	52 49 46 46 [RIFF]
Flash Shockwave	.swf	46 57 53 [FWS]
Flash Video	.flv	46 4C 56 [FLV]
Mpeg 4 video file	.mp4	00 00 00 18 66 74 79 70 6D 70 34 32 [....ftypmp42]
MOV video file	.mov	6D 6F 6F 76 [....moov]

Windows Video file	.wmv	30 26 B2 75 8E 66 CF
Windows Audio file	.wma	30 26 B2 75 8E 66 CF
PKZip	.zip	50 4B 03 04 [PK]
GZip	.gz	1F 8B 08
Tar file	.tar	75 73 74 61 72
Microsoft Installer	.msi	D0 CF 11 E0 A1 B1 1A E1
Object Code File	.obj	4C 01
Dynamic Library	.dll	4D 5A [MZ]
CAB Installer file	.cab	4D 53 43 46 [MSCF]
Executable file	.exe	4D 5A [MZ]
RAR file	.rar	52 61 72 21 1A 07 00 [Rar!...]
SYS file	.sys	4D 5A [MZ]
Help file	.hlp	3F 5F 03 00 [?_..]
VMWare Disk file	.vmdk	4B 44 4D 56 [KDMV]
Outlook Post Office file	.pst	21 42 44 4E 42 [!BDNB]
PDF Document	.pdf	25 50 44 46 [%PDF]
Word Document	.doc	D0 CF 11 E0 A1 B1 1A E1
RTF Document	.rtf	7B 5C 72 74 66 31 [{ tf1]
Excel Document	.xls	D0 CF 11 E0 A1 B1 1A E1
PowerPoint Document	.ppt	D0 CF 11 E0 A1 B1 1A E1
Visio Document	.vsd	D0 CF 11 E0 A1 B1 1A E1
DOCX (Office 2010)	.docx	50 4B 03 04 [PK]
XLSX (Office 2010)	.xlsx	50 4B 03 04 [PK]
PPTX (Office 2010)	.pptx	50 4B 03 04 [PK]
Microsoft Database	.mdb	53 74 61 6E 64 61 72 64 20 4A 65 74
Postcript File	.ps	25 21 [%!]
Jar File	.jar	50 4B 03 04 14 00 08 00 08 00

There are more than 30 files contained in this evidence bag:

http://asecuritysite.com/evidence.zip

Now, using your Python program, see if you can match the magic number, and then change the file extension, and see if you can view them.

File	Type	What it contains …
file01		
file02		
file03		
file04		
file05		
file06		

file07		
file08		
file09		
file10		
file11		
file12		
file13		
file14		
file15		
file16		
file17		
file18		
file19		
file20		
file21		
file22		
file23		
file24		
file25		
file26		
file27		
file28		
file29		
file30		
file32		
file33		

file34		
file35		
file36		
file37		
file38		
file39		
file40		