**Introduction**

Presented by: MAthieu Cruts Sören rogiers

Software Development method: Extreme Programming

Steganography Project Plan

Version 1.0

April 1, 2014

Building a command line application which hides data in tilled images created from 1 single image. The tiles will always be squares.

The data to be hidden will be a file that has been encrypted using a symmetric algorithm. The password can be derived from the image, once all the tiles have been put together.

**Planning (User story)**

When the user wants to embedd a file into a picture (Png,Gif,Jpg) he opens the commandprompt and types :

* Java programName path\_to\_inputFileImage path\_to\_fileToEmbed path\_to\_outputDirectory numberOfTiles=10\&rotation90=10%\&customManipulation=20%&customMethod=Sample.manipulate().

After executing the command he can find his images with the hidden data in the output folder.

**Iteration**

1. Crop an image (.PNG, .JPEG, .GIF) to a square

* Using BufferedImage (of Abstract class Image), read through Java’s ImageIO library
* Check width and height of original image -> crop to a square
* Use a Rectangle to intersect with the dimensions of the input image (java.awt.Rectangle.intersection())
* Use a new SubImage (BufferedImage) that represents the square

🡺 Obtained through: java.awt.image.BufferedImage.getSubimage()

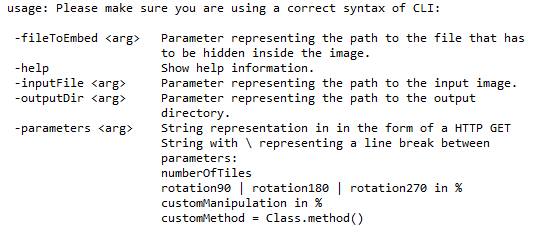
* Using Graphics2D to draw the new Image.

1. Creating the tiles

* Tiles will be created by using the Graphics2D class (java.awt) which will chop a single BufferedImage up into a given amount of tiles, which will then be stored in an array of new BufferedImages
* The new Tiles can be written as single JPEG’s using javax.imageio.ImageIO.write()

1. Arguments & Parameters

* The command line arguments will be handled by the Apache CLI library (org.apache.commons.cli.CommandLine).
* Every argument will be stored in an Options object (org.apache.commons.cli.Options) which also holds a description for the user.
* A HelpFormatter (org.apache.commons.cli.HelpFormatter) will also be provided in case the user has problems with the syntax of running the command. The image below holds the output of the –help commando:



* The parameters will be in the form of a HTTP GET string and all of the parameters will have default values as well. The example below shows a correct parameter syntax:

numberOfTiles=10-20 \

&rotation90=10%\

&customManipulation=20%&customMethod=Sample.manipulate()

* An example of a fully formed valid command to run the program would then be:

java programName path\_to\_inputFileImage path\_to\_fileToEmbed path\_to\_outputDirectory numberOfTiles=10\&rotation90=10%\&customManipulation=20%&customMethod=Sample.manipulate()

1. Extra Features

The data to be hidden will be a file that has been encrypted using a symmetric algorithm.

The bytes of the file will be segmented and then hidden into the chopped tiles of the image, the first part of the data will go in the top-left tile and so on in a row-wise fashion. The password can be derived from the image, once all the tiles have been put together.

In the byte of data, the bits have a rank where the leftmost bit is the most significant one and the rightmost is the least significant one. If we want to change some data in an image, we want it to be as unobtrusive as possible, or even invisible. Thus we want to apply our changes to the least significant bit of some of the bytes. In this way we change each byte, a maximum of 1 in value.

An example approach would be to alter one bit of the RGBA values of 2 adjacent pixels of the image using a modulus operator based algorithm:

Byte of data to hide = (bit1, bit2, bit3, bit4, bit5, bit6, bit7, bit8)

Pixel A = R1, G1, B1, A1

Pixel B = R2, G2, B2, A2

(R1, G1, B1, A1) – [(R1, G1, B1, A1) % 2] + (bit1, bit2, bit3, bit4)

(R2, G2, B2, A2) – [(R2, G2, B2, A2) % 2] + (bit5, bit6, bit7, bit8)

Example operation with 2 white pixels (255, 255, 255, 0):

Byte of data to hide = (0, 1, 0, 0, 1, 0, 0, 0)

Pixel A:

(255, 255, 255, 0) – [(255, 255, 255, 0) % 2] + (0, 1, 0, 0)

= (255, 255, 255, 0) – (1, 1, 1, 0) + (0, 1, 0, 0)

= (254, 255, 254, 0)

Pixel B:

(255, 255, 255, 0) – [(255, 255, 255, 0) % 2] + (1, 0, 0, 0)

= (255, 255, 255, 0) – (1, 1, 1, 0) + (1, 0, 0, 0)

= (255, 254, 254, 0)

Pixel A’ = (254, 255, 254, 0)

Pixel B’ = (255, 254, 254, 0)

Finally, we will also be using a false footer added to the image to confuse the investigators.

**Unit Testing**

Unit tests are one of the corner stones of Extreme Programming . But unit tests XP style is a little different. First we’re going to create or download a unit test framework to be able to create automated unit tests suites. Second we will test all classes in the system. Trivial getter and setter methods are usually omitted. We will try to create our tests first before writing code.

The biggest resistance to dedicating this amount of time to unit tests is a fast approaching deadline.

The harder the test is to write the more you need it because the greater your savings will be. Automated unit tests offer a pay back far greater than the cost of creation.

In order to have a complete unit test suite when you need it you must begin creating the tests today.

Building a single universal unit test suite for validation and regression testing enables frequent integration. It is possible to integrate any recent changes quickly then run your own latest version of the test suite. When a test fails your latest versions are incompatible with the team's latest versions. Fixing small problems every few hours takes less time than fixing huge problems just before the deadline.

**References**

Apache Commons CLI Library : http://commons.apache.org/proper/commons-cli/

StackOverflow: stackoverflow.com/

Image cropping: http://www.java-tips.org/

DreamInCode: http://www.dreamincode.net/forums/topic/27950- steganography/