CIS192 Project Overview

2012-04-24

1 Source

Code is available at https://github.com/zwass/Sneaky

2 Group

- Nick Watson (nwatson)
- Zachary Wasserman (zwass)

3 Description

We built a system to encode hidden messages into images. As discussed in CIS551 (Computer and Network Security), one can slightly modify the pixels of an image in order to encode data, without making changes visible to a casual observer. This process is called "watermarking", and is a type of steganography, which is the practice of concealing a message to hide its existance. This is different from standard cryptography, which relies on mathematically proven methods to transform plaintext to ciphertext, which is plainly visible but hard to decrypt without some secret knowledge (e.g. a key).

Our scheme allows for the encoding of a relatively large amount of data onto a source image. For example, we were able to encode the entire script of Hamlet onto a portrait of William Shakespeare. Using a parity-bit encoding, we can allow decoding without even requiring a reference image.

Encoding and decoding are quite fast. First we take the input data and convert it to a bitstring. We then traverse the image, slightly modifying the red value of each pixel to encode a single bit. On decoding, we can examine the parity of the red-values, and from it recover the message. Using a medium-powered laptop computer, we are able to perform these actions on the order of a few seconds.

We attempted an optimization by using bitwise and arithmetic operations instead of string operations, but this did not provide a significant speed-up. This is most likely due to the fact that Python is too far-removed from hardware to gain the same time benefit from these operations as a lower-level language like C would.

Our software allows for a fun and somewhat covert transmission of data. Obviously, there is no actual encryption, but by appearing to be normal images, the encoded images that we post are likely to afford "security through obscurity."

Lists are the primary data structure harnessed by our software. We use the pixel matrix provided by the Python Imaging Library (PIL), and single dimensional lists for our encoded data and bitstrings. Network communication is mostly assisted by the TwitPic and Tweepy API wrappers for TwitPic and Twitter, which provide abstractions over the OAuth and REST based APIs provided by the services.

The biggest challenge in creating this software was working with the PIL to perform the actual encoding and decoding of messages onto the images. Any single error in our encoding resulted in an impossible to decode message. Ultimately we found that starting small and working up to the more complicated encoding scheme, and larger images and messages was helpful in correctly implementing the project.

Working on this project afforded us a broad range of experience of developing Python applications. We learned how to manipulate images and files, as well as interacting with third party APIs using the assistance of publicly available Python libraries.

Had we had more time, we would have liked to build a GUI for working with the images and messages. This seemed like it could have been an entire project in itself, so we instead chose to focus on functionality.

An additional feature that might be fun to have is the ability to do both cryptography and steganography simultaneously. Before encoding the message into the image, it could be encrypted using RSA or AES. This would make it virtually impossible to determine whether an image actually contained a message, since running the decoding algorithm would appear to output gibberish if the incorrect key were given.

4 Features

- Encoding and decoding of hidden data in watermarked images
- Only one image is required to decode a message. The encoding scheme uses a parity bit to determine the hidden data.
- Integration with Twitter
  - Automatically encode and post messages to Twitter
  - Decode messages from images in a Twitter stream

5 Libraries

- Python Imaging Library (http://www.pythonware.com/products/pil/)
- Python Requests (http://docs.python-requests.org/en/v0.10.7/index.html)
- Python TwitPic (https://github.com/macmichael01/python-twitpic)
- Tweepy (http://code.google.com/p/tweepy/)

6 Responsibilities

6.1 Nick

- Parity bit encoding/decoding
- Twitter integration

6.2 Zach

- Basic encoding/decoding
- Twitter integration