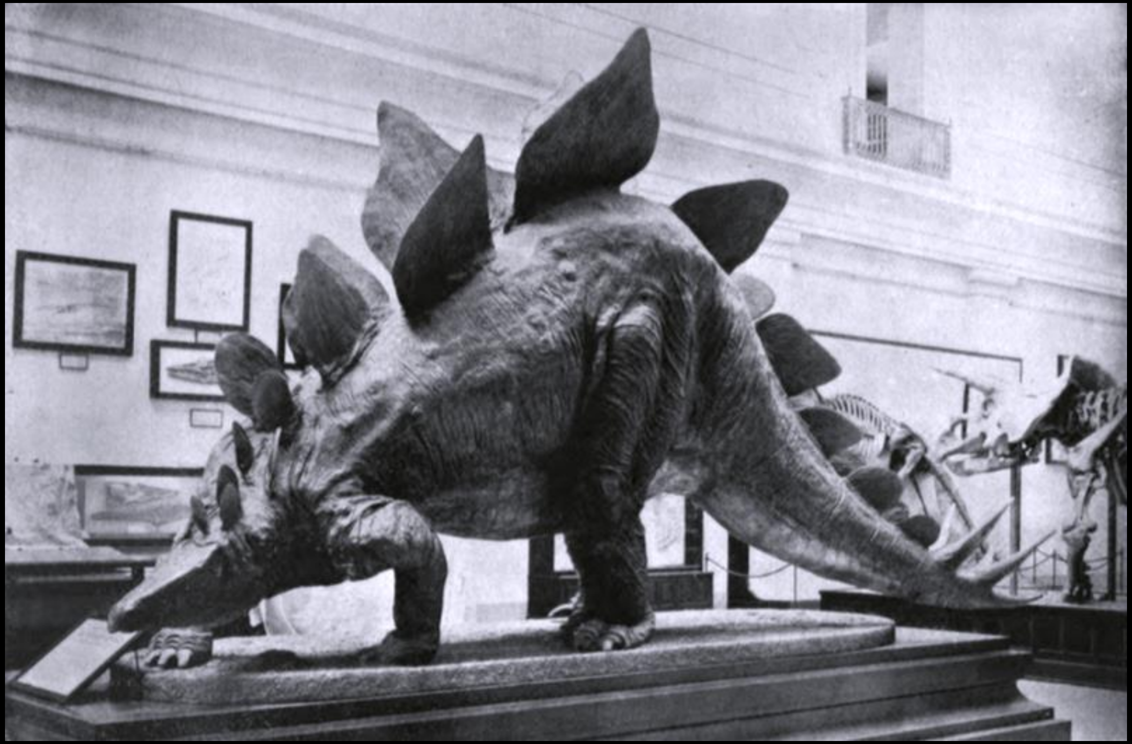
Intro to Steganography

WiCyS Illinois



Workshop format

- A broad overview of steganography
- Why we care about it
- Python example
- How to approach steganography CTF challenges

What is Steganography?

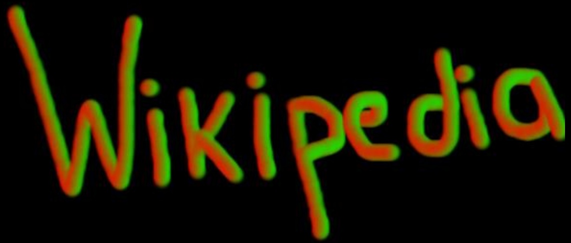
- The practice of hiding something within another thing
- An example you may have tried before: writing with lemon juice
- Of course, today we'll focus on digital steganography

But why steganography? Can't we just hide stuff with encryption?

- Yes, but....
- Encryption usually looks like it's obviously encrypted
- Some techniques exist for trying to identify steganography, but even so, it will probably catch less attention than something encrypted which you're obviously trying to hide

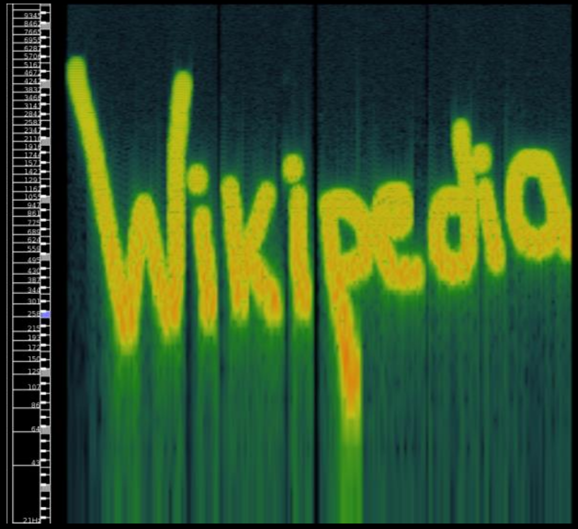
Some digital examples of steganography

- Hiding an image inside of another image
- Hiding a text-based message in an image
- Hiding an image in an audio file



An image spelling out Wikipedia

Image is converted to audio file



Audio spectrogram of the resulting file – looks familiar!

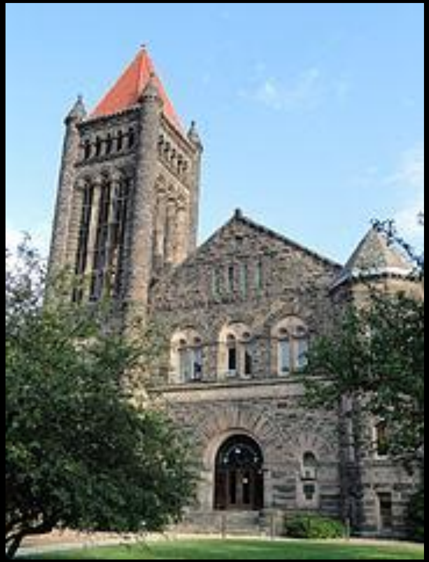
By Skimel - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=82251192

By Skimel - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=82251009

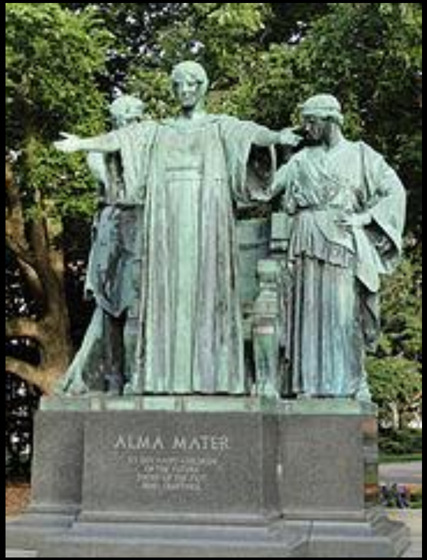
Source for the following slides and code: https://towardsdatascience.com/steganography-hiding-an-image-inside-another-77ca66b2acb1

Example: hiding an image inside of another image

We will hide Altgeld...



...Inside of Alma!

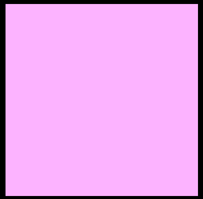


First: some light info on how images work

- Everything is constructed of pixels
- Different color systems can be used: printing tends to use cyan/magenta/yellow/black to create all other colors, red/green/blue (RGB) is common for digital images
- For RGB representation: each pixel is a 6-bit hex value representing levels of (red, green, blue)
- In decimal, each color level can be represented by a number from 0 to 255 (inclusive)

Second, some light info on binary

#fcb3ff



Amount of red = #fc
= 11111100 in binary
= 4 + 8 + 16 + 32 + 64 + 128
= 252 in decimal

Amount of green = #b3
= 10110011 in binary
= 1 + 2 + 16 + 32 + 128
= 179 in decimal

Amount of blue = #ff
= 11111111 in binary
= 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128
= 255 in decimal

- The left-most bit holds the largest value, and the right-most bit holds the smallest value because each bit is equal to x * 2^i (s.t. x = 0 or 1, i = 0 on the rightmost bit, increments by 1 to the left)
- Each of these bits are summed up
- So, the bits on the right make less of a difference in the final value than those on the left

Now, why it applies to us

- We want to encode one image in another image's pixels.
- Changing certain bits leads to a big change in the original pixel color, while other bits don't make it look visually different.
- This means we should change the **least-significant bits of the original image and** hide the **most-significant bits** of the other image inside them!

An example of how this works

Say this is a pixel from the image we are storing our second image in:

R(11001010)
G(00100110)
B(11101110)

And, say this is a pixel from the image we are hiding:

R(00001010)
G(11000001)
B(11111110)

The final, combined pixel would be:

R(11000000)
G(00101100)
B(11101111)

(The final pixel simply replaces the least significant 4 bits of the original image with the most significant 4 bits of the hidden image)

An example of how this works

The final, combined pixel would be:
R(11000000)
G(00101100)
B(11101111)

So, we could extract the hidden pixel and end up with:
R(00000000)
G(11000000)
B(11110000)

We had to pad the least significant bits with zeros since we don't know what they are. This leads to a slight loss of image quality, but that's ok.

Python code implementing this idea

- See on Google Colab: https://colab.research.google.com/drive/12BhmgV0xNzAttI2n4hEygUemvTV0UVxb

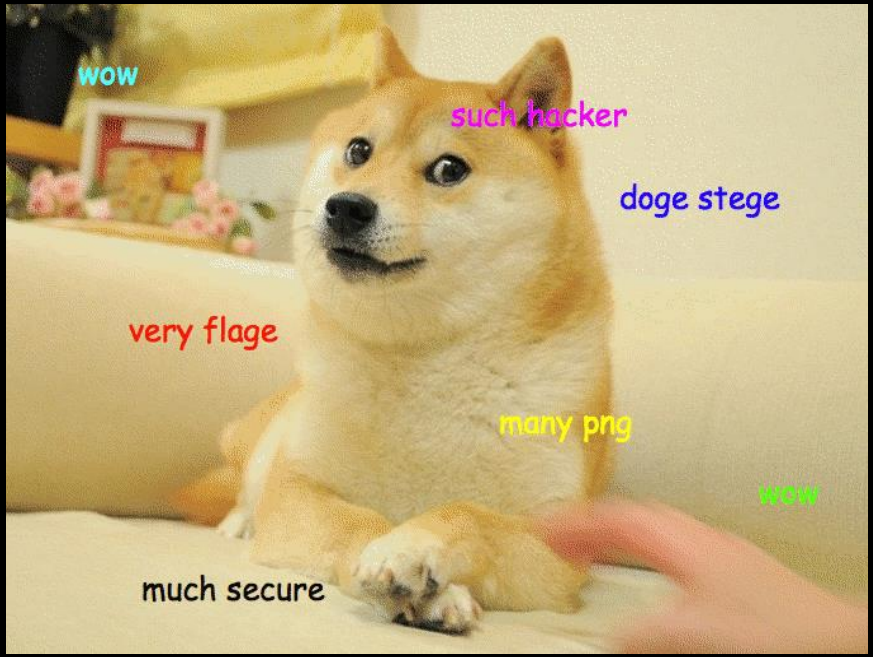
Ok, so how do I approach steganography CTF problems?

- Probably a good idea to start with understanding file metadata
- Go from there – view the image, Google it, look at the header, file signature, etc.
- Try using known steganography tools to aid your search for the flag!

For greater detail on these ideas and some specific tools to use, see: https://fareedfauzi.gitbook.io/ctf-checklist-for-beginner/steganography

CTF problem example

- This image is from PlaidCTF 2014; it contains a flag!



If you would like to try solving it…

This link provides the image and an overall text-in-image guide:

- https://ctfs.github.io/resources/topics/steganography/invisible-text/README.html

Or, see a solution write-up here:

- https://github.com/ctfs/write-ups-2014/tree/master/plaid-ctf-2014/doge-stege