

## Steganography Examples with Outguess

Without getting into the history or politics of steganography I've set this page up as a (hopefully) light reading reference for those who are interested in the 10,000 foot view. There are plenty of elaborate sites out on the web with detailed examples and definitions of steganography, this is not one of them.

In summary, steganography is simply a means of hiding the existence of communication. In contrast to encryption, in which communications are scrambled, steganography attempts to hide the fact that a message even exists.

A classic example of early steganography from the late 6th century BC involves a slave who had his head shaved, a message tattooed to his head, and then once his hair grew back he was sent out to deliver the message.

In modern steganography messages are likely to be encrypted, and then hidden in some electronic media. For example a secret message may be encrypted and then hidden in an image file.

In my current line of work I've often wondered how many of the random images I see contain hidden messages. I don't really play with steganography much these days but in the past I have run some scripted tests against large repositories of images. Many others have done the same thing and published their studies, my tests were simply for personal curiosity. My favorite steganography tool is Outguess. The author of outguess, Niels Provos, has also written the stegdetect and stegbreak tools (as well as many other excellent contributions to community).

The following example assumes you have a basic understanding of computers. Having a UNIX or Linux background is helpful but not required.

In the example we are just going to hide a plain text message inside of an image. Plain text means just that, no encryption, just some text you created in your favorite text editor (notepad, vi, etc). (Note that MS Word is NOT a text editor, but it can be used as one.)

Here we have a photograph of a coyote hiding behind some leaves while standing on a stuccoed block wall. This photograph is saved as a JPEG file named "coyote-a.jpg". Here is the original image:



## Steganography Examples with Outguess

Now, we are going to hide the contents of a file called "secret-message.txt" inside a copy of the image. The "secret-message.txt" file contains the sentence "This is a super secret message.". We are going to use a key - or passphrase - to essentially password protect our hidden message. More on that later.

Our key is 'super secret passphrase'. The new image file will be called 'steged-coyote-a.jpg'. The original image is untouched.

Using outguess makes this very simple. Here is the command we used to do the deed:

```
outguess -k 'super secret passphrase' -d secret-message.txt coyote-a.jpg
steged-coyote-a.jpg
```

The -k (-k for key) flag says to use 'super secret passphrase' as our key.

The -d (-d for data) flag says to hide the contents of the secret-message.txt file inside of a new file called steged-coyote-a.jpg for which we use the coyote-a.jpg file as a reference.

Hard to read due to line warps but the complete example is here:

```
# outguess -k 'super secret passphrase' -d secret-message.txt coyote-a.jpg steged-
coyote-a.jpg
Reading coyote-a.jpg....
JPEG compression quality set to 75
Extracting usable bits: 83084 bits
Correctable message size: 43152 bits, 51.94%
Encoded './secret-message.txt': 256 bits, 32 bytes
Finding best embedding...
0: 158(54.9%)[61.7%], bias 163(1.03), saved: -3, total: 0.19%
1: 142(49.3%)[55.5%], bias 149(1.05), saved: -1, total: 0.17%
2: 141(49.0%)[55.1%], bias 145(1.03), saved: -1, total: 0.17%
3: 138(47.9%)[53.9%], bias 134(0.97), saved: -1, total: 0.17%
5: 138(47.9%)[53.9%], bias 131(0.95), saved: -1, total: 0.17%
8: 139(48.3%)[54.3%], bias 125(0.90), saved: -1, total: 0.17%
12: 131(45.5%)[51.2%], bias 125(0.95), saved: 0, total: 0.16%
17: 131(45.6%)[51.2%], bias 117(0.89), saved: 0, total: 0.16%
41: 127(44.4%)[49.6%], bias 116(0.91), saved: 0, total: 0.15%
56: 120(41.7%)[46.9%], bias 116(0.97), saved: 1, total: 0.14%
125: 129(44.8%)[50.4%], bias 105(0.81), saved: 0, total: 0.16%
125, 234: Embedding data: 256 in 83084
Bits embedded: 288, changed: 129(44.8%)[50.4%], bias: 105, tot: 82730, skip: 82442
Foiling statistics: corrections: 59, failed: 0, offset: 14.500000 +- 9.211794
Total bits changed: 234 (change 129 + bias 105)
Storing bitmap into data..
Writing steged-coyote-a.jpg...
#
```

The resulting image 'steged-coyote-a.jpg' looks like this:

## Steganography Examples with Outguess



Now, both of the coyote images shown here are reduced in size but if you click on them they will expand. To the casual observer there is no difference between the two images. Go ahead and download them to your computer and look at them in Photoshop or Gimp or you favorite image editor. You wont find anything.

To extract our hidden message we can use outguess, if we know the key. The command used to retrieve our hidden message is not much different than the one used to hide it.

```
# outguess -k 'super secret passphrase' -r steged-coyote-a.jpg message.txt
Reading steged-coyote-a.jpg....
Extracting usable bits: 83084 bits
Steg retrieve: seed: 125, len: 32
# cat message.txt
This is a super secret message.
#
```

We used -r (-r for retrieve) to get our secret message out of the file called steged-coyote-a.jpg and write this to a file called message.txt.

But what if we don't know a message is hidden? That is pretty much the whole point of steganography. In some way steganography is similar to geocaching as things are hidden in plain sight and it takes a keen eye and some experience to find those hidden gems.