

Hiding in plain sight

Using steganography to avoid observation

Skill Level: Intermediate

[Sean-Philip Oriyano](mailto:sean.oriyano@gmail.com) (sean.oriyano@gmail.com)

IT Instructor
oriyano.com

02 Jun 2009

Steganography is the technique of hiding information within some format in a way that makes it difficult to detect by one who doesn't know it's there. In the computer age, steganography has become quite advanced and allows for information to be hidden in all types of data files, such as images, documents, and space on a hard disk. Learn how potential attackers could employ this technique, and gain some insight on steganalysis, the detection of its use.

Steganography is an age-old process. This powerful technique has gotten quite a lot of ink dedicated to it post the 9/11 terrorist attacks as a method that could enable terrorists, subversives, and others to communicate in a way that is incredibly difficult to detect. This article examines the process of steganography and shows how you can detect its use.

Before I get started, let me first differentiate this process from one of the other processes used to alter data and keep it away from those not authorized to view it: *encryption*. Encryption can alter a piece of clear-text, or unencrypted information, into cipher-text, or encrypted information. What does encryption *not* do? Well it doesn't prevent someone from knowing that data is being transmitted, as encrypted data can be intercepted and examined. But without the all-important key, it cannot be easily deciphered. Compare this with *steganography*, where data—encrypted or otherwise—is hidden within something else in such a way that those who are unaware will only see the main item and not the data hidden within it. Why use one technique over the other? Simply put, if you transmit an encrypted message, it can attract unwanted attention from those from whom you want to protect it, while steganography can be used to send communications right under the nose of a third

party with little chance of raising suspicions.

Steganography in action

Let's take a look at how this whole process works, as it can be tricky to understand without a bit of a primer.

For this example, consider a common, standard image format like JPEG. JPEG files are popular because of their versatility and support, as well as (most importantly) their ability to support 16.7 million colors (the maximum amount of color the human eye can perceive). Now consider a standard digital photograph taken with an above-average or high-end camera of any given subject, such as a sunset, a beach, or the Las Vegas Strip. Any of these subjects will contain a tremendous amount of color and detail, as well as—depending on the camera—a varying amount of what's known as *white noise*.

So, what is this white noise? It is the background or randomized information contained in any piece of data. In this case, think of it as the imperceptible or unused information in the image. Someone employing a steganographic technique can use this noise to hide "secrets" by altering the information in ways that cannot be easily noticed. In this image, consider that each pixel is made up of a combination of numbers used to specify the color that will be displayed.

In any standard image, a pixel has a sequence of three numbers associated with it that dictate the mixture of the red, green, and blue (RGB) channels for the pixel (and therefore the color displayed). In an RGB value, any given channel can contain a value anywhere from 0 (no color) to 255 (full color). In the context of the digital world, each of these values is stored in a sequence of 8 bits, with the Most Significant Bit (MSB) being 128 and the Least Significant Bit (LSB) being 1. Let's take a look at a standard pixel:

```
Original pixel value: 255, 255, 255 (White)
Same pixel value in binary: 11111111, 11111111, 11111111
```

Now, alter this pixel to store data:

```
New pixel value: 255, 255, 255 (White)
Same pixel value in binary: 111111rr, 111111gg, 111111bb
```

In my new example, I marked the LSBs in each channel to call attention to them. In the real world, I could alter the LSB of each of these channels across several pixels to store the codes associated with letters. Because most letters only need 8 bits, I have more than I need to reflect a letter in each pixel of an image. Keep in mind that while I am using the four LSBs of each pixel to describe a letter of text, the same

change will alter the color of the pixel; however, because of how small a change is made to store a single letter in each pixel, the resulting change in color is equally small—too small for the naked eye to see. Unless you have the original image, noting the change between the new and original images is a difficult task (although software is available to help with this detection process).

When you understand how a pixel can be altered in such a way to store text information (or other data, for that matter), you can move to the next step, which is figuring out scale. Consider a JPEG image 1024 x 768 pixels in size—not uncommon by any means. If you do the calculations, you can determine that an image of this size contains around 800,000 pixels—that's a lot of pixels that can carry information. For an image 1200 x 1600? This image size contains over a million pixels that can each be altered in this way. That's enough to carry the text of a novel with room to spare.

Steganography tools

So, how do you perform this process? Well, the reality is that a number of tools—both online and downloadable—are available to perform this process. Additionally, several tools are available to carry out detection of information that has been hidden using steganography—a process known as *steganalysis*.

Currently, you can locate tools for performing steganography functions on different types of media. Here, I introduce a handful of tools—primarily, image-based tools—designed to carry out the technique on some common types of media.

Hiding Glyph

The Hiding Glyph application is used to store a file or group of files within a standard, non-compressed, 24-bit (or greater) bitmap image. The benefit of this application is that you can select any image that contains a reasonable variation of color as a location to hide your files and make them free from detection. Furthermore, without having both the original and the altered image, detection—not to mention actual extraction of the information—becomes almost impossible.

Vecna

Vecna is an interesting application that hides data in an image using [the method I described earlier](#). Vecna has the advantage of being a Java™-based application, and as such can use an image format that the Java language can read as a location to hide data. After the information is embedded in the image, the resulting data is output as a PNG file, with alpha channel information used to carry the data.

This application stores its hidden information by using a rather interesting method: dispersing the added information around the image in a random fashion. Vecna

employs its own random number generator to achieve this level of relative chaos in the data. It is also worth mentioning that the techniques this application uses to hide data can be used to hide all types of data in an image, including text, Portable Data Format (PDF), and even other images.

TrueCrypt

TrueCrypt is a popular tool for performing encryption of a volume of data, but it can provide an additional layer of protection by hiding a volume inside an already-encrypted volume. You can use TrueCrypt to create this initial encrypted volume, then take the free space within the encrypted volume to hide additional data. TrueCrypt states that this feature gives a measure of plausible deniability, as one can easily detect that there is an encrypted volume. But even if someone manages to crack the key on the volume and open it, there is relatively little chance of detecting the hidden volume, even if that person suspects it is actually there.

The power of TrueCrypt is something worth noting properly, so let me mention a couple of things. First, TrueCrypt can encrypt and hide files in the same way as images—no surprises here, as the technique is relatively similar to what I discussed for images. The power of TrueCrypt comes into play when you note just how much can be protected and hidden. Indeed, TrueCrypt can not only hide files but entire operating systems.

F5

F5 is another steganography application that was developed some time ago and has since seen some "face lifts" to make it more usable. Basically, F5 is another tool for hiding data such as .txt or .doc files in images such as JPEGs.

mp3stego

Now we're breaking some new ground. I won't get into the "how" of this tool, but I will bring it to your attention as yet another tool for employing the steganographic process. You can use mp3stego to hide data (as the name implies) within MP3 files of a size proportional to the original data. Basically, the process is this: You obtain a .wav file, hide your data within it, then compress the .wav file into an MP3 file. The benefit of this format is that it can be used to easily hide data of any type in a plain and simple-looking format.

Just envision the possibilities of this process in contrast to the image-based techniques mentioned earlier. MP3 files and images both represent different ways of distributing hidden data. An image file can be posted on a Web site, where someone would be unlikely to suspect that some secret data is present—for example, posting some secret data in a pornographic image (not exactly the first place you would expect to find hidden secrets). An MP3 file, in contrast, could have data encoded into it and be posted on BitTorrent, for example. A recording of a lecture from a

conference could have data coded into it and uploaded, and chances are, if someone downloaded it, he or she would be bored long before even suspecting there was something hidden in the file.

Steganos Privacy Suite

This application package is a commercial software suite that includes a toolbox of steganographic utilities as well as other tools to cover your tracks. Steganos Privacy Suite has the ability to either select an existing file and embed data within it, or create an image or sound file to carry the data. In fact, this suite can interface with scanners or an attached microphone to create the carrier file. Additionally, if you are not sure of the best type of file in which to store your secret data, Steganos can help you scan a hard disk for the best way to hide your information.

Further obfuscation

Want to make things more interesting? If you want to keep your data from being discovered, or at least make it more difficult to be detected, you could add *padding* to your hidden secret. In this technique, detection is thwarted by the addition of bogus data, basically muddying the waters and making the detective determine what is the real data and what is not. Of course, it should be noted that padding additional data increases the likelihood that someone will look in the first place for hidden information.

Detection of steganography

Of course, for every measure, there is a countermeasure, and steganography is no different. In this case, the countermeasure is a technique known as *steganalysis*. The goal of steganalysis is quite simple: Detect suspect data, determine whether information is hidden within the data, and recover that data.

On the surface, this process may sound curiously like a companion technique known as *cryptanalysis*. But the reality is that the processes are not the same. With cryptanalysis, the fact that data is encrypted is obvious: You need only observe the data to determine whether it is encrypted. With steganalysis, the presence of data is only suspected, not confirmed.

How does detection work?

In its simplest and most basic form, steganalysis is performed through the use of statistical analysis techniques. This technique uses methods that compare known, unmodified files with those that are suspect, the idea being that the "fingerprints" of known files can be compared with the suspect files. Through this statistical comparison you can, in theory, detect variations from the normally generated files. The key here is that the known, good files need to originate from the same source as

the suspect files (that is, the digital camera or scanner), or as close a match as possible for the comparison to be effective. Some variations in this technique look for deviations from known compression algorithms in play on JPEGs, MP3s, and other files, as the compression algorithm is publicly known.

One more advanced technique in use is known as *Noise Floor Consistency Analysis*. Without making this sound like a doctoral thesis in mathematics, in this method, comparison is made at the bit level between the MSB and the LSB components of a piece of data. Because the MSB and LSB should have a basic correlation between them, the alteration of the LSB by some steganographic techniques can break this relationship, making detection possible. In fact, some applications that perform this technique can detect hidden data within a highly reliable degree of confidence.

Something to keep in mind with steganalysis techniques is that another layer of complexity may be added to the detection of hidden data. Consider this: If you want to add another element of protection to your hidden data, you could combine it with encryption before embedding the data. In this case, if detection is successful, decryption must also be performed.

Steganalysis tools, like many security tools, are highly specialized, often with a highly specialized price tag. However, if you would like to experiment with some of the techniques, there is an open source project, Stegsecret (see [Resources](#)), which will begin your exploration.

Why steganography?

This is the big question for a lot of individuals seeing this process for the first time. Let me discuss some of the applications of this technique. Keep in mind that like just about anything you can name, this whole process can be used for legal or illegal purposes depending on who is using it and for what. Legitimate applications of this process include the enforcement of intellectual property such as copyrights by applying what are known as *watermarks* to an image. In contrast, some of the more nefarious reasons include hiding information for illegal reasons. In fact, organizations such as Al-Qaeda and others are known to use this process to hide information in harmless images hosted on Web sites.

So, how can you prevent the use of steganographic techniques for the transmission and storage of illegal information? Well, this is a problematic issue to say the least. In the right hands, one could fairly easily hide information in "plain sight" and get away with it. Remember, if done correctly, information could be hidden in any image, document, or other piece of data and never even be suspect. Although you could never successfully confirm how widespread this technique is in illegal or criminal circles, it is probably safe to say it is in use by criminals, terrorists, and other bad folks for all kinds of mischief. As those on the "light side of the Force," we can use

company security policies and a code of honor (so to speak) to prevent the misuse of this technology. Unfortunately, those on the "dark side" don't have such obstacles.

Conclusion

Steganography is an old and incredibly versatile and effective technique for obscuring information that is actually in plain sight. Although methods for detecting hidden data do exist, they cannot be entirely relied upon, as none is 100 percent effective. Any attempt to detect and thwart this technique must combine technology and vigilance to blunt the effectiveness of the process.

Resources

Learn

- For a great [background on steganography](#), check out Wikipedia. Wikipedia also provides good information about [cryptography](#).
- Get the latest security news and information at [securityfocus.com](#).
- The [SANS Institute](#) is your one-stop shop for security information, certification programs, and research.
- The [developerWorks Web development zone](#) is packed with tools and information for Web 2.0 development.
- [IBM technical events and webcasts](#): Stay current with developerWorks' Technical events and webcasts.
- Personalize your developerWorks experience with [My developerWorks](#).

Get products and technologies

- [TrueCrypt](#) is an open source tool that provides data encryption for Windows®, Mac OS X, and Linux® systems.
- Image-based steganographic tools include [Hiding Glyph](#) and [Vecna](#).
- [mP3stego](#) allows you to hide data in MP3 files.
- [Steganos Privacy Suite](#) is a commercial toolset that provides data protection, password management, and more.
- Explore steganalysis with [Stegsecret](#), an open source (GNU/GPL) tool.

About the author

Sean-Philip Oriyano

Sean-Philip Oriyano has been actively working in the IT field since 1990. Throughout his career, he has held positions such as support specialist to consultants and senior instructor. Currently, he is an IT instructor who specializes in infrastructure and security topics for various public and private entities. Sean has instructed for the U.S. Air Force, U.S. Navy, and U.S. Army at locations both in North America and internationally. Sean is certified as a CISSP, CHFI, CEH, CEI, CNDA, SCNP, SCPI, MCT, MCSE, and MCITP, and he is a member of EC-Council, ISSA, Elearning Guild, and Infragard. You can reach Sean at sean.oriyano@gmail.com.