Global Information Assurance Certification Paper

Current Steganography Tools and Methods
Erin Michaud
GSEC Practical, Version 1.4b
April, 2003

Abstract

This paper will provide a review and analysis of several freeware tools that employ some of the more common methods of hiding information in digital files, demonstrating how one can easily embed secret messages in some of the more commonly exchanged image, audio and text file formats. Overviews of different methods of steganography for each file type will be followed with the analysis of tools that provide this functionality. Afterwards, a discussion of methods for detecting and disabling the methods we used will be covered. These discussions will provide the reader with a good understanding of the more common, steganographic techniques and good baseline of understanding for the exploration of more complex methods.

Steganography Overview

Steganography is by no means a modern practice. Literally meaning "covered writing" it is the practice of hiding messages within other messages in order to conceal the existence of the original. Examples of its use can be found throughout history, dating as far back as ancient Greece. However, with the digital media formats in use for data exchange and communication today providing abundant hosts for steganographic communication, interest in this practice has increased. Couple this fact with the multitude of freely available, easy to use steganography software tools on the internet, the ability to exchange secret information without detection is available to virtually anyone who desires to do so, and provides unique challenges and opportunities for the security professional.

For the security professional, this means that data you are paid to protect could be leaving your control without your knowledge. Conversely, one of the emerging uses for steganographic techniques is digital watermarking, which provides an organization with a way to ensure the integrity of data they wish to disseminate by embedding copyright or other information in a digital file. Regardless of whether it is used for good or ill, an understanding of current methods of data hiding should be part of every security professional's knowledge base.

Terminology and Methodology for Tool Review

Some specific terminology in the field of steganography has developed to make clear the differences between files to be hidden, those that they get hidden in and

the resulting combinations of the two. Sellars' provides an excellent definition of the terms that will be used throughout this paper:

"The term 'cover' is used to describe the original, innocent message…[t]he information to be hidden in the cover data is known as the 'embedded' data. The 'stego' data is the data containing both the cover signal and the embedded information." (Sellars, p.3)

For the tools reviewed here, the embedded data, or hidden message is a small text file called mysecret.txt, that contains the first verse of the poem "Mary Had a Little Lamb". The cover files are text, image or audio files that will simply be named cover, with their appropriate file type extension (e.g. cover.bmp). Similarly, the files containing our secret message will be named stego, again with the appropriate file type extension.

Data Hiding in Text

Steganography in text files can be accomplished through various techniques. Methods that can be applied to both the soft and hard copies of a document include line-shift coding, word-shift coding and feature coding as well as syntactic/semantic methods. The first three of these systems rely on visually changing the formatting or look of the file, by modifying spacing between lines, spacing between words, or modifying features of certain letters respectively (Sellars, p.7-8). An example of this would be using one or two spaces between words to indicate binary digits, and using these ones and zeros to code a message. Syntactic and semantic methods of steganography in text files utilize modification of "diction and structure of text without significantly altering meaning or tone" (Bender, et.al, p.334). This is accomplished by rephrasing sentences or using synonymous pairs of words, each of which constitutes a particular value. For example alternating between using the words "happy" and "content" in a document could indicate binary ones and zeros. All of these techniques provide robustness of duplication in that they can be decoded using the image of the document (i.e. bitmap or photocopy) or by accessing the document in its digital format directly. Their weakness comes from the fact that these types of file modifications may be visible to the eye or in the case of syntactic/semantic methods may change the nuances of meaning of the given text (Bender, et.al, p.334).

Our tool analysis for text file steganography will cover a product that makes use of white-space or open-space encoding. White space steganography is the manipulation of the spaces between words or in the case of our tool at the end of lines of text.

Concealing Messages in Text Files Using Open Space Methods: Snow

Description:

Snow is a free for non-commercial use program available at http://www.darkside.com.au/snow and is authored by Matthew Kwan. According to Bender, et.al, "soft-copy text is in many ways the most difficult place to hide data." This is due to the fact that extra characters or punctuation in a document can easily be noticed by the reader. The Snow program utilizes an open space method of embedding data in a text file – it adjusts the trailing spaces of each line of text to encode its message. The author of Snow describes it as "a program for concealing messages in text files by appending tabs and spaces on the end of lines, and for extracting messages from files containing hidden messages" (Kwan). Snow relies on the fact that most text viewers and editors do not by default show these characters when viewing, thus masking the fact that there is extra data in the file.

Snow is run from the windows command line and is available for unix/linux platforms, and there is a java version available as well.

Usage:

According to the manual page for snow, "the data is concealed in the text file by appending sequences of up to 7 spaces, interspersed with tabs", typically allowing 3 bits of stored data for every 8 columns in the cover file. The program has the handy ability to tell you how much data it can fit in the desired cover file: Issuing the command "snow –S cover.txt" produces the following output:

> File has storage capacity of between 1763 and 2012 bits.
> Approximately 235 bytes.

Our embedded data, mysecret.txt is only 107 bytes, so we know it will easily fit within this cover file. If space is of concern, the program also offers a flag to compress the data, however the author notes that "if the data is not text, or if there is a lot of data, the use of the built-in compression is not recommended", and suggests that the user pre-compress the data with more robust compression tools such as gzip or winzip.

Snow also provides the ability to encrypt the data to be hidden with a password protected key. It uses the authors own ICE encryption protocol, allowing for passwords or pass phrases (if provided in quotes on the command line) of up to 1170 characters.

To embed our secret data in our cover file we issue the command:

> snow –C –f mysecret.txt –p mypasswd cover.txt stego.txt

This command compresses the message contained in mysecret.txt, encrypts it with the password-protected key using “mypasswd” and embeds it in cover.txt, creating the stego file called stego.txt. The output from the command informs the user that it compressed the original message by 41.87% and that the message used approximately 25.14% of the available space in the cover file.

The extraction process is just as straightforward, issuing the command

snow –C –p mypasswd stego.txt

will output the contents of our secret message file to the screen.

Summary:

The stego.txt, when opened with a common text editor, like Microsoft Word, looks identical to the original, despite having gained 644 bytes in size with the addition of the secret message. Also, by telling Microsoft Word to show special formatting marks, one can easily see the inserted tabs and spaces in the stego document. However, thanks to the availability of the built in encryption scheme, an extra layer of protection is provided, meeting the requirement imposed by the Kerckhoff principle in cryptography which states “the security of the system has to be based on the assumption that the enemy has full knowledge of the design and implementation details of the steganographic system” (Sellars, p.6). Meaning, even if one had knowledge that there was encoded data in the file, they would need to crack the password and/or the encryption scheme in order to extract the message. Another benefit of this technique is that it can be used on any text. However, one must keep in mind that the embedded data can only be retrieved from the digital text file, not from a printed or scanned hard copy.

Data Hiding in Image and Audio Files

Typically, using image files as hosts for steganographic messages takes advantage of the limited capabilities of the human visual system (Sellars, p.9). Encoding extra data in an image file changes pixels in the image, but these changes would remain imperceptible to the human eye. Some of the more common method for embedding messages in image files can be categorized into two main groups, image domain methods and transform domain methods. The image domain methods modify their host files at the bit level, changing the file bit by bit to encode their message. One of these methods is the least-significant bit method, which will be explored in detail during the tool analysis below. The transform domain methods manipulate the algorithms and transformations inherent in the creation of the image itself, like the transformation used in JPEG compression. Advantages of using this category of methods include more robustness than the image domain methods, and ability to work on a larger range of image formats (Johnson & Jajodia, p.3).

Hiding data in audio files is a bit more challenging than hiding them in image files, as the human auditory system is more sensitive than its visual system (Sellars, p.14). In addition to the least-significant bit method which works on the same principal that is does for image files, and which will be explored in the following tool analysis section, there are three other primary methods for using audio files to host embedded messages. Phase coding works by substituting the phase of an audio segment with a reference phase that represents the data. This method "is one of the most effective coding methods in terms of the signal-to-noise ratio" allowing virtually inaudible coding (Bender, et.al p.325). Spread spectrum techniques are designed to encode data by spreading it across a wide range of the frequency spectrum, whereas "normal" audio communication typically condenses information in as narrow a band as possible. The primary disadvantage of this method is that it has a tendency to introduce noise to the original audio file. Echo data hiding inserts additional data into audio files by introducing an echo. Data can be hidden by varying the initial amplitude, the decay rate and the delay of the echo. This system provides minimal degradation of the original, however does not work well on files that have a lot of line noise or gaps of silence (Sellars p.16).

Concealing Messages in Image and Audio Files Using LSB Methods: Steghide

Description:

Steghide, authored by Stefan Hetzl and available at http://steghide.sourceforge.net/index.html, is a steganography tool available under the Gnu Public License that uses the least significant bit (LSB) method of hiding messages. It can be used to embed data in bitmap and JPEG image files, as well as wav and au audio files. The release notes also mention future plans to add the ability to use gif, mp3 and avi files. The least significant bit technique relies upon the fact that a digital image is made up of arrays of color and intensity values, and is considered one of the most well -known image steganography methods. With an audio files, this technique works by "replacing the least significant bit of each sampling point by a coded binary string" (Bender et.al, p. 324). Regardless of the file format, this method simply embeds the bits of the message into the least significant bit values of the image or audio files. Changing the LSB "does not result in a human-perceptible difference because the amplitude of the change is small" (Lin, Delp, p.3). However, one must choose the cover file with some care. Depending on the channel capacity of an audio file, for example, some audible noise could be introduced (Bender et.al, p.325), and potentially render the file suspect to a listener. Similarly, using certain image types can increase the chance of visual evidence of image tampering. It is best to use images with a large number of colors, as the chance for visible corruption would be less than if one used an image with a very small color palette. For this reason, grayscale images are often recommended, as subtle changes in gray pixels would go completely unnoticed to the human eye (Fridrich, et.al, p.22)

Steghide is a windows command line tool also available for unix/linux platforms. The version in use below is 0.4.6b.

Usage:

The Steghide homepage describes the program as “a steganography program which embeds a secret message in a cover file by replacing some of the least significant bits of the cover file with bits of the secret message” (Hetzl). It offers features like encryption of message data before embedding (using the blowfish encryption algorithm and MD5 hashing for key creation from the pass phrase), the pseudo-random distribution of the hidden bits, and the ability to embed a crc32 checksum of the plain data to verify integrity during extraction.

To embed the mysecret.txt file in our cover image, cover.jpg, producing stego file stego.jpg, issue the command:

```
steghide embed –pf mysecret.txt –cf cover.jpg –sf stego.jpg
```

The user is prompted to enter a pass phrase, and prompted again to confirm it. The default behavior of steghide is to use pseudo-random interval length to determine placement of hidden bits. This behavior provides some protection against pattern detection attacks. Encryption via blowfish/MD5 hashing is also default behavior, as well as addition of the hidden message file’s checksum for later integrity checking. Command line options are available to customize and/or disable these options if desired, however, the defaults provide the most secure options. Unlike the snow program, Steghide does not provide the ability to prescreen the cover file to see how much data could be hidden, nor does it provide any built-in compression. However, one could use standard file compression tools condense data to be hidden prior to embedding it.

To extract our secret, we use the command:

```
steghide extract –sf stego.jpg
```

After providing the proper pass phrase, it extracts mysecret.txt. We have the option also of using the –pf flag to specify and output filename other than the original mysecret.txt.

The commands used to embed data in bitmap, wav and au files are identical to those above, simply use cover files of the appropriate type.

Summary:

Steghide provides a very simple means of adding steganographic data to multiple file types. Both image types tested (JPEG and bitmap) showed no visual

clues that there were modifications made. Similarly, listening to a modified audio file in wav format provided no aural indications of tampering. The only file type to show a file size difference between original and modified version was the JPEG file. Its use of pseudo-random placement of modified bits combined with built in encryption of the data also help to reduce chances of attack.

Similar to the Snow program's open space method of embedding data, Steghide's LSB method has some inherent weaknesses. The primary weakness being that it is vulnerable to many possible file modification processes. "Scaling, rotation, cropping, addition of noise, or lossy compression to the stego image is very likely to destroy the message." (Lin, Delp, p.3).

Detecting Steganography

Steganalysis is "the practice of attacking steganographic methods by detection, destruction, extraction or modification of embedded data" (Lin, Delp, p.4). The definition of success in steganalysis depends upon your intent. For the security professional charged with protecting his employer's data, a successful result would be proving the existence of hidden data being sent, and not necessarily the ability to extract it. For the data thief, wishing to perhaps use a digital image that contains a protective watermark, success would be not only detecting the existence of the watermark, but would also require destroying it without damaging the integrity of the desired cover file.

As in cryptography, there are five formal categories of detection techniques. In a "stego-only" attack, the attacker only has the stego file, and no other information. A "known-message" attack gives the attacker the knowledge of the secret message that is hidden in the file. With the "known-cover" method, one has both the original innocent cover file as well as the resulting stego file. A "chosen stego" attack provides the attacker with the extraction tool to reach the data, and the "chosen message" technique assumes the attacker has the stego tool itself, and can embed and detect messages at will. (Lin, Delp, p.4)

A stego only attack, while considered the most difficult attack in that one has the least information to go on, is far from impossible, especially if ones goal is to merely detect that there is a hidden message and not necessarily have the need to extract it. For text files using empty space methods, like the one used with the Snow tool, merely opening the document with an editor that shows formatting codes would indicate that there were oddities in the formatting. The message could be easily destroyed through simply removing the extra spaces and tabs. Similarly, for other text methods such as line and word shift methods, visual inspection of the text itself could indicate anomalies. While this method is more immune to destruction if one only as a hard copy of the document, it is also easily destroyed with any text editor if one has the digital copy of the file.

For image and audio files with messages embedded with LSB methodology, detection with only the stego file available is a little more difficult. Detection in this case would rely on the appearance of visual or audible distortions or patterns. A known cover attack, where one has both the original and modified files would usually result in better success of stego detection of image and audio files. Anomalous patterns and excess noise in the stego file are much more easily detectable when comparing it to the original, particularly if the file format makes use of compression (as in JPEG files) that would show up as excess noise even in an innocent file (Johnson, Jajodia p.5). The Steghide tool reviewed attempts to minimize this type of attack by using pseudo-random placement of the encoded bits throughout the file. Destruction or distortion of LSB encoded messages in image files can be simply a matter of zeroing out the LSB fields of the file in question, image conversion, cropping or the application of other image formatting changes. For audio files, methods to damage the hidden message include the introduction of a random relative amplitude signal and reconstructing the file by ignoring bad signals (Vidas, p.5).

Both of the tools we looked at provided a bit of extra protection against the "known message" or "chosen stego" types of attack, in that they both make use of pass phrase protected keys to encrypt the data before embedding it. The robustness of each tools encryption algorithm and the chosen pass phrase would dictate the ease of message extraction. However, that only provides protection against someone extracting the full message, and serves no protection against destruction or detection of the message's existence.

Conclusions

Interest in the use of steganography in our current digital age can be attributed to both the desire of individuals to hide communication through a medium rife with potential listeners, or in the case of digital watermarking, the absolute necessity of maintaining control over one's ownership and the integrity of data as it passes through this medium. This increased interest is evidenced in the sheer number of available tools to provide easy steganographic techniques to the end user, as well as the proliferation of research and press on the topic.

The intent of this paper was to cover some of the more common methods of data hiding using widespread file formats and easily available tools as an introduction to the primary concepts of steganography. These discussions should serve as a starting point to the exploration of more complex steganographic techniques involving, for example, the use of network packets and unused hard disk space as cover medium, or the more complex methodologies used on our standard image and audio files.

References:

Bender, W., et al. “Techniques for Data Hiding”, IBM Systems Journal, Vol. 35, Nos. 3&4, 1996: 313-336

Fridrich, Jessica, et.al. “Detecting LSB Steganography in Color and Grayscale Images”, Magazine of IEEE Multimedia Special Issue on Security, Nov. 2001: 22-28
http://www.ws.binghamton.edu/fridrich/Research/acm_2001_03.pdf (March 2003)

Hetzl, Stefan, “The Steghide Homepage”, Version 0.4.6b
http://steghide.sourceforge.net (March 2003)

Johnson, Neil F. and Jajodia, Sushil. “Steganalysis of Images Created Using Current Steganography Software”, Lecture Notes in Computer Science, Vol. 1525, Springer-Verlag, 1998.
http://link.springer.de/link/service/series/0558/papers/1525/15250273.pdf (March 2003)

Kwan, Matthew, “The SNOW Homepage”
http://www.darkside.com.au/snow/index.html (March 2003)

Lin, Eugene T., and Delp, Edward J. “A Review of Data Hiding in Digital Images”. April 1999.
ftp://skynet.ecn.purdue.edu/pub/dist/delp/pics99-stego/paper.pdf (March 2003)

Sellars, Duncan. “An Introduction to Steganography”
http://www.cs.uct.ac.za/courses/CS400W/NIS/papers99/dsellars/stego.html (March, 2003)

Vidas, Alexandra. “Data Hiding: A Policy and Futures Overview of Digital Watermarks and Steganography”
http://www.unc.edu/~vidaa/inls281 (March 2003)