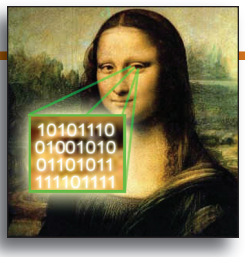

Detecting Steganography on a Large Scale

by William Ella

Introduction

If asked, most people today would not recognize the word steganography. Even within the academic and professional societies of computer science, many are unfamiliar with the term. Likewise, "steganography" is not even included within most spell-checking programs [1]. How could such an emerging field in computer science remain hidden for so long?

Steganography is the science of hiding information; this is often confused with the related science of **cryptography**, which is similar to a virtual "padlock" that openly challenges an eavesdropper to break into a message. Unlike cryptography, steganography challenges an eavesdropper to notice the presence of a message. In practical terms, this means that steganography can be anywhere; there is potential for a hidden message wherever information lies. The earliest example of steganography references the time of the Romans. A general shaved the head of a trusted slave, tattooed a message on top of his head, and then sent the slave across the country after his hair grew back. When the slave reached his destination unopposed, he shaved his head to reveal the hidden message [5]. Other examples of classic steganography involve the usage of invisible ink and microdots in World War II [2]. In this article, I will first establish the foundations of information hiding, then describe some elementary methods used in steganography and steganalysis, and finally explain the results of a research experiment I performed in order to understand how to undertake large-scale steganalysis.

The Methodology of Information Hiding

The two basic types of modern information hiding are steganography and **watermarking**. In general, the goal of watermarking is to put a secure stamp on a transmitted file; this is often seen in copyrighted multimedia such as sample pictures, online videos, and other types of intellectual property. The presence of the watermark usually does not need to be hidden, as long as it cannot be removed without destroying the original file. The practice of watermarking is similar to cryptography. Steganography, on the other hand, rests on the complete undetectability of a message. Unlike watermarking, the host file is often unimportant for steganographic usage, so it uses an innocent file as a cover for sending hidden data. However, if a message is even expected within this file, then the integrity of that usage of steganography has been broken, regardless of whether or not the message can be decoded [6].

There are three basic tenets behind hiding information. The first is **capacity**, which is the amount of information that can be embedded within the cover file. An information hiding algorithm has to be able to compactly store a message within a file; it does no good to have a message that cannot be found but is also severely hampered by its size. Next is **security**, which refers to how easily a third-party can detect hidden information within a file. Intuitively, if a message is to be hidden, an ideal algorithm would store information in a way that was very hard to notice. Finally, **robustness**, the amount of modification a message can withstand before being destroyed by a third-party. In regards to both steganography and watermarking, steganography is more concerned with capacity and security, and watermarking mainly focuses on robustness [5].

Different Media and Techniques

With modern techniques in steganography, hidden digital messages have the potential to be anywhere. Information can be embedded into many different types of digital files. Prominent examples are images, music, executables, and word-processing documents. Although the techniques may differ from file to file, the main principle of steganography stays the same. The idea is to hide information within the redundant data of a file. With this in mind, even html code, network traffic, and fax headers can be media for carrying secret messages [1].

The image is the most popular type of file for hiding information by far; its presence on the Internet is quite ubiquitous. Because of the image's prominence, most research is focused on hiding information within different file types such as BMP, GIF, and JPEG. Most information hiding for images include one of the following four techniques: modifying the least-significant bits of a file, masking the data in noise, scattering the data throughout the file, or embedding data with the compression algorithms. The least-significant bit modification finds all of the least important color data in an image and replaces it with the hidden message. In more sophisticated algorithms, bits are selected based on the characteristics of human vision, making it practically impossible to notice any visual imperfections in the image. Masking data in the noise distributes hidden data throughout the image file. This can be achieved in many ways; one interesting example is the "patchwork method," in which pairs or patches of pixels are randomly selected. The first half of the pair has its pixel values lowered by a slight constant, while the latter half is raised by the same amount. Scattering data throughout the file, also known as spread spectrum techniques, use transformation algorithms to evenly spread message data throughout an entire cover file. Finally, compression algorithms embed data by manipulating the techniques used to make an image file smaller in size [6]. All of these techniques can be generalized to other file types as well [7].

Detecting Hidden Information

Though information can be soundly hidden within innocent files, every technique leaves an inadvertent trail in the cover file. **Steganalysis** is

the science of finding the traces of those trails to detect the presence of hidden information. General and specific algorithms are the two main types of attacks used to detect hidden information. Specific algorithms target a certain type of steganography, such as precise formulations of least-significant bit modification, in the hope of directly detecting the presence of a hidden message. These techniques are very good for attacking files that have messages hidden with programs that are easily available to find, because their algorithms are generally well-known. General algorithms, however, are considered to be "blind" in the sense that they attempt to find an embedded message without knowing the process that was used to hide the data [7].

Because specific algorithms are limited by nature to previously existing steganography techniques, the area of general algorithms is more fruitful in steganalysis research. Furthermore, general algorithms can be broken down into two different types of attack. The first type is the empirical attack, which looks for noticeable changes within the cover file. This can be as simple as a person scanning an image with the naked eye or as involved as employing a computer to decompose an image into its bit planes to find abnormalities. However, as steganographic algorithms become more sophisticated, empirical attacks lose much of their utility because they are limited to what can be observed directly from the cover file. Therefore, a second and more powerful type of attack is statistical analysis. To overcome the problems of empirical attacks, statistical analysis examines a file to see if its makeup has any unusual disparities from a normal file of its type. Popular examples of this type of analysis include checking the palette ordering of an image, detecting image signatures that seem unusual for native files of that type, and finding a great amount of periodicity among coefficients that could indicate patterns of embedded data [6].

Steganography and Wikipedia

As with many other tools, steganography can be used for both positive and negative purposes. Just as easily as two pen pals can share a fun private message, criminals and terrorists also have the ability to secretly communicate using steganography. With the billions of images being passed back and forth on the Internet, many things could easily slip under the radar. For example, examine the two similar images in Figure 1. They could belong to a varied collection of sites, ranging from topics such as travel to photography or even weather services.

Figure 1: A text file is hidden within one of these images.

Is it obvious as to which one contains the hidden information? Using the free S-Tools steganographic suite, I took an image and imported a text file into it in less than a minute. With any steganography program that has a well-designed user interface, almost anyone can do this without any formal training. The fact that any of the billions of images on the Internet could potentially hide a dangerous message is a large problem. Even with the original image right next to it, there is almost no visual way to tell a dangerous message from an innocent one. Although there is a large amount of steganography research on finding new algorithms for either hiding or recovering data, it seems that there is relatively little research focusing on how to apply steganalysis to real situations. With the large amount of data flowing across the Internet, governments and private companies need to find a feasible way to sift through it all in order to find dangerous messages and agendas. In an attempt to understand how steganalysis on such a massive scale could be undertaken, I designed an elementary search on Wikipedia, a popular user-driven encyclopedia, in an attempt to find hidden information within its images. Because anyone can edit the pages on Wikipedia, it is not only the perfect model for a large database with potential hidden information lurking in its corners, but is also an example of a seemingly innocent site that should be a legitimate concern for various organizations.

Experimental Setup and Raw Data

In order to model a large-scale database scan, the experiment has two distinct steps:

1. Use the program Wikix to download a set of images from Wikipedia.
2. Use the program StegAlyzerSS in order to scan the images for hidden data.

The testing computer was my laptop, which dual-boots Windows XP and Ubuntu 7.10 with a 1.86GHz Pentium M chip and 2GB of RAM. I ran five different image-scanning trials, ranging from 196–13,147 files in each experiment (with 13,147 files being approximately 5.23GB of data). Because the Wikipedia image database was quoted to be about 406GB of data as of October 2007 [4], Wikipedia's database is estimated to have over one-million images. Figure 2 shows the number of files that were scanned in each trial.

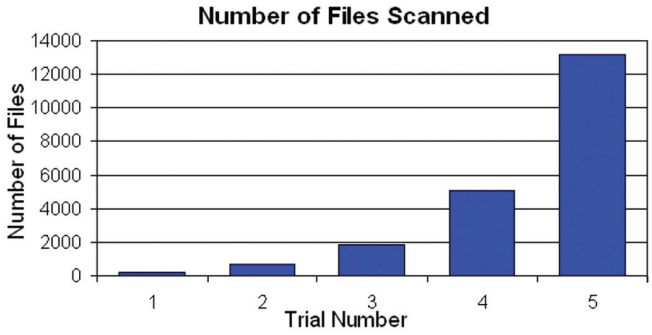

Figure 2: The number of files scanned in each trial.

The file-scanning algorithm ran linearly. On average, it scanned about 9.4 files/second. For about 13,000 files, it took roughly 27 minutes. When extrapolated to the entire image database, it would take about 31 hours to scan. The scanning time for each trial is illustrated in Figure 3.

StegAlyzerSS looks for three different trails left by steganography programs: the signature of a particular algorithm, appended information after the end-of-file character, and disturbances to the least-significant bits of the image file. In every trial, StegAlyzerSS did not find any signatures, and, excluding the 5th trial, no traces of least-

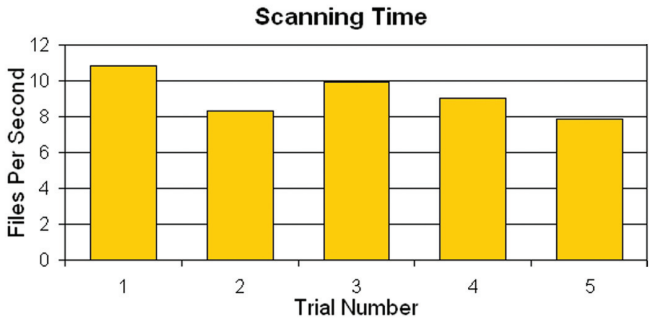

Figure 3: The average scanning time of each trial.

significant bit modification were found (only 3 were found in Trial 5). However, because image manipulation programs and cameras leave information after the end-of-file character, 260 images were found to have appended information in Trial 5. Extrapolated to the entire dataset, there could be over 20,000 images to scan through. Figure 4 indicates the number of files with appended information in each trial.

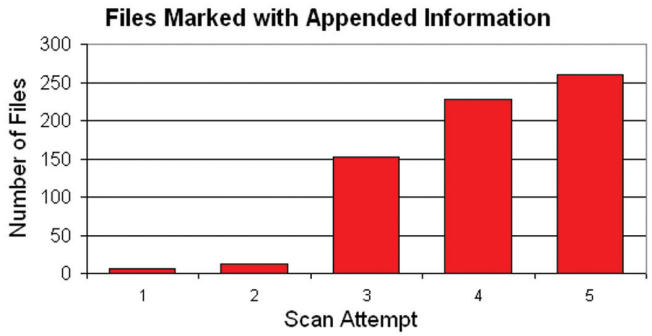

Figure 4: The number of files with information after the end-of-file character.

Data Analysis

Because there were no confirmed instances of steganography found in the scans, this experiment emphasizes how hard blind steganalysis can be. On one hand, there just might be not be any images with hidden information, but on the other hand, it could be that I missed the crucial image. However, the scans greatly helped to elucidate the original intent to model a large scanning process. As the data shows, scanning of the Wikipedia database would take less than 1.5 days on the test computer. On a real research machine, this time would be significantly reduced. Also, with a high-speed Internet connection, the entire image database could be downloaded within a day. Thus, with the proper resources devoted to the project, the Wikipedia database could easily be continually monitored for evidence of dangerous messages. However, this comes with the caveat that, like virus scanning software, the strength of this process is entirely dependent on the strength of the detection algorithm behind the scanning program. With my project, the number with least-significant bit modification was minimal, but that proportion could change when additional files are scanned. The number of files with appended information grew at a rate that was far too large for a person scanning the information. Therefore, an algorithm would need to be developed in order to automate that process. However, once individual problems are addressed, scanning any large database, whether it be Wikipedia or a company's internal storage, can be constantly monitored with ease.

Project Conclusions

Though I have shown that large scans of databases can be accomplished, this is not the optimum way to ensure images are without steganographic data. The scanning of large databases is analogous to scanning every box at a port; although everything is checked, it takes a lot of effort to find potential dangers. For instance, Provos and Honeyman searched two-million images on Usenet without being able to find anything conclusive [5]. Thus, it is desirable to have something that is more similar to the airport system, where files would be scanned before they reach their final destination. This process would make scanning much more manageable by finding messages before they are acted upon. Lau proposed a system for governments to scan outgoing Internet traffic for images, and, when one is found, it would be scanned for steganography[3]. If conducted in an ethical manner, this system could be used in duly-authorized situations. Governments and large organizations could employ both of these techniques at once to fully safeguard against dangerous usage of steganography.

Future Directions

This initial experiment has greatly helped me understand how to perform real-world steganalysis, but it has also raised further questions. To understand the properties of a normal image set, I plan to perform additional data analysis on the images scanned, looking for statistics such as general noise found in different types of image data and how that affects the scanning process. Studying such an "organic" database of images in this work leads me to question how it would work on a real database, such as a dynamic database, like a popular image forum. It would be interesting to use a few more scanning programs to find any discrepancies in scanning behavior. In addition, I plan to implement an airport-style system to see how effective it is when compared to a database scan. Finally, I would like to generalize the scanning process beyond Wikipedia, so that almost any site can be thoroughly searched for usage of steganography.

When compared to other related fields, such as cryptography, steganography has just begun to develop. New algorithms and attacks are emerging every day as steganographers and steganalysts create more sophisticated hiding techniques. Potential uses, such as data protection through watermarks and private communication are becoming prominent as steganography becomes better recognized among the mainstream computer science community. The irony of a mostly unknown field concerned with hiding information will not be overlooked for much longer.

Project Acknowledgments

I am deeply indebted to many different individuals for help with my project. First, I need to thank Jim Wingate, the director of the Steganography Analysis & Research Center at Backbone Security, for allowing me to use a free version of their scanning program, StegAlyzerSS. I would also like to thank Jeffrey Vernon Merkey for creating the excellent, free, open-source Wikix program, which enabled me to download the images from Wikipedia and Andrew Brown for the open-source S-Tools steganography program. In addition, I need to thank Yousef Ourabi, because he helped troubleshoot the image download process. Finally, I would like to thank Dr. Ernest Ackermann for supporting me through this endeavor.

References

1. Cole, E. 2003. *Hiding in Plain Sight: Steganography and the Art of Covert Communication*. Wiley, New York, NY.
2. Kessler, G. 2002. Steganography: Hiding data within data. **http://www.garykessler.net/library/steganography.html**.
3. Lau, S. 2003. An analysis of terrorist groups potential use of electronic steganography. SANS Security Essentials.
4. Ourabi, Y. 2007. Download all wikipedia images with WikiX. **http://yousefourabi.com/semantic-web/download-all-wikipedia-images-with-wikix**.
5. Provos, N. and Honeyman, P. 2003. Hide and seek: An introduction to steganography. *IEEE Security & Privacy Magazine 1*, 3. 32-44.
6. Wang, H. and Wang, S. 2004. Cyber warfare: Steganography vs. steganalysis. *Comm. ACM 47*, 10. 76-82.
7. Wayner, P. 2002. *Disappearing Cryptography*. Morgan Kaufman, San Francisco, CA.

Biography

*William Ella (**billy.ella@gmail.com**) is an undergraduate student pursuing two majors, computer science and mathematics, at the University of Mary Washington in Fredericksburg, Virginia.*