

**Copyright © 2003, The Society of Photo-Optical Instrumentation Engineers.**

**This paper was presented during the Electronic Imaging Symposium, on January 24, 2003, in Santa Clara, California. Please use the following format to cite this paper:**

**Jeffrey A. Bloom and Rafael Alonso, “SmartSearch Steganography”, in *Security and Watermarking of Multimedia Contents V*, Edward J. Delp III, Ping Wah Wong, Editors, Proceedings of SPIE Vol. 5020, (2003).**

# SmartSearch Steganalysis

Jeffrey A Bloom and Rafael Alonso  
Sarnoff Corporation, Princeton, NJ

## ABSTRACT

There are two primary challenges to monitoring the Web for steganographic media: finding suspect media and examining those found. The challenge that has received a great deal of attention is the second of these, the steganalysis problem. The other challenge, and one that has received much less attention, is the search problem. How does the steganalyzer get the suspect media in the first place? This paper describes an innovative method and architecture to address this search problem.

The typical approaches to searching the web for covert communications are often based on the concept of “crawling” the Web via a smart “spider.” Such spiders find new pages by following ever-expanding chains of links from one page to many next pages. Rather than seek pages by chasing links from other pages, we find candidate pages by identifying requests to access pages. To do this we monitor traffic on Internet backbones, identify and log HTTP requests, and use this information to guide our process.

Our approach has the advantages that we examine pages to which no links exist, we examine pages as soon as they are requested, and we concentrate resources only on active pages, rather than examining pages that are never viewed.

**Keywords:** steganography, data hiding, steganalysis, sniffer, web search

## 1. INTRODUCTION

Two technical developments have combined to create a new threat to security. First, it has become easy to share information via the World Wide Web. Any number of parties may communicate with one another from any number of locations anywhere in the world using commonly available computers and networks. Second, at the present time there are several publicly available steganographic tools for hiding data in images (See for example [1, 2]). These conditions enable anyone to exchange data covertly by embedding the information in seemingly innocuous Web images or video.

This combination offers criminals and terrorists an almost ideal opportunity to use the Web as a covert communications channel. Terrorists who do not want to be tracked can use a series of public Web access facilities (e.g., libraries or Web cafés) to anonymously download steganographically-marked images and communicate without attracting attention. Furthermore, the sender of the message need only make the marked image available for a short period of time on a page to which no other page has a link; thus, no current search engine will ever see the marked image. Perhaps most dangerous of all, this covert channel is ideally suited for insider threats.

To counter this new security threat we propose to develop an innovative method to monitor the web for video and image files that have been steganographically modified by publicly available software. Sarnoff's SmartSearch Steganalysis system will be able to discover hidden messages that may be placed on the web for only a short time – as little as twenty four hours - and withdrawn after use.

There are two primary challenges to monitoring the Web for steganographic media: finding suspect media and examining those found. The challenge that has received a great deal of attention is the second of these, the steganalysis problem. Given an audio file, a digital image, or a video clip, steganalysis tools seek to determine the

likelihood that the work has been modified to communicate a hidden message. A great deal of progress has been made in this area and significant research efforts continue. The other great challenge, and one that has received much less attention, is the search problem. How does the steganalyzer get the imagery or audio in the first place? This paper describes an innovative method and architecture to address this search problem. The system described here has not yet been built and therefore, no experimental results will be presented.

The typical approaches to searching the web for covert communications are often based on the concept of “crawling” the Web via a smart “spider” (see for example [3]). Such spider web crawlers find new pages by following ever-expanding chains of links from one page to many next pages. We believe the standard web search approaches have a number of disadvantages for steganalysis applications:

- These approaches are brute force in that the number of links is enormous and all pages discovered by tracking links must be examined for images or video containing embedded data. Available computing resources are diluted chasing links instead of devoting efforts to gathering pages that may contain embedded data.
- Link-based spider searchers miss pages to which there are no links.
- Pages that suddenly appear and quickly disappear are mostly always missed by these approaches. This is precisely the type of pages one would expect a terrorist organization to use.
- No spider, which we know of, can crawl the entire web, only a small portion of it.

Rather than seek pages by chasing links from other pages, we find candidate pages by identifying requests to access pages. To do this we monitor traffic on Internet backbones, identify and log HTTP requests, and use this information to guide our process. This approach has the following advantages:

- We examine pages to which no links exist.
- We examine pages as soon as they are requested, thus we examine “new” content quickly.
- We concentrate resources only on active pages, rather than examining pages that are never viewed.

As an optimization, we can compile an exhaustive list of all Internet hosts, and allow users to specify the trusted and untrusted sites. The system default is that no sites are deemed trustworthy, but if users are confident in exempting certain sites or whole domains, processing requirements will be decreased.

This paper describes the SmartSearch Steganalysis architecture, provides estimates on the computational and bandwidth requirements of the various components, and proposes technical approaches that allow the system to be feasibly implemented.

## 2. SMARTSEARCH STEGANALYSIS OVERVIEW

Our technique is designed to periodically examining potential host images on the Web as often as once every 24 hours for the presence of hidden messages. This system will detect steganographic markings that have been created through the use of the most popular data hiding tools. Hence, the two main challenges that we are tackling are (1) sweeping the Web on a daily basis and (2) testing images for evidence of the use of several different types of steganographic techniques.

Sweeping the entire Web is a Herculean task, one not achieved by even the fastest search engines after several days of work. Our approach consists of monitoring *requests* for pages rather than crawling the Web following *links* to pages. Further, we detect the *first* data request (in every 24-hour period) to web sites, and select only those pages for further analysis. Since the number of unique page requests is several orders of magnitude smaller than the total number of Web pages, our approach applies the system resources to examine data that is actually being used rather than wasting resources on data that is never seen. As an optional optimization, we also avoid extra work by ignoring page requests to trusted sites, such as the .gov domain, the .mil domain, and the top few levels of domains associated with major companies (e.g., www.sarnoff.com).

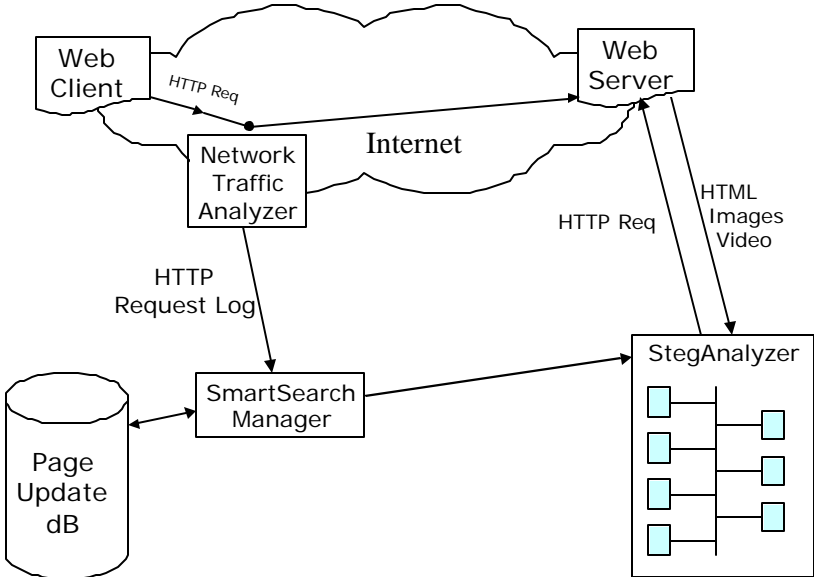
Image requests are duplicated so that we obtain a copy of each image. These images are input to a fast and efficient parallel steganography detector based on the most current techniques in the literature. This tool will detect the presence of messages embedded by most of the data hiding image software publicly available on the Web.

At the core of the system is a monitoring system that identifies requests for images and thus analyzes the entire *active* Web every 24 hours. The *active* Web is that portion which is viewed. While a full-blown implementation of this system might run on the backbone of the Web, the initial test bed will run on our own corporate intranet to verify the technical performance of SmartSearch Steganalysis. We can then estimate the viability of our approach for the entire World Wide Web by extrapolating our results (for example, by scaling the system’s performance on Sarnoff’s Intranet for size, data capacity, frequency of use, etc.).

**3. DETAILED ARCHITECTURE OF SMARTSEARCH STEGANALYSIS**

Our SmartSearch Steganalysis approach is comprised of three primary components: the *Network Traffic Analyzer*, the *SmartSearch Manager*, and the *Steganalyzer*. The architecture is sketched in Figure 1. With reference to this sketch, we can describe the roles of each of the three components by considering the following concept of operations scenario. A terrorist uses a publicly available data-hiding tool to embed information in an image, which she then makes available on the Web. In Figure 1, the image is placed on the “Web Server.” This web server may be a multi-user server (e.g., an eBay server), it may be a private server running in someone’s personal computer, or it may be a publicly accessible server at a government agency. Furthermore, the page that contains the marked image may be part of a web of interconnected pages or it may be a single page with no outside links pointing to it.

A colleague of the terrorist, presumably a terrorist himself, represented in the figure as “Web Client,” will download the web page containing the message-laden image and decode the message. This client may be physically located in a publicly accessible area such as a public library or an Internet café. The communication will be successful as long as the two parties agree on the URL of the communication channel, the steganographic tool (likely a small, easily downloaded tool), and the secret key.



**Figure 1 - System Architecture**

There are multiple Network Traffic Analyzers sitting on the backbone of the Internet. As Internet traffic passes through one of these analyzers, it is parsed and HTTP requests are identified and logged. These logs are then sent to the SmartSearch Manager, which compares each request to a Page Update Database. The manager identifies those requests that correspond to pages that have not been examined in the past 24 hours. The URL’s associated with these potential host pages are sent to the Steganalyzer which acquires any imagery embedded in the suspect pages and assesses the likelihood of steganographic contents.

### 3.1 Network Traffic Analyzer

The front line tool for SmartSearch will consist of a number of network traffic analyzers. The number of network traffic analyzer needed depends on the size and the topology of the network being monitored. The number required may be estimated by using an Internet topology model (e.g., see [4]); such an exploration is beyond the scope of this paper. Each analyzer will reside on or near an Internet router and will examine all network traffic traveling through that router. The job of the analyzer will be to identify HTTP requests and forward a log of these requests to the SmartSearch Manager. We expect that the vast majority of HTTP page requests will fit within a single network packet; reconstruction of requests that span multiple packets, although possible, will significantly increase the complexity of the network traffic analyzer.

There are several challenges in the design and implementation of the network traffic analyzers. An obvious one is keeping up with the flow of HTTP requests. To get a sense of the computational difficulty of the task, we can examine “packet sniffer” software tools. We should point out that a basic packet sniffer can be implemented rather simply. For example, Stein demonstrated a simple packet sniffer at the WWW6 Conference in Santa Clara (April 1997) that consists of 39 lines of Perl code [5]. For more details on packet filtering techniques see the recent book by Chappell [6].

There are also several engineering issues to be resolved in implementing an efficient traffic analyzer. We will mention only two. For example, if the analyzers monitor each of the input lines of the routers on the backbone of the Internet, the analyzers will observe requests originating from a given portion of the Web demanding data from the rest of the Web. One will obtain a more efficient implementation if the network analyzers are placed so that they monitor the output lines of the backbone routers. If so, the analyzers will sniff all data requests going to a portion of the Web. If this is the case, the Page Update Database shown in our architecture diagram may be easily partitioned and each network analyzer need only send its log to the single SmartSearch Manager responsible for the relevant portion of the Web. A second important engineering issue is how to best protect the privacy of the individuals whose traffic is being monitored. Care must be taken to limit external access to the internal data structures of the network analyzer to minimize the possibility of attack.

At this point one should note that some of the functionality of the network analyzers is similar to that of other Internet-oriented software. Examples include commercially available security software such as firewalls, as well as several (usually) shareware tools such as packet sniffers. It is likely that we will be able to adapt such software to our needs instead of authoring a totally new system. There is an additional reason to consider basing the network analyzers on other existing tools. It may be the case that, in practice, the expense of widely fielding network analyzers may only be warranted if the analyzers carry out additional tasks. For example, the analyzers may be required to engage in additional security-oriented tasks (e.g., detection of and protection from denial of service attacks). If so, we can both make use of an existing tool and increase the benefits of our system.

### 3.2 SmartSearch Manager

As shown in the system architecture diagram, the SmartSearch Manager will receive the HTTP request logs from the network analyzers. The request logs can come to the SmartSearch manager in two ways: either in a periodic “batch” mode or in a continuous “stream” fashion. The batch mode is preferred if the communication link between the network analyzers and the SmartSearch manager is limited in bandwidth. In the batch mode the network analyzers will also delete duplicate HTTP requests, trading off increased workload at the analyzers for decreased communication requirements. Alternatively, the stream mode is preferred if the overall system is being designed to have the fastest response time possible (i.e., to be able to detect the presence of steganographically marked pages as soon as possible) or if it is desirable to limit the computational load at the network analyzers.

In both batch and stream modes the SmartSearch manager needs to delete duplicate HTTP requests (even in the batch mode, since the duplicates may come from different network analyzers). At this point the SmartSearch manager may optimize system performance further by distinguishing between “trusted” web pages (perhaps entire domains) and untrusted pages. As pointed out earlier, the system will function correctly, albeit perhaps at the cost of degraded performance, if no page is considered above suspicion.

The most important optimization step takes place at this point. The SmartSearch Manager discriminates between those untrusted pages that have been visited recently (for example, within the last 24 hours) and those that have not

been visited recently. The time interval that defines “recently” is application dependent; we refer to this interval as the system’s “sweep cycle.”

To distinguish newly requested URLs, the SmartSearch Manager employs the services of the Page Update Database. This database contains information about each web page that has been requested. When a page is first requested, an entry is made in the database. A timestamp indicating when this page was last submitted for analysis is checked when the page is again requested and, if beyond the sweep cycle, the time stamp is updated and the page sent for analysis. Periodically, the SmartSearch manager will update the database to remove obsolete data associated with request older than the sweep cycle.

As mentioned above, it may be desirable to partition the Page Update Database. If this is done, the performance requirements of the SmartSearch Manager are eased somewhat. However, the partitioning imposes a new requirement on the SmartSearch manager: removing unnecessary entries duplicated in multiple databases (this task is not onerous, and may be carried out by the Page Update Database itself if that component is implemented via a commercially available distributed database).

There are several implementation alternatives to be considered for the Page Update Database. The simplest alternative is to use a relational DBMS; such a system will have more functionality than needed for this application, but will require less customization. A standard relational database may be too slow for some applications, hence a main memory DBMS (See [7]) may be a better choice for faster performance. Finally, developing a main memory data structure combined with persistence procedures results in probably the fastest performance, but requires the most implementation effort.

### **3.3 Steganalyzer**

The Steganalyzer will retrieve content from the Web as directed by the SmartSearch Manager. Initial implementation will analyze only still imagery, however the architecture allows for easy expansion to analyze audio video files as appropriate steganalysis algorithms are implemented.

Once retrieved, the suspect image is then analyzed by a number of steganalysis algorithms as might be found in the literature. The primary algorithms are discussed below. A key requirement in developing the Steganalyzer is fast performance. In order to provide scalability in the face of a heavy processing load, the Steganalyzer will be implemented on a cluster computer and will employ scalable distributed data structure technologies on a parallel-processing platform. The steganalysis work we propose could be done in an obviously parallel fashion (for example, we could simply run  $N$  different steganalysis algorithms on  $N$  different computers). However, based on our previous experience developing parallel software for clusters, we can achieve greater throughput by implementing a parallel tool.

Steganalysts often concentrate on steganography techniques that claim to be secure against unauthorized detection. These represent, after all, the interesting challenge. Thus, steganalysts often take Kerckhoff’s assumption that the security of the communication method depends only on a secret key and that knowledge of the hiding algorithm without the key does not compromise the security. We cannot however assume that all steganographers will follow this reasoning and so we will implement detectors for easily detectable data hiding methods as well as “secure” steganographic techniques (at least those intended to be secure).

With regard to such “secure” techniques, we begin with the RS steganalysis of Fridrich, Goljan, and Du [9]; PoV analysis of Westfeld and Pfitzmann [10]; universal blind detection of Farid [11]; and a quality-based method of Avciba, Memon, and Sankur [12].

RS steganalysis [9] is designed to detect LSB-based steganography. The relative smoothness in a particular bitplane is measured over groups of pixels and changes to this smoothness are observed as bits are flipped (negated). The technique has proven to be quite accurate at identifying not only the fact that a steganographic algorithm has been used, but in estimating the number of bits embedded. This approach can be applied to transform coefficients or palette indices as well as to pixel values.

PoV analysis [10] is also intended to detect LSB-based steganography. Pairs of values (PoV) differ only in the LSB. Thus, the process of embedding a secret message into the LSB transforms pixel values into the other member of their pair. Assuming that the bits embedded are equally likely to be zero as they are to be one, such embedding

changes the relative frequencies of the elements of the pairs. The resulting distributions are unlike those of natural imagery. This approach is also applicable to domains other than the pixel domain.

Farid's universal blind detection technique [11] is based on high-order statistical models. A statistical feature vector is extracted from the image and input to a linear classifier. The classifier must first be trained. This technique has been shown to work well for LSB-based steganography and may provide good results for other techniques.

The quality-based method [12] extracts quality features designed to predict compression, blur, and noise artifacts characteristic of many steganographic and watermarking methods. A large set of training data is mapped to this feature space and a regression analysis is used to define the decision boundary.

The framework is flexible enough to incorporate new steganalysis algorithms as they become available. For example, we are currently working on techniques for video steganography as well as quality-based detectors for imagery based on Sarnoff's JNDMetrix image quality assessment tools.

#### 4. SUMMARY

While promising approaches to steganalysis have been developed recently, tools for obtaining suspect images for analysis are lacking. Typical approaches based on following links, similar to those used by search engines to index the Web, are likely insufficient today and certainly so tomorrow. We have presented a novel approach that implicitly partitions the Web into an active part and an inactive part. Inactive web pages are those that are available, perhaps with links to them, but are not viewed. A typical personal home page, for example, would usually be labeled inactive. When a page is requested, it's label changes to active. Active pages become inactive after a 24-hour period during which they are not requested. Our approach analyzes only the active part of the web, which we estimate to be orders of magnitude smaller than the entire. The primary advantages are that resources are only expended on analyzing web pages being viewed making the process computationally feasible, pages to which there are no links can be analyzed (if they are being viewed), and pages that are short lived will be analyzed as soon as they are requested.

#### REFERENCES

1. Church of the Swimming Elephant, <http://www.cotse.com/tools/stega.htm>
2. StegoArchive.Com, <http://www.members.tripod.com/steganography/stego/software.html>
3. N. Provos and P. Honeyman, "Detecting Steganographic Content on the Internet", Proceedings of ISOC Network and Distributed System Security Symposium (NDSS'02), 2002.
4. S. Bhattacharjee, K. L. Calvert, E. W. Zegura, "How to Model an Internetwork", INFOCOM '96, Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation, Vol. 2, pp. 594 -602, 1996.
5. Lincoln Stein, "Security", a tutorial presented at the Sixth International World Wide Web Conference, Santa Clara CA, April 1997 (available at <http://stein.cshl.org/~lstein/talks/WWW6/sniffer>).
6. L. Chappell, *Packet Filtering: Catching the Cool Packets*, Podbooks.com, San Jose, 2002.
7. H. Garcia-Molina & K. Salem, "Main Memory Database Systems: An Overview", IEEE Transactions on Knowledge and Data Engineering, December 1992.
8. J. Fridrich and M. Goljan, "Practical Steganalysis of Digital Images – State of the Art", Security and Watermarking of Multimedia Contents, vol. SPIE-4675, pp. 1-13, 2002.
9. J. Fridrich, M. Goljan, and R. Du, "Reliable Detection of LSB Steganography in Color and Grayscale Images", Proc. of the ACM Workshop on Multimedia and Security, pp. 27-30, 2001.
10. A. Westfeld and A. Pfitzmann, "Attacks on Steganographic Systems", Third International Workshop on Information Hiding, IH'99, Springer-Verlag, LNCS 1768, 2000.
11. H. Farid, "Detecting Hidden Messages Using Higher-Order Statistical Models", IEEE International Conference on Images Processing, 2002.
12. I. Avcibas, N. Memon, B. Sankur, "Steganalysis Using Image Quality Metrics", Accepted for Publication, IEEE Transactions on Image Processing, May 2002.