A Framework for Avoiding Steganography Usage Over HTTP

Jorge Blasco[a], Julio Cesar Hernandez-Castro[b], José María de Fuentes[a], Benjamín Ramos[a]

[a] *Computer Science Department, Carlos III University of Madrid, Av. de la Universidad 30, 28911 Leganés*
[b] *School of Computing, University of Portsmouth, Buckingham Building, Lion Terrace, Portsmouth PO1 3HE, UK*

Abstract

Steganographic techniques allow users to covertly transmit information, hiding the existence of the communication itself. These can be used in several scenarios ranging from evading censorship to discreetly extracting sensitive information from an organization. In this paper, we consider the problem of using steganography through a widely used network protocol (i.e. HTTP). We analyze the steganographic possibilities of HTTP, and propose an active warden model to eliminate any covert communication channel. Our framework is meant to be useful in many scenarios. It could be employed to ensure that malicious insiders are not able to use steganography to leak information outside an organization. Furthermore, our framework could be used by web servers administrators to ensure that their machines are not being abused, for example, as anonymous steganographic mailboxes. Our experiments show that steganographic contents can generally be successfully eliminated, but that dealing with high payload carriers such as large images may introduce notable delays in the communication process.

Keywords: steganography, covert channels, http, active warden, sanitization

Email addresses: jbalis@inf.uc3m.es (Jorge Blasco), Julio.Hernandez-Castro@port.ac.uk (Julio Cesar Hernandez-Castro), jfuentes@inf.uc3m.es (José María de Fuentes), benja1@inf.uc3m.es (Benjamín Ramos)

1. Introduction

Steganography is the science that studies the techniques to hide the existence of messages (Johnson and Jajodia, 1998). The ability of sending secret messages can be useful for several purposes. On one hand, in a country under a totalitarian government steganography could be used to circumvent censorship (Feamster et al., 2002), and in a more general setting it could be instrumental for whistleblowers. On the other hand, steganography can also be used to commit malicious or criminal activities. In fact, it could be used by an employee stealing sensitive information from an organization in a case of industrial espionage. Before transferring this valuable information, the employee may hide it into innocuous looking documents. In this way, any security check or network monitoring tool would not detect sensitive information leaving the organization. Steganography can also help exchange illegal content (such as child pornography) using public resources like web servers or P2P networks as repositories, without the knowledge of the owners of those resources.

Recently, the usage of steganography reached public media coverage when a group of spies from Russia were uncovered (McGreal, 2010). As the FBI report describes (Kac, 2010), the spies, who infiltrated some United States Government agencies, used a steganographic program to conceal their intelligence reports into digital images. Those were later uploaded to public web servers, so the Russian intelligence at Moscow could download them and extract the secret messages (intelligence reports) after the use of a preshared key. Another malicious usage of steganography that has risen recently is to command & control botnets (Kartaltepe et al., 2010). By employing steganography, botnet owners can benefit from social network sites and transform them into infrastructures to covertly deliver their commands. In this way, botnet administrators have a centric, fast, reliable and easy method to distribute their commands to multiple bots.

Avoiding such a kind of malicious steganography usage is an important issue for organizations which hold large amounts of sensible information, or by system administrators that do not want their services to be used for unauthorized purposes.

The contribution of this paper is twofold: First we present a framework which limits the transmission of hidden information through Hyper Text Transfer Protocol (HTTP). Our system hinders the usage of steganography on HTTP message body entities such as images, text, etc. Additionally, it

avoids the usage of the HTTP protocol structure itself for steganographic purposes (i.e. modifying HTTP headers to hide information as proposed by Dyatlov and Castro (2003)). Although steganography can be used to create a covert channel through any kind of network protocol, we have focused on the restriction of HTTP for several reasons:

- The restriction of HTTP traffic through firewalls and rule based systems is infeasible as it is essential for Internet communications. HTTP provides access to multiple kinds of services such as news, search engines, web mail, social networks, multimedia, etc. but also provides enterprise services such as Business to Business services, reference, documentation, etc. which are essential for organizations and cannot be simply blocked.

- HTTP allows users to access plenty of information and consumption services (YouTube, Flickr, etc.). These kind of services can be easily used by steganography users as cover repositories and anonymous mailboxes to upload, store and download hidden information (Burnett et al., 2010). In this regard, URL filtering software could be used to restrict the amount of sites a potential steganography user is able to connect. Due to the great amount of these, it is however unlikely that the security administrator will be able to block them all.

- The usage of HTTP provides a higher anonymity level, in comparison with other common organization wide network protocols such as SMTP. Although HTTP communications are not anonymous, only the web server administrator or network nodes between the user and the web server may posses enough information to identify who accessed the server resources (the possible recipients of the hidden messages). Other protocols such as SMTP explicitly specify the recipient of the information when communicating to an intermediary server, so they are easier to trace.

- Finally, the usage of third party web servers as part of the steganographic communication involves a violation of the terms of these web servers. This kind of abuse may induce unnecessary overloads that should be actively avoided by system administrators.

Our proposal enables the normal usage of HTTP connections, while prevents the existence of covert channels through these. This would allow organi-

zations to avoid information theft through HTTP. Additionally, our scheme may allow service providers to ensure clients perform authorized usage of their services (i.e. they are not used to covertly store unauthorized material). Although our proposal is focused on HTTP it may be easily adapted to other protocols such as SMTP, FTP, etc.

The rest of the paper is structured as follows. We describe the basics of Steganography in Section 2. The related work is summarized in Section 3. In Section 4, we analyze the steganographic capabilities of HTTP. The design of Stego-Proxy is explained in Section 5. Section 6 depicts our evaluation and summarizes our results. Finally, Section 7 gathers the conclusions and future work lines.

2. Steganography

The first model of steganography was described by Simmons (1998) as the prisoners' problem. Simmons described two prisoners (Alice and Bob) who want to plot an escape plan. As they are not in the same cell, they must communicate through a warden (Wendy), that will analyze any communication between them. If Wendy ever suspects that Alice and Bob are planning to escape he will put them into isolation cells and the escape will be frustrated. In this scenario, Alice and Bob will not be able to just use cryptography, as encrypted messages will raise suspicions on Wendy. In order to achieve their goal, Alice and Bob should hide their secret messages into innocuous looking ones (called covers), so Wendy will only see unremarkable messages exchanged between prisoners.

However, if Wendy is aware of the existence of some kind of steganography she may be able to detect the presence of hidden messages or even further destroy the covert channel between Alice and Bob. On one hand, if Wendy just analyzes the messages and forwards them to its recipient, then Wendy is a *passive warden*. In this case Wendy verifies if the cover contains hidden contents or not. On the other hand, if Wendy has high suspicions of Alice and Bob planning an escape through their messages, but she is not able to obtain proof, she may modify slightly the exchanged messages trying to perturb any hidden information. In this case, Wendy is an *active warden*. Even further, Wendy may be able to insert some information impersonating Alice or Bob, thus performing a man in the middle attack. In this case Wendy is a *malicious warden*. Although steganographic algorithms should be robust to active warden attacks, steganographic researchers have mainly

focused on the imperceptibility of hidden information while resistance against active attacks has been mostly addressed in other information hiding areas like watermarking (Cox et al., 2008).

Thanks to the widespread adoption of digital devices and electronic documents, steganography has attracted the interest of researchers in the last years. Fisk et al. (2003) defined the concepts of structured and unstructured carriers. A carrier specifies the features or characteristics used to hide the information in the cover. In this regard, a structured carrier is defined as a carrier which structure is well defined (XML files, PDF files, network protocols, etc.), while unstructured carriers do not have a defined structure (images, video, natural language, etc.). HTTP is a specially interesting and relevant case because it encompasses both carrier types, as information can be hidden into the structure of the HTTP message (see Section 4) or into the content uploaded or downloaded by the user (images, video, documents, etc.).

2.1. Steganography in Unstructured Carriers

The increasing concerns about copyright violations of multimedia works motivated the first research efforts on information hiding techniques for unstructured carriers. Least Significant Bit (LSB) techniques are the best known technique, based on hiding information into the least significant bits of covers (van Schyndel et al., 1994). Depending on the data representation, least significant bits can be the last bits of the RGB composition or the last significant bits of the DCT transform, etc. The amount of information embedded in the image generally sets the amount of distortion that results in the cover image. Due to the size of image files, image steganography provides high capacity. A comprehensive description of several image steganography techniques can be found in (Chandramouli et al., 2004).

Burnett et al. (2010) propose the usage of image steganographic algorithms to create a covert channel through an image sharing website (i.e. Flickr). Authors built a software that is capable to hide secret messages into images that are then uploaded to Flickr. Users who want to read a secret message have just to configure a client application to access the Flickr profile of the sender and use a pre-shared password. images are uploaded through HTTP connections. Although authors propose their system mainly to avoid censorship, it could also be used to exfiltrate sensitive information from an organization network.

Audio steganography techniques allow to hide information in compressed (MP3, etc.) or uncompressed (WAV, etc.) audio files. In this regard, the concept of LSB steganography can be also used in the audio domain. Changing the least significant bit on each audio sample allows to encode information without generally creating an audible difference on the audio file. Depending on its configuration, an audio file may hold up to 44100 audio samples for every second, making uncompressed audio a very high capacity carrier (Bender et al., 1996). Audio steganography can be performed also in compressed audio files like MP3s (Petitcolas, 1998).

Besides image and audio, another widely used way to represent information is text. The most relevant proposals for text steganography have been based on the concept of mimic functions (Wayner, 1992). Using mimic functions, NICETEXT (Chapman and Davida, 1997) is able to transform a secret message M into a seemingly innocuous text T which contains sentences in natural language. Grothoff et al. (2005) propose to embed information into the noise and errors produced by automatic translation systems. In this way, new errors and noise detected on the steganographic text would be attributed to the automatic translation system. Particular language and expressions used in some contexts can be also used to introduce hidden information. Shirali-Shahreza (2006) proposes to take advantage of the language used on short text messages to hide secret messages.

Timing and order of packets in network protocols are also unstructured carriers that can be used to hide information. In this regard, Ahsan and Kundur (2002) propose to hide information in the order of TCP and IP packets. As TCP provides with a reassembly mechanisms, they studied how to embed information in such a way that packets are rearranged depending on the secret to transmit. A generalization on the steganographic possibilities of ordered channels was presented in (Chakinala et al., 2007).

2.2. Steganography in Structured Carriers

Although network protocols can be used as unstructured carriers (i.e. timing channels), the network protocol structure also allows to create covert communication channels. Fisk et al. (2003) identified some methods to embed hidden information on TCP, UDP, ICMP or IP header fields. Fields such as options or padding could be used to insert additional hidden information. A complete survey of structured network covert channels can be found in (Zander et al., 2007).

Document formats such as Microsoft Word (Park et al., 2009) and PDF files (Zhong et al., 2007; Lee and Tsai, 2010) can be used to hide information. In the former article, authors propose to actually hide information using redundancy in the Microsoft Office 2007 document format (OOXML), which is based on XML. Specifically, authors use unknown relationships and parts, which are elements of the OOXML that are not shown by any Office program, but are saved and can not be modified within the Office suite. In the latter proposal, information is embedded in imperceptible changes on character, word, and line spacings.

3. Related Work - Fighting Against the usage of Steganography

Even though steganography is not a threat by itself, its malicious usage entails a threat that must be addressed. Detection and/or elimination of steganography are the most common methods to fight its usage. Detection techniques try to tell whether steganography has been used or not Elimination techniques try to actively destroy or alter the hidden information, as the active warden in Simmons' scenario.

3.1. Steganography Detection

Steganalysis studies the security of steganographic algorithms. A steganographic algorithm is considered secure if it is not possible to statistically distinguish between a cover with hidden information and a clean one (Cox et al., 2008). Once the mere existence of hidden information has been detected, the purpose of steganography has been defeated, so the algorithm is considered broken. There exist several types of steganalysis. A targeted steganalysis uses the knowledge about the steganographic technique to detect stego-objects created with that specific technique, while blind steganalysis aims to distinguish if a file has some hidden information with no information about the used steganographic technique. Usually, steganalysis techniques are based on statistical models of clean files, used to design classifiers that are able to detect files with hidden information because they differ from this underlying model.

Blind and targeted steganalysis techniques have been greatly studied on digital images (Fridrich and Goljan, 2002). In audio, authors of Geetha et al. (2006) identify audio quality metrics as a statistically distinguishable feature between stego-objects and cover audio files. Using a genetic algorithm, they are able to build a distinguisher that told apart up to 80 % of stego-audio

files. Targeted approaches such as (Hernandez-Castro et al., 2010) have also been proposed, enabling in this case to detect hidden information embedded through *MP3Stego*.

Steganalysis techniques also target faulty implementations of steganographic algorithms. Bell and Lee (2010) found that most steganographic algorithm implementations modify the headers of the cover used when embedding information. This is produced because the steganographic algorithm has not been correctly implemented, removing most of the meta-data of the original cover. Authors prove that most steganographic applications create stego-objects with new metadata instead of using original meta-information, thus greatly easing the detection of stego-objects by these characteristic new headers. Both *MP3Stego* and *Outguess* are vulnerable to this kind of attack (Petitcolas, 1998; Provos, 2001).

An architecture for implementing steganalysis techniques was addressed in (Liu et al., 2009). Authors propose a system to implement several steganalysis techniques to detect steganographic content transmitted through network protocols in real time. Their approach is tested against video covert channels, but authors acknowledge the limitations on the scalability of their system when targeting multiple steganographic algorithms. Another related proposal is (Ulieru et al., 2004). In this, authors propose an agent based architecture to detect and extract hidden information from websites.

Another way to detect the usage of steganography is by finding the presence of steganographic programs used to hide or embed information. Zax and Adelstein (2009) identified the main forensic artifacts of steganographic applications. These, along with forensic software such as Encase[1] or The Coroner's Toolkit[2] may allow the detection of steganography applications, thus providing a way to alert and even take countermeasures such as forbidding the execution of those applications. This approach is useful if there exists the possibility of monitoring the suspicious computer (i.e. a malicious employee inside an organization).

3.2. Steganography elimination

Steganography elimination techniques try to eliminate possible covert channels without disrupting legitimate communication. The active warden

[1] http://www.guidancesoftware.com/

[2] http://www.porcupine.org/forensics/tct.html

of Simmon's scenario uses these kind of techniques to hamper covert communication between Alice and Bob. The main issue in these kind of techniques is how to eliminate the hidden information without changing the perception of the transmitted object, as the purpose is to break the cover communication, but not to render the legitimate channel unusable. The most common active warden technique is overwriting possible hidden information carriers with random noise.

The process of eliminating the covert channel is also called steganographic sanitization. Whitehead (2005) sanitized image data by overwriting redundancy sources of images (LSB, DCT, and DWT). Their approach showed no visual impact on the sanitized images. The requirements of the sanitization process (average of 175 milliseconds per image), made it suitable for use on networks.

Fisk et al. (2003) introduced the concept of Minimal Requisite Fidelity (MRF). MRF measures the degree of fidelity that is both acceptable for the end user and the communication (i.e. routers, network cards, system kernels, etc.). On structured carriers, MRF is a measure that quantifies the amount of information that can be modified without destroying the semantics of the modified object while on images and videos (unstructured) it can be calculated through human perception. In their work, Fisk et al. identified features of TCP/IP packets that may lead to covert communications such as window size, packet order, source ports, padding bits, etc. They described a sanitization process to delete the hidden information that was inside the boundaries of the MRF for TCP, UDP and ICMP communications.

Additionally, Schear et al. (2006) introduced a framework to avoid information leakage through public web servers. Their system uses a warden that must allow a document to be published into a public web server. The warden (which may be a human or a machine) is in charge of checking if the document contains sensitive information (as defined by the organization security policy). Any document not previously vetted is filtered by a gateway that has direct communication with the warden. To avoid information leakage through a compromised web server, authors propose to sanitize HTTP connections, rewriting sent headers. This system does not overwrite possible steganographic carriers, but tries to detect hidden information before vetting a document.

On other contexts, commercial proxies, such as SafeSquid[3], allow content and URL filtering based on security policies defined by the organization. This kind of proxies are focused on limiting the resources the employee access (i.e. social networks, personal sites, etc.) from inside the organization as well as protecting them from accidentally downloading viruses, etc. from malicious web sites. Nevertheless, to the best of the authors knowledge there is not any implementation of such proxies that eliminates steganographic content of HTTP connections.

Although previous approaches have studied methods and techniques to avoid the usage of steganography, none of them have proposed an actual solution that may avoid its usage in real scenarios (such as communicating covertly through HTTP). Previous presented approaches have allowed improvements on the active warden scenarios, but they are not able to protect against covert channels through HTTP communications, as they only take into account some of the possible features where information might be hidden. We propose a design that allows to eliminate covert channels created through an application network protocol (specifically HTTP). Our design could be easily adapted for other network application protocols. This could serve to ensure that the HTTP protocol is not being used to covertly transmit non negligible amounts of information, as well as to make certain that some services, which may be used to transmit and store hidden information, are used according to the terms of use accepted by the user.

4. Covert Channels over HTTP

The Hyper Text Transfer Protocol (HTTP) is defined in RFC 2616. HTTP is a widely used application protocol that supports transmission of any object that can be defined through a MIME type. It is a stateless protocol that works over TCP connections. It works on a request - response basis. Each time a client requests a resource, the HTTP server replies with a response for that resource. The version 1.1 of the protocol allows to maintain the TCP connection between different pairs of request-responses.

HTTP request and responses are very similar in their structure. A message is composed by a message start line, a set of headers and a message body. The message body is separated from the headers by an empty line

[3]http://www.safesquid.com

ended with \r\n. The message start line depends whether the message is a request or a response. On requests, the start line is named *request line* and is composed by the method to use, the universal identifier (URL) of the requested resource and the HTTP version (now 1.1). On responses, the start line is named *response line* and is composed by a status code followed by a description of that code and the HTTP version used. HTTP headers are pairs of header name - value separated with ':'. The message body is used to transmit any data associated with the request or the response. HTTP uses MIME types to describe the message body contents.

HTTP can be used as a covert channel to allow the exchange of information between two users. Depending on the users and the intended applications, there are two main possibilities. In the former, a user establishes a covert communication between him and a web server. In this case information can be hidden in both the content transmitted and the structure of the HTTP messages. In the latter, a user establishes a covert communication between him and another user. In this case, both users use the HTTP server as a hosting service (intermediary) for exchanging their messages. The HTTP server is usually unaware of the fact that it is being used for an unauthorized purpose. In this scenario, information is exchanged using the body of the HTTP messages, as the other party does not have access to the HTTP headers. This model was described in some depth by Jones (2001). Researchers have studied and proposed several methods to combine both of the previously described approaches.

Feamster et al. (2002) suggest the usage of an HTTP covert channel to avoid censorship on the web. In this, a web server authorized by the censors, but outside their control, works as a conscious intermediary between the client and unauthorized web servers. To perform a request to censored content, the client sends an innocuous request to the accomplice web server. The unauthorized URL is hidden inside innocuous requests URL. The accomplice web server access the censored content and hides it (using image steganography) on an image that is sent back to the client as response to the innocuous URL request.

Dyatlov and Castro (2003) describe different characteristics of HTTP messages that may be used to transfer information in a covert fashion. These include modifications on header order, structure and contents. As an example, Horenbeeck (2006) proposes the usage of Entity tag headers (which value is determined by web server implementations) to send information to clients. Entity tag headers are used to know if a specific website content requires to be

downloaded again (through the usage of the "if-no-match?" header). Clients may also modify the entity tags they sent to transmit cover information to the web server.

Besides previous presented approaches, an HTTP packet can hide information in the following elements:

- HTTP Version: Both client and server may modify the version of the HTTP protocol they are using to communicate in order to transmit some bits of information. However, this kind of behavior would be easily detected by a careful observer, as it would not be usual to change the HTTP version of the protocol during consecutive requests from the same machines, so this approach is not recommended.

- Headers: HTTP headers allow to hide information in a variety of ways. Modifying the order of HTTP headers may allow to hide certain amounts of information. Additionally, the usage of new non-standard headers, or the modification of some of the headers contents (such as the inclusion of uppercase, etc.) would also allow to codify some bits of hidden information.

- Content: Section 2 describes some techniques to hide information in carriers such as images, audio, text or document files. As HTTP allows to transmit these in the message body, hiding information in such carriers would allow to establish a covert channel through HTTP. This approach is used in (Burnett et al., 2010) with images uploaded to Flickr as carriers.

5. StegoProxy Design & Implementation

The proposed system aims to forbid the usage of covert channels through HTTP. It would be useful when dealing with malicious insiders, or users who abuse computer networks to covertly transmit illegal contents.

5.1. Working Scenario

To illustrate the aims of our system, we consider the following scenario (Figure 1). A disgruntled employee *Alice* who has access to organization's sensitive information wants to transmit it to an unauthorized recipient(*Bob*). *Alice* may directly send the sensitive documents to *Bob* by mail, but traces at the mail server would indicate that *Alice* contacted an outsider and sent

him sensitive information. This would ease *Alice*'s prosecution. Or maybe her company has implemented a Data Loss Prevention solution that will detect her attempt and stop it. To overcome these drawbacks, *Alice* could use a steganographic program to conceal the sensitive information and send it as an innocuous looking message (i.e. image) to *Bob*. If the malicious behavior is discovered, an audit may find *Alice* suspect, as she has sent strange messages to an unknown recipient. However, *Alice* could upload the hidden sensitive information (in an innocuous image) to a public web server (i.e. Twitpic[4]) (1). If *Bob* (the intended recipient for *Alice* hidden messages) is the administrator of the web service used by *Alice*, he will just have to extract the hidden information from *Alice* messages (3b). If *Bob* is another web service user, he will have to access as any other user but also extract the hidden information (3a). In this case, his identity will only be revealed to the web service where the hidden content is stored. Therefore, its identity remains anonymous to *Alice's* organization, as many other users may have accessed the same information (2).

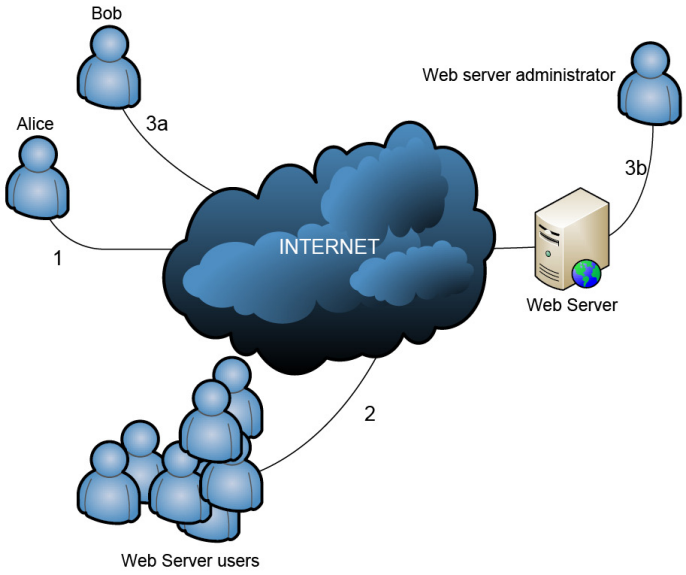

Figure 1: Working scenario diagram

[4]This service is used by Twitter users to share images and other multimedia content such as videos. http://www.twitpic.com

Based on the previous scenario, our system can be placed at a network infrastructure for two purposes: destroy any possible hidden communication outside a specific network, or eliminate any kind of covert communication coming into a network. In the former, a malicious insider might be using the network to transfer sensitive information outside an organization. We need to assume that the insider has not enough privileges to change packet routing, which is a reasonable assumption in most cases. In the latter, a computer resource such as a public web server may be used to transfer that sensitive information. As these two networks will probably be controlled by different organizations, they both must protect against these kinds of attacks separately. The design of our system takes this issue into account as it allows to filter both incoming and outgoing HTTP connections. Besides the previous scenario description, we also assume the following constraints:

- Systems used inside a controlled network can not be fully accessed or controlled. This is reasonable because of law restrictions on employees' privacy. Thus, it is only possible to control network communications.
- Usages of unknown or not explicitly allowed network protocols inside the controlled network can be forbidden. Otherwise, these protocols could be used to establish additional covert channels.
- Protocols such as SMTP, IMAP, etc. are not taken into account. Nevertheless, our framework could be easily adapted to work under those protocols.

5.2. Design

StegoProxy eliminates steganographic content by overwriting possible carriers of steganographic information. For such a purpose our framework is based on the concepts of *Steganographic Unit* and *Minimum Request Fidelity*. We define a *Steganographic Unit* as the minimum amount of semantic information that allows the transmission of hidden information, thus, the creation of a covert channel. Any part of an HTTP message that can be changed without changing the semantics of the message can be considered as a steganographic unit. HTTP steganographic units were described in Section 4. Steganographic units shall not be confused with the information carriers. As an example, two unstructured carriers such as text can refer to different steganographic units: the user agent definition and text from a web page.

The concept of Minimum Request Fidelity (MRF) was introduced by Fisk et al. (2003) and describes the maximum changes an object can support until it can not be considered semantically the same object. As HTTP allows both structured and unstructured carriers, our system will have to take into account the MRF for both kind of carriers.

Depending on the organization's aim, StegoProxy can be deployed in different network placements. If the main goal of StegoProxy is to stop possible information leakages through HTTP connections, it should be placed as a transparent proxy at the Internet gateway (Figure 2). To reduce the added overload, several StegoProxies can be deployed to work in parallel. In this way, all HTTP messages from inside the organization will have to pass through StegoProxy. On the other hand, if the aim is to ensure that a web service is not being used as an anonymous mailbox for covert communications, the proxy may be placed after the web server captures the request, but before it is processed and passed to the web application (as in Figure 3).

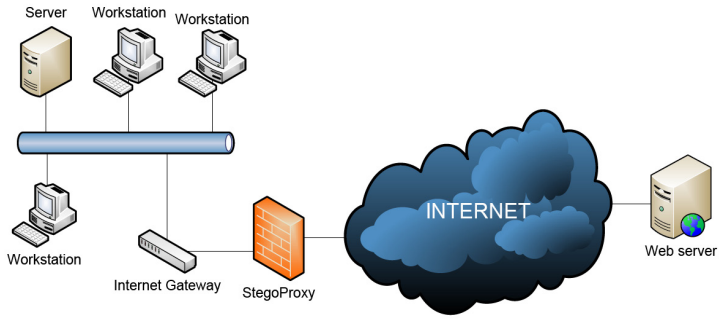

Figure 2: StegoProxy placement to avoid covert HTTP channels inside an organization

5.3. Components

As shown in Figure 4, our system comprises three main components: the HTTP inspector, the steganographic unit sanitizers and the HTTP assembler.

HTTP Inspector

This component analyzes incoming packets and divides the HTTP messages into steganographic units SU_i that will be later processed. Steganographic units are created using a database that can be expanded when new carriers are discovered on HTTP communications. Steganographic units can

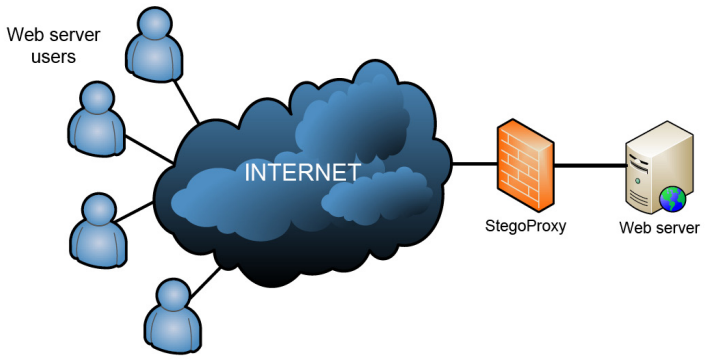

Figure 3: StegoProxy placement to avoid the abuse of a webserver as a steganographic mailbox

transport hidden information using more than one carrier. As an example, a text may have embedded information in the number of spaces it has or in the usage of uppercase letters.

Stegno Unit Sanitizers

A Sanitizer S_j removes the steganographic information that may be transmitted through a carrier j. In order to remove all possible steganographic communications through the carrier j a sanitizer may implement more than one sanitization process. We define a *targeted sanitization process* (as in targeted steganalysis) as the process that removes, from a steganographic unit SU_i, information hidden through a specific steganographic algorithm. A *blind sanitization process* is defined as the process that removes information hidden in a specific carrier j independently of the algorithm used. If a sanitizer includes more than one sanitization process, they must be executed sequentially over each steganographic unit (Figure 4). Both kinds of sanitization processes must take into account the MRF when performing their modifications to the steganographic unit. Additionally, as a steganographic unit can hide information in more than one information carrier, some steganographic units may have to pass through several sanitizers. In such a case, sanitizers are also applied sequentially.

HTTP Assembler

This component assembles all sanitizated steganographic units and builds back the HTTP message. The sanitized HTTP message is then delivered to its original recipient.

5.4. HTTP Message Sanitization Process

Formally, the process performed by StegoProxy is described in the following and in Figure 4:

- The HTTP message M is disassembled by the Inspector I in steganographic units: $I(M) = \{SU_1, SU_2, \ldots, SU_i\}$.
- Each steganographic unit may hold different carriers $SU_i = \{C_1, C_2, \ldots, C_j\}$.
- A Sanitizer S_j deletes hidden information on a specific carrier j. A sanitizer is composed by a set of sanitization processes (targeted or blind) which are applied sequentially $S_j = \{S_{j_1}, S_{j_2}, \ldots, S_{j_k}, \}$.The process of applying a sanitizer S_j to a steganographic unit removes that carrier from its steganographic payload: $S_j(SU_i) = S_j(\{C_1, C_2, \ldots, C_j\}) = \{C_1, C_2, \ldots, S_j(C_j)\} = \{C_1, C_2, \ldots, C_{j-1}\}$
- A steganographic unit SU_i is passed through its corresponding sanitizers $S_1, S_2, \ldots, S_j$ according to their carriers, resulting in a sanitized steganographic unit SSU_i. Sanitizers are to be applied sequentially: $SSU_i = S_j(\ldots S_2(S_1(SU_i))\ldots)$. Sanitizers may be not deterministic (i.e. adding random noise to steganographic units).
- Once all steganographic units have been sanitized, the Assembler A ensembles again the HTTP message using all steganographic sanitized units: $A(SSU_1, SSU_2, \ldots, SSU_i) = M_s$

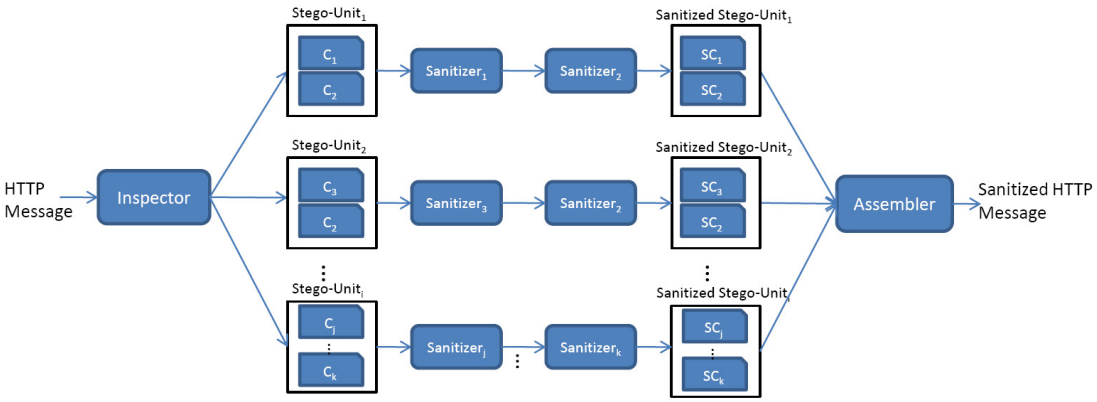

Figure 4: StegoProxy components and process diagram

5.5. Implementation

In order to evaluate the validity of our proposal, we have implemented StegoProxy as an HTTP Proxy. This implementation, in Java, is a proof-of-concept tool, not focused on providing optimum performance, despite the fact that StegoProxy is able to handle several HTTP request at the same time, as a standard HTTP Proxy[5]. Our implementation works according to the description given in Section 5.4: each HTTP message is disassembled in steganographic units. Each Unit is sanitized, and the resulting message is reassembled and sent to its destination. Our implementation allows filtering both incoming and outgoing HTTP messages.

5.5.1. Implemented Sanitizers

StegoProxy design allows the usage of sanitizers for any kind of MIME types and textual information in HTTP Headers. Nevertheless, our implementation is focused on eliminating steganographic content in images, which is the most popular steganographic carrier. Specifically we focus on eliminating least significant bit (LSB) steganography on BMP, JPG and GIF images (Petitcolas et al., 1999) together with steganography based on GIF shuffling (Kwan, 2003). Additionally, we have included a sanitization process for different headers such as Server, User Agent, etc. Obviously, our implementation can benefit from the design of more sanitizers, something we plan to continue doing for improving the tool.

LSB BMP. Least significant bit techniques modify the least significant bits of a certain part of an image to embed information. Embedded information (which is usually encrypted) introduces statistical changes in the least significant bits that makes it possible to detect it (Fridrich et al., 2001). However, several approaches such as randomization of the embedding location and reducing the image capacity can be used to difficult the detection of steganographic content using LSB in BMP images (Mielikainen, 2006).

Our LSB BMP sanitizer rewrites randomly the 2 least significant bits of the three components of random selected pixels. One quarter of the image pixels are randomly rewritten. Under these constraints, the probability of an image pixel of not being modified by our implemented sanitizer would be $p_{not\ modified} = p_{embedding} * (1 - p_{overwritting})$ where $p_{embedding}$ is the probability

[5]StegoProxy is available at http://stegoproxy.sourceforge.net/

of embedding information into one pixel of the image and $p_{overwritting}$ is the probability of overwriting the least 2 significant bits of a pixel.

Our sanitizer overwrites a quarter of the total number of pixels $p_{not\ modified} = p_{embedding} \cdot 0,75$. Therefore, in this scenario the probability of an image not having any pixel with hidden information modified is $p^{pixels}_{not\ modified} = (p_{embedding} * 0,75)^{pixels}$ where $pixels$ is the number of pixels in the image.

LSB JPG. Least significant bit steganography in JPG format uses the same philosophy as the LSB on BMP images, but instead changes the least significant bits on the DCT transforms to embed information. In our sanitizer, we overwrite the 2 least significant bits of the bitmap generated by the BMP image as in the LSB BMP, and compress again the generated image. This modifies the DCT coefficients eliminating the possible hidden information. Modifications to the least significant bits on JPG images have more impact than modifications performed on same bits of BMP images. This is perfectly natural, due to the smaller degree of redundancy due to the higher compression.

GIF. The Graphics Interchange Format (GIF) enables 256 color images. In a GIF image, first a set of 256 colors is defined as an index. Each pixel is mapped to the color index using a byte. When using LSB GIF steganography, the least significant bits used to define a color in the index are changed, thus, changing all the appearance of all pixels pointing to that color. As GIF images usually include a very specific set of colors, the detection of steganographic images using this technique is difficult. We avoid the usage of GIF steganography by overwriting with random information the least significant bits of the defined colors.

Another steganographic technique used in GIF images is modifying the order of the 256 color index. Modifying color index can be used to encode information and does not change the real appearance of the image, as pixels are modified to point to the new color index. To defeat this steganographic technique, we randomly shuffle the color index, eliminating any possible information encoded in the order of the colors.

6. Evaluation

We have evaluated the efficacy and efficiency of StegoProxy. In terms of efficacy, we evaluate if StegoProxy is able to remove steganographic content

from images sent through HTTP to web services. In terms of efficiency, we have measured the delay that the HTTP sanitization process introduces in the HTTP message exchange process.

6.1. Security Analysis

The proposed framework allows to sanitize both incoming and outgoing HTTP connections. Under the presented scenario, an attack to our proxy can be considered successful if any user is able to establish a covert communication channel through the HTTP protocol. We have identified three main attacks that may lead to this situation: denial of service attacks, usage of non identified or removable information carriers and circumventing StegoProxy.

A denial of service (DoS) attack tries to interrupt the authorized access of legitimate users to a resource or server. A DoS could overload StegoProxy in such a way that it is not able to sanitize connections in real time, drastically slowing the navigation speed. To avoid being unable to use a fundamental network protocol, StegoProxy should be disabled, being the HTTP traffic unchanged. This kind of attack, when performed outside the organization, could be detected using security solutions such as Intrusion Detection Systems (IDS). An attack of such kind from inside the organization would be infeasible, as it would leave traces on network systems that could lead easily to the inside attacker.

Another way to stop StegoProxy from hampering the creation of covert channels through HTTP would be to use HTTP fields which modification is beyond the MRF for HTTP. In (Feamster et al., 2002), authors propose the usage of the Uniform Resource Locator (URL) of each request to insert hidden information. As any change in the URL is beyond the MRF, StegoProxy can not protect against this kind of covert channel. Server responses used images as hidden information carrier. Our current framework is able to eliminate hidden content from that carrier. Nevertheless, eliminating the server response covert channel does not forbid a user to transmit sensitive information outside the boundaries of the organization without being noticed. In (Feamster et al., 2002) the URL covert channel was used to hide censored URLs. Using their scheme authors required an average of 4 HTTP requests to hide an URL. Therefore, the transmission of sensitive information would require a huge number HTTP request to the same server.

In our scenario description (Section 5.1) we specify that the attacker does not have control over the network configuration. This does not allow him to configure his system to avoid the StegoProxy. Nevertheless, HTTP messages

could be encapsulated through other protocols such as DNS (Horenbeeck, 2006). Using other protocols for such a purpose could be the same as using them as covert channels, as this kind of behavior is not authorized by the organization. Applying the proposed framework to those protocols or restricting its usage through firewall rules would enforce that all HTTP messages pass through StegoProxy.

6.2. Experimental Setup

Our experimental setup covers the first mentioned scenario. That is, a malicious employee tries to steal information from its organization using a steganographic program to hide information into images and then transmit those images to a web service. In this scenario, the proxy will be placed at the organization gateway filtering all the outgoing traffic. The second scenario, in which a web server is protected against unfair use, is equivalent to the first one with the only difference that instead of sanitizing requests, StegoProxy processes web server responses.

To perform our tests we used 5 images. For each image, we embedded a random message using "Steghide" and "Outguess" (Table 1). All generated images were uploaded to Twitpic[6] with and without using the stegoproxy. Figure 5 shows the original image used in our experiments along the versions with hidden information through the aforementioned steganographic programs.

We measured the time it took to complete the POST request when using and not using StegoProxy. For the experiments performed with StegoProxy, we also calculated the time it took to dissemble the HTTP requests, to sanitize and to ensemble them again. Figure 6 shows graphically the aforementioned time gaps. Each image upload was repeated 5 times. In our experiments, StegoProxy runs on a Core 2 Duo machine at 3 Ghz while requests are generated by other computers in the same network segment.

6.3. Results

All images sent to Twitpic were later downloaded to check for hidden content. We were unable to extract the hidden content from images sent through StegoProxy. Results show that StegoProxy is able to eliminate

[6]Twitpic is a image service for twitter that enables to store images linked to a Twitter account.

Table 1: Test images features

	Original Size	**Outguess Size**	**Steghide Size**	**Pixel Size**
Image 1	2,2 MB	837,7 KiB	2,2 MB	2592x1936
Image 2	2,0 MB	746,3 KiB	2,0 MB	2592x1936
Image 3	2,5 MB	938,8,3 KiB	2,5 MB	1936x2592
Image 4	2,1 MB	820,1 KiB	2,1 MB	2592x1936
Image 5	1,9 MB	708,5 KiB	1,9 MB	2592x1936

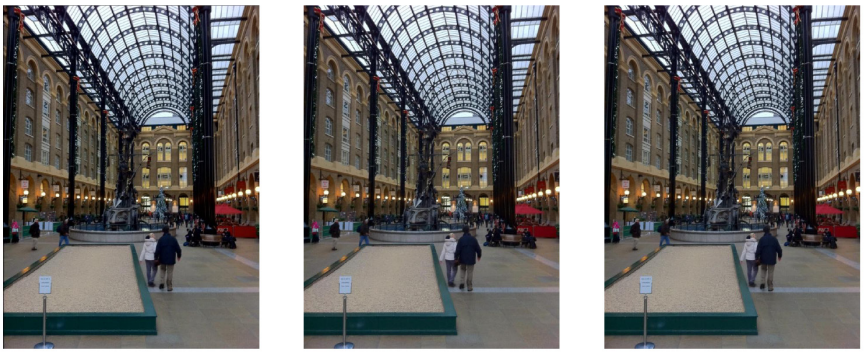

Figure 5: Example of image used during the experiments (Image 3) in its original form, with embedded information through Outguess and with embedded information through Steghide.

steganographic content from images transferred through the network without introducing any human perceptible distortion on sanitized images. However, the sanitization process increased significantly the time required to upload the images to Twitpic.

Figure 7 shows the average time measured with the three kinds of images used during the experiments. Approximately, the image upload process takes up to three times more. Although Outguess images are smaller in file size, the sanitization times is very similar to other images as the sanitization process does not depend on the file size but on the number of pixels.

Although this result may hinder the feasibility of our approach, images

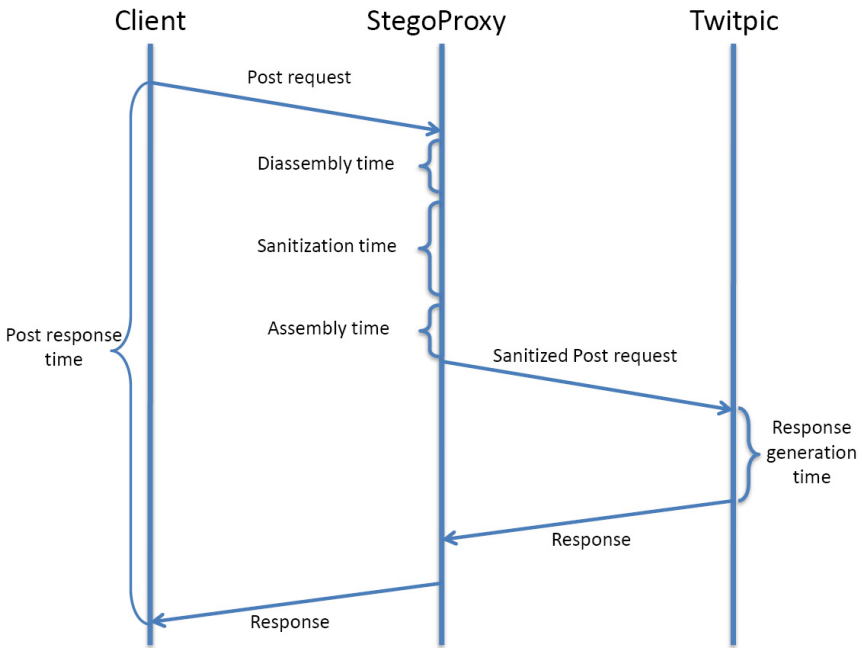

Figure 6: HTTP message process during the experiments and measured time gaps

used during the experiments were up to 2,5 MB. Using smaller images would reduce the amount of time required for sanitization. Additionally, an improved implementation or its usage on a specific purpose machine would obviously improve the measured results by a large margin. It is also important to note that StegoProxy can be parallelized in multiple machines.

7. Conclusions and Future Work

In this work we address the usage of steganography over popular network protocols such as HTTP. We have proposed a framework to limit the usage of steganography and covert channels through HTTP. Using steganography over HTTP could allow malicious employees to steal sensitive information from their organizations. Steganography over HTTP could also be used to control botnets and transmit user sensitive information. Depending on the scenario, HTTP steganography can take advantage of certain web services such as photo sharing sites as remote storage, using the aforementioned services for unauthorized purposes. However, current security policies can not block this kind of covert channels effectively, as the HTTP protocol is almost essential

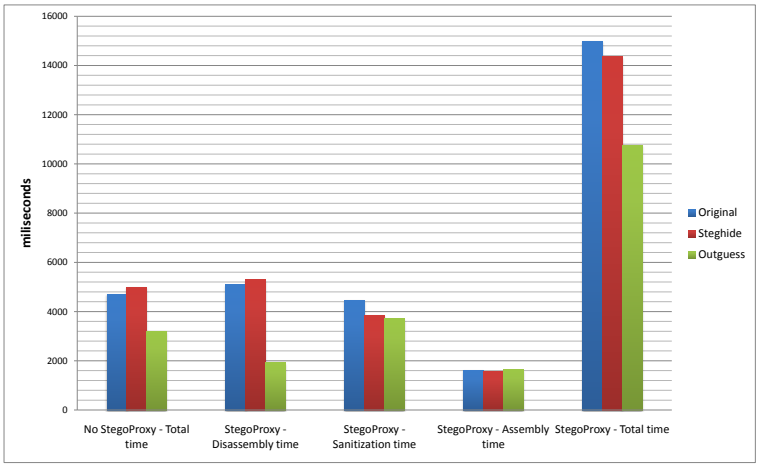

Figure 7: Average times measured during StegoProxy evaluation.

for network communications and Internet connectivity.

We have proposed a general framework that reduces the steganographic possibilities of the HTTP protocol. Our framework allows to implement different sanitizers that eliminate hidden content from any kind of information transmitted through HTTP. Although our framework reduces drastically the amount of information that can be sent covertly, it does not eliminate all covert channels. Low bandwidth covert channels are still available. Our approach allows to control both incoming and outgoing HTTP requests, being able to mitigate the malicious insider and the computer misuse risk.

Our evaluation shows that the usage of StegoProxy introduces perceptible delays to users communications. Nevertheless, our implementation is far from perfect. An improved (non Java-based) implementation and a better hardware environment would significantly increase the overall performance of our implementation. On the other hand, our main aim was achieved: We were unable to recover hidden information from images sent through StegoProxy.

Our future work will involve the extension of this approach to other communication protocols such as SMTP, P2P protocols, etc. Additionally, implementation of new sanitizers would help to provide protection against more

information carriers.

We believe this work could be useful to protect organizations from attacks such as information theft, also providing unaware intermediaries (such as web servers) a mean to protect against an unauthorized usage of their services.

References

United States of America vs Anna Chapman and Mikhail Semenko. 2010.

Ahsan, K., Kundur, D.. Practical data hiding in TCP/IP. In: Proc. Workshop on Multimedia Security at ACM Multimedia. volume 6; 2002. `http://ee.tamu.edu/~deepa/pdf/acm02.pdf` Accessed on April 2011.

Bell, G., Lee, Y.K.. A Method for Automatic Identification of Signatures of Steganography Software. IEEE Transactions on Information Forensics and Security 2010;5(2):354–358.

Bender, W., Gruhl, D., Morimoto, N., Lu, A.. Techniques for data hiding. IBM Systems Journal 1996;35(3):313–336.

Burnett, S., Feamster, N., Vempala, S.. Chipping away at censorship firewalls with user-generated content. In: Proceedings of the 19th USENIX conference on Security. USENIX Association; USENIX Security'10; 2010. p. 29–45.

Chakinala, R., Kumarasubramanian, A., Manokaran, R., Noubir, G., Rangan, C., Sundaram, R.. Steganographic communication in ordered channels. In: Information Hiding. Springer Berlin / Heidelberg; volume 4437 of *Lecture Notes in Computer Science*; 2007. p. 42–57.

Chandramouli, R., Kharrazi, M., Memon, N.. Image steganography and steganalysis: Concepts and practice. In: Digital Watermarking. Springer Berlin / Heidelberg; number 2939 in Lecture Notes in Computer Science; 2004. p. 204–211.

Chapman, M., Davida, G.. Hiding the hidden: A software system for concealing ciphertext as innocuous text. In: Information and Communications Security. Springer Berlin / Heidelberg; volume 1334 of *Lecture Notes in Computer Science*; 1997. p. 335–345.

Cox, I.J., Miller, M., Bloom, J., Fridrich, J., Kalker, T.. Digital watermarking and steganography. 2nd ed. Morgan Kaufmann, 2008.

Dyatlov, A., Castro, S.. Exploitation of Data Streams Authorized by a Network Access Control System for Arbitrary Data Transfers: Tunneling and Covert Channels over the HTTP Protocol. Technical Report; Gray-World; 2003. `http://gray-world.net/projects/papers/covert_paper.txt` Accessed on February 2011.

Feamster, N., Balazinska, M., Harfst, G., Balakrishnan, H., Karger, D.. Infranet: Circumventing Web Censorship and Surveillance. In: Proceedings of the 11th USENIX conference on Security. USENIX Association; USENIX Security'02; 2002. p. 247–262.

Fisk, G., Fisk, M., Papadopoulos, C., Neil, J.. Eliminating steganography in internet traffic with active wardens. In: Information Hiding. Springer Berlin / Heidelberg; volume 2578 of *Lecture Notes in Computer Science*; 2003. p. 18–35.

Fridrich, J., Goljan, M.. Practical steganalysis of digital images-state of the art. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. volume 4675 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*; 2002. p. 1–13.

Fridrich, J., Goljan, M., Du, R.. Reliable detection of LSB steganography in grayscale and color images. In: Proceedings of ACM Workshop on Multimedia and Security. 2001. p. 27–30.

Geetha, S., Sindhu, S., Gobi, S., Kannan, A.. Evolving GA classifiler for audio steganalysis based on audio quality metrics. In: Intelligent Sensing and Information Processing, 2006. ICISIP 2006. Fourth International Conference on. ICISIP; 2006. p. 105–108.

Grothoff, C., Grothoff, K., Alkhutova, L., Stutsman, R., Atallah, M.. Translation-based steganography. In: Information Hiding. Springer Berlin / Heidelberg; volume 3727 of *Lecture Notes in Computer Science*; 2005. p. 219–233.

Hernandez-Castro, J., Tapiador, J., Palomar, E., Romero-Gonzalez, A.. Blind steganalysis of mp3stego. Journal of Information Science and Engineering 2010;26(5):1787–1799.

Horenbeeck, M.V.. Deception on the network: thinking differently about covert channels. In: Proceedings of 7th Australian Information Warfare and Security Conference. 2006. p. 174–184.

Johnson, N., Jajodia, S.. Exploring steganography: Seeing the unseen. IEEE computer 1998;31(2):26–34.

Jones, E.. Legitimate sites as covert channels: An extension to the concept of reverse http tunnels. 2001. `http://gray-world.net/papers/lsacc.txt` Accessed on February 2011.

Kartaltepe, E.J., Morales, J.A., Xu, S., Sandhu, R.. Social Network-Based Botnet Command-and-Control: Emerging Threats and Countermeasures. In: Proceedings of the 8th international conference on Applied cryptography and network security. ACNS'10; 2010. p. 511–528.

Kwan, M.. GIFShuffle. 2003. `http://www.darkside.com.au/gifshuffle/` / Accessed on February 2011.

Lee, I.S., Tsai, W.H.. A new approach to covert communication via PDF files. Signal Processing 2010;90(2):557–565.

Liu, Y., Corbett, C., Chiang, K., Laboratories, S.N., Archibald, R., Ghosal, D.. SIDD: A Framework for Detecting Sensitive Data Exfiltration by an Insider Attack. In: System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on. 2009. p. 1–10.

McGreal, C.. FBI breaks up alleged Russian spy ring in deep cover. 2010.

Mielikainen, J.. LSB matching revisited. Signal Processing Letters, IEEE 2006;13(5):285–287.

Park, B., Park, J., Lee, S.. Data concealment and detection in Microsoft Office 2007 files. Digital Investigation 2009;5(3-4):104–114.

Petitcolas, F.A.P.. MP3Stego. 1998. `http://www.petitcolas.net/fabien/steganography` Accessed on February 2011.

Petitcolas, F.A.P., Anderson, R.J., Kuhn, M.G.. Information hiding: A survey. In: Proceedings of the {IEEE}. IEEE; volume 87; 1999. p. 1062–1078.

Provos, N.. Outguess. 2001. http://www.outguess.org/ Accessed on February 2011.

Schear, N., Kintana, C., Zhang, Q., Vahdat, A.. Glavlit: Preventing Exfiltration at Wire Speed. In: Proc. 5th Wksp. Hot Topics in Networks (HotNets). 2006. .

van Schyndel, R., a.Z. Tirkel, , Osborne, C.. A digital watermark. In: Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference. IEEE Comput. Soc. Press; volume 2; 1994. p. 86–90.

Shirali-Shahreza, M.. Stealth Steganography in SMS. In: Wireless and Optical Communications Networks, 2006 IFIP International Conference on. 2006. p. 11–13.

Simmons, G.. The history of subliminal channels. IEEE Journal on Selected Areas in Communications 1998;16(4):452–462.

Ulieru, M., Paul, R., Khan, M., Potdar, V., Chang, E.. e-Forensics Steganography System for Terrorist Information Retrieval. Proceedings of NetObject Days 2004 2004;19:28–30.

Wayner, P.. Mimic Functions. Cryptologia 1992;16(3):193–214.

Whitehead, A.. Towards eliminating steganographic communication. In: Proc. 3rd Annual Conference on Privacy, Security, and Trust. 2005. .

Zander, S., Armitage, G., Branch, P.. A survey of covert channels and countermeasures in computer network protocols. IEEE Communications Surveys & Tutorials 2007;9:44–57.

Zax, R., Adelstein, F.. FAUST: Forensic artifacts of uninstalled steganography tools. Digital Investigation 2009;6(1-2):25–38.

Zhong, S., Cheng, X., Chen, T.. Data hiding in a kind of PDF texts for secret communication. International Journal of Network Security 2007;4(1):17–26.