UNIVERSIDAD CARLOS III DE MADRID

Doctorado en Ciencia y Tecnología Informática

PhD Thesis

Information Leakage and Steganography: Detecting and Blocking Covert Channels

Author:
Jorge Blasco Alís

Supervisors:
Dr. Julio César Hernández Castro
Dr. Pedro Peris López

Computer Science Department

May 21, 2012

I was just guessing
At numbers and figures
Pulling the puzzles apart

Questions of science
Science and progress
Do not speak as loud as my heart

Coldplay - The Scientist

To my parents

Agradecimientos

Escribir estas líneas es a la vez lo más fácil y lo más difícil de todo este camino. En estos últimos años han sucedido muchas cosas. He reído y he llorado. He sentido la soledad, pero también el más cálido de los abrazos y la mejor de las compañías. Durante este camino, cada una de las personas con las que he compartido un instante de mi vida me ha aportado un granito de arena para convertirme en la persona que soy ahora. Pero no sólo eso. Sé con certeza que si no fuese por cada uno de vosotros, nunca habría llegado al final del camino.

En primer lugar tengo que dar gracias a mis tutores Julio y Pedro. Julio, me acuerdo perfectamente cuando decidí empezar esta aventura. Te preguntaba en una de nuestras reuniones si existía la posibilidad de hacer un doctorado, qué era estar en la universidad y si merecía la pena. Acertaste de pleno y sinceramente, te doy las gracias por haberme hecho elegir este camino. Gracias por tu guía y por apoyarme tanto en los ratos buenos, como en los ratos malos, que realmente es cuando más lo he necesitado. Gracias por esos viajes, por esa experiencia increíble de Inglaterra y por ser además de mi tutor un gran amigo. Pedro, gracias por llevarme hasta el final de este camino como lo has hecho. Me has ayudado más de lo que crees a llegar a este punto. Lo mejor de todos estos años es que no sólo he aprendido de vosotros como tutores, también como personas, pero sobre todo como amigos.

Tengo que dar gracias también al grupo de investigación con el que he trabajado durante todos estos últimos años y en especial a Arturo por darme la oportunidad de iniciar este camino en la universidad. Gracias también a Benjamín, Anabel, Almudena, Esther y Javi. Sin daros cuenta, todos me habéis enseñado algo durante estos años con vosotros. Juan, quería darte las gracias de forma especial por toda la ayuda que siempre me has ofrecido y todo lo que he aprendido de ti. Agustín, gracias por acogerme y hacerme sentir cada día como si estuviese en casa. Jose, gracias por saber sacar una sonrisa hasta en los momentos más difíciles. Tienes un corazón casi tan grande como tus pies. Isra, gracias por estar ahí siempre y por decirme siempre lo que piensas.

Chema, empezamos este camino juntos sin conocernos y creo que lo acabamos de la mejor forma posible. Ni en mi visión más optimista de las cosas pensaba que podría encontrar un compañero tan bueno. Ha sido un gran placer compartir todo este camino contigo, pero no te vas a librar de mi tan fácilmente. Sé que nos queda muchísimo camino por recorrer, gracias. Sergio y Noelia, gracias por escucharme y por hacerme sentir tan querido a vuestro lado. Realmente sois capaces de hacer sentir a la gente como si realmente valiese millones. Guille, simplemente, ¿Dónde carajos has estado tú todo este tiempo? Gracias por ayudarme a levantarme y por apoyarme cuando me encontré aquella piedra en el camino. No sé que habría sido de mí sin tu ayuda.

Tampoco me puedo olvidar de toda la gente que he tenido el placer de conocer en la Universidad durante este tiempo. Teresa, Pablo, Mario, Dani y Juanma gracias por acompañarme en durante el camino que pronto también va a acabar para vosotros. Kike, gracias

por ser el amigo que está ahí siempre que salgo del despacho. Ester, no podía imaginar que en tan poco tiempo una persona pudiese llegar a marcarme tanto, gracias de corazón.

A los de siempre también os tengo que dar las gracias. Luismi, Josete, Borja, Fer y Jimmy. Habéis estado siempre ahí, aunque yo no estuviese y eso dice mucho de vosotros. Mil gracias. Susana, contigo compartí un periodo de mi vida muy importante en el que aprendí mucho de ti y de mi mismo. Sin ti hoy no estaría escribiendo estas líneas. Gracias.

No me quiero dejar tampoco a aquellos que en estos años me habéis aportado esos pequeños granitos de arena. Gracias Esther, Alvarito, Charlie, Javitchu, Luis Recuenco, Luis Martín y otros tantos que me habéis enseñado muchas cosas sin saberlo.

También tengo que dar las gracias a mis familias inglesas. Gracias a mi familia del sur: Nacho, Cris, Pablo y Andreu. No os imagináis la suerte que he tenido de encontraros tan lejos, pero a la vez teneros tan cerca. Gracias a mi familia del norte: Javi, Pili, Diego y Erika. Pasen los años que pasen, siempre estáis ahí como el primer día. Sois únicos. Y por último, gracias a mi familia londinense: Pedro e Irene. Os descubrí de casualidad y me habéis acogido, escuchado y ayudado sin pedir nada a cambio.

Evidentemente no me puedo olvidar de los que pusieron el primer granito de arena para que yo estuviese hoy aquí. Mamá, Papá gracias por haberme educado y convertido en quien soy. Gracias por esos cinco minutos de reflexión por mis trastadas y gracias por todo el cariño y apoyo incondicional que me habéis dado siempre. Sin vosotros por encima de todo, no sería quien soy hoy. Gracias a Luis y a Judit, por todo vuestro apoyo y por esos dos regalos inesperados que nos habéis dado a todos. Tampoco me puedo olvidar de María y Nachete, gracias por darnos la alegría y la vida que nos falta a los demás a veces. Gracias Yaya y todos mis mallorquines por vuestro apoyo y ánimos.

Por último me gustaría dar las gracias a dos personas que no ya están ni estarán, pero que aunque se hayan ido, les llevo en el corazón como si aún estuviesen aquí. Una por enseñarme lo que son las ganas de vivir. Gracias Javivi. La otra por enseñarme a ser mejor persona y quererme siempre con todo su corazón. Gracias Avi.

Gracias a todos por convertirme en la persona que soy hoy.

Jorge

Abstract

This PhD Thesis explores the threat of information theft perpetrated by malicious insiders. As opposite to outsiders, insiders have access to information assets belonging the organization, know the organization infrastructure and more importantly, know the value of the different assets the organization holds. The risk created by malicious insiders have led both the research community and commercial providers to spend efforts on creating mechanisms and solutions to reduce it. However, the lack of certain controls by current proposals may led security administrators to a false sense of security that could actually ease information theft attempts.

As a first step of this dissertation, a study of current state of the art proposals regarding information leakage protections has been performed. This study has allowed to identify the main weaknesses of current proposals which are mainly the usage of steganographic algorithms, the lack of control of modern mobile devices and the lack of control of the action the insiders perform inside the different trusted applications they commonly use. Each of these drawbacks have been explored during this dissertation.

Regarding the usage of steganographic algorithms, two different steganographic systems have been proposed. First, a steganographic algorithm that transforms source code into innocuous text has been presented. This system uses free context grammars and to parse the source code to be hidden and produce an innocuous text. This system could be used to extract valuable source code from software development environments, where security restrictions are usually softened. Second, a steganographic application for iOS devices has also been presented. This application, called “Hide It In” allows to embed images into other innocuous images and send those images through the device email account. This application includes a cover mode that allows to take pictures without showing that fact in the device screen. The usage of these kinds of applications is suitable in most of the environments which handle sensitive information, as most of them do not incorporate mechanisms to control the usage of advanced mobile devices. The application, which is already available at the Apple App Store, has been downloaded more than 5.000 times.

In order to protect organizations against the malicious usage of steganography, several techniques can be implemented. In this thesis two different approaches are presented. First, steganographic detectors could be deployed along the organization to detect possible transmissions of stego-objects outside the organization perimeter. In this regard, a proposal to detect hidden information inside executable files has been presented. The proposed detector, which measures the assembler instruction selection made by compilers, is able to correctly identify stego-objects created through the tool *Hydan*. Second, steganographic sanitizers could be deployed over the organization infrastructure to reduce the capacity of covert channels that can transmit information outside the organization. In this regard, a framework to avoid the usage of steganography over the HTTP protocol has been proposed. The presented framework, diassembles HTTP messages, overwrites the possible carriers of hidden information with random noise and assembles the HTTP message again. Obtained results show that it is possible to highly reduce the capacity of covert channels created through HTTP. However, the system introduces a considerable delay in communications.

Besides steganography, this thesis has also addressed the usage of trusted applications to extract information from organizations. Although applications execution inside an organization can be restricted, trusted applications used to perform daily tasks are generally executed without any restrictions. However, the complexity of such applications can be used by an insider to transform information in such a way that deployed information protection solutions are not able to detect the transformed information as sensitive. In this thesis, a method to encrypt sensitive information using trusted applications is presented. Once the information has been encrypted it is possible to extract it outside the organization without raising any alarm in the deployed security systems. This technique has been successfully evaluated against a state of the art commercial data leakage protection solution. Besides the presented evasion technique, several improvements to enhance the security of current DLP solutions are presented. These are specifically focused in avoiding information leakage through the usage of trusted applications.

The contributions of this dissertation have shown that current information leakage protection mechanisms do not fully address all the possible attacks that a malicious insider can commit to steal sensitive information. However, it has been shown that it is possible to implement mechanisms to avoid the extraction of sensitive information by malicious insiders. Obviously, avoiding such attacks does not mean that all possible threats created by malicious insiders are addressed. It is necessary then, to continue studying the threats that malicious insiders pose to the confidentiality of information assets and the possible mechanisms to mitigate them.

Resumen

Esta tesis doctoral explora la amenaza creada por los empleados maliciosos en lo referente a la confidencialidad de la información sensible (o privilegiada) en posesión de una organización. Al contrario que los atacantes externos a la organización, los atacantes internos poseen de acceso a los activos de información pertenecientes a la organización, conocen la infraestructura de la misma y lo más importante, conocen el valor de los mismos. El riesgo creado por los empleados maliciosos (o en general atacantes internos) ha llevado tanto a la comunidad investigadora como a los proveedores comerciales de seguridad de la información a la creación de mecanismos y soluciones para reducir estas amenazas. Sin embargo, la falta de controles por parte de ciertas propuestas actuales pueden inducir una falsa sensación de seguridad en los administradores de seguridad de las organizaciones, facilitando los posibles intentos de robo de información.

Para la realización de esta tesis doctoral, en primer lugar se ha realizado un estudio de las propuestas actuales con respecto a la protección de fugas de información. Este estudio ha permitido identificar las principales debilidades de las mismas, que son principalmente la falta de control sobre el uso de algoritmos esteganográficos, la falta de control de sobre dispositivos móviles avanzados y la falta de control sobre las acciones que realizan los empleados en el interior de las organizaciones. Cada uno de los problemas identificados ha sido explorado durante la realización de esta tesis doctoral.

En lo que respecta al uso de algoritmos esteganográficos, esta tesis incluye la propuesta de dos sistemas de ocultación de información. En primer lugar, se presenta un algoritmo esteganográfico que transforma código fuente en texto inocuo. Este sistema utiliza gramáticas libres de contexto para transformar el código fuente a ocultar en un texto inocuo. Este sistema podría ser utilizado para extraer código fuente valioso de entornos donde se realiza desarrollo de software (donde las restricciones de seguridad suelen ser menores). En segundo lugar, se propone una aplicación esteganográfica para dispositivos móviles (concretamente iOS). Esta aplicación, llamada “Hide It In” permite incrustar imágenes en otras inocuas y enviar el estego-objeto resultante a través de la cuenta de correo electrónico del dispositivo. Esta aplicación incluye un modo encubierto, que permite tomar imágenes mostrando en el propio dispositivo elementos del interfaz diferentes a los de a cámara, lo que permite tomar fotografías de forma inadvertida. Este tipo de aplicaciones podrían ser utilizadas por empleados malicios en la mayoría de los entornos que manejan información sensible, ya que estos no suelen incorporar mecanismos para controlar el uso de dispositivos móviles avanzados. La aplicación, que ya está disponible en la App Store de Apple, ha sido descargada más de 5.000 veces.

Otro objetivo de la tesis ha sido prevenir el uso malintencionado de técnicas esteganografi-cas. A este respecto, esta tesis presenta dos enfoques diferentes. En primer lugar, se pueden desplegar diferentes detectores esteganográficos a lo largo de la organización. De esta forma, se podrían detectar las posibles transmisiones de estego-objetos fuera del ámbito de la misma. En este sentido, esta tesis presenta un algoritmo de estegoanálisis para la detección de información oculta en archivos ejecutables. El detector propuesto, que mide la selección de instrucciones realizada por los compiladores, es capaz de identificar correctamente estego-objetos creados a través de la herramienta de *Hydan*. En segundo lugar, los "sanitizadores" esteganográficos podrían ser desplegados a lo largo de la infraestructura de la organización para reducir la capacidad de los posibles canales encubiertos que pueden ser utilizados para transmitir información sensible de forma descontrolada.. En este sentido, se ha propuesto un marco para evitar el uso de la esteganografía a través del protocolo HTTP. El marco presentado, descompone los mensajes HTTP, sobrescribe los posibles portadores de información oculta mediante la inclusión de ruido aleatorio y reconstruye los mensajes HTTP de nuevo. Los resultados obtenidos muestran que es posible reducir drásticamente la capacidad de los canales encubiertos creados a través de HTTP. Sin embargo, el sistema introduce un retraso considerable en las comunicaciones.

Además de la esteganografía, esta tesis ha abordado también el uso de aplicaciones de confianza para extraer información sensible de las organizaciones. Aunque la ejecución de aplicaciones dentro de una organización puede ser restringida, las aplicaciones de confianza, que se utilizan generalmente para realizar tareas cotidianas dentro de la organización, se ejecutan normalmente sin ninguna restricción. Sin embargo, la complejidad de estas aplicaciones puede ser utilizada para transformar la información de tal manera que las soluciones de protección ante fugas de información desplegadas no sean capaces de detectar la información transformada como sensibles. En esta tesis, se presenta un método para cifrar información sensible mediante el uso de aplicaciones de confianza. Una vez que la información ha sido cifrada, es posible extraerla de la organización sin generar alarmas en los sistemas de seguridad implementados. Esta técnica ha sido evaluada con éxito contra de una solución comercial para la prevención de fugas de información. Además de esta técnica de evasión, se han presentado varias mejoras en lo que respecta a la seguridad de las actuales soluciones DLP. Estas, se centran específicamente en evitar la fuga de información a través del uso de aplicaciones de confianza.

Las contribuciones de esta tesis han demostrado que los actuales mecanismos para la protección ante fugas de información no responden plenamente a todos los posibles ataques que puedan ejecutar empleados maliciosos. Sin embargo, también se ha demostrado que es posible implementar mecanismos para evitar la extracción de información sensible mediante los mencionados ataques. Obviamente, esto no significa que todas las posibles amenazas creadas por empleados maliciosos hayan sido abordadas. Es necesario por lo tanto, continuar el estudio de las amenazas en lo que respecta a la confidencialidad de los activos de información y los posibles mecanismos para mitigar las mismas.

Table of Contents

List of Figures

List of Tables

Acronyms

DLP	Data Loss Prevention
PII	Personable Identifiable Information
AES	American Encryption Standard
BLP	Bell-LaPadula
TEA	Tiny Encryption Algorithm
HTTP	Hyper Text Transfer Protocol
SMTP	Simple Mail Transfer Protocol
P2P	Peer to peer
FTP	File Transfer Protocol
TCP	Transmission Control Protocol
ROC	Receiver operating characteristic
TPR	True Positive Rate
FNR	False Negative Rate
LSB	Least Significant Bit
SPA	Simple Pair Analysis
MRF	Minimal Requisite Fidelity
IDS	Intrusion Detection System
BLP	Bell-LaPadula
PDF	Portable Document File
IT	Information Technology
VPN	Virtual Private Network
XML	Extensible Markup Language
LSB	Least Significant Bit
PNG	Portable Network Graphics
JPEG	Joint Photographic Experts Group
FPGA	Field Programmable Gate Array

Chapter 1

Introduction

Its just a path you haven't taken,
A story never told,
Don't try to start something unless you care
Something unless you care for it
Care for it
Delorentos - Care for

1.1 Motivation

Nowadays an organization patrimony is not only measured in terms of its physical possessions, but also in terms of the information they hold. Buildings, computers, data storage devices and networks are assets that provide the means to perform business activities, but information assets enable opportunities, competitive advantages, innovation and awareness that can be translated into increases in the organization economic growth. As any other organization asset, information is also subject to threats. These include attacks from people outside or inside the organization that may try to affect the availability, integrity or confidentiality of the information assets held by the organization.

An insider can be defined as a user with legitimate access to the organization's computers and networks [3]. Although insiders usually work on achieving the organization's goals, malicious insiders are becoming an increasing threat. A malicious insider is an insider who intentionally misuses an authorized level of access to affect the confidentiality, integrity or availability of the organizations' assets, including both data and systems [4]. Unlike attackers from outside the organization, malicious insiders do not have to bypass most of the security mechanisms deployed on the organization (security controls, information security systems, closed circuit television, etc.) to attempt a successful attack [5]. Additionally, they usually know the organization's computer infrastructure, the way data is stored across it and, more significantly, its value [6]. Malicious insiders can interfere with the availability of the organization services or can put at risk the confidentiality of sensitive information such as personal data or intellectual property. A recent survey conducted over more than 500 organizations, stated that 51% of them had to deal with insider related security events [7].

One of the threats information assets are exposed to is information leakage, a threat that comprises the unauthorized disclosure of confidential information. This includes any kind of information that is essential to achieve the organization's goals, such as blueprints, source code, financial or investments plans, etc. Client's private records are also considered sensitive, as they enable the organization to access its clients and have to be protected because of privacy regulations. Although information leakage can be originated by both insiders (accidentally or maliciously) and outsiders, information leakage incidents originated by users from within the organization are often much more severe than incidents caused by outsiders [7]. Additionally insiders are more prone to commit crimes that involve the information assets the organization holds, as can be seen on data gathered in Table 1.1 [1].

Committed crime	**Percentage (non-exclusive)**
Unauthorized access to / use of corporate information	63 %
Unintentional exposure of private or sensitive data	57 %
Virus, worms, or other malicious code	37 %
Theft of intellectual property	32 %

Table 1.1: Most common e-crimes committed by insiders [1]. Crimes are non-exclusive (i.e. an insider can access or use corporate information to steal intellectual property from the organization)

Information leakage events usually result in loss of competitiveness, economic fees imposed by governments and loss of reputation [8]. In fact, the economic losses produced by information theft related events have been studied extensively in several reports and surveys. One, conducted over United States banking and financial institutions, stated that 91% of the organizations that suffered from these threats experienced financial losses, which on 30% of the cases exceeded 500.000 US dollars [9]. As an example, a T-Mobile employee sold hundreds of thousands of personal data records of United Kingdom customers to rival firms [10]. Those records were used to offer new contracts to T-Mobile customers, causing not only a major economic impact due to reputation loss and privacy concerns, but also because of clients that actually switched to another provider. Other reports analyzing the risks derived from information leakage [11] estimate that the average economic cost per incident was 2.2 million Euros during 2008[1].

In order to mitigate the risks created by information leakage incidents, security providers have developed a new kind of security tools usually known as DLP (Data Loss or Leakage Protection) solutions. Vendors often claim that these tools are able to prevent both accidental and malicious information leakage [12]. DLP solutions analyze network packets, files stored in computers and data in use to ensure sensitive data is treated according to the organization's policies and regulations. In this scenario, techniques such as steganography are of special interest. Steganography techniques allow the creation of covert channels by hiding information inside other carriers that look unsuspicious. In this way, malicious insiders could evade current information leakage protection mechanisms.

As the awareness on the importance of the information leakage thread rises, the efforts spent in creating solutions to the information leakage problem have also increased. In fact, the amount of money that organizations are spending on these solutions has increased during the last years and is expected to continue its increase in the next years [1]. Additionally, as can be seen in Figures 1.1 and 1.2[2] researchers have increased their efforts in studying

[1]This report only accounts for information leakage events that are related to private data, not other economic assets such as blueprints, etc. Unfortunately, organizations try to avoid these kind of events being publicly known.

[2]This information has been gathered on 16th February 2012 from Google Scholar (http://scholar.google.com) and Sciverse (http://www.sciverse.com) using the search expressions specified in the Figure and restricting the search to the paper content and engineering and computer science areas.

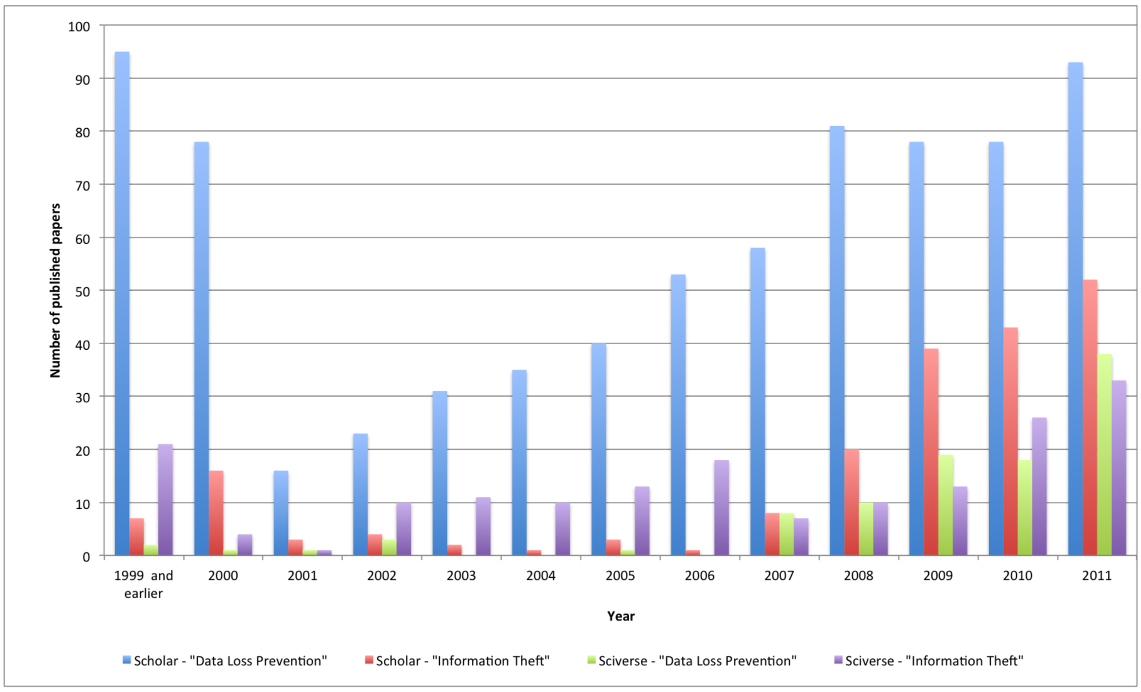


Figure 1.1: Evolution of the number of papers published regarding "Information theft" and "Data loss prevention" in the last years

information leakage and steganography. Of special interest is the evolution of the increase in the appearance of the terms "information theft" in the very last years.

The main challenge addressed by this thesis is to **analyze and improve the current mechanisms to prevent malicious information leakage**. To achieve this, the PhD dissertation addresses the security of current information leakage protection techniques and mechanisms. The security of such systems is analyzed by exploring the impact of different techniques such as the creation of covert channels by means steganography over them. If these techniques can be successfully used to extract information from an organization, new techniques to protect and avoid such attacks should be designed and implemented. Therefore and to offer solutions to the security problems detected, this dissertation also addresses the design and evaluation of mechanisms to protect against techniques that could be used by malicious insiders to steal sensitive information from an organization.

1.2 Research Question

The work described in this PhD dissertation addresses the following *Research Question*:

Are the current information leakage protection mechanisms enough to protect against malicious insiders trying to steal information from an organization?

Addressing this question inevitably requires to explore the following secondary research questions:

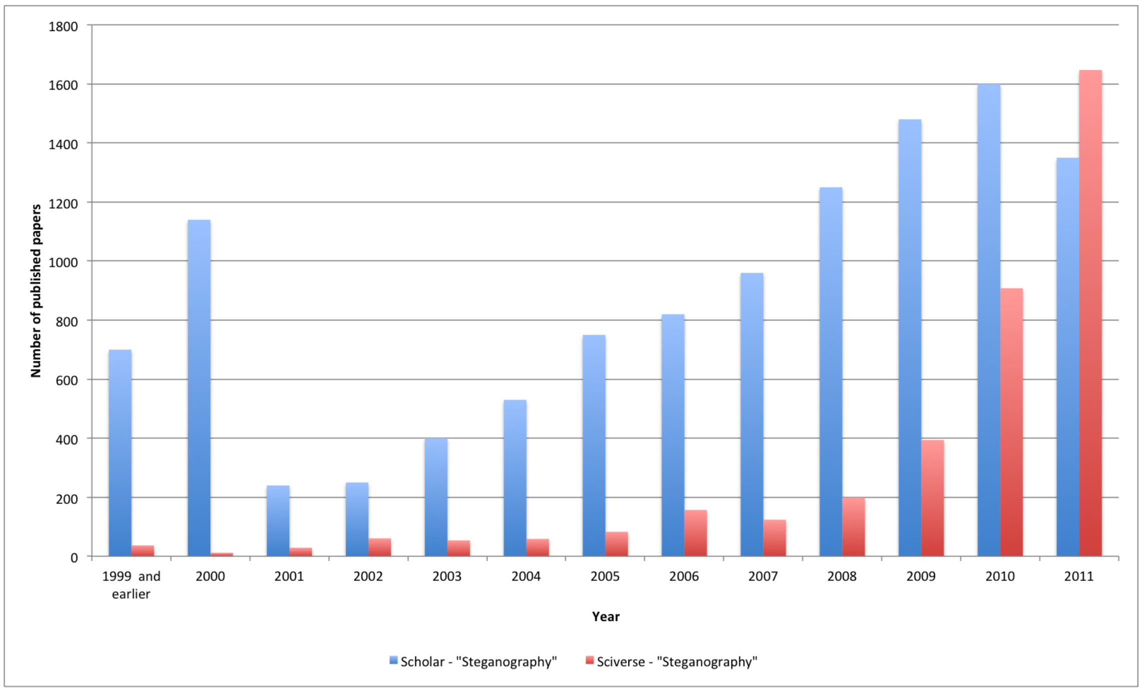


Figure 1.2: Evolution of the number of papers published regarding "Steganography" in the last years

- What kind of computer or not computer resources can be used by a malicious employee to perform a malicious activity such as information theft?
- Is it possible to transform information in such a way that its usage and transmission passes unnoticed to information leakage prevention mechanisms?
- Is it possible to use the trusted resources and applications in a computer system to leak information?
- It is possible to control the required leakage vectors to hinder information leakage attempts?

Each of these questions is addressed during this thesis dissertation. In Chapter 7 a summary of the answers found in this thesis is presented.

1.3 Goals

Although information leakage protection solutions have gained a lot of popularity during the last years, they are still in their early development phase. The proposed research questions serve as a starting point to address this.

In order to do so, this PhD thesis examines information leakage problem from the malicious insider point of view. In particular, **this thesis focuses on addressing the leakage of sensitive information insiders have already access to**. The goals of these PhD thesis haven extracted from the research questions defined in Section 1.2. Specifically, the goals of these PhD are the following:

- **Analyze the security of current proposals regarding information leakage protection systems.**
- **Develop new protection mechanisms to enforce the security of data within organizations and hinder the malicious leakage of information.**
- **Design, implement and evaluate new methods and steganographic algorithms that could be used to create covert channels with the purpose of leaking information from a supposedly protected organization.**
- **Design, implement and evaluate new active and passive techniques to fight against the usage of steganography.**

In order to achieve these goals, a work plan and methodology was defined during the PhD proposal preparation.

1.4 Methodology and Planning

The research activities performed during this PhD dissertation can be divided into three main blocks. The first block comprises the study and analysis of current proposals regarding information leakage and steganography. This analysis has been used as a starting point for the second block of this thesis which comprises the creation of attacks to evade current solutions to protect against information leakage. This block includes the design of steganographic methods and other techniques to avoid information leakage protection mechanisms. Finally, the third block of this research period has been focused on developing the mechanisms and techniques to improve the current information leakage protection techniques in order to avoid the previously designed attacks.

The research work involved in this thesis has been organized in different phases. Figure 1.3 shows a Gantt chart that depicts the thesis schedule. Description of each of the phases along its tasks is given in the following lines.

1.4.1 Phase 1: Review and Analysis of State of the Art Proposals

This phase involves the review and analysis of state of the art proposals related to steganography and information leakage protection. This phase focuses first on surveying the current proposals regarding information leakage protection. Additionally, steganographic and steganalytic algorithms and their implementations are studied. Activities included in this phase have been performed through all the thesis duration, reviewing new proposals regarding the thesis scope. The preparation of the PhD proposal document is included as a part of this phase.

Task 1.1: Information Gathering

This task involves the process of gathering information related to the subject of study across multiple scholar sources. Although, this task is intended to be executed during the first months of the thesis preparation, it has been carried out during the whole research period, assimilating new proposals offered by other authors in the same topics.

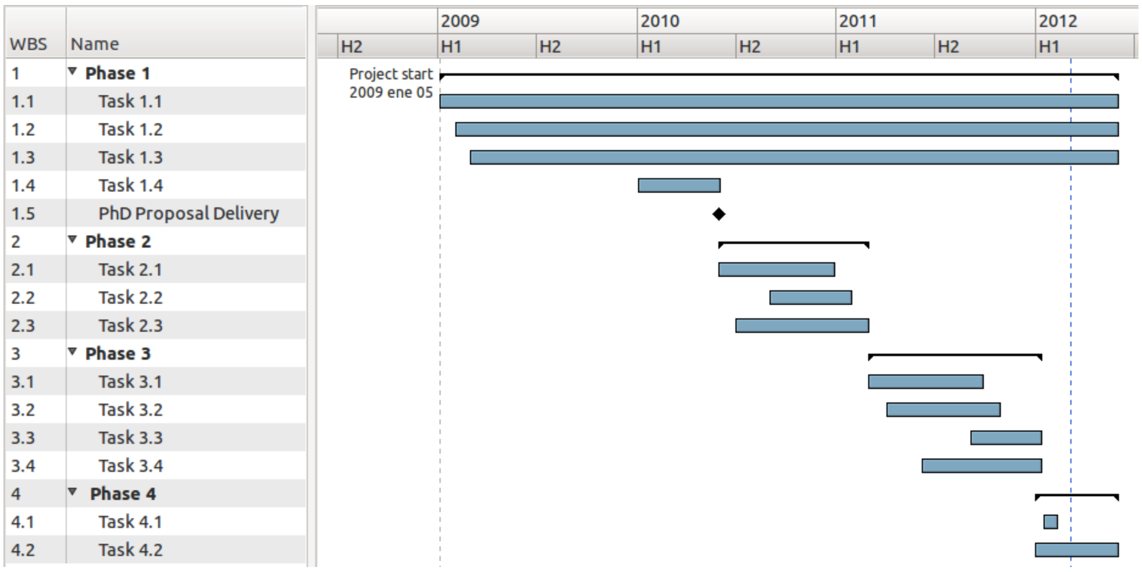


Figure 1.3: Gantt chart depicting the research plan

Expected duration Activity to be performed across PhD.

Expected outcome Reference collection and bibliography.

Task 1.2: Analysis of the State of the Art Proposals Regarding Information Leakage Protection

This task includes the analysis of systems and architectures proposed for the problem at hand (information leakage). The result of this analysis is presented in Chapter 2 and Section 6.2. Both academic and commercial proposals to fight against information leakage have been analyzed.

Expected duration Activity to be performed during the whole PhD.

Expected outcome Review of current proposals regarding Information Leakage Protection.

Task 1.3: Analysis of State of the Art Steaganography Techniques

This task gathers and analyzes current state of the art techniques on steganography and steganalysis. The suitability of these algorithms to create covert channels to extract information from organization environments has been studied. Additionally, steganalysis, active warden and computer forensics techniques to avoid and detect usage of current steganographic algorithms have been studied.

Expected duration Activity to be performed during the whole PhD.

Expected outcome Review of current proposals regarding steganography, steganalysis and related techniques.

Task 1.4: PhD Proposal Preparation

Results of previous tasks were used for the preparation of the PhD Proposal document.

Expected duration 5 months.

Expected outcome PhD Proposal.

1.4.2 Phase 2: Design and Analysis of Steganographic Algorithms and Other Attacks for Information Theft

This phase aims to analyze the threat that covert channels created by means steganography and other attacks arise into organizations' sensitive data. During this phase, new steganographic algorithms that could be used, among other applications, to steal information from organizations have been designed. Additionally, evasion of information leakage protection techniques by other means is addressed in this phase.This phase also involves the implementation of the developed algorithms, evaluating its feasibility in organization environments.

Task 2.1: Design of Steganographic Algorithms and Other Techniques for Information Theft

This task is about designing new steganographic algorithms focused on overpassing current protection techniques against information theft. A specific focus on the usage of trusted applications for information theft purposes is given. A study of its possibilities and digital artefacts (evidence) generated by those applications will be performed.

Expected duration 7 months.

Expected outcome Design of new steganographic algorithms and attacks to information leakage protection systems.

Task 2.2: Implementation and Evaluation

The designed attacks and steganographic algorithms have been implemented as part of this task. This has served to experimentally evaluate the performance, robustness, security and capacity of the different proposals.

Expected duration 5 months.

Expected outcome Results and feedback for task 2.1.

Task 2.3: Documentation

All documentation generated during Phase 2 has been carried out in this task. This includes the software registration processes that was carried out during this PhD.

Expected duration 8 months.

Expected outcome Research papers and documentation.

1.4.3 Phase 3: New Protection Mechanisms for Information Leakage Protection

The purpose of this phase is to improve the current protection mechanisms to avoid information leakage. In this way, detected vulnerabilities in current information leakage protection mechanisms have been addressed. In this phase new steganalysis, active warden and computer forensic related techniques have been proposed. Implementation and evaluation of proposed techniques have been also carried out during this phase.

Task 3.1: Design of Techniques to Avoid the Usage of Steganography

This task addresses the design of new techniques to avoid the usage of steanography inside organization environments. In this task active warden techniques as well as other techniques such as steganalysis and mechanisms to avoid the usage of steganography inside organization environments have been addressed.

Expected duration 7 months.

Expected outcome Techniques to avoid the usage of steganography in organization environments.

Task 3.2: Design of New Techniques to Detect and Avoid Information Leakage in Organization Environments

In this task, techniques to avoid attacks detected in Task 2.1 have been addressed. Designed mechanisms serve to complete the protection mechanisms included in current information leakage protection solutions.

Expected duration 7 months.

Expected outcome A set of techniques and mechanisms to improve current information leakage protection solutions.

Task 3.3: Implementation and Evaluation

Proposed techniques have been implemented and experimentally evaluated. This phase also includes the experiment design. Specifically, attacks designed in Task 2.1 have been evaluated against the new developed protection mechanisms.

Expected duration 4,5 months.

Expected outcome Results and feedback for Task 3.1.

Task 3.4: Documentation

Results gathering and documentation has been carried out in this Task.

Expected duration 7 months.

Expected outcome Documented results and research publications.

1.4.4 Phase 4: Thesis Preparation

During this phase results obtained on previous stages of the thesis have been gathered to built the dissertation document. Although the contributions on the previous stages can be incorporated directly in the thesis document, it has been required to spent time gathering and consolidating the work done during the whole research period. This kind of work is out of the scope of the previous stages so it will be carried out separately.

Task 4.1: Results gathering and Document Definition

Previous documentation and publications have been gathered. Additionally, with the gathered information, a PhD dissertation structure has been defined.

Expected duration 1 months.

Expected outcome Necessary material and PhD Template document.

Task 4.2: Thesis Writing and Revisions

Previous defined structure has been filled up with the results of the different stages and reviews. All the generated publications and documentation have been modified and integrated into the dissertation.

Expected duration 5 months.

Expected outcome PhD dissertation.

1.5 Contributions

The work performed during this PhD has resulted in several contributions in the field of information leakage protection and steganography. The first contribution of this work is **a survey in current information leakage protection systems**. Although academic authors and information security providers have developed several approaches to deal with information leakages (accidental or malicious), no survey has been performed. This survey has served to identify the pitfalls of current approaches, serving as a starting point for the next contributions of this thesis.

Based on the previous contribution a set of attacks and techniques to avoid current information leakage protection mechanisms have been developed, producing the following contributions:

- **A method to transform source code into natural language text** [13]. This method would allow to transform information that is usually identified as sensitive (source code) into innocuous literary texts fragments from books that can not be identified by current DLP solutions as sensitive.

- **A steganographic application for mobile phone devices** [14]. As part of this PhD, an application for mobile devices (specifically for iOS devices) was developed. The application, which source code has been registered, has been downloaded more than 5000 times worldwide (since 25th September 2011).

- **A method to evade information leakage protection mechanisms through the use of trusted applications** [15]. An attack over current DLP solutions that could be used by malicious insiders to steal information inadvertently has been identified. In this attack, a malicious insider uses a trusted application in the organization environment to modify sensitive data in order to render the information leakage protection mechanisms unusable.

To prevent previous attacks, new methods and mechanisms to protect against information leakage and most specifically steganography have been developed. The main contributions under this scope are:

- **A new method to hinder the usage of steganography through HTTP** [16]. Using an active warden role, incoming and outgoing HTTP messages can be sanitized. This reduces the capacity of covert channels created by means of steganography. The proposed method could be easily adapted to other network protocols.

- **A new steganalytic method to detect the existence of hidden information in executable files** [17]. The proposed method is able to detect if a tool named *Hydan* has been used to hide information inside executable files.

- **A set of indicators and mechanisms that could be used to improve current information leakage solutions to avoid information leakage attempts using trusted applications** [15]. If implemented, these indicators could be used to detect if an employee is maliciously using a trusted application to steal information from the organization.

1.6 Evaluation Methods

Several of the contributions and proposals of this thesis have required an evaluation. Steganographic methods proposed in Chapter 4 have been evaluated in terms of security and robustness. Additionally, the developed mobile application has been evaluated in terms of efficiency. Both properties have been evaluated experimentally. Several stego-objects were created with both tools ("CSteg" and "Hide It In") and compared with models of clean objects.

To evaluate the usage of trusted applications to leak information, an attack using a cipher was implemented into a spreadsheet program. Several aspects of the proposed evasion technique have been evaluated. A security evaluation (from the point of view of the attacker) was performed. The goal of this evaluation was to compare the security of the proposed algorithm implementation with standard implementations of cryptographic algorithms. Additionally, the performance and efficiency of the evasion method have been evaluated. Spreadsheet encryption times have been compared with encryption times of a C standard implementation. Complexity in terms of space (memory usage) was analytically obtained. Finally, a experimental test against a commercial DLP solution was performed in order to prove the validity of the proposed attack.

Protection mechanisms developed to fight against information leakage have been experimentally evaluated. The framework to hinder the usage of steganography through the HTTP protocol was implemented in Java. First, a statistical evaluation of the amount of hidden information that could survive the sanitization process used by the framework was performed. Then, several experiments were carried out. In these experiments, random information was hidden using several steganographic programs and uploaded to Twitter[3]. During the experiments the security and performance of the Java implementation were evaluated.

The steganalysis technique presented in Chapter 5 has been experimentally evaluated using executable files included in an Ubuntu distribution. A set of executable files was used to hide information using “Hydan”. The proposed technique was used to compare the set of executable files with a set of non modified executable files. The experimental evaluation served also to establish a threshold to maximize the effectiveness of the steganalytic technique.

Details about each of the evaluation processes can be found in the corresponding Chapters of this document.

1.7 Document Organization

This dissertation assembles the work related to information leakage protection and steganography performed during the entire PhD studies. Some of the work presented in this dissertation comes from papers that have been published during this period of time. Specifically, Chapter 3, Chapter 6 and Chapter 5 take most of its contents from published papers that have been adapted to give an improved reading experience. The remainder of this dissertation is divided in six different Chapters and a reference Section. They are organized as follows:

- **Chapter 2** introduces the reader to the field of information leakage protection. This Chapter includes a survey of current proposals to deal with information leakage. Both academic approaches and commercial solutions have been studied. The purpose of this Chapter is to summarize all proposals related to information leakage protection. This Chapter serves as a starting point for the rest of the work performed in this thesis.

- **Chapter 3** introduces the reader to the fields of steganography. The formal model of steganography is defined along the most important characteristics of steganographic algorithms. Additonally, covert channels are defined and steganalysis techniques are presented. A review of steganographic and steganalysis proposals is given.

- **Chapter 4** presents contributions of this thesis related to steganography. A steganographic method to transform C source code into text is presented. Additionally, a steganographic application for smartphones is also introduced. Both contributions could be used to evade information leakage protections mechanisms and to steal information.

- **Chapter 5** gathers the protection mechanisms that have been developed to fight against the attacks described in Chapters 4 and 6. Specifically, a set of techniques to detect the exfiltration of information through the use of trusted apps is presented. This techniques could be included in future developments of information leakage protection solutions.

- **Chapter 6** describes a method to evade current information leakage protection solutions by means of trusted applications. The proposed method is experimentally evaluated against a commercial information leakage protection solution. Additionally, the security of the proposed method (from the point of view of the malicious insider) is also analyzed.

- **Chapter 7** summarizes the work achieved during this PhD. This Chapter presents the concluding remarks regarding the areas of steganography and information leakage protection. A summarized answer to the research questions presented in Section 1.2 is

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

given. Finally, open questions left out of the work of this thesis and further research on both areas directly related to this dissertation are also presented.

- **Appendix A** describes the works published during the research period related to this PhD. This Chapter includes works done regarding this dissertation, but also works done in collaboration with other authors during this research period.

Chapter 2

Information Leakage Protection

In the city
Where I'm from
Lions in cages just for fun.
But you will passé around your cage
And wait for night to come
Wolf Gang - Lions in Cages

2.1 Introduction

In this chapter the threat of information leakage is introduced along an analysis of the current proposals to fight it. As information leakage is a threat created by insiders, proposals that address the insider threat (in a general form) have also been taken into account. The review performed in this chapter aims to check if current proposals, address the threats created by malicious insiders that steal information they have access to by means of covert channels.

The first step to address the information leakage threat is to acknowledge it. The Information security management and organizational methodologies for information security that are used to implement security mechanisms must address and acknowledge the threat of information leakage in order to recommend the necessary controls to avoid them. As these methodologies drive deployment of security mechanisms in an organization, a not proper analysis of the threat will produce pitfalls in the rest of the process.

Although information leakage is a threat of enough entity to be treated separately from other insider threats, several proposals have focused on detecting malicious insiders in general [18, 19, 20, 21, 22, 23]. These systems try to detect malicious insiders whether they try to sabotage the organization infrastructure, steal information, or perform other attacks to the organization. In fact, information theft is a threat that can be originated by malicious insiders and outsiders. In the case of malicious outsiders, they previously have to gain access to the organization infrastructure by some mean (social engineering, software vulnerabilities, etc.). Therefore, once an outsider has gained access to the organization (and overpass the security at the edge of the organization) she can be considered as an insider. Nevertheless, the characteristics of this insider are very different from the real insider. In one hand, the intentions of this insider are clearly malicious. On the other hand, she is not acquainted with the organization infrastructure and security policy. Therefore, in her attempt to perform malicious activities she will perform tasks that real insider will not perform. This kind of insiders are called masqueraders, and have been widely studied in the literature [20, 22, 24, 25, 26, 27, 28, 29, 30, 31].

Section 2.4.2 addresses current proposals in this regard.

Finally, this Chapter analyzes the current proposals that have been specifically developed to fight the information leakage threat. This includes the researchers' proposals, but also the commonly known DLP solutions offered by information security providers.

2.2 Information Leakage

An information leakage event happens when an information asset is revealed to unauthorized parties. Although this thesis is focused on malicious leakages (i.e. information theft), the purpose of this Section is to present a general view of the information leakage threat. This Section briefly describes the basic concepts related to information leakage such as the value of information assets, how they can be identified, its possible states and the leakage vectors that can be used to steal them.

2.2.1 Information as an Asset

Nowadays organizations hold huge amounts of assets in different forms and shapes, being information one of their most valuable assets due to the high cost and difficulties to produce or replace it.

Organizations usually hold two different kinds of information assets, personal information assets and business information assets. *Personal information assets* are defined as any kind of asset that includes Personable Identifiable Information (PII), which is any kind of information that could be used to uniquely identify, contact or locate a single person. Unfortunately, the definition of PII depends on countries legislations, varying from one country to another.

Business information assets can be defined any kind of information that is essential to achieve the organization objectives. These includes blueprints, source code, financial, investments plans, etc. This kind of assets are tremendously heterogeneous, as they can be in many different shapes (text file, design, presentation) and formats (PDF, DOC, DOCX, PPT, OPT, CATIA, etc.).

2.2.2 Sensitive Information Identification and Classification

Identifying sensitive information hold in an organization is the first step to protect against its leakage. This can be an extremely slow and time consuming task if it is performed manually. Automatic information identification and classification techniques allow the identification and classification of sensitive information across the organization infrastructure.

Additionally, information assets may suffer modifications due to business process or an attack by a malicious insider. In this case, identification techniques have to take into account that information may suffer this kind of changes in order to effectively identify information assets. In the worst scenario, an information asset may have been hidden using steganography inside an innocuous carrier. In this case, techniques to identify sensitive information are useless. Steganalysis techniques, which are presented in Section 3.5.1, should be used to identify such files and avoid information thefts. In this regard, information assets can be classified as *clear information assets*, which express sensitive information explicitly; *modified information assets*, which contain sensitive information, but not explicitly; and *hidden information assets*, which contain sensitive information, but inside other innocuous works (which may not be identified as sensitive).

Although not specifically designed for identifying sensitive information, the use of the following techniques has been observed in the literature and commercial products.

Keyword Matching and Pattern Recognition Techniques

In some cases, information assets may include certain words or properties that may allow their identification as sensitive documents through keyword matching (names, surnames, the word confidential, etc.). Keyword matching may be also used to search through file metadata for sensitive information.

Pattern recognition techniques are part of automaton theory and computational theory introduced by Kleene in [32]. Nowadays regular expressions are a common mechanism to identify patterns through textual files, being implemented in most programming languages.

Keywords and patterns can be used only when information assets contain explicit sensitive information (they can not be used with modified or hidden information assets). However, keywords and patterns must be carefully selected, as they may detect as sensitive documents that are not (false positives) [33].

Document Fingerprinting

When used to identify sensitive information, fingerprinting, is a technique which tries to extract signatures from documents in such a way that those signatures can be later used to identify versions, and other documents with similar contents. As the goal of this kind of fingerprinting is identifying similar documents, usually these kinds of fingerprints are also called approximate fingerprints [34]. Fingerprinting techniques may serve to detect clear information assets and modified information assets. Nevertheless, the level of malicious modifications this techniques allow has not been deeply studied.

Classical document similarity measures are based on document indexing and similarity research [35, 36]. The *Vector Space Model* proposes the use of vectors to represent documents, in such a way, that comparison of documents is performed by comparing the vectors represented by them. To compare vectors, classical concepts of vector theory such as the *inner product* or the cosine angle between vectors are used (Equation 2.1).

$$Similarity(A,B) = cos(A,B) = \frac{A \bullet B}{\mid A \parallel B \mid} = \frac{\Sigma_{i=0}^{n} W_{A_i} W_{B_i}}{\sqrt{\Sigma_{i=0}^{n} W_{A_i}^2}\sqrt{\Sigma_{i=0}^{n} W_{B_i}^2}} \qquad (2.1)$$

Where A and B are the documents which similarity is being checked and W_{X_i} is the component value i of vector (document) X. As the components of the vector (term frequency) become similar, the cosine value will approach to one, as the vector angle will approach to zero. Although is one of the classical measures of document similarity, the cosine measure is not used as it has several drawbacks [37].

Furthermore, unsupervised techniques such as Self Organizing Maps (SOM) have been also used to cluster similar documents [38]. A Self Organizing Map is built using the frequency of normalized words inside each document to be clustered.

The usage of cryptographic hash functions has also been explored to detect document similarities [34, 39, 40, 41, 42]. To be able to work properly, these techniques require a preprocessing step in which the structure and the formatting of the document is separated from the content (which will be the part compared). Hash functions based techniques usually divide the document into chunks. To find a similar document, those chunks are compared against the chunks of the possible similar document. Depending on the number of hash values that match it can be considered as co-derivative, similar, etc.

Forensic Hash Functions

Forensic hash functions (also known as fuzzy hashes) are a family of hash functions which main aim is to determine the similarity and correlation between different pieces of information [43]. Although mechanisms for similar document identification were previously proposed by Brin et al. [41], these were first addressed as a forensic tool by Kornblum [43]. Regarding information leakage protection, these kinds of techniques could be used to trace sensitive information, even after it has been modified.

Examples of forensic hash functions are *SSDep* and *SDHash*. *SSDep* uses rolling hashes and stores the 6 least significant bits of each of the hashes that are in accordance with the file context (the part of the file being hashed). To obtain similarity between two files, the edit distance between the two signatures (the amount of changes to go from one signature to the other) is measured. Kornblum proved that for small modifications of files, signature comparisons helped in identifying files as similar. When comparing two random files, the probability of having two equal signature characters is 2^{-6}. Thus, as the signatures are greater (for greater files) the probability of having similar signatures decreases. More details regarding *SSDep* can be found in [43].

SDHash is another forensic hash function proposed by Roussev [44]. In this case, for each part of the file, the 64 byte sequence with the lowest number of occurrences is hashed and stored in a bloom filter. File signatures are constructed using several bloom filters, requiring approximately 2% of the original file size. To check file similarity, Bloom filters are compared using a Hamming distance based approach. This allows to detect common 64 byte sequences (with a low probability to appear) between the two files.

In [45] an evaluation of previous forensic hash functions is performed. Tests regarding the identification of embedded objects inside other files, and block correlation between files are performed. Overall, *SDHash* performed better in all tests. In fact, the authors conclude that the capacity of *SSDeep* to detect similarities is very dependent on finding large blocks of common data between files, while SDHash is less dependent on this fact. However, in the best case (*SDHash*) the minimum amount that two files have to share to correlate them is 16 KB in the worst scenario. If information is fragmented in smaller sizes and embedded into other objects, it would not be recognized as related by these tools.

Watermarking

Watermarking techniques embed information in the document about the document itself. Watermarking techniques have been widely explored, mostly in multimedia content, due to its possibilities regarding copyright protection [46]. Nevertheless, watermarking has been explored in many other domains such as CAD Designs [47], executable code [48, 49], XML files [50], etc. Although adding watermarks is not a suitable solution for identifying sensitive documents, it could be used to monitor modified documents and to define document properties such as sensitiveness, confidentiality, etc.

An overview of the previous presented identification and classification techniques is shown in Table 2.1.

Technique	Detects variable content	Tracks changes	Modifies content
Keyword matching	No	No	No
Pattern recognition	Yes	No	No
Document fingerprinting	Yes	Yes	No
Forensic hash functions	Yes	Yes	No
Watermarking	Yes	Yes	Yes

Table 2.1: Information identification and classification techniques overview

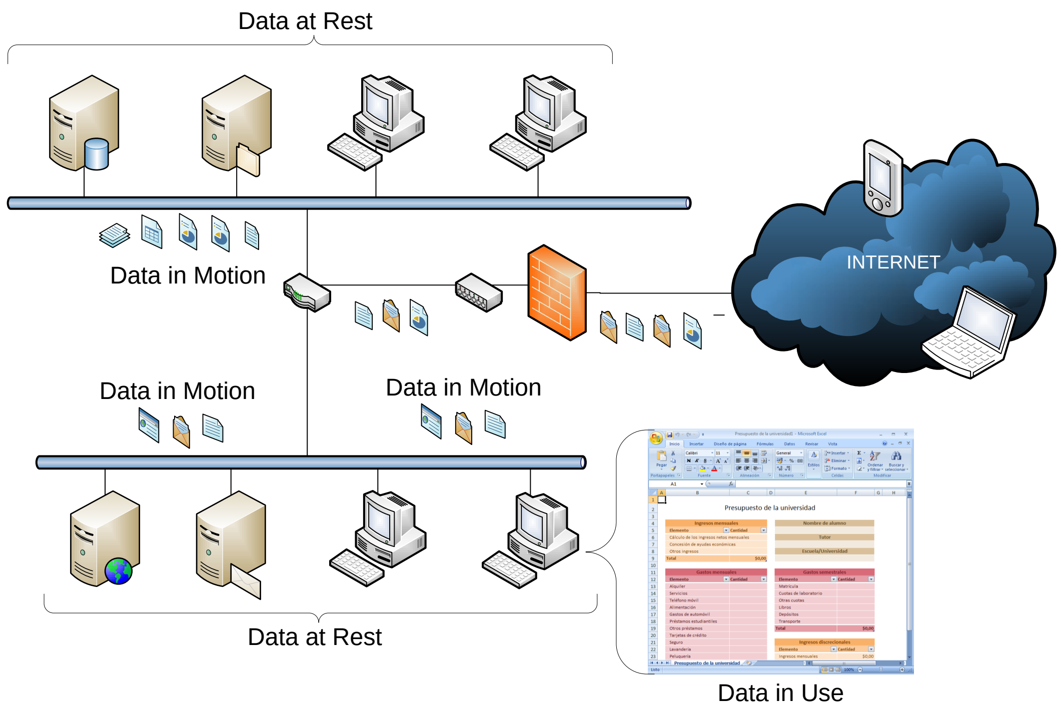


Figure 2.1: Information types within an organization

2.2.3 Information State and Organization

An organization IT infrastructure is usually composed by servers, desktops and mobile devices (laptops and smartphones) connected through the organization's network (and the different branches connected through the Internet via a Virtual Private Network). Every day information assets flow through the organization's network (information in motion), are used by employees (information in use) and are stored in each of the organization's systems (information at rest), as shown in Figure 2.1. As information can be found in each of these states and each state is treated differently by information systems, different controls and countermeasures should be implemented to avoid information leakages.

Information at Rest

Information assets that are stored, but not in use, at the organization infrastructure are considered information at rest. Information assets in this state are usually stored in hard drives, memory cards, solid state disk and any other physical support for digital data. Logically, in servers this information is usually stored in databases, file repositories, data warehouses, etc. On desktops and mobile devices (laptops, smartphones and portable drives) it is usually stored directly in the file system in form of documents (pdf, Word, Excel, OpenOffice, etc.), graphic designs (CAD, Catia, etc.), text files (source code, etc.) and binaries (images, etc.).

Information in Use

Any information asset being used at any workstation or server is considered information in use. Although information being used is in most cases also stored in a storage device, the information being used by the computer program is stored in the system's memory, creating new threats that must be addressed. Furthermore, the access and manipulation possibilities of the programs using the information must be taken into account when designing protection mechanisms to avoid information leakage. These programs can also be used to create new information assets that are stored in the system's memory before being properly saved.

Information in Motion

As business process are performed, organization members need to access information assets located in some remote locations of their own organization and share them with other team mates. Any information asset that travels through the organization network is considered information in motion. These assets, although being also stored in some other location (thus being considered as information at rest), can be sent through the network, being possible to produce an uncontrolled copy of a sensitive document (if the information reaches an unauthorized or uncontrolled device without the proper protection mechanisms). One of the challenges on controlling information in motion is to be able to control the wide number of protocols that can be used to send information such as HTTP, HTTPS, SMTP, P2P protocols, FTP, etc.

2.2.4 Information Leakage Vectors

In terms of computer science, a leakage vector can be defined as the means by which some information asset is revealed to unauthorized users, whether the leakage was produced intentionally or not. Although information assets have been historically leaked and robbed by physical means, the widespread adoption of IT has eased the way information is copied and transmitted, allowing new ways of information leakage. In the following lines a description of both kinds of leakage vectors is given, focusing on the ones produced by information systems, as they are the ones to be studied in this dissertation. These leakage vectors must be used also when techniques such as steganography are used to hide the information theft attempt.

2.2.4.1 Physical Leakage Vectors

Although nowadays most of the information inside an organization is stored electronically, the containers of this information (hard drives, USB drives, cds, etc.) are physical devices susceptible of being stolen physically. The other most important information asset container is paper, which can be stolen easily, as it is not feasible to implement restriction on organizations' infrastructure to control the flow of information in printed formats [51].

Avoiding leakages by means of these requires to implement physical security measures, that do not allow the extraction of organization's equipment. Nevertheless, to enforce the information security leaving the organization, some mechanisms such as hard drive encryption may be implanted on information systems.

2.2.4.2 Information Systems Leakage Vectors

The same mechanisms insiders use to perform their usual business task can be also used to steal information assets. Techniques that transform sensitive information with the purpose of not being detected are not considered as leakage vectors. A leakage vector specifies the way information can be disclosed from a computer system. In order to avoid leakages and information theft, protection mechanisms and countermeasures must be placed into these leakage vectors.

Leakages Through the Organization's Network

Organizations' networks are an essential part of organizations IT infrastructure, enabling collaboration and communication between employees, clients, providers, etc. An organization network can be used for inside to inside, outside to inside and inside to outside communications. Inside to inside communications cover all communications that happen within the organization network. These includes corporate mails and messaging, internal server access, etc. Although these kinds of communications usually do not increase or produce information leaks, information transferred by these mechanisms can be sent to unauthorized recipients inside the organization, producing an information leak inside the organization.

Outside to inside communication covers any communication which starting outside the organization has its destination at some point inside the organization network infrastructure. These communications include Virtual Private Networks (VPN), connections to public services such as Web, FTP, etc. and incoming mails. This kind of communications are susceptible to produce information leakages, as sensitive information can be misplaced on a public access server, as described by [52]. Furthermore, these servers can also be hacked by outsiders giving them access to information stored in them. Classical security tools such as IDS, IPS, Firewalls and Antivirus protect information systems from these kinds of external threads.

Inside to outside communication involves any connection started within the organization boundaries which destination is outside the organization. These usually include a wide range of services (protocols such as HTTP, SMTP, FTP, P2P, instant messaging, etc.) that are used by employees to communicate to the outside world. All these services can be used to steal information as an insider can post sensitive information into a social network, send a sensitive file through email or upload it to an FTP server. Furthermore, a malicious insider could distribute the sensitive information asset through a P2P network. Additionally, a malicious insider might use a steganography technique that takes advantage of a network covert channel (such as the ones described in Chapter 3) to steal information.

Leaks Through the Organization's Printer

Printing information converts an electronic file into a sensitive piece of paper. In this regard, printing should not be considered as a leakage vector itself, but is one of the main procedures that can be used to transform digital information into a physical container such as paper, allowing its physical extraction, which in most of the cases may leave few evidence.

Leaks Through Organization's Portable Media

Usage of external hard drives, USB drives, etc. has spread in recent years, as they allow easy transportation of files, backups, etc. Besides the loss of this kind of devices that can lead to information leakage, these devices can be used by insiders to steal information from the organization, saving copies of sensitive files to unsupervised portable storage systems. These removable drives can be plugged into computers usually through USB ports (but any other method such as Bluetooth, infrared, etc. can be also used).

Leaks Through Personal Mobile Devices

Along with external devices, the usage of smartphones and other personal mobile devices has increased tremendously during the last years. These devices are capable of performing complex operations and even can be compared in terms of computational power with some personal computers. Although these devices allow to improve the employee's productivity, they also create new leakage vectors. These devices are usually configured to access the organization infrastructure (even if they are personal devices), but are not configured according the security policy of the organization (for example, forbidding the use of the camera). In this way, they are a new leakage vector that should be taken into account.

	Network	**Printer**	**Port. Media**	**Mobile Device**
Allows encryption	Yes	No	Yes	Yes
Requires physical access	No	Yes	Yes	No
Leakage object	Data	Paper	Data	Data
Monitored	Yes	Yes	Yes	Partially
Lockable	Yes	Yes	Yes	Partially

Table 2.2: Summary of information systems leakage vectors

Table 2.2 shows a summary of the previous described leakage vectors. Both researchers and information security providers have spent lots of efforts in developing systems and techniques to mitigate leakages through these leakage vectors. In the next Sections, the main proposals are described and analyzed.

2.3 International Standards and Recommendations

Although this dissertation is focused on technical aspects of information security, managing information security is a complex task that involves not only technical controls but also process management and other areas of the organization. In this regard, the 27001 of the ISO/IEC standard specifies a set of recommendations, processes, methodologies and controls that any information security management systems should take into account [53]. Although every information security management system adapts to the organization needs, this standard allows to define a baseline that can be evaluated both internally and externally.

Different works have analyzed the suitability of the 27001 standard to handle insider threats. These works do not analyze the information theft threat specifically, but all the threats that an insider comprises to the security of the information systems of the organization.

In [54], Humphreys analyzes the 27001 standard regarding the threat posed by insiders (including the information theft as an insider threat). The author identifies a set of controls that are defined in the standard and should be used to address insider threats. These includes technical controls such as "Access Control" (point A.11 of the standard) and non technical controls such as "Human Resources Control" (point A.8 in the standard). However, the authors misses other controls and recommendations in the standard that should also be used to improve the awareness against insider attacks, and more specifically against information leakage attacks. Specifically, network security management controls (A.10.6 in the standard) should be used to control and handle all the network exchanged data. In the same way, exchange of information (A.10.8) and monitoring controls (A.10.10) should also be used to monitor information flow and detect possible transmissions of sensitive information outside the organization boundaries.

Nevertheless, as pointed out by other authors [55], these standards present some weakness and more importantly, do not describe clearly the controls that a security management system should implement to avoid information leakage. In fact, recommendations about the CISSP security certifications [56] warn that the insider threat is not usually taken into account when developing the information security policy. Nevertheless, no specific recommendations to protect against these insider threats are given.

Besides the ISO/IEC standards, researchers have also proposed methodologies to exclusively prevent from the insider threat. In [57] the authors propose a methodology that allows to produce a specific set of requirements that address the insider threat. Additionally, authors design two deliverable documents to provide awareness in organizations environments against the insider threat. However, the use of such methodologies is compatible with the use of information leakage prevention systems. In fact, these methodologies and standards do not provide actual implementable mechanisms to fight against information leakage (and generally insider threat), but gather a set of recommendations that in most of the cases will involve the implementation of one of the previous mentioned systems.

2.4 Malicious Insider Detection Systems

Research in malicious insiders aims to find the mechanisms, procedures and techniques to detect malicious insiders in the scope of an organization. Malicious insiders may misuse IT systems, steal information, commit sabotage or any other action harmful to the interests of the organization.

Insider detection is an information security area that has gained a lot of interest in the last years (see Section 1.1). In fact, it is a field with extensive research that has offered several proposals to deal with insiders. As part of the work required to answer the research question presented in this thesis a review of current proposals to fight malicious insiders has been performed. Although, this dissertation focuses specifically in the information leakage threat, works that deal with the insider threat (without specifying which one) have also been considered.

The first works to propose solutions to handle the insider threat range from 2001. Since then, several different proposals have been presented. The performed review has allowed to identify a set of properties that can be used to characterize each proposal. First, a temporal property has been identified. While a group proposals focuses on trying to alert from the insider risk prior any possible security breach, other proposals detect the insider incident at the time it happens. Second, presented proposals differ on the granularity of the insider threat they detect. While some proposals may detect a specific insider threat, others just detect the malicious insider as a general threat.

Analyzed proposals have been categorized using the previously presented characteristics. Proposals have been categorized using the temporal property. This organization allows to present in more detail proposals that focus on detecting the information leakage threat. This categorization is a contribution of this dissertation and has not been taken from any other author.

2.4.1 Predictive Proposals

Predictive techniques try to identify malicious insiders before they can commit any action that is harmful for the organization. Reviewed proposals in this regard include the information leakage threat as part of the malicious insider threat. That is, they do not identify the specific threat created by the insider. The output of these techniques can be used to improve the security controls of the organization, modify assigned privileges or help the human resources department during hiring processes. Schultz et al. [19] defined a set of indicators (mostly behavioural) to asses the risk posed by insiders. These, which are not clearly defined and can not be quantified objectively, included verbal behaviour, personality traits and computer usage among others. Applying a weight to each indicator, a general risk assessment could be given for each employee. The main problem of this proposal is the difficulty to generate methods to measure the used indicators. Additionally, selecting the weights for each indicator seems a very difficult task. Existing insider security events could be used to define a set of weights, but it is not clear whether those can change between organizations or not.

In [18], the authors propose a system that gathers information only from the IT infrastructure, such as file system events, memory and other IT related possible events. Based on the collected information, the system creates a “threat profile” for each user in the organization infrastructure. The granularity of the predictions given by the system enables only to differentiate between accidental and malicious threats. The final assessment is given taking into account, not only the IT related events, but also the role of the user in the organization and its level of access. Authors explain that their proposal is not currently implemented (and therefore not evaluated) because is not considered adequate and requires a better architecture description, algorithm development and integration with other security systems.

Other proposals use concepts brought from criminology such as criminal profiling. These are based on predicting the probable characteristics of a criminal offender based on the behaviours exhibited by previous offenders in the commission of a crime [58]. In [59], Nykodym et al. presented guidelines to construct insider profiles including some examples such as the saboteur and the spy. As an example, they state that spies are usually confident persons, who are comfortable in their positions and steal just for money, not for personal revenge or hate. The main problem of proposals in this regard (as well as other prediction techniques) is the difficulty to test that their assessments are right and also the lack of experimental results that back up the proposed frameworks. Table 2.3 summarizes the previously presented techniques to predictively detect malicious insiders.

	Magklaras et al. [18]	**Schultz et al.** [19]	**Nykodym et al.** [59]
Year	2001	2002	2005
System?	Yes	No	No
Methodology?	No	Yes	Yes
Detect. Granularity	Malicious or accidental threads	Mal. insider	Mal. insider profiles

Table 2.3: Summary of predictive proposals to detect malicious insiders

2.4.2 Reactive Proposals

Reactive techniques detect the malicious insider at the moment she commits the attack against the organization. These systems can be seen as a kind of intrusion detection system in the way that they detect possible security incidents that violate the organization security policy (in this case they detect internal incidents). Although some of the reviewed proposals do not differentiate between the threats created by malicious insiders, some address directly the information leakage threat. In fact, Data Leakage Protection (DLP) solutions developed by information security providers are included in this category of proposals. Nevertheless, due to its relevance in the context of this PhD, they are analyzed in a separate Section.

Schonlau et al. [20] built a simulation network in which 70 insiders (20 of them malicious) performed different actions (simulating working hours) during several days. Using the UNIX *acct* auditing tool, they collected over 15000 executed commands per each user. 5000 first commands were left intact while the rest of the commands were mixed with malicious insiders commands. Detection methods should have to be able to detect correctly those injected blocks. Schonlau dataset has been widely used by researchers studying insider detection mechanism [25, 26, 27, 28, 29, 30, 31].

Similar systems have also been proposed in Windows based systems [60]. In this case, registry entries are used as audit data to perform detection. To detect possible attacks by insiders, a weighted vote system is used. During training, each collected property of the windows registry votes (with a weight) to decide if there is an insider attack going on. If the assessment is wrong, the weights of each of the wrong votes are adjusted. As authors state, for each employee an instance of the system must be trained with that user's data. After training, if the system detects abnormal behaviour (through the weighted voting system) it will raise an alarm.

The works of Schonlau and Shavlik addressed a specific kind of insider which is called masquerader. These are outsiders who, by some mean, have gained the access to insider credentials. Although the purpose of this kind of insider is obviously malicious, they are very different to the insiders studied in this PhD. Masqueraders usually do not know the organization infrastructure and moreover the information they have access to.

As well as IDS techniques have been used to detect malicious insiders, other techniques to detect malicious attacks from outside the organization have been adapted to work with insiders. One proposal in this regard is derived from the concept of *honeypots*. A honeypot is a vulnerable information system created with the purpose of learning how intruders attack information systems. In 2003 Spitzner [21] introduced the term *honeytoken*, which describes a piece of fabricated information that seems valuable to an information system attacker (insider or outsider).

Bowen et. al [22, 24] took the concept described by Spitzner and designed a system which placed different decoy documents (honeytokens) along all the organization infrastructure. Additionally, several agents were placed along the organization to detect the usage of these documents. To be able to distinguish these from regular documents, authors used three different approaches. First, all documents were inserted with a random HMAC. Only the HMAC of fabricated documents was stored, so when that document was being opened or transmitted it raised an alarm. Additionally, documents without the random HMAC also raised an alarm. Second, decoy documents included information about bogus accounts that were constantly monitored. Finally, decoy documents loaded remote information (such as images) from a specific server. Therefore, when an insider opened such a document, a connection from the insider to the alert server would be generated. Although this system can detect some attempts to steal information, the main goal of the system is to detect insiders that go outside of their scope of action. In fact, this system is not able to detect insiders who steal documents they usually have access to (but are sensitive too).

While this approach is specially suitable for masquereders, who don't have knowledge of the organization valuable information, malicious insiders who have already knowledge that such systems are operating inside the organization will be more suspicious of finding valuable information unprotected, reducing its efficacy.

Monitoring network usage can serve to detect sensitive files leaking or raising alerts of suspicious behaviours by some employees. The most complete proposal in this direction was presented by The MITRE Corporation. They developed a system called ELICIT [61, 23], that takes into account network related events to detect insiders. ELICIT's goal is to detect employees who perform actions inside their privileges, but outside their actual scope, violating the *need-to-know* policy [62]. ELICIT translates user generated network traffic into information-use-events, that describe document level operations, such as searching, reading, writing, deleting, printing, etc. These events and a social network based on contextual information were used to built *suspicion detectors*, which detected specific activities that might be suspicious for insiders. Detectors were created by humans experts based on previous real insider cases. Overall, authors defined 76 detectors which feed a Bayesian Network, weighted by humans. The Bayesian Network produced a value which assessed the probability of user activity being malicious. ELICIT was though only for detecting users who violated *need-to-know*. As a consequence, neither users misusing their usual accessible files and the usage of steganography would be detected (as in other previously analyzed proposals). Table 2.4 summarizes the main previously presented techniques to reactively detect malicious insiders.

	Schonlau et al. [20]	**Caputo et al.** [61]	**Bowen et al.** [22]
Year	2001	2008	2009
System?	Yes	Yes	Yes
Input	Unix commands	Social network and network related events	Decoy documents and related events
Detect. Granularity	Masqueraders	*Need-to-know* violations	*Need-to-know* violations

Table 2.4: Summary of main reactive proposals to detect malicious insiders

2.4.3 Information Leakage Detection Proposals

Besides studying the malicious insider threat as a whole, researchers have also addressed specific threats imposed by insiders. This Section analyzes those related to the information leakage thread specifically. Works that focus on other insider threats are not described nor analyzed as they are outside the scope of this PhD.

2.4.3.1 Research Community Proposals

Although, the proposals that analyze the malicious insider threats acknowledge the threat of information leakage, the amount of research addressing it specifically is nowadays small.

In [63], the authors propose an architecture (called SSID) to implement several steganalysis techniques to detect steganographic content transmitted through network protocols in real time. The proposed architecture is divided into three layers. The first layer is in charge of extracting the network traffic and classifying it depending on the application used to generate it (network protocol). The second layer extracts content from the network packets according to a content database. This layer also checks if some of the extracted content matches the critical repository (sensitive information) database. Finally, extracted content is also passed through a steganalysis process. If hidden content is found, this layer tries to extract if from the cover and compare it against the critical repository database. The approach is tested against video covert channels, but authors acknowledge the limitations on the scalability of their system when targeting multiple steganographic algorithms.

Additionally, [52] 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 in the organization security policy). Any document not previously vetted is filtered by a gateway that has direct communication with the warden. All documents that are not allowed to leave the organization are divided into chunks of 1024 bytes, calculating and storing its hash value. In order to decided if a document is allowed to leave the organization, the network stream is also divided into chunks of 1024 and its hash is compared with the critical data hashes repository. To avoid information leakage through a compromised web server, authors propose to sanitize HTTP connections, normalizing headers of all HTTP messages sent by the web server. This sanitization which disables covert channels through the HTTP protocol is performed at the gateway. Nevertheless, the content of the HTTP messages is not sanitized, as it is vetted by the warden. Although steganalysis algorithms are not considered for the warden, they could also be implemented as part of the vetting process.

Multilevel security models allow the classification of information into different levels in such a way that only users in possession of the necessary clearance level can get access. The Bell-LaPadula model [64], which was originally designed for military environments, establishes a security level to every subject and document. The BLP model establishes two mandatory premises for access control. First, no process may read information from a higher security level. Second, no process may write information to a lower security level. Taking this into account, if a process is able to read a sensitive information file, it means that the subject has clearance to read sensitive information files. In such a case, and following the BLP model, she would not be able to downgrade the security level of the file using that process. This, along with mechanisms to avoid extraction of sensitive files could be used to stop information leakages.

Although models such as BLP can serve to fight against information leakage, they also create a series of problems that make its use in organizations infeasible [65]. First, considering that the security level of a process is upgraded according to the files it accesses and the security level of the subject, if a sensitive file is opened, the next opened files will have to be written at the same level, independently of the level they came from. This leads to an overall increase of the security level of all files that are opened with that process. Therefore, besides stopping potential data leaks, this model can lead to employees being stopped from doing their jobs. Additionally, in order to work properly, applications and systems have to pass through major (and often costly) modifications [66].

Process colouring is a technique devised to detect and trace the propagation of worms across a system [67]. Each process susceptible of being exploited is uniquely coloured. Process interaction (read, write, etc.) with other objects such as files or processes results in the propagation of the color. In this way, if a worm infects a process, all interactions made by the infected process can be traced back to the original source. Although the presented technique is focused on processes, it could be adapted to track sensitive information. Assume that, apart from assigning colors to processes, each sensitive file is given a unique color. Using process colouring, each process will get coloured by the colors of the sensitive files they interact with (colors can be accumulated). If a coloured process writes into another file, that file will also gain the process colors. To avoid leaks of sensitive information, the security policy could define a set of colors that cannot be transmitted outside the organization by any means.

The usage of this kind of technique involves certain drawbacks. First, process colouring, as presented by Jiang et al., is a technique that works over processes. A color is assigned to each process, but documents and other operating system elements are colorless until a process interacts with them. In order to track sensitive information, additional colors should be initially assigned to sensitive files. Besides, even after assigning colors to sensitive files it is unclear if once an application is closed, colors should remain on its processes. For example, an employee could use a text editor to modify a sensitive file. Later on, the same user could open again the same application and use it to create a personal document. Additionally, depending on the amount of sensitive information to manage, the numbers of colors can saturate the system, hindering the sensitive information tracing process [68].

2.4.3.2 Commercial Solutions

The information security industry has acknowledged the importance of the information leakage threat by developing specific solutions to fight it. Data Leakage Protection (DLP) solutions are commercial products composed by a set of components (sensors, information discovery agents, content filtering agents, etc.) which are deployed in the organization infrastructure to avoid information leakages. DLP solutions are not only used to avoid information leakage originated by malicious insiders, but also accidental leakages that may be produced due to insider mistakes. Table 2.5 shows a list of the top solutions, as classified by Gartner [2]. Additionally, there exists an open source DLP solution that has been recently released and at the time of the writing of this document and is being actively maintained[1].

DLP solutions try to cover the information leakage problem as a whole, not only detecting possible leakages but also providing employee education, forensic evidence, management systems and response procedures. These solutions first identify the sensitive information an organization holds. Once configured, they monitor leakage vectors to avoid the transmission of previously identified sensitive information. In this regard, each time a piece of data is placed into a leakage vector, it is checked against the sensitive information database. This includes keyword matching, pattern recognition and fingerprinting checks. If any of them marks the information as sensitive, the DLP solution will block the information transmission. Additionally, some of them allow to restrict the set of applications an insider can execute, only allowing to executed applications that are trusted by the organization. Furthermore, these solutions also

[1] http://code.google.com/p/opendlp/

Vendor	Solution Name
Websense	Data Security Suite
Trend Micro	Leak Proof 5.0
RSA	Data Loss Prevention Suite
Symantec	Data Loss Prevention 9
McAfee	Total Protection
Palisade Systems	Palisade DLP
Verdasys	Digital Guardian 5
Computer Asociates	CA DLP v6
Fidelis Security Systems	Fidelis XPS
GTB Technologies	GTB Data Loss Platform
Safend	Safend Data Protection Suite
Trustwave	Trustwave DLP

Table 2.5: List of most important DLP vendors by Gartner [2]

usually include information discovery appliances which are used to analyze and automatically detect sensitive information in organization's servers and workstations.

Besides the increase of concern and research efforts by IT security industry, there are still issues to be solved regarding these new kinds of solutions. Reducing the number of false positives produced by actual DLP solutions is a challenging task for security administrators, as they produce a high overhead in the security analyst due to the necessary manual review of false positive cases [69]. Additionally, none of these solutions takes into account the usage of steganography or other technique to transform information to hide information prior to extracting it from a organization. Therefore, they do not provide the necessary protection mechanisms against some kind of malicious insiders. A review of current evasion techniques against these kinds of solutions is described in Section 6.2.

2.5 Drawbacks of Current Proposals

This Chapter has presented the problem of information leakage and analyzed the current proposals (both academic and commercial) to fight it. Some of those proposals address the information leakage threat as part of the malicious insider threat, while others address it specifically. As reviewed, current approaches to mitigate information theft (or the malicious insider threat) do not take into account specific challenges and techniques that may use malicious insiders to steal information. In the following lines the detected drawbacks of current research in information leakage protection are summarized:

- Most of the current proposals do not consider the possibility of using techniques such as steganography to hide sensitive information. These techniques allow to hide information in such a way that proposed systems would not be able detect sensitive files being stolen. Current identification techniques for sensitive information (Section 2.2.2) do not allow to identify steganographic objects as sensitive, even when they include sensitive information. Current forensic hash functions do not allow to identify embedding of small data fragments as the ones generated by steganographic algorithms during the embedding procedure. While forensic hash functions require at least 16 KB of common data between compared files, steganographic algorithms divide the secret to embed in much smaller bit streams. The use of these techniques has been acknowledged in [70].

- Proposals considering the usage of steganography to steal information only address this problem partially [52]. As a consequence, organizations may have a false sense of security that may ease the information theft.

- Current proposals classify applications used during business activities as trusted or not because their usefulness to perform business activities, not the actual usage given by the employee. This means that some applications that could be used for malicious purposes are not considered as such, and therefore are not properly controlled. As an example, an application such as Excel could be used to bypass current DLP solutions and extract information from an organization. This is described in more detail in Chapter 6.

- Advanced Mobile devices such as smartphones are not controlled by current proposals. Although the threat to information confidentiality of this kind of devices seems fairly evident, there is no proposal to enforce the proper usage of these devices in an organization environment. In fact, as stated in [71], only 34% of surveyed organizations have a mobile device security strategy defined. Additionally, the restrictions imposed by their operating systems makes difficult to install the same kind of agents that are installed in desktop workstations.

These drawbacks show that current proposals to fight against information leakage (or malicioius insiders) are not ready to stop some attacks committed by malicious insiders. The following Chapters of this dissertation, show the existence of these drawbacks, but also propose mechanisms to mitigate them.

Chapter 3

Steganography and Covert Channels

Hold, Hold your talk now And let them all Listen to your silence
The Ting Tings - Silence

3.1 Introduction

Steganography is the art and science that tries to hide the existence of messages [72]. Steganography shall not be confused with cryptography, which main aim is to conceal the content of the message by performing different transformations on it, so only authorized persons can read it. The use of steganography is specially suitable when there is a necessity to hide that the communication is taking place. In this regard, cryptographic techniques are only able to hide the content of the messages transmitted, not its existence.

Steganographic techniques were first mentioned by *Herodotus* [73]. He described the story of *Demaratus*, an exiled Spartan who wanted to warn them about a Persian invasion led by *Xerxes*. *Demaratus* sent a warning message to Sparta written on a wooden table covered by wax. The message was able to pass all the guards and controls, and to safely arrive to Sparta, enabling them to better prepare the defence.

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 [74], 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 data leakage prevention 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.

3.2 Principles of Steganography

The first informal model for steganography was described by Simmons [75] 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 she will put them into isolation cells and the escape will be frustrated. In this scenario, Alice and Bob would not be able to use directly cryptography as messages between them will raise suspicions on Wendy. In order to achieve their goal, Alice and Bob should hide their messages into innocuous ones, so Wendy will see meaningless messages between prisoners.

However, if Wendy is aware of the existence of steganography she may be able to detect the presence of hidden messages, destroy the covert channel between Alice and Bob or even further impersonate them. On one hand, if Wendy just checks the messages and forwards them to its recipient, then Wendy is a *passive warden*. In this case, Wendy only verifies if the message contains hidden content 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 message contents trying to perturb the hidden information. In this case, Wendy is an *active warden*. Finally, if Wendy is able to embed information into the message in the same way Alice and Bob do, she may be able to impersonate both of them. In this case, Wendy is a *malicious warden*. These scenarios must been considered when designing steganographic algorithms, so its quality can be measured by means of the difficulty to detect its content and the possibility that hidden information is not lost even if the transmitted object suffers some modifications.

3.2.1 Definitions

The prisoners problem depicted by Simmons shows the different elements that take part in a steganographic communication. These elements are formally described next (Figure 3.1).

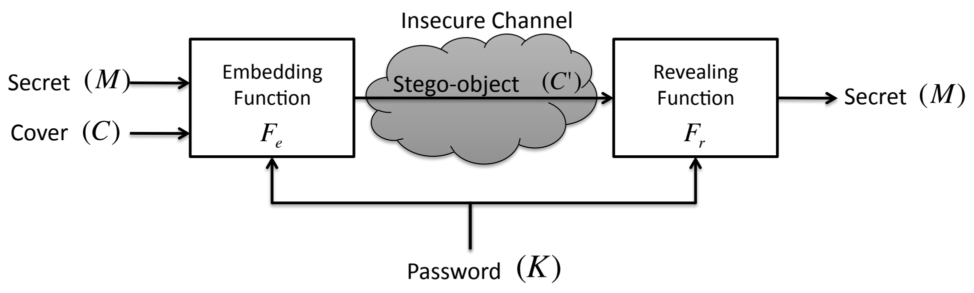


Figure 3.1: Elements of a steganographic system

Let M be the *secret message* to be covertly sent. Let C be the innocuous message (*cover*) used to embed the secret. Let K_S be a previously shared *password* known by both the sender and the recipient of the message. The *embedding function*, $F_e(M, C, K_S) = C'$, takes as input the cover C, the secret message M and the password K_S and outputs a *stego-object* C' which looks like the original cover and should be statistically indistinguishable from the set of all possible covers $S_C = C_1, C_2, \ldots, C_n$. As part of the embedding process, F_e usually encrypts the secret message before embedding it into the cover. The stego-object is sent to its recipient through an insecure channel that may be controlled by an adversary (Wendy). On reception, the *revealing function*, $F_r(C', K_S)$, is applied. F_r takes as input the stego-object C' and the password K_S and outputs the secret message M. The cover used to transmit the message is usually discarded, as its only purpose is to transmit the message covertly.

Besides the previously presented model, there exists a slightly modified model of steganographic system that instead of requiring a cover, automatically generate them for each secret to hide. These systems are called steganography systems by cover synthesis [46]. These systems do not take a cover as an input but a definition of a model of acceptable covers. Usually, these kind of systems are able to generate covers that can hold up any amount of information, while standard steganographic systems usually have finite capacity.

3.2.2 Desired properties for Steganographic Systems

According to the prisoners problem described by Simmons, to establish a proper covert communication channel the steganographic algorithm used by Alice and Bob should posses some desired properties. These are capacity, security and robustness.

3.2.2.1 Capacity

The capacity of a steganographic system defines the amount of practical information that can be embedded into a cover. That is, the amount of the secret that can be hold by the cover. The capacity of a steganographic system can be measured in terms of the ratio of bits of secret that can be embedded per bit of cover. The capacity (or ratio) of a steganographic algorithm can be defined then as Equation 3.1.

$$R(stego_{system}) = \frac{bits_{secret}}{bits_{cover}} \tag{3.1}$$

In the case of steganographic systems by cover synthesis, $bits_{cover}$ shall be replaced by $bits_{stego-object}$.

3.2.2.2 Security

All security requirements for cryptographic systems should also be considered for steganographic systems. This means that the security of the steganographic algorithm should not rely itself on the algorithm, which should be public, but on the secrecy of the key. This restriction does not apply only to the difficulty of recovering the embedded content without the key. In steganography, it should not be possible to distinguish a clean cover from a stego-object if the key is unknown. If it is possible to distinguish a cover from a stego-object, the purpose of steganography has been defeated.

For this to happen, the modifications performed to the cover in order to embed the secret should not modify the statistical properties (or model) of the set of possible covers. The area that studies the security of steganographic systems is known as steganalysis. This area is described in more detail in Section 3.5.1.

3.2.2.3 Robustness

The purpose of robustness in a steganographic algorithm is to be able to resist attacks from active wardens. If a modification of a cover file has generated a valid cover that is supposedly indistinguishable from a innocuous cover, the warden can introduce few modifications to that cover creating also an undistinguishable cover. If the algorithm is not robust, the embedded information will be lost. At least, and to avoid the malicious warden (who impersonates Alice or Bob) the algorithm should be able to detect modifications to the transmitted data.

Although this property is considered as important for steganographic algorithms, the literature and research in steganography has underestimated it, as other authors acknowledge [46].

3.3 Covert Channels

A covert channel can be defined as an already existent communication channel that allows to inadvertently transmit information between two processes outside its original purpose [76]. Although the covert channel is supposed to transmit information in such a way that its transmission is not detectable, it is possible to detect the existence of those covert channels. According to [77], there exist several classifications for covert channels regarding its characteristics. Regarding the usage of storage for creating the covert channel, they can be classified in timing or storage covert channels.

A *timing covert channel* can be defined as any channel that allows covert communication by modulating the time between messages transmitted and received [78]. A hard drive could modify the time it takes to read a block of data to transmit sensitive information to a process installed on the host machine. In the same way, a maliciously modified keyboard could modify its response time in such a way it transmits hidden information (i.e. keys pressed) to a malicious operating system. Network protocols such as TCP and IP [79, 80, 81] can also be used to encode secret messages. Depending whether a packet is received or not in a specific time interval, a different bit value is transmitted. A generalization on the steganographic possibilities of ordered channels was presented in [82].

A *storage covert channel* is any covert channel that involves the direct or indirect writing of information into a storage device by a process. In this regard, most steganographic algorithms can be used to created storage covert channels (as they produce stego-objects that will be later sent through the network or read by another process). In fact, covert channels created by steganographic applications allow transmission of sensitive information without leaving explicit traces that the sensitive information has been transmitted. These kind of technologies can be used by malicious insiders to steal information from a supposedly protected organization rendering the insider protection mechanisms useless [70]. The easy access to these kind of applications (hundreds of them are available on the Internet) allows non skilled malicious insiders to use these methods to steal information, extending the possible base of suspects.

Although, eliminating the existence of covert channels is infeasible, their capacity can be limited by normalizing or restricting the changes that can be introduced by the covert channel. In this regard, normalization techniques and active warden techniques (that may include for example noise addition to timing protocols) can be used for such a purpose. These are addressed in Section 3.5.4.

3.4 Steganography in Digital Media

Due to the great interest on information hiding techniques to protect intellectual property (mostly granted by the digital content industry), steganography in images, video and audio has been widely explored by the research community [72]. Fisk et al. [83] defined the concepts of structured and unstructured carriers. A carrier specifies the feature 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, PDF, network protocols, etc.), while unstructured carriers do not have a well defined structure (images, video, natural language, etc.).

Although this definition may be similar than the definition of cover, it is necessary to distinguish the concepts of cover and carrier. The cover is the object where the information is hidden, while the carrier is the specific feature of the cover used to hide information. In fact, a specific cover can include several kinds of carriers. As an example, within a HTML document (the cover), information can be hidden using HTML tags (structured carrier) or the natural language text included in the document (unstructured carrier).

The goal of this Section is to show the great amount of carriers and mechanisms that enable the use of steganography. Therefore, there is not a deep analysis of each of the presented techniques. However, all of them could be used (in the appropriate context) by a malicious insider to extract information from an organization.

3.4.1 Steganography in Unstructured Carriers

Least Significant Bit (LSB) techniques are among the best known steganographic techniques for images. These are based on hiding information into the least significant bits of covers [84]. 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 usually provides high capacity. A comprehensive description of several image steganography techniques can be found in [85].

An example of steganographic system based on images that can be used for information leakage is presented in [86]. The authors 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, ACC, 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 in 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 [87]. Audio steganography can be performed also in compressed audio files like MP3s [88].

Besides image and audio, another widely used way to represent information is text. The authors of [89], 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. As an example, [90] proposes to take advantage of the language used on short text messages to hide secret messages.

3.4.2 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. identified some methods to embed hidden information on TCP, UDP, ICMP or IP header fields [83]. 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 [91].

Games can also be considered structured carriers, as they follow a fixed set of rules. The usage of games to create covert communications is extremely suitable due to the difficulty of modelling the players behaviour and the fact that it is difficult at first to suspect of a game as a covert channel. A general methodology to use games as a covert channel was described in [92] (with an implementation for the game of Go).

Document formats such as Microsoft Word [93] and PDF files [94, 95] can also be used to hide information. In the former article, the 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 inside imperceptible changes on character, word, and line spacings.

3.4.2.1 Cover generation schemes

Most steganographic systems embed the secrets to hide in redundant and not semantically significant parts of the covers. However, there exists other kind of steganographic systems that directly generate the cover with the secret embedded (steganographic systems by cover synthesis). The embedding function of these systems uses a set of rules to generate the stego-object in such a way that it is similar to the set of covers. The most relevant proposals in this regard are in the area of text steganography. These are reviewed in Section 4.2.1.

3.5 Fighting Against the Usage of Steganography

The prisoners problem depicted by Simmons can be used to extract the main mechanisms and techniques to fight against the usage of steganography. Specifically, most of these techniques can be extracted from the warden behaviour of Simmons' scenario. This is, she can try to detect the presence of the covert channel (acting as a passive warden) or destroy it (acting as an active warden).

3.5.1 Steganalysis

Steganalysis is the area of steganography which analyzes the security of steganographic algorithms. It may be possible to think that the goal of steganalysis techniques is also to extract the hidden information from stego-objects. Nevertheless, as previously mentioned, that is not the focus of steganalysis. Steganalysis studies methods to detect whether an object carries or not hidden information (as the passive warden in the Simmons' model). Therefore, in order to perform steganalysis over digital objects it is required to control the medium used to transmit those objects [75].

A Steganographic algorithm is considered secure if it is not possible to distinguish stego-objects (generated by the steganographic algorithm) from clean objects (covers with no hidden information) with a higher probability than a random guessing [96]. Once hidden information has been detected, the purpose of steganography has been defeated, as in the warden case, so the algorithm is considered broken and insecure (its usage to communicate covertly would be detectable) [46].

Depending whether the steganalysis focuses on one steganographic algorithm or just tries to detect hidden information, steganalysis techniques can be classified as targeted or blind. Targeted steganalysis aims to distinguish clean objects from stego-objects created with one specific algorithm while blind steganalysis tries to distinguish if a cover hides information independently of the algorithm used to embed the information.

3.5.2 Definitions

The application of steganalysis can be described as a decision problem, in which given a specific object it is required to state if it hides a secret message or not. In the case of steganalysis, the steganalysist is prone to two different kinds of errors:

- A steganalist may decide that a certain object does not hold secret information when the object actually holds that secret information. This kind of error is called *false negative* (Table 3.1). The probability of committing such an error is called the *missing probability* β [97].

- A steganalysist may decide that a certain object holds secret information when analyzing an object that does not contain any hidden information. This kind of error is called *false positive* (Table 3.1). The probability of committing such error is called *false positive probability* α [97].

	Classified as innocuous	**Classified as stego-image**
Innocuous	True Negative	False Positive
Stego-image	False Negative	True Positive

Table 3.1: Confusion matrix for classification decisions when performing steganalysis

Taking this into account, the *detection probability* can be defined as the probability of detecting hidden information inside objects that really hold hidden information and its value is $1 - \beta$ and it is. The goal of steganalysis techniques is to reduce both α and β, to be able to detect as much stego-objects as possible, but without classifying innocuous objects as steganographic ones.

However, steganalysis techniques usually do not output a binary decision, they output a value that measures some feature in which the steganalysis is based (such as the used capacity, amount of secret bits, etc.). In these cases, a decision threshold must be defined to establish which results will be considered as stego-objects and which not. Obviously, variations of this threshold will have a repercussion on α and β. If such is the case, ROC (Receiver operating characteristic) curves can be used to adjust the steganalsysis decision (Figure 3.2). These curves are used to compare the possible values of the true positive rate and the false positive rate, depending on the decision system configuration.

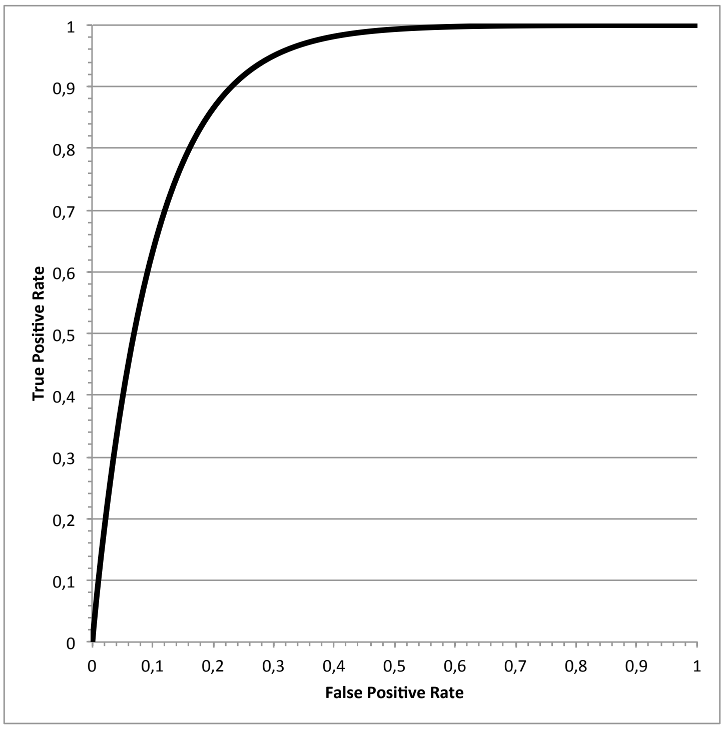


Figure 3.2: Example of ROC curve

The True Positive Rate (TPR) is defined as the amount of detected true positives among the total amount of positives (Equation 3.2).

$$True\ Positive\ Rate = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3.2}$$

The False Negative Rate (FNR) is defined as the amount of false negatives among the total amount of positives (Equation 3.3).

$$False\ Negative\ Rate = \frac{False\ Negatives}{True\ Positives + False\ Negatives} \tag{3.3}$$

As a point in the ROC curve is selected (a threshold or configuration for the decision system), TPR and FNR change, changing also α and β. Although the ROC curve can be used to compare the behaviour of the decision system depending on its settings (threshold), the optimum settings highly depend on the problem domain. Some problems may require to optimize the TPR while others require a trade-off between the TPR and FNR.

3.5.3 Steganalysis in Digital Media

Blind and targeted steganalysis techniques have been greatly studied in digital images [98]. The great amount of algorithms available for image steganography has also generated a lot of works in image steganalysis. Regarding LSB steganography, there exist several proposals that address its detection. In [99], Fridrich et al. propose the usage of a technique called RS Analysis. This technique allows to detect, not only the presence of hidden information, but also its amount. Based on a similar principle, authors of [100] present another method to measure the amount of hidden information. This method is named Simple Pair Analysis (SPA). Both methods, RS Analysis and SPA are explained in more detail in Section 4.3.3.1.

Another kind of image steganalysis is performed by Bell et al. in [101]. In this case, instead of performing statistical analysis of image sets, authors use the signatures generated by steganographic software to detect stego-objects. As authors point out, steganographic algorithms implementations include a series of errors that allow to detect the presence of hidden information very easily. The main error identified by the authors is that most steganographic software creates a new file where the cover data is copied before embedding. Instead of incorporating the cover file metadata, all stego-objects have the same metadata included by the steganographic application. This allows to easily create a signature of the metadata inserted by the steganographic application. Authors obtained a 100 % of True Positive Rate with programs such as Invisible Secrets [102], Outguess [103] and STools [104]. The True Positive Rate for MP3Stego [88] (not an image steganographic program, but also vulnerable to this attack) was 90%.

Audio steganalysis has also used both approaches (targeted and blind). In [105], authors identify audio quality metrics as a statistically distinguishable feature between stego-objects and cover audio files. Using a genetic algorithm, they were able to build a distinguisher that told apart up to 80 % of stego-audio files. Targeted approaches such as [106] have also been proposed, enabling in this case, to detect hidden information through the tool *MP3Stego*.

The security of text steganography has been addressed by studying the statistical properties of normal texts. Although tools such as NICETEXT allow to generate texts with sense, they usually do not follow all statistical properties of natural language texts, enabling a fast and reliable detection, as the presented in [107].

Although, previously presented techniques enable to detect different varieties of hidden content inside different covers, most of those works are based on statistical models constructed using a finite set of samples. Making such generalizations from a finite set is always a risky decisions. The selection of the finite set (for example the set of images where to hide information) may have very important implications on the result of the decision system. This means, that even after testing a given detector, it is not possible to ensure that its detection rate, etc. will hold for any kind of object that passes through it. However, in most of the cases, the usage of finite sets is the only way to provide some kind of detector for hidden information.

3.5.4 Active Warden Techniques

Active warden techniques take advantage of the low robustness of steganographic systems (See Section 3.2.2.3). These techniques try to limit the covert channel bandwidth by normalizing it or overwriting the possible hidden information. The main issue of this 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 allowed channel unusable. The most common active warden technique is to hide random information into the transmitted message in such a way that there will be a high probability of overwriting the original hidden message.

The process of eliminating the covert channel is also called steganographic sanitization [70]. In 1997 Petitcolas released *Stirmark* [108], a tool for testing watermarking algorithms robustness. Although *Stirmark* is a watermarking evaluation tool, most of the transformations done by the software could be used as active warden attacks to steganographic systems. In fact, any tool used to test the robustness of watermarking algorithms could be used to delete steganographic content from transmitted objects. In the same context in [70], the authors sanitized image data by overwriting all possible hiding areas of images (LSB, DCT, and DWT). Their approach showed no visual impact on the sanitized images. The little process required for the sanitization process (average of 175 milliseconds per image), made it suitable for use on networks.

Nevertheless, as previously mentioned, modifications performed to transmitted objects are limited in such a way that the object still maintains its semantic value. In [83], the authors proposed the term Minimal Requisite Fidelity (MRF) to delete hidden information from structured carriers such as TCP. MRF measures the degree of fidelity that is both acceptable for the end user and the communication. On structured carriers MRF can be quantified, as changes on the carrier structure may preclude the comunication (in case of network protocols) or do not allow the document interpretation (in case of html documents or other structured documents). On images and videos (unstructured), the MRF has to be calculated through human perception. In their work, authors identify features of TCP and IP packets that may lead to covert communications such as the 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.

As opposite to stegangalysis techniques (passive warden), active wardens can actively eliminate possible covert channels, thus stopping possible data leakages through steganographic algorithms. In fact, they can be applied to any file independently of being a carrier or not. Nevertheless, parameters such as scalability, transparency, efficacy and efficiency have to be taken into account when designing and implementing such systems in an organization environment.

3.5.5 Computer Forensics

Outside the steganography field, Computer forensics are also used to fight and detect usage of steganography. In this regard, researchers [109] have identified main forensic artefacts of steganographic applications. These artefacts along forensic software such as Encase[1] or The Coroner's Toolkit[2] may allow the detection of steganography applications providing a way to alert and even take countermeasures such as forbidding the execution of those applications.

3.6 Summary

Steganography allows to hide information into innocuous covers to establish covert communication channels. During the last years, there has been a great interest in developing steganographic techniques. In fact, as previously described, there has been extensive research in developing steganographic algorithms for a very different variety of covers, going from the classical images and audio covers to more “exotic” ones such as games, network packets, etc.

However, steganalysis and active warden techniques are developed at the same time steganography techniques are proposed. This, fosters the security of steganograhic systems, but also can be used to avoid and detect its usage in malicious scenarios. In this regard,

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

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

Chapter 5 describes a steganlysis technique that could be used to fight against the usage of steganography (specifically applied to steganography in executable code).

Chapter 4

Attacks on ILP Systems Using Steganography

I want to break free
I want to break free
From your lies
You're so self satisfied
I don't need you
Queen - I Want to Break Free

4.1 Introduction

The usage of steganography in organization environments creates a risk to the confidentiality of sensitive information. Using steganography, this information could be hidden into innocuous covers and extracted from the organization premises.

In fact, the usage of steganography recently reached public media coverage when a group of spies from Russia were uncovered [110]. As the FBI report describes [111], the spies, who infiltrated into 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 pre-shared key.

In this Chapter some steganographic techniques that could be used to bypass current information leakage protection mechanisms are presented. Specifically, a steganographic scheme to hide C source code and a mobile steganographic application are presented.

4.2 Steganography of source code in text

In order to avoid information leakage, one of the first policies implemented by organization is to restrict the applications that can be executed on an insider workstation [69]. These is done by defining a set of trusted applications that can be executed on the computer system. Any other application not granted by the system administrator (or the DLP agent) can not be executed or will raise an alarm. In this way, steganographic or cryptographic applications can not be used to hide sensible information assets. Nevertheless, there exist some work environments in which these restrictions are not feasible. As an example, programmers and software designers require more flexibility, as they require to test the programs they are creating. Additionally, the ability to create programs and execute them without restrictions offers more opportunities to leak information without being noticed.

In this Section, a steganographic system to transform source code into text is presented. An instantiation of this model could be used by malicious insiders to transform high valuable source code into text to extract it from the organization. This program takes advantage of the reduced restrictions in software designers and programmers workstations. Transformed text can be exfiltrated, as it is not recognized as a valuable asset by the organization data leakage protection systems. An example of such attack to extract information is the one that affected Google and other major software companies in the US, which was named Operation Aurora [112]. In this, compromised computers downloaded images hiding malicious programs, which looked for source code to send it outside the organization.

The software that implements this steganographic system for the C programming language (*Csteg*) has been uploaded to SourceForge.net [1]. *Csteg* has been developed under *GPLv2* license, so its source code can be modified and improved by any interested person.

4.2.1 Related Work

Although several proposals modify text structure to embed information, the most relevant proposals for text steganography have been based on the concept of mimic functions [113]. Using mimic functions, NICETEXT [114] is able to transform a secret message M into a seemingly innocuous text T which contains sentences in natural language.

Other text steganography systems based on context-free grammars have been developed in the past: *Spammimic* [115] can convert a short text message into an email Spam message. In 2001 Marttila [116] conceived a system to hide source code inside a text generated by a context-free grammar, but limited its use to only one cryptographic algorithm. Martilla's *c2txt2c* is very limited as it was not able to hide another cryptographic algorithm except for Blowfish [117]. Although this system is not a really steganographic system it is based on the same principles an could be used to create a steganographic system. In fact, this system has been the origin of a steganographic system to hide source code presented in this thesis (See Section 4.2).

4.2.2 General Model

The presented steganographic system is not based in the standard cover replacement mechanisms of steganogrpahy but on cover synthesis mechanisms such as the one presented by Wayner [118]. This is, it does not modify a cover to embed the message, but uses a context-free grammar to generate stego-objects (texts in this case) based on a certain input. This approach allows hiding files of any size. Although this approach uses context-free grammars, it is not based on the work of Wayner. Instead, the proposed model is based on the approach by Martilla [116]. In this, a grammar describing the C programming language is used to transform source code into text.

Using context-free grammars to produce stego-text has its drawbacks. As the information to be hidden increases, producing meaningful texts is harder. Martillas work produced a very short amount of texts and only accepted C source code as input. Stego-texts generated by the proposed system depend on the introduced parameters (the same secret does not always

[1] http://sourceforge.net/projects/csteganography

generates the same stego-text). Additionally, the proposed scheme allows to hide any kind of source code that can be described through a context-free grammar. As in Martilla's work, the usage of this steganographic system is constrained to hide source code and can not hide any other kind of file.

In this case, the steganographic system definition is not as described in Figure 3.1. The proposed steganographic system requires three elements as input: A grammar description (G), a plain text to use as cover (C) and a secret (S) which is a source code file (Figure 4.1):

- The *grammar description* G, describes grammar of the programming language to hide.
- The *cover-text* C, is the text that will be used as base to generate the stego-texts. In order to produce meaningful stego-texts, the content of the text file should have sense and meaning. Additionally, a cover-text can be divided in fragments $C = \{f_1, f_2, \ldots, f_n\}$, which are parts of the text that are different each other. Fragments can be selected based on grammatical rules (words, sentences, paragraphs, etc.) or by other mechanisms.
- The *secret* S, is the information to be hidden. It must be inside the language generated by G. That is $S \in L(G)$.

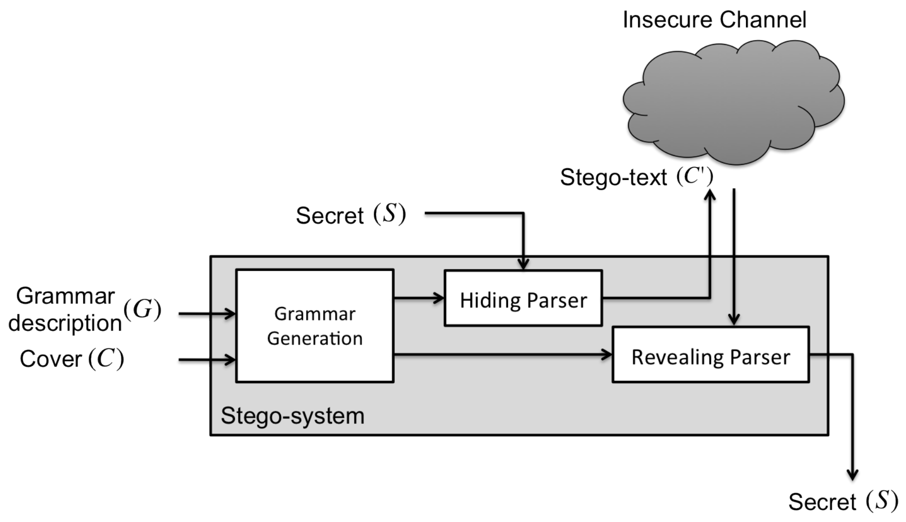


Figure 4.1: Proposed steganographic system (by cover synthesis) to hide source code into text

The proposed steganographic system uses the plain text C and the grammar description G to build a parser that will translate the source code into text. In this way, source code files (secrets S) of any length can be converted into innocuous text (stego-text C'). In order to produce a different stego-text with the same secret, the plain text file C must be modified.

To recover the original source code, the steganographic system needs the stego-text C', the same input text file C that was used to hide the source code and the grammar description G (Figure 4.1). In this way, C is required to obtain back the original source code. During the recovery process a grammar to translate the innocuous text C' into the original source code is generated. The steganographic system and the grammar generation process is explained in detail in the next Sections.

4.2.2.1 Hiding Grammar Generation

The hiding grammar is used as input to a parser generation process that will result in an executable file. That executable file will receive as input the source code to be hidden, generating as output the innocuous text file (stego-text). After the stego-text is generated, the executable file can be deleted.

Let $G = \{\Sigma_N, \Sigma_T, P, S\}$ be a context-free grammar describing the programming language to be hidden. $\Sigma_N = \{S, N_1, \ldots, N_n\}$ is the set of non terminal symbols required to form the language. $\Sigma_T = \{t_1, t_2, \ldots, t_m\}$ is the set of terminal symbols (words) that are part of the language (e.g in C *if*, *else*, *while* *{*, *}*, etc). Let $P = \{P_1, P_2, \ldots, P_k\}$ be the finite set of productions that generate the programming language. The hiding grammar generation process adds to each of the productions in P an output rule to be executed during the parsing process. In this way, every time that a production is activated (when reading a source code file to be hidden), a certain output is generated.

The cover-text file C is used to generate the output of the parsing process. The output for each production P_i of the grammar is extracted from the cover-text file (Figure 4.2). To avoid conflicts in the recovery process, fragments f used to produce output cannot be repeated. Each of the fragments can be attached to a terminal symbol so it will be its output in the stego-text. Therefore, not all kind of cover-texts C are suitable for use in the proposed stegosystem, only those with a number of fragments greater than the number of terminals in G. This is, $|C| \geq |\Sigma_T|$. Nevertheless, is not necessary to have only one fragment per terminal symbol. A derivation with just one terminal symbol may use more than one text fragment to produce the output. In this case, the output for that derivation will be the concatenation of these fragments. The resulting grammar is closely related to the programming language to be hidden.

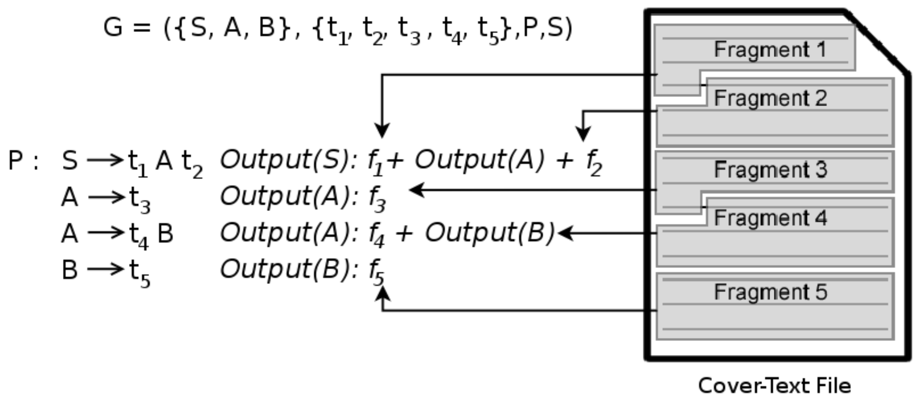

G = ({S, A, B}, {t1, t2, t3, t4, t5},P,S)
P : S → t1 A t2 Output(S): f1 + Output(A) + f2
A → t3 Output(A): f3
A → t4 B Output(A): f4 + Output(B)
B → t5 Output(B): f5
Fragment 1
Fragment 2
Fragment 3
Fragment 4
Fragment 5
Cover-Text File

Figure 4.2: Hiding grammar construction

Programming languages have a high redundancy. This can be exploited to mount a steganographic attack based on frequency analysis. To reduce the feasibility of these attacks, some rules of the grammar may have different number of outputs from the text file. This output is selected randomly between the random fragments selected from the cover-text file (Figure 4.3). In this case, the number of required fragments in C increases. In order to select the amount of outputs per production an analysis of the programming languages to hide should be performed. In this case, an analysis of the usage of the C programming language has been done. It can be found in Section 4.2.3.

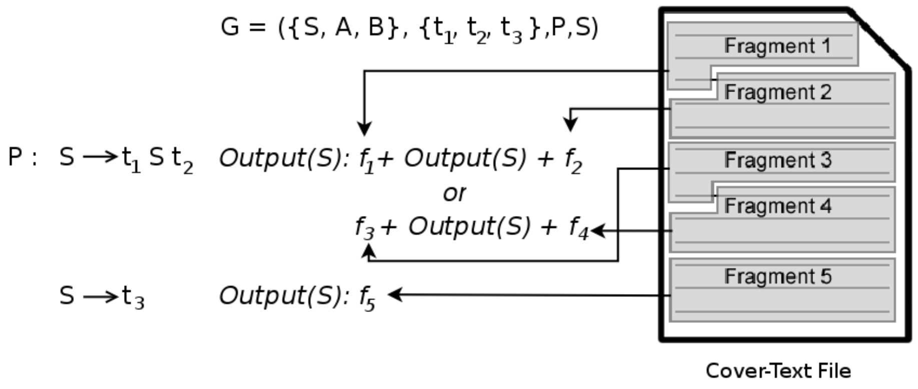


Figure 4.3: Random fragments in hiding grammar

4.2.2.2 Creating the Recovery Grammar

The recovery grammar is used to create a parser that will read the stego-text C' and output the original source code S. In order to create the recovery grammar, the programming language grammar G and the cover-text file C are required. In this case, fragments f extracted from the plain text file are used as terminal symbols of the grammar. This is, using G a new context-free grammar is created $G' = \{\Sigma_N, \Sigma'_T, P, S\}$ being $\Sigma'_T = \{f_1, f_2, \dots, f_j\}$ being j the required number of fragments for the used grammar (including terminal symbol concatenations and random productions added in the hiding grammar). In this case, the output produced by each of the grammar rules includes the original programming language terminal symbols (Figure 4.4).

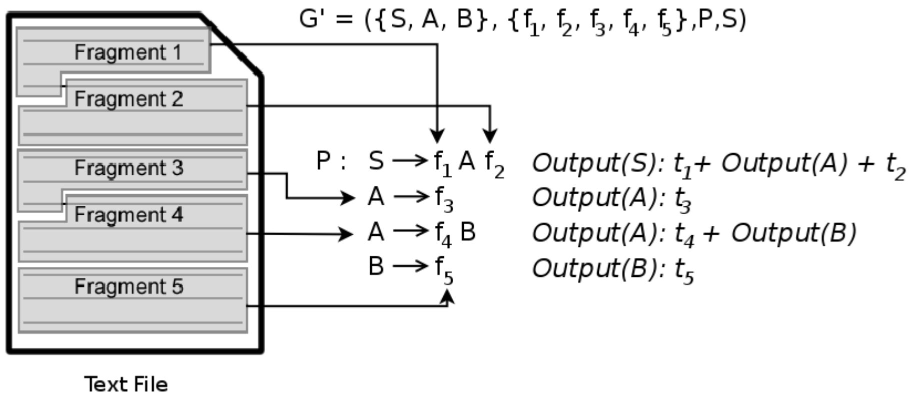

G' = ({S, A, B}, {f1, f2, f3, f4, f5},P,S)
Fragment 1
Fragment 2
Fragment 3
Fragment 4
Fragment 5
P : S → f1 A f2 Output(S): t1+ Output(A) + t2
A → f3 Output(A): t3
A → f4 B Output(A): t4 + Output(B)
B → f5 Output(B): t5
Text File

Figure 4.4: Recovery grammar construction

If additional randomness was included in the hiding grammar, its productions would be able generate different outputs randomly. In this case, it is also required to modify the set of productions P of G. For each of the random outputs generated by a production P_i, it will be necessary to add a new production in P'. This is, if a production P_i in P included l random outputs, l new productions $\{P'_{i_0}, \dots, P'_{i_l}\}$ will have to be added. All l productions will generate the same output (Figure 4.5).

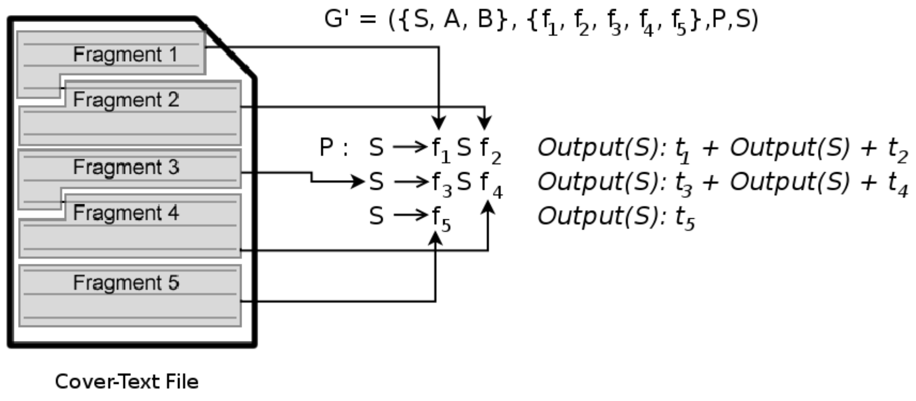


Figure 4.5: Random fragments in recovery grammar

4.2.2.3 Specific Programming Languages Issues

A context-free grammar describing the programming language is not enough to build stego-text files. Source code is usually split into different files. Some programming languages, like C and C++, may include preprocessor directives in their code. All these must be processed before the parsing process starts. Each programming language may need different approaches to solve its specific issues. Decisions taken to solve this issue for the C programming language are described in Section 4.2.3.1.

4.2.3 Hiding C Source Code

The previously described general stego-system has been implemented for the ANSI C programming language. This implementation has been called *Csteg*. The stego-system has been implemented using Ruby and C. Parsers are produced through Flex [119] and Bison [120] parser generators. Cover-text files have been retrieved from Project Gutenberg[2].

The description of the programming language is given in a special format that is read by a Ruby program. The Ruby program creates the hiding and recovering grammars ($G\ and\ G'$) and output rules in Flex and Bison formats. In order to create G', terminal symbols from from G are replaced by fragments extracted from cover-text. In the current implementation of *Csteg*, a fragment can be a word, a phrase or a paragraph from the cover-text. This is, the terminal set of G' is composed by words phrases or paragraphs from the used cover-text. Parser generation process is described in Figure 4.6.

4.2.3.1 Fragment Extraction

As described in Figures 4.3 and 4.5 the fragments in the cover-text are extracted in order to create a hiding and recovery parser (Figure 4.6). Cover-texts used for hiding source code require at least the same number of fragments as terminal symbols in the programming language to hide, $|C| \geq |\Sigma_T|$. As in other programming languages, C has some structures that may be more used than others. The frequency of appearance of these structures may lead to a specific frequency appearance of cover-text terminal symbols.

[2]http://www.gutenberg.org - Accessed on April 7th 2012

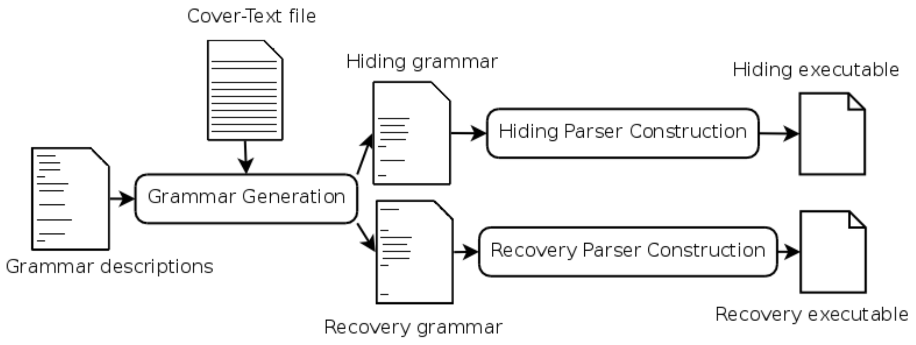


Figure 4.6: Parser generation process

To hinder frequency-based steganalysis, random derivations have been added for the most used C structures. To select the productions to be randomized, an analysis of the most used structures of a group of C source code files with cryptographic purposes has been performed. Depending on the source code files to be hidden, a new source code analysis may be required. Files have been gathered from different sources including the "eStream Project" [121], and "Applied Cryptography" [122]. The overall number of C source code files used is 414. Measures have been made with tools taken from [123]. Comments have not been accounted.

Token type	**Appearance in** %
Punctuation mark	51,59
Identifier	30,02
Numerical literal	11,63
Reserved word	4,77
String literal	1,29
Preprocessor directive	0,7

Table 4.1: Frequency of C tokens in cryptographic software

Tests (Table 4.1) show that most of the terminal symbols in the analyzed source code files are punctuation marks (comma, plus, minus, parenthesis, etc.). Identifiers and literals are very used too. These kinds of tokens cannot be hidden simply by a grammar substitution. Specific hiding strategies for these are described later. Reserved words appear in less than 5 % of the tokens (*if, else, while, int, char, etc.*). Preprocessor directives occurrence is not significant, only 0.7 %. The frequency of punctuation marks and reserved words has been further analyzed.

Inside the punctuation mark terminal symbol set (Table 4.2), comma (",") is the most used punctuation token (nearly 21 % of the punctuation tokens are commas). As the comma, semicolon and parenthesis are the most repeated punctuation tokens. Equals symbol and square brackets have a percentage of appearance above 5 %. The rest of 49 different punctuation tokens do not have a frequency of appearance greater than 3 %, having 40 of them a frequency less than 1 %.

Additionally, the most used reserved word in the analyzed source code is the "*if*" clause (Table 4.3). Also, the most used data types are integers and characters.

The performed analysis shows that inside each group of possible tokens (punctuators and

Token	Frequency	Token	Frequency
,	21,52	->	2,05
;	13,21	.	1,82
(	12	*	1,73
)	12	|	1,34
=	5,41	#	1,19
]	4,8	v++	1,11
[	4,8	+	1
{	2,21	*v	0,92
}	2,21	Other	10,68

Table 4.2: Frequency of punctuation mark tokens in cryptographic software

Word	Frequency	Word	Frequency
if	14,84	static	2,93
int	13,79	register	2,84
unsigned	9,25	case	2,83
char	8,84	while	2,60
for	8,30	break	2,54
void	5,85	sizeof	1,54
else	5,09	extern	1,21
return	5,02	short	1,14
long	3,74	struct	0,98
const	3,49	Other	3,15

Table 4.3: Frequency of reserved words in cryptographic software

reserved words) there are only a few tokens which are commonly used. The rest of the tokens are used rarely compared with the common tokens (Figure 4.7).

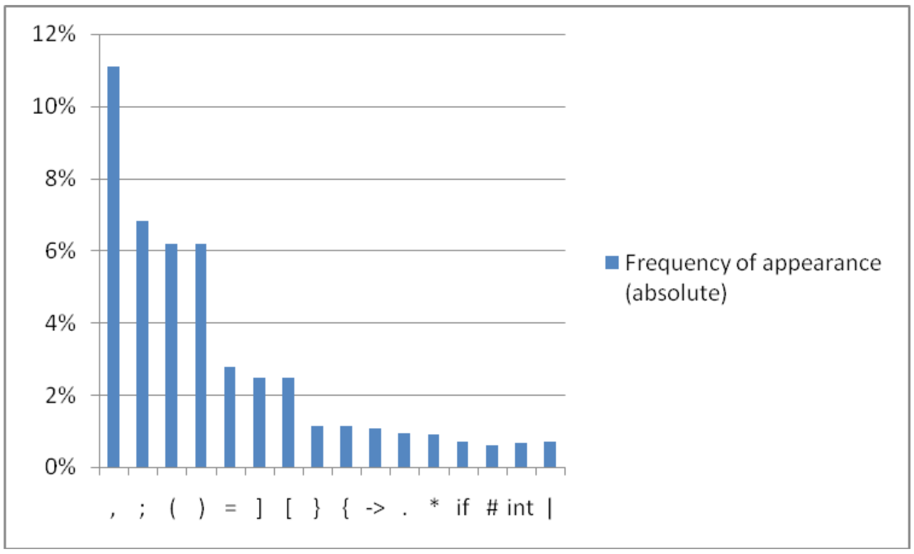


Figure 4.7: Frequency of most used tokens in the analyzed source code

To reduce the difference in the redundancy of fragments linked to most used programming language terminal symbols, random derivations have been introduced. Each time a symbol from a group is derived, its output will be chosen randomly from a group of cover-text fragments.

Most used tokens in each of the analyzed sets have been grouped defining boundaries on its frequency (Tables 4.2 and 4.3). For each group, a number of random cover-text derivations has been added. The number of derivations added in a group is directly related with boundaries selected for that group. Four groups have been created for each set (punctuation marks and reserved words). The highest appearance group will correspond to the one with more random derivations. The last group will not have any random derivation on it. Groups created for punctuation marks are shown in Table 4.4.

Group	Additions	Freq. gap	Tokens
1	20	100%-14%	,
2	14	14%-6%	; ()
3	4	6%-2%	= [ ] { } ->
4	1	2%-0%	Other

Table 4.4: Number of random derivations per punctuation mark

The same procedure has been used for reserved words. The results are shown in Table 4.5.

Group	Additions	Freq. gap	Tokens
1	10	100%-9%	if, int, unsigned
2	8	9%-5%	char, for, void, else, return
3	3	4%-1%	long, const, static, register, case, while, break, sizeof, extern, short
4	1	(1%-0%]	Other

Table 4.5: Number of random cover-text derivations per reserved word

There are elements inside the source code that can not be hidden with substitutions from the cover-text. These elements have been hidden by using special techniques:

- Identifiers are replaced by names. Each time an identifier is found, a symbol table is looked up. If the identifier is not in the symbol table, it is stored in it and associated with an English name. The output in this case is the identifier concatenated with the associated name. If the identifier is in the symbol table, nothing is added to the table and the output produced is just the English name associated with that identifier.

- A *String* literal can use any kind of character. A simple Caesar cipher to hide the content of this kind of literals has been implemented. The ciphered string will be imperceptible inside the stego-text.

- Numerical literals that appear in the code are translated into "poems" with a substitution algorithm. The digits of the literal are substituted by poem words. Each of the different digits has ten different words to choose randomly for its substitution. Numbers hidden this way can be in decimal, octal or hexadecimal format.

Table 4.6 shows an example of hiding a short piece of C source code into text using sentences from the book "The First Men In The Moon" by H. G. Wells.

Source Code	**Generated Text**
`int example (){` `return 46323+56743;}`	In this he descended towards constantly more luminous caverns of the moon. What did that matter now? A sort of languor had possession of my limbs and mind, I did not believe for a moment that we should ever find the sphere in that vast desiccated wilderness. I returned to my scenario. It was all amazingly petty, no doubt, but what was there remaining for me to do? Whatever I did I was resolved that I would keep myself level and right side up. It was horrible and delightful, and as wild as a nightmare, to go flying off in this fashion. James And then, as soon as I had the door locked on them all, I tore off the little man's clothes again, shied them right and left, and got into bed forthwith. example Had I indeed heard a sound? I felt the pressure of Cavor's hand upon my arm. My grip upon Cavor's foot became convulsive at the sight of him, and we remained motionless and peering long after he had passed out of our range. And at last, as I reached a little landing that was separated only by ten steps or so from the supreme seat, the woven splendour of the music reached a climax and ceased, and I was left naked, as it were, in that vastness, beneath the still scrutiny of the Grand Lunar's eyes. The silence, the rhythmic disappointment of the silence, came as a fresh shock. He stretched out his hand for his blanket, thrust his head through its central hole, and wrapped it about him. this is a poem:"sits truck right of mountain right of" It was dawn, a gray dawn, rather overcast but showing here and there a long patch of greenish gray. I tried to whisper to Cavor that I could hardly go without food much longer, but my mouth had become too dry for whispering. this is a poem:"sadly cat green licks among" It was dawn, a gray dawn, rather overcast but showing here and there a long patch of greenish gray. And behold, they had smashed like wax and scattered like chaff, and fled and vanished like the creatures of a dream! I rubbed my eyes, doubting whether we had not slept and dreamt these things by reason of the fungus we had eaten, and suddenly discovered the blood upon my face, and then that my shirt was sticking painfully to my shoulder and arm. If, for example, a Selenite is destined to be a mathematician, his teachers and trainers set out at once to that end.

Table 4.6: Example of generated text using a short piece of C source code with sentences from "The First Men In The Moon" by H. G. Wells

4.2.3.2 Parser Generation

Terminal symbols extracted from the cover-text $C = \{f_1, \ldots, f_j\}$ along the programming language grammar G are used to generate a parser description compatible with Flex and Bison. Each time an operation (hiding or recovery) is performed only one parser is generated. A hiding operation generates a C parser producing a stego-text as output. A recovery operation generates a stego-text parser that produces C source code as output. To produce the parsers the source code generated by Flex and Bison is compiled.

4.2.3.3 Covering Process

The generated C parser is able to parse any kind of C source code. Nevertheless, it can not properly handle C preprocessor directives. These are not hidden but executed generating a unique file including all the source code to be compiled. A special preprocessor has been implemented. This preprocessor reads all the source code files involved in the hiding process and creates an auxiliary file that will be the one passed to the parser to produce the stego-text (Figure 4.8). The preprocessor reads the source code file and processes the preprocessor directives. The resulting auxiliary file contains C source code but no preprocessor directives. File inclusion directives are handled differently in comparison to an usual C preprocessor. When the preprocessor finds an *#include* directive it checks if the file included is a system library (with $<>$) or a user library (with " "). System libraries are not included in the auxiliary file. If system libraries were included the lines of code to be hidden would significantly increase. Reference to system libraries is hidden like a string literal. On the other hand, user libraries source code is included in the auxiliary file.

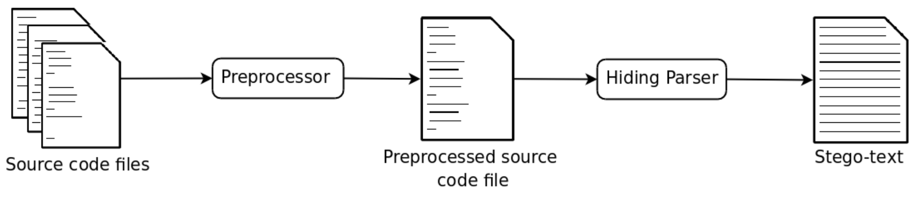


Figure 4.8: Covering process

The use of a C preprocessor means that most of the preprocessor directives included in the code are lost in the recovery process. The hiding process will make that the recovered source code will not be exactly the same source code that was hidden, but it will have exactly the same functionality. Additionally, depending on the preprocessor directives found in the source code, it may only be possible to execute it on a machine with the same architecture and software configuration than the one used to hide the source code.

4.2.3.4 Recovering Process

The generated stego-text parser reads the stego-text files and generates C source code files (Figure 4.9). In order to regain the functionality of the original source code, a post processing is needed. If the preprocessed file was the result of some file inclusions, the source code files which were included are created. This restores the original source code file structure.

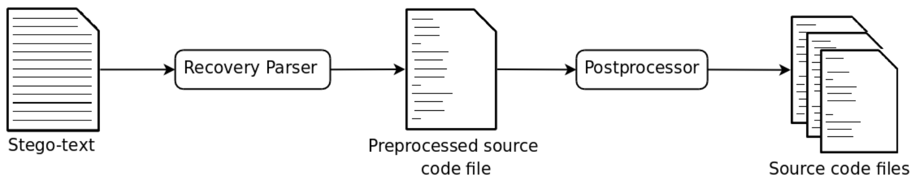


Figure 4.9: Recovering process

4.2.4 Evaluation

An experimental evaluation has been performed on the C implementation of the presented model. In this evaluation, the accordance of generated text to English has been measured. Additionally, the redundancy of generated stego-texts (could be used as a detection feature) has been analyzed.

4.2.4.1 Experimental setup

Overall, twelve experiments have been performed to test *CSteg*. Each consisted on the concealment and the recovery of the source code of a cryptographic algorithm. Four different algorithms (Table 4.7) have been chosen: Anubis, 3DES, IDEA and DECIM. Each of the algorithms has been hidden using three different types of terminal symbols from the cover-text file (words, phrases and paragraphs). Different books from Project Gutenberg's repositories were used as cover-texts.

	Anubis	**3Des**	**Idea**	**Decim**
Size in Kilobytes	37.904	16.389	2.263	45.561
Files	2	2	2	6

Table 4.7: Source code file properties

4.2.4.2 Experimental Results

The size of the source code files, the generated stego-text, and the recovered source code have been measured. The compression ratio, using *bzip2*, between the three sets of files has been also compared. To measure the quality of stego-texts produced, character and digraph frequency have been computed and compared against a reference model for English. Size of source code refers to the sum of the size of all the source code files of the program.

The size of stego-texts generated (Figure 4.10) strongly depends on the kind of terminal symbol selected. Stego-texts generated with words from the cover-text are (naturally) smaller than stego-texts generated with paragraphs. Obviously, the average size of the terminal symbols in the second case is bigger. The hiding process has removed all preprocessor directives except file inclusions, which remain unchanged. Comments have been lost. These are the reasons why the size of the recovered files (Figure 4.10) is generally smaller than the size of the original source code files.

The capacity (or ratio) of *CSteg* for each of the tested files (Table 4.8) has been computed. This can be used as a redundancy estimation. To perform this calculation the size of the stego-object has been divided by the size of the recovered source code. The size of the original source code has not been used because comments and some preprocessors directives from the original source code are lost in the recovery process. The proposed stego-system, as expected, has introduced a lot of redundancy into the stego-object.

Compression algorithms help to evaluate redundancy on files. Stego-text compression ratio (Figure 4.11) compared with original and recovered source code files indicates again that stego-text files have a considerable redundancy. Stego-text files generated with words have much less redundancy than those generated with phrases and paragraphs. Differences on the compression ratio between phrases and paragraphs are not significant. A redundancy

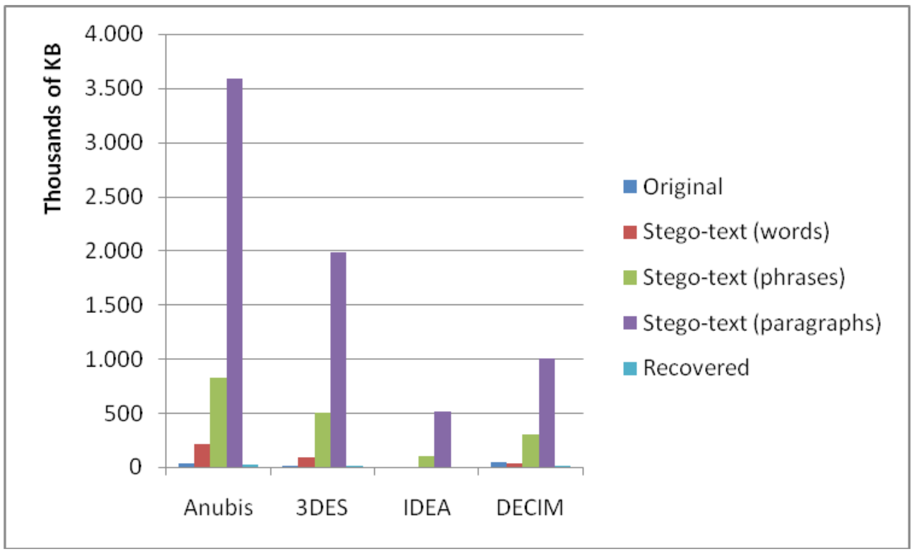


Figure 4.10: File size chart

	Anubis	3Des	Idea	Decim
Words	0,120	0,146	0,182	0,231
Phrases	0,032	0,027	0,015	0,030
Paragraphs	0,007	0,006	0,003	0,009

Table 4.8: Capacity of *CSteg* per test

check by an attacker of the stego-text would rise suspicions. This security drawback could be reduced by adding more random stego-text derivations. In the ideal situation the stego-text would not have any repetition. In this case, the compression ratio would not be the same as the source code but it would be less suspicious. Word generated stego-texts have the lower compression ratio, but they only are a conjunction of generally meaningless words. Due to the minimum differences between phrases and paragraphs stego-texts, it should be a better choice the use of paragraphs to build the stego-texts.

In order to pass unnoticed to an automatic warden, generated stego-texts should (at least) comply with the frequency distribution of characters in English. The frequency of appearance of characters and digraphs in all stego-texts generated has been measured. Unfortunately, distribution of letters in texts depends on the length, language, and the text itself. Frequency of appearance of characters usually is similar between text of the same language and length, but it can be easily manipulated, as in the case of lipograms[3]. Aside from these rare cases, frequency distributions for characters and digraphs based on 235 different books from Project Gutenberg have been constructed. This frequency distribution will not describe the frequency that should follow any kind of text but it will give a good approximation of the characteristic frequency distribution for texts in English.

The character and digraph frequency distribution of each kind of texts have been compared against the distribution obtained from Project Gutenberg texts (Figures 4.12 and 4.13). Frequency distribution of characters in stego-texts generated with words from a Project Guten-

[3]A lipogram is a text which does not use a specific letter. A notable example is "La Disparition" [124]. None of the 300 pages contained the letter "e". The English version, "A void", did not include the letter "e" either

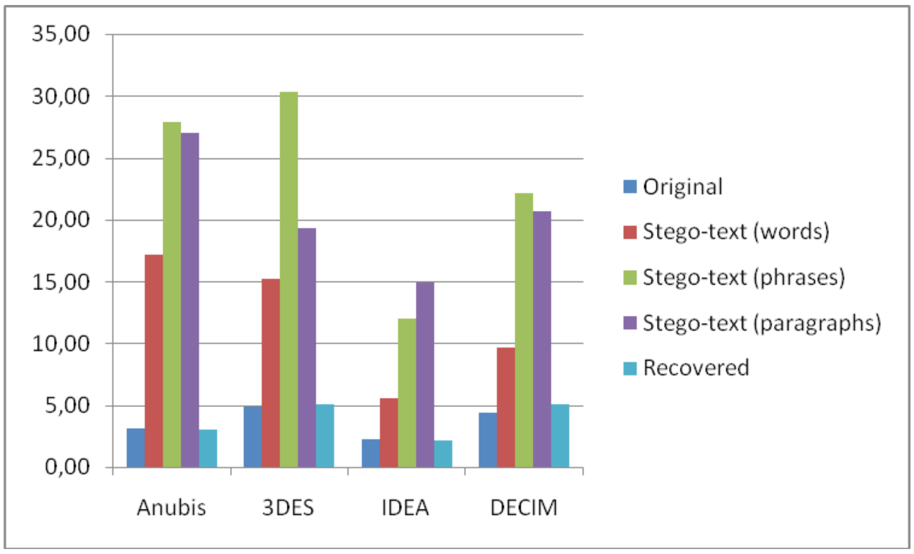


Figure 4.11: Compression ratio chart

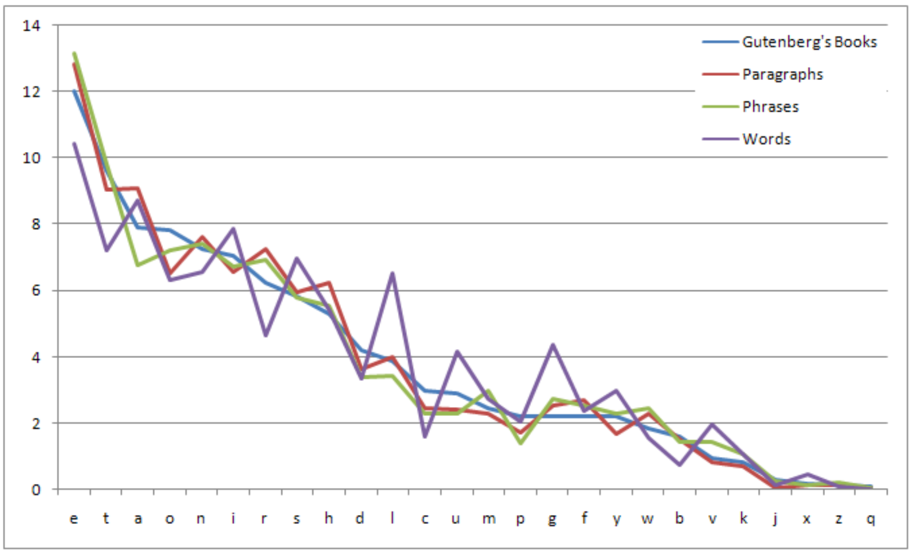


Figure 4.12: Comparison of character distribution

berg book differs a lot from the reference distribution. Stego-texts with paragraphs and phrases are much closer to the reference distribution. Digraphs frequency is consistent with this observation. Stego-texts generated with phrases and paragraphs fit better the reference digraph distribution.

4.2.5 Summary

The proposed stego-system scheme allows to convert any kind of source code file (if the required grammar is available) into text. The usage of this application in environments where source code files are handled could allow malicious insiders to transform them into text in order to inadvertently extract them from the organization. The reduced security constraints

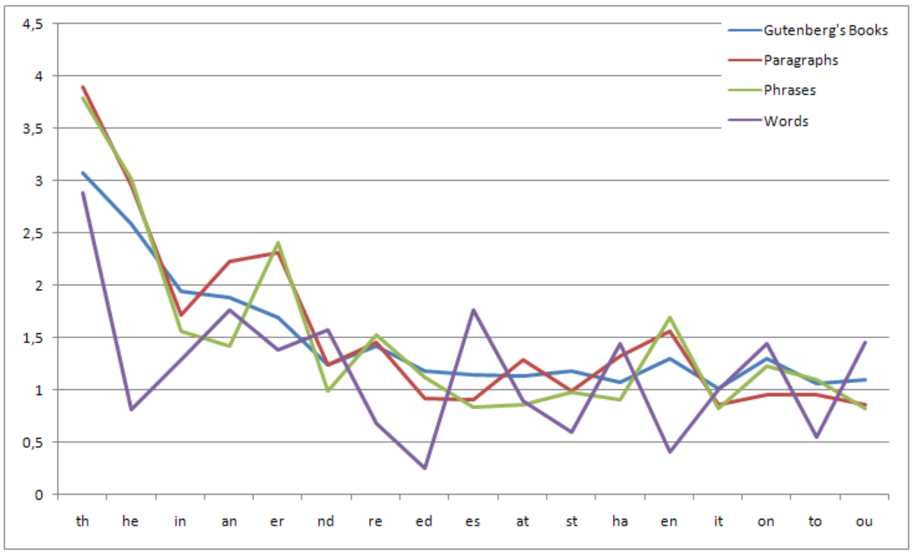


Figure 4.13: Comparison of digrams distribution

inside programmers and software designers workstations could ease this attack.

The proposed stego-system has been implemented to hide ANSI C source code files into text. Due to the generative nature of the stego-system, it can hide files of any length. Nevertheless, as the size of the embedded data increases, the redundancy of the stego-text also increases, reducing the security of the system against redundancy-based steganalysis. Stego-texts carry the same functionality of the original source code files. Embedded files lose comments and most of the preprocessor directives, but data lost does not change the functionality of the recovered source code.

The generated stego-texts include meaningful text fragments. A human can find common structures of the human language. This is an important issue that will help the stego-text to pass unnoticed by a passive warden (a DLP solution). On the other hand, grammar parsers are very sensitive to changes. Unfortunately, the stego-system inherits this drawback. A change in the stego-text by an active attacker will probably cause the embedded data to be lost.

The proposed scheme lacks of some security features that makes its real use infeasible for an attacker that requires a minimum level of security. Additional features that would improve the security of the proposed system would be the addition of a password to introduce more randomness in the hiding process. The password could be used to randomize the fragment extraction and output generation process. Additionally, the stego-system should be modified in order to hide any kind of file, not only source code files. This constraint make its use feasible, only in environments related to software development.

4.3 Stegnography for Mobile Devices

Research on steganographic algorithms has usually focused on algorithms development, but not on the devices that execute those algorithms. In fact, from the point of view of the information leakage problem, there exist some restrictions that can be applied to employees workstations that would not allow them to execute steganographic applications. In this regard, mobile devices such as smartphones and tablets can be a new way to implement these steganographic algorithms.

These devices have great computing capabilities and constant internet access. Additionally, its popularity have spread their usage, not only as an organizational device to help in business process, but also as a personal device. Although mobile devices include a range of features that make them usable in organization environments, their operating systems also pose some difficulties to avoid threats such as information leakage. In fact, it is very difficult to control the capabilities of mobile devices depending on their context, and this may lead to uncontrolled actions performed by their employees. As previously mentioned, from a survey of 500 organizations in the US, only the 34% of them had some kind of mobile managing system implemented [71].

In order to prove the feasibility of using steganography in smartphones, a steganographic application has been developed for the iOS platform[4]. The purpose of the application is to prove that it is possible also to execute steganographic applications in other devices, and not only in users' workstations. The developed application, which is called "Hide It In" [14], is available since December 2010 for free in the Apple App Store. Additionally, its source code is registered in the "Comunidad de Madrid" registry for inventions and copyrighted material. It has been downloaded more than 5.000 times with an average of more than 100 downloads per week (Figure 4.14). The purpose of this application is to prove that it is possible to implement widely used steganographic algorithms in other devices less controlled than insiders' workstations such as mobile devices.

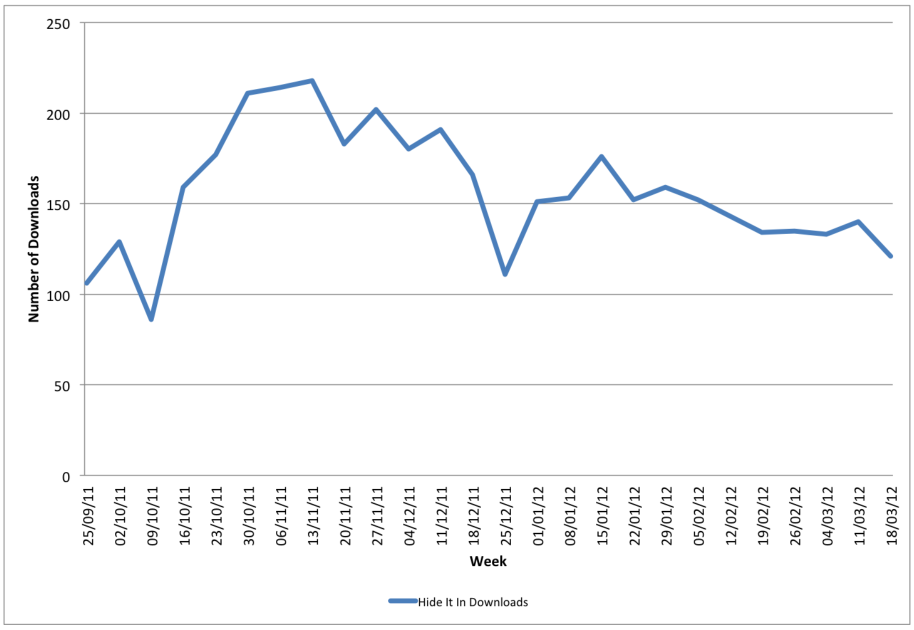


Figure 4.14: Weekly downloads of "Hide It In" since 25th September 2011

[4]http://www.apple.com/ios/ - Accessed on April 8th 2012

4.3.1 Related Work

“Hide It In” is not the only steganography application currently available for iOS devices. In this Section, other steganographic applications for iOS devices are described. Each reviewed application (which icons are shown in Figure 4.15) has been downloaded and tested. Although, there exist more applications than the ones reviewed in this Section, only those that worked during tests have been included. The results of the performed review are depicted in Table 4.9.

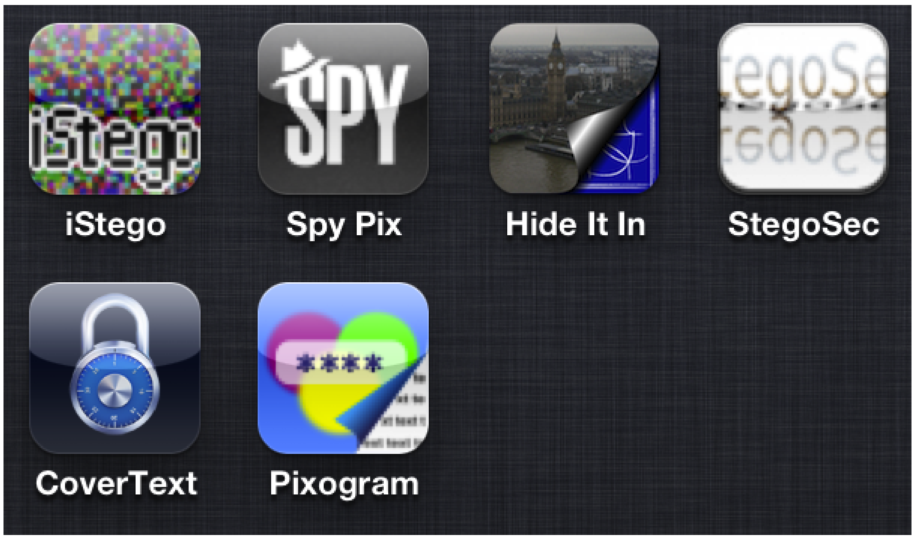


Figure 4.15: Icons of reviewed steganographic applications for iOS (including “Hide It In”)

iStego

This application is capable of hiding text or images into image covers [125]. For such a purpose, it uses the LSB algorithm, generating a PNG image with the result. It does not encrypt the secret to hide. Additionally, it does not perform capacity checks. If the secret does not fit the cover, only a portion of the secret will be hidden into the cover. It only allows to hide images inside the smartphone photo library.

Spy Pix

This application uses the LSB algorithm to hide images (from the photo library or taken with the camera) into other images (also from the photo library or taken with the camera) [126]. *Spy Pix* does not encrypt the secret to embed. However, it allows to configure the amount of least significant bits used to embed the secret. Generated secrets can be exported by email.

StegoSec

This application by Raffaele de Lorenzo is a reduced implementation of Outguess [103] capable of hiding text messages into images [127]. StegoSec hides the introduced secret into the LSB of the discrete cosine transform coeficients. Therefore, stego-objects are generated in JPEG format. In order to hide text secrets it is necessary to define a password that will be used during the information hiding process.

CoverText

This application allows to hide text messages (up to 140 characters) into images [128]. Specifically, it hides textual information into JPEG images, but does not require any password. This means, that anyone using the application can extract the embedded text from an image.

Pixogram

Pixogram allows to hide a textual message into an image [129]. Pixogram uses a proprietary algorithm to embed up to 20.000 characters per image. Stego-images generated with Pixogram are stored inside the application and can be sent using the device configured email accounts.

	iStego	**Spy Pix**	**StegoSec**	**CoverText**	**Pixogram**	**Hide It In**
Secret	Text and img.	Img.	Text	Text	Text	Img.
Cover	PNG Image	PNG Image	JPEG Image	JPEG Image	PNG Image	PNG Image
Password	No	No	Yes	No	Yes	Yes
Encryption	No	No	AES 128	-	Proprietary	AES 256
Export Method	Email	Email	Email	Email	Email	Email
Cover Mode	No	No	No	No	No	Yes

Table 4.9: Summary of current iOS steganographic apps and its features

4.3.2 Capabilities

The current version of “Hide It In” allows to hide images captured by the iOS device camera inside other images from the Camera Roll of the device. Those images can be sent by mail using any account configured in the device operating system. Currently, the application has three modes: hiding, revealing and cover mode.

4.3.2.1 Hiding information

“Hide It In” implements a modified version of the LSB algorithm. “Hide It In” uses PNG files as covers. Images from the camera roll are converted to JPEG before embedding. Additionally, the cover image size is decreased to reduce its final size. If the image does not fit the selected PNG file, a JPEG recompression is performed. These actions over the cover and image secret are performed due to restrictions imposed by iOS devices. In the case of iOS devices, if an image being sent through the mail application is bigger than a certain threshold (400 KB), the device may ask the user to send a smaller version (optionally). In such a case the hidden information is lost.

Each pixel of the PNG image used as cover is composed by three colour components and one alpha component, each of them of one byte. The application embeds information to be transmitted in the two least significant bits of each colour component, but does not modify the alpha channel. This is due to compression changes made by the iOS system that may delete the information sent in that channel. Before embedding, it encrypts the secret with AES 256. The embedding function is described in Algorithm 1.

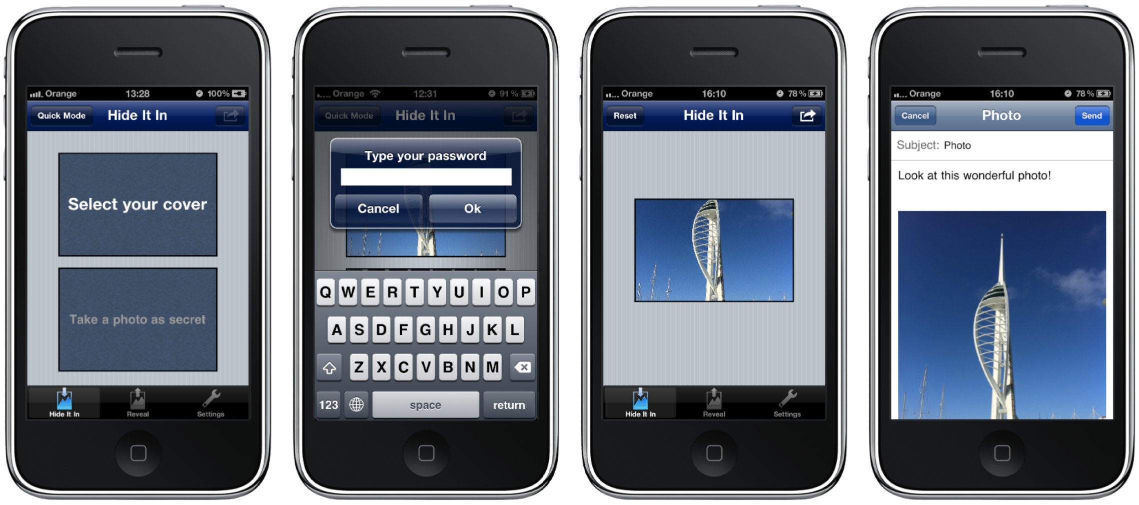
Algorithm 1: Embedding Function of Hide It In

```
# Secret encryption
Data: S, secret image to be hidden; C, PNG image used as cover; P, password used
      during encryption;
begin
    # Cover preparation
    Set C ← Normalize(S)
    # Secret preparation
    Set S ← Normalize(S)
    Set S ← JPEG(S)
    while Size(S) > Capacity(C) do
        Set S ← ReduceJPEGQuality(S,0,75)
    Set SignificantBits ← 2
    Set CurrentByteCover ← 0
    Set CurrentSecretBit ← 0
    # Secret encryption
    Set EncryptedSecret ← $AES_{256}$(SHA1(P), S)
    # Empedding process
    for CurrentSecretBit < SizeInBits(S) do
        # If the current component is not the alpha component then
        if (CurrentByteCover+1) mod 4 $\neq$ 0 then
            Set CurrentSignificantBit ← 0
            for CurrentSignificantBit < SignificantBits do
                Set Mask ← 0xFF and GetBit(CurrentSecretBit,S)
                Set C[$CurrentByteCover$] ← C[$CurrentByteCover$] and Mask
                Set CurrentSecretBit ← CurrentSecretBit + 1
                Set CurrentByteCover ← CurrentByteCover + 1
                Set CurrentSignificantBit ← CurrentSignificantBit + 1
end
```

In order to hide an image, the user of the application is required to perform the following steps (Figure 4.16):

- The user selects a secret taking a picture with the camera.
- The user selects a cover from his photo library.
- The user inputs a password to encrypt the secret before the embedding takes place.
- The secret is embedded into the cover.
- The user selects to send the stego-picture by mail, introducing the recipient, subject and mail body. The stego-image is automatically attached.

If the user performs all the previous mentioned steps, she would able to send the cover with the embedded secret to an email address outside the organization control. Now it is just necessary to extract the information back.

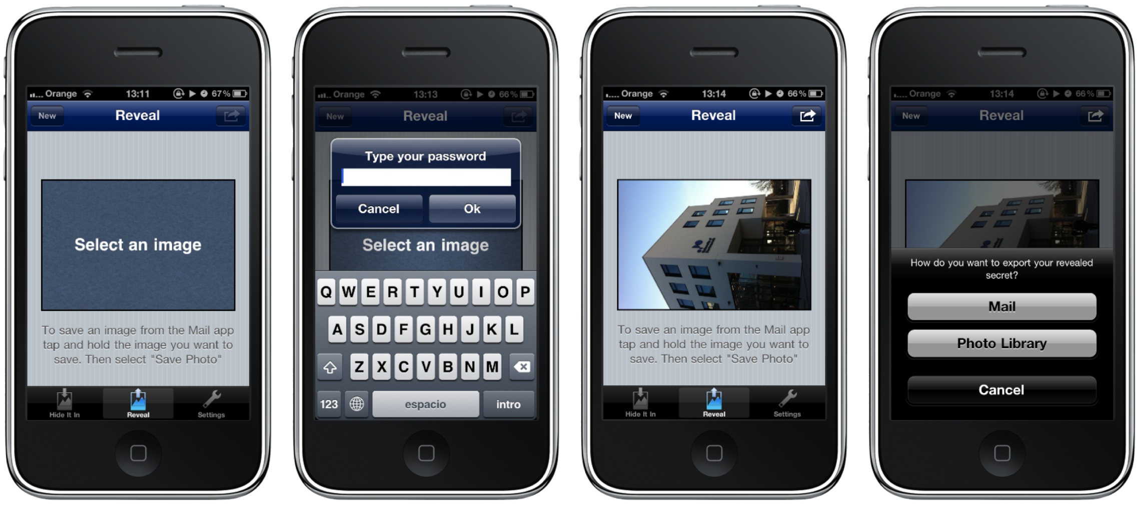


Look at this wonderful photo!

Figure 4.16: Process followed by the user to hide an image and send it outside the organization.

4.3.2.2 Revealing information

In order to extract the secret from the stego-image, the user has to open the stego-image with "Hide It In". To do so, it is just necessary to open the mail containing stego-image and introduce the previously used password. The revealing function will first extract the least significant bits from the three image components and then decrypt the stream of bits using the provided password. The obtained image can be saved into the mobile phone photo library or sent again by email. This process is depicted in Figure 4.17.



To save an image from the Mail app tap and hold the image you want to save. Then select "Save Photo"

How do you want to export your revealed secret?

Figure 4.17: Process followed by the user to obtain back a secret image from a stego-image

4.3.2.3 Cover mode

In the case of iOS, taking a photography with the device camera forces the device to display a view with the image the camera lens is capturing. Although this behaviour is useful for the normal usage of the camera, it may be dangerous for an insider using their mobile phone in an hostile environment. However, current frameworks for mobile application development allow developers to customize the appearance of their applications. In fact, it is possible for a developer to overlap the camera view shown to the user with a different view.

“Hide It In” includes a cover mode that allows to make pictures without showing the camera view. To use this mode, the user has first to configure some default parameters. These include the password to be used during encryption, a mail address to send the stego-object, a subject and body for the email message and finally, an image that will be used as cover and also shown instead of the camera view.

4.3.3 Evaluation

In this Section, the security of the generated stego-images and the performance of the application are analyzed.

4.3.3.1 Security

Although the security of the LSB steganographic systems has been widely studied and they are considered as broken [99], an analysis of the security of the proposed applications has been performed. Steganalysis techniques try to detect whether objects hide any kind of information or are just innocuous. To evaluate the security of “Hide It In”, two different sets of images are used, a set with hidden information (stego-images) and a set with no hidden information. The steganalysis procedure should be able to detect stego-images with more than a 50 % of accuracy (random classification).

The performed steganalysis is based on feature extraction. That is, a set of features are extracted from each image. Feature values are used to tell apart stego-images from innocuous ones. These techniques are also explored in Chapter 5, but the security analyisis performed in this Section requires to introduce them.

Image Set

Images generated with “Hide It In” are supposed to be sent through an iOS device. In the information leakage scenario, an adversary would take a picture of some secret with her device and try to send an email with the stego-image using the same device. In this case, the adversary (the organization trying to avoid information leaks) would have to detect whether the image sent with that device includes or not hidden information. Therefore, the image set has been restricted to images taken with iOS devices (specifically an iPhone 4). The set of images used for the security analysis is 210 of which 145 are clean images taken with the iPhone 4 and the rest, 65, are stego-images that are used as cover, images taken with the same device.

Features to analyze

Several features can be used for steganalysis. In this case, a selection of techniques that obtained successful results with LSB algorithms has been selected. On [99], a technique called RS Analysis to detect LSB steganography in images was presented. RS Analysis is based on the concepts of Regular and Singular groups. Let G be a group of adjacent pixels:

$$G = \{p_1, \ldots, p_n\}$$

Let $f(G)$ be a function that measures the level of regularity or smoothness of a group of pixels. Finally, Let F be an operation that flips the least significant bit of a pixel component. Authors define a *regular group* as a group of pixels that follows the constrain:

$$f(F(G)) > f(G)$$

That is, the regularity of the group of pixels G once F has been applied is greater than the regularity of the original group of pixels (they differ more from each other). A *singular group* is defined as the opposite. That is, a group of pixels which regularity decreases after applying F:

$$f(F(G)) < f(G)$$

Authors found out that, when flipping 50% of the LSB image pixels (which is the same as embedding a random message in the image) the number or regular groups and singular groups tends to balance (its difference gets reduced). Authors results stated that messages requiring less more than 0.005 bits per pixel where detectable with RS Analyisis (for gray scale images). Nevertheless, other results raise this limit to 0.01-0.02 bits per pixel [130]. Additionally, this last study states that RS Analysis is highly sensitive to image compression. In fact, if images to be analyzed have been previously compressed, the accuracy of the steganalysis drops significantly.

Another technique to detect LSB steganography is Sample Pair Analysis [100]. In this, authors state that in natural images, the probability of a pixel sample pair whose difference is $2m+1$ being m, $0 \leq m \leq 2^{b-1} - 2\, b - 1$ having its even component greater or smaller than the even component is the same. This statement is corroborated with the analysis of 29 natural images. Most significant bits are equal (being b the amount of bits per sample). Sample pair analysis performs very similar to RS Analysis. In fact, as authors state, both steganalysis are based on the very similar principles and assumptions.

Both, RS Analysis and SPA have been used as features used to detect whether images hold hidden information or not. Regarding RS Analysis, the number of R and S groups for each of the three colour components (red, green and blue) has been calculated. This values are used to estimate the hidden message length, and the percentage of modified bits. Additionally, the same analysis is performed with overlapping groups. Finally, a mean of the two kind of features provided by RS Analysis (i.e. percentage of flipped bits and estimated message length) has been calculated. Overall, RS Analysis generates 14 features to aid in hidden information detection.

Sample Pair Analysis also performs its analysis for each of the colour components of the image. Sample pairs of pixels are selected across and down the image, and its difference is measured. Assuming a distribution of differences as in [100], the message length and percentage of embedded pixels are estimated. In the same way as in RS Analysis, the mean of both kinds of values has been calculated. This allows to obtain 8 more features to help in determining the existence of hidden information inside an image.

Along previously presented features, randomness tests are useful to analyze the random behaviour of bit streams. These kind of measures are widely used to analyze the security of cryptographic algorithms in terms of the randomness of their output [131]. The least significant bits of each pixel component (with the exception of the Alpha component) have been used to embed a random bit stream (the encrypted secret). Therefore, randomness tests over these streams may produce different results when comparing to innocuous images (these streams are supposed not to be random). For each image, 14 different bit streams are extracted. Besides using the whole image as a bit stream itself, bit streams are also extracted from last significant bits of each component, and a combination of these. For each bit stream, 7 different randomness measures are calculated. These measures are obtained by executing ENT against the different generated bit streams [132]:

- Entropy: the amount of information per byte as defined by Shannon [133].
- Optimal compression ratio: the compression ratio in which the bit stream would be compressed with an optimal compression algorithm.
- Chi-square test: a chi-square test is used to test the distribution of bytes in the bit stream against a theoretical random distribution of bytes.
- Arithmetic mean: the bit stream is split in bytes and the arithmetic mean is calculated.
- Montecarlo π value: the bit stream is split in bytes and each set of 6 bytes is used as a 24 bit coordinate for a point. Measuring the distance of each point to the circle center allows to calculate an approximate value of π. For random streams, the value obtained with this method should approach to the real value of π. The error ratio for the obtained value is also calculated.
- Serial correlation: this feature measures the relation of a byte of the bit stream with the previous byte in the bit stream.

The combination of randomness measures with the different bit streams that can be analyzed, generates up to 98 different features. Added to the previous features, there is an overall of 120 features.

Results

Features have been extracted and gathered in a feature file to be processed by a data mining software (i.e. Weka [134]). Nevertheless, a visual analysis of the obtained results allows to distinguish a set of features that alone could be used to detect images with hidden information. In the case of “Hide It In” 62 out of 120 features can be used on their own to detect the presence of hidden information. Table 4.10 shows a selection of them and its boundaries.

Although some general features such as the image entropy can be used to distinguish stego-images, all features based on least significant bits of the image provide more accurate boundaries. This is an expected result, as the least significant bit stream of each component is supposed to be random, while the randomness of the image (or a color component) is increased by the embedding, but can not be considered as truly random.

Features from RS analysis that perform secret length estimation produce false positives and negatives, while those that estimate the percentage of hidden information work pretty

Feature	**Description**	**Detection rule**
Entropy	Image entropy	$>$ 7,982098
Correlation	Image correlation	$>$ -0,001399
ChiSquare Red	Red component χ^2 test	$\leq$ 2483,09
ChiSquare Green	Green component χ^2 test	$\leq$ 1968,15
Correlation Red	Red component correlation	$\leq$ -0,016456
Entropy Green	Green component entropy	$>$ 7,991523
Correlation Green	Green component correlation	$\leq$ -0,018153
Entropy LSB	Entropy of LSB stream of the three color components	$>$ 6,07
Compression LSB	Compression of LSB stream of the three color components	$=$ 0
ChiSquare LSB	χ^2 test of the LSB stream of the three color components	$<=$ 812,4
Mean LSB	Mean value (for bytes) of the LSB stream of the three color components	$>$ 124,6198
MonteCarlo LSB	π value estimation of the LSB stream of the three color components	$\leq$ 3,250071205
MonteCarlo Error LSB	π value error of the LSB stream of the three color components	$\leq$ 3,45
Ratio RSA Red No groups	Amount of hidden information (in %) for the red component with non overlapped RS analysis	$>$ 85,264
Ratio RSA Blue No groups	Amount of hidden information (in %) for the blue component with non overlapped RS analysis	$>$ 80,76514
Ratio RSA Green No groups	Amount of hidden information (in %) for the green component with non overlapped RS analysis	$>$ 93,30064
Ratio RSA Red groups	Amount of hidden information (in %) for the red component with overlapped RS analysis	$>$ 87,12815
Ratio RSA Blue groups	Amount of hidden information (in %) for the blue component with overlapped RS analysis	$>$ 86,96491
Ratio RSA Green groups	Amount of hidden information (in %) for the green component with overlapped RS analysis	$>$ 86,25562
Mean Ratios RSA	Mean value of previous RS analysis values	$>$ 89,80835

Table 4.10: Features that can be used on their own to detect “Hide It In” stego-images including its required detection boundary

	Classified as innocuous	Classified as stego-image
Innocuous	111	3
Stego-image	1	64

Table 4.11: Confusion matrix for the stego-image classifier based on SP analysis features

well. In fact, each of them can be used on its own to identify stego-images (Table 4.10 rows 14 to 21).

Surprisingly, none of the features extracted using Simple Pair Analysis can be uniquely used to detect stego-images. A C4.5 tree [135] has been build using this kind of features. The well known Weka data mining software has been used [134]. The resulting classifier has been tested using cross validation. The obtained tree is represented in Figure 4.18.

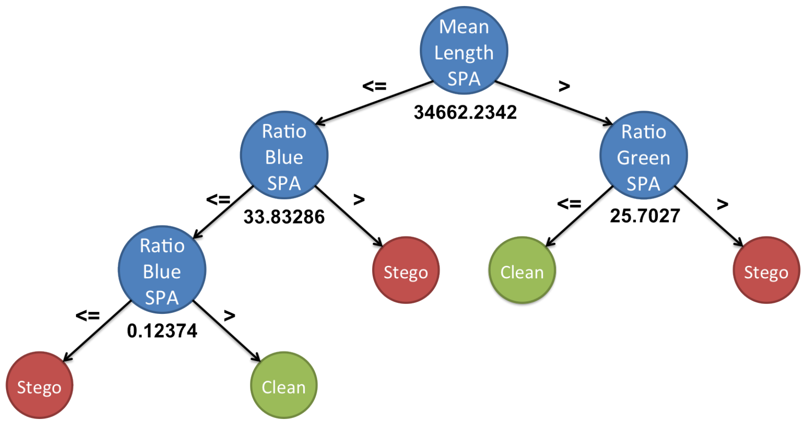


Figure 4.18: C4.5 classifier built (using Weka) with Simple Pair steganalysis features

The obtained classifier has a true positive rate of $TPR = 0.985$ and a false positive rate of $FPR = 0.026$, being its confusion matrix the one represented in Table 4.11. Although the resulting classifier is accurate, simpler classifiers obtained with randomness features obtain better results.

4.3.3.2 Performance

One of the goals of this work is to prove that steganographic algorithms can be implemented and executed in a reasonable time in a mobile device. For such a purpose, a performance evaluation of “Hide It In” has been performed.

In order to test the performance of the application, the standard behaviour of the application has been modified. Instead of hiding only one picture per execution, it performs 65 hiding operations with the same secret, but different passwords. Overall, ten executions with different secret bit streams have been performed. Tests have been conducted on an iPhone 4 (which includes an A4 1 Ghz processor and 515 MB of RAM). Measures have been obtained trough “Instruments”, an application provided by Apple to profile the software performance and other related metrics. All tests were directly run on the device.

The performance of the whole process has been measured in terms of time. The amount of time required to obtain the stego-image from the cover, secret and password is measured by the overall hiding time. This time is more deeply analyzed in the next paragraphs. The hiding process generates stego-images as programming language objects in the system memory. In order to write them into the file system, it is required to obtain its PNG representation. This time has also been measured. Finally, the time required to write the stego-image to the file system has been measured. The time required by the smartphone in each of the previous mentioned steps (for each run) is shown in Figure 4.19.

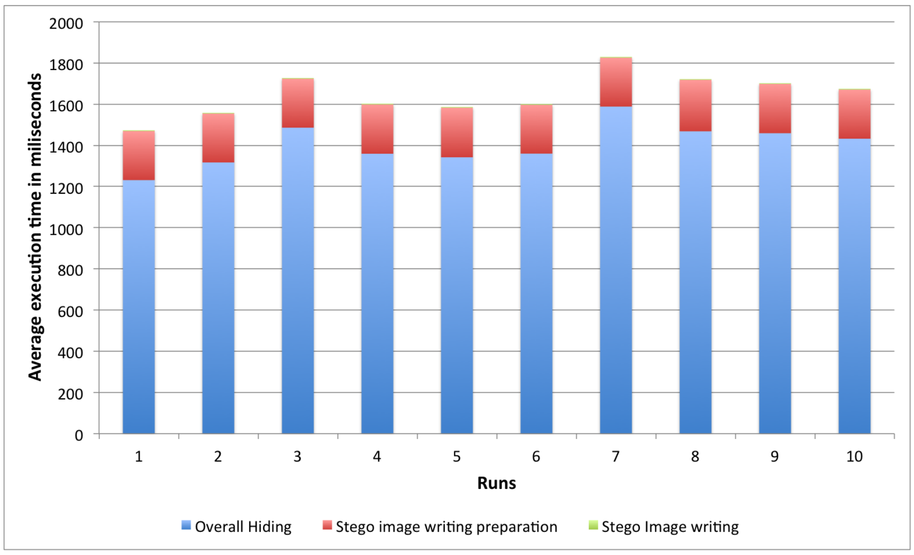


Figure 4.19: Distribution of the mean execution time required to create an stego-image in ten runs of the performed tests

The mean times required for each phase are depicted in Table 4.12. The first thing to notice is that the time required to complete one stego-image generation is in mean less than two seconds. This is a reasonable amount of time for a smartphone and is also consistent with responsiveness restrictions imposed by the operating system (5 seconds). Additionally, the time to write the image into the device is negligible. This is probably because the small amount of data to be written in the device and the fact that the storage device is a flash drive (much faster than standard hard drives). The time required to prepare the stego-image to be written in PNG format is less (in mean) than 250 ms. This is the approximately the 14,56 % of the time spent during the whole process. However, the current implementation of the application does not focus on performance. This is a good result. Even when the software development has not focused on performance, the execution time results show that it is feasible to use the application in a reasonable time.

Execution phase	**Mean time required in milliseconds**	**Ratio in %**
Overall hiding	1401 ms	85,20 %
Stego-image writing preparation	239 ms	14,56 %
Stego-image writing	4 ms	0,24 %

Table 4.12: Overall time required to hide an image and store the resulting file in the device filesystem

Additionally, the performance of the secret hiding process has been analyzed. In this case, the time required to hide a secret has been divided into different phases according the tasks required by the hiding algorithm:

- The *secret preparation time* is the time required to prepare the secret, to be hidden. This includes an image resize to reduce the file size and a compression in JPEG format. As mentioned earlier, more than one JPEG compressions may be required in order to fit the secret into the cover. In performed tests, all images selected as secrets were able to fit all the covers.
- The *cover preparation time* is the time required to resize the cover to its final image size. Additionally, this includes also the byte extraction from the image object.
- The *AES encryption time* accounts the amount of time used to encrypt the whole secret using a randomly provided key.
- The *secret embedding time* measures the time required to embed the encrypted secret into the normalized cover object.
- The *stego-image preparation time* accounts the amount of time required to create the image object after the embedding has taken place. In the performed tests, this time is negligible.
- The *logging time* measures the time the system has spent with logging messages while performing tests.

Figure 4.20 shows average percentage of time spent by each of the previous defined phases in while hiding a secret into an image. Of all the time required to hide a secret into a cover, more than 50 % of that time is spent in cover and secret preparations. If this check was not performed by the operating system, these steps would be skipped, but additional time would be spent in embedding more information in a greater cover.

Table 4.13 shows the average amount of time spent in each of the previous defined phases to hide one secret. Time required to prepare the secret is the highest, as additional compressions may be performed in the secret image to reduce its size to fit the cover.

Execution phase	**Mean time required in milliseconds**
Secret preparation	460 ms
Cover preparation	308 ms
AES encryption	10 ms
Secret embedding	603 ms
Stego-image preparation	0 ms
Logging	1 ms

Table 4.13: Detailed (by phase) view of the mean time required to embed a secret and produce a stego-image object

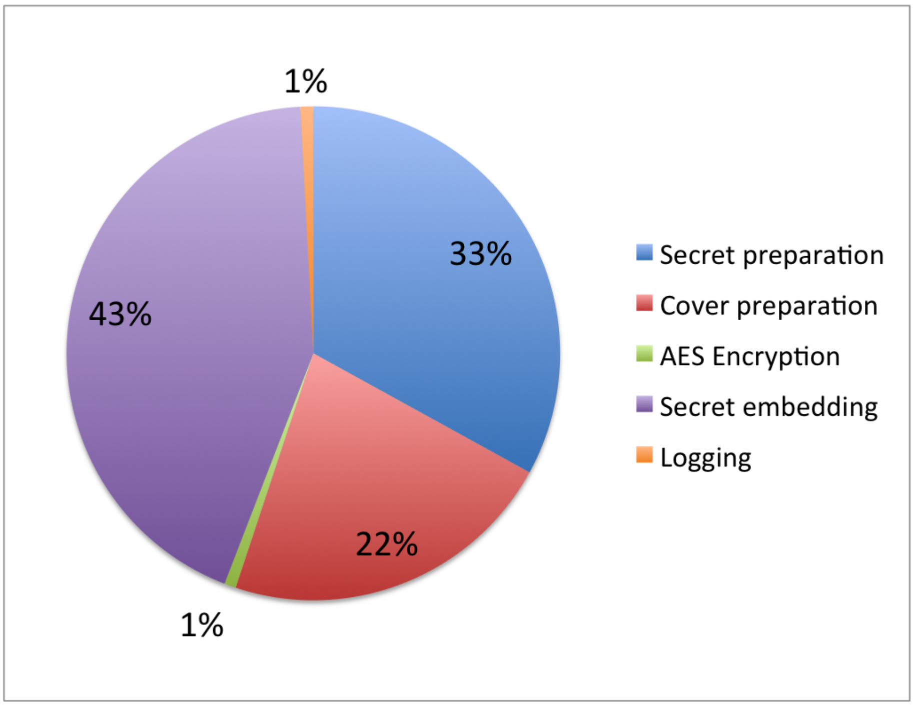


Figure 4.20: Detailed distribution of the time required to create an stego-image in ten runs of the performed tests

4.3.4 Summary

“Hide It In” is an application developed for iOS devices that allows to embed images into innocuous ones to send them by email. Its current version allows to hide pictures taken with the device camera up to 135 KB. If the size of the picture is greater the image is recompressed with JPEG. Performance tests show that executing steganographic applications in mobile devices is feasible and can be done in a reasonable time. Even further, some programming APIs allow to covertly take images with the camera, hindering the detection of malicious usage of the devices. Additionally, the difficulties to manage the mobile devices of the whole organization (along the personal devices) makes this kind of devices very difficult to control. This can led into the usage of mobile devices as a new leakage vector inside organizations.

However, the steganographic algorithm implemented in “Hide It In” is not secure. This means, that detection mechanisms could be incorporated into the organization infrastructure to detect (with high accuracy) the transmission of stego-images created with this software. The feasibility of incorporating these mechanisms is not inside the scope of this PhD, but it is an issue that should be taken into account to continue this research line. Note that “Hide It I” is only a proof of concept. In fact, the steganographic algorithm provided by “Hide It In” can be modified to improve its security. New performance tests and security analysis should be performed in this case.

4.3.4.1 Improvements

The current version of "Hide It In" demonstrates that it is possible to implement usable steganographic applications in new devices such as smartphones with a reasonable performance. Nevertheless, some of the features offered by the current version should be improved:

- *Improve the security of the steganographic algorithm*: Although the current version of the application is a proof of concept tool to demonstrate the feasibility of using steganography in smartphones, the security of the steganographic algorithm should be improved. This could be achieved implementing more secure steganographic algorithm proposals or by designing a new one.
- *Increase the amount of data to hide*: Incrementing the amount of information to be hidden in each image would reduce the security of the stego-system. Therefore, in order to increase the amount of information to be hidden, more images would be required as cover. Once a user selects the secret, the number of images required to hide that secret would have to be selected by the user. Once all the embedding has been performed, the user would be able to send all the stego-images by mail. When recovering the secret, the user would have to select all the images (stego-images) that contain a part of the secret.
- *Allow to hide other file types*: The current version of the application only allows to hide images taken from the camera. However, the current embedding and revealing functions do not constraint the kind of file that can be embedded inside an image. In this regard, it could be possible to allow the application to use other file types such as PDF, Word documents, spreadsheets, etc. as secrets.
- *Use new leakage vectors*: In the current version of the application the only option to extract the secret from the organization is by sending an email (or by keeping the image in the phone). New options to extract the message from the organization such as posting to social networks could be implemented. In addition to this, new phone capabilities such as cloud synchronization could allow to automatically extract the image from the organization.

Chapter 5

Detecting the Usage of Steganography to Avoid Information Leakage

You hid there last time
You know we're gonna find you
Sick in the car seat,
Cause you're not up to going
Out on the main streets,
Completing your mission
Two Door Cinema Club - Undercover Martyn

5.1 Introduction

Chapter 4 describes steganographic techniques that can be used to hide sensitive information into digital media objects. Current DLP solutions analyze network traffic (also known as data in motion) or files being used at workstations (also known as data at rest and data in use) to detect sensitive data [69]. In this way, if sensitive information is detected being transmitted outside the organization control (through a leakage vector), an alert is raised and the information transmission is stopped.

However, as shown in Chapter 2, techniques to identify and classify information allow DLP solutions only to track information that can be read explicitly. In fact, if information is transformed in such a way that it is not recognized as sensitive, leakage prevention mechanisms cannot prevent its transmission outside the organization. If the malicious insider uses steganography or trusted applications to perform those changes she will hinder the detectability and traceability of her attack (Chapters 4 and 6 [14, 13]). Therefore, steganalysis techniques have to be developed, not only to study and improve the security of steganographic algorithms, but also to provide mechanisms to detect its malicious usage.

In the case of organizations holding sensitive information, steganalysis techniques could be implemented at both network and workstation level. Besides analyzing data looking for sensitive information, data would be checked with the implemented steganalyzers (distinguishers) to detect the existence of hidden information. In this case, any file transmission that includes a piece of information that is detected by a steganographic distinguishers as suspicious of having hidden information should be interrupted, logged into the system and deeply analyzed before allowing its final transmission.

In this Chapter, a detector for a steganographic algorithm called *Hydan* is proposed. This embeds secret information into binary executable files. Although implementing distinguishers can serve as a tool to avoid the extraction of hidden information through the usage of steganography [52], false positives have a negative effect on employees' productivity. In this regard, active warden techniques, that try to eliminate or reduce the bandwidth of covert channels created by means of steganography, can serve to reduce the amount of sensitive information that can be covertly transmitted without reducing the employee workload due to false positives. In this Chapter, an active warden technique to avoid the usage of steganography over HTTP is presented [16]. This proposal could be easily adapted to other network protocols if required.

5.2 Steganalysis of Hydan

Hydan is a steganographic tool which embeds messages into executable files [136]. Stego-objects generated with *Hydan* have exactly the same functionality and same size as the original program. This allows embedding information in commonly used programs without raising suspicions regarding its functionality or size. As *CSteg*, this program could be used to extract sensitive information in software development environments (where system and security restrictions are softened). In this Section a steganalysis of *Hydan* is presented. The proposed steganalysis is able to tell apart files that hold hidden information through the use of *Hydan*.

5.2.1 Previous Work

Hydan [136] is the first documented tool and scheme that directly uses executable files as covers. Previously, other techniques have been used to insert hidden information into source code files. The goal of these was only to watermark source code files for copyright protection purposes. Source code watermarking techniques involve access to source code, where programmers insert copyright marks and integrity checks right inside their code. Information inserted in this way can be used to prove the integrity and authorship of the program [137].

Besides *Hydan*, other works have also addressed steganography techniques to secretly embed information inside executable files [138, 139]. In [138], the authors identify four different executable code modification techniques that could be used to embed secret information. *Instruction Selection* consists in replacing some of the instructions in the executable file for others with the same functionality. Depending on the instructions selected, a different value can be encoded. This approach is similar to the one used by *Hydan* to embed information. *Register Allocation* modifies the registers used by some instructions to embed information in the same way. *Instruction Scheduling* consists in changing the order of non-dependant instructions. In this case, the information is embedded on the order of those non-dependant instructions. Finally, *Code Layout* uses the order of big blocks of instructions (functions, etc.) to embed information. All executable code modifications are done in such a way that the resulting code still retains the exact same functionality.

The authors implemented all the proposed techniques in a tool called *Stilo*. A steganalysis of *Stilo* is proposed in the same paper based on a concept named *Code Transformation Signature*, which is defined as the set of characteristics that can be used to detect the presence of hidden information into *Stilo* executable files. The authors described the *Code Transformation*

Signatures for *Stilo* and proposed a group of countermeasures to avoid them. The authors also mentioned *Hydan*, but they did not perform a steganalysis nor reveal the corresponding *Code Transformation Signatures*. In this Chapter, the main properties (its *Code Transformation Signatures*) that can be used to detect executable-files with hidden information through *Hydan* are described. Based on those properties, a very efficient distinguisher is proposed.

Apart from this work, authors of [139] propose to hide information in special sections of Windows Portable Executable files. In their work, Shin et al. proposed the usage of the ".text" section of an executable file to embed as much information as required. This section, which is where the program code is hold, is modified to hold as much information as demanded (modifying also the pointers that point to the beginning of the real code). Although this approach allows to embed as much information as required, it does not provide any kind of security, as file size will increase exactly with the size of the secret.

5.2.2 Basics of Hydan

Hydan uses the "redundancy" on the instructions sets of executable files to introduce hidden information. Specifically, *Hydan* uses the concept of *functionality-equivalent instructions*. A set of *functionality-equivalent instructions* is a group of instructions in which any instruction of the group can be replaced by other without loss of functionality.

Consider two instructions I_a and I_b which operate over certain registers or memory addresses (input), producing a change of state on any of them (output). Let be x_i the input to a given instruction (register, literal or memory address) and let $F(I, X = \{x_1, x_2, \ldots, x_n\})$ be the result over X of executing the assembler instruction I over X. Therefore, two instructions I_a and I_b are functionally equivalent, if $F(I_a, X = \{x_1, x_2, \ldots, x_n\}) = F(I_b, X')$ being $X' = \{f_1(x_1), f_2(x_2), \ldots, f_n(x_n)\}$ and f_i are a functions that modify the value of x_i. Therefore, $\{I_1, I_2, \ldots, I_n\}$ is a set of *functionality-equivalent instructions* if $F(I_1, X) = F(I_2, X') = \ldots = F(I_n, X'^n)$.

For example, to add a certain amount to a specific register it is possible to use *add, r1, 8* or , equivalently, use *sub, r1, -8*. In this case, the *add* instruction could encode the bit value 0, and the *sub* instruction may encode the bit value 1. Depending on the size of the *functionality-equivalent instructions* sets it is possible to encode more than one bit with one instruction. A set of four *functionality-equivalent instructions* would allow to codify 2 bits (00, 01, 10 and 11). Generally, with a set of n equivalent instructions it would be possible to encode $\lfloor \log_2(n) \rfloor$ bits. Table 5.1 describes the *functionality-equivalent instructions* groups and number of instructions in each of the groups for the *x86* set, which is the most common and the one used by *Hydan*.

The embedding process of *Hydan* is done in two steps. First step encrypts the message to be hidden using *AES* or *Blowfish* with a password provided by the user. In the second step, the encrypted message is embedded into the executable file. Specifically, *Hydan* works as follows: Once the message has been encrypted, *Hydan* searches for possible places to introduce information. Then, *Hydan* generates a random number seeded with the password entered by the user. This number is used to select which of the selected places of the executable file will be used to hide the information. With this mechanism, the password will be needed to recover the data and different passwords will lead to different placements of the embedded information. The recovery process first extracts the encrypted message from the executable file. Then, the message is decrypted using the provided password.

With *Hydan*, it is possible to embed (on average) 1 bit of information per 110 bits of executable code ($R_{Hydan} = 1/110$). In fact, it is possible to embed different ratios of information,

Group	**N. Inst.**	**Group**	**N. Inst.**	**Group**	**N. Inst.**
toac8	5	*toac32*	5	*rrcmp8*	2
rrcmp32	2	*toasxc8*	7	*toasxc32*	6
addsub8	2	*addsub8-2*	2	*addsub32-1*	2
addsub32-2	2	*addsub32-3*	2	*xorsub8*	4
xorsub32	4	*add8*	2	*add32*	2
adc8	2	*adc32*	2	*and8*	2
cmp8	2	*cmp32*	2	*mov8*	2
mov32	2	*or8*	2	*or32*	2
sbb8	2	*sbb32*	2	*sub8*	2
sub32	2	*xor8*	2	*xor32*	2
and32	2				

Table 5.1: Groups of *functionality-equivalent instructions* used in *Hydan* and number of instructions in each set

but El-Khalil proposed the specified one as the better trade-off between security and capacity [136].

Hydan changes perceptibly the content of the executable files with hidden information. Therefore, if these changes lead to a specific signature, it could be possible to build a system that is able to distinguish a *Hydan* executable file from any other executable file. This signature may be shown in many different ways. Next Section discusses the possible methods to detect a *Hydan* modified executable and proposes a very efficient distinguisher to detect a *Hydan* covert-channel.

5.2.3 Steganalysis of Hydan

Changes introduced by *Hydan* into assembler code can modify different properties of the original executable file. *Hydan* does not change the size of the stego-object, but it changes the code itself. If the original program is available it will be possible to check through integrity checks (CRCs [140], hash functions [141], etc.) if the executable file has been modified, but these cannot be used as proof of the existence embedded information. Other properties such as execution time, flag activation and copyright marks checks, can prove that executable code has been modified, but again cannot be used as proof of the existence embedded information.

However, most compilers often produce similar sets of instructions. Thus, if a compiler has to select between two instructions with the same functionality it will usually select the same instruction. This property of most compilers allows building a profile of *clean* applications based on the probability distribution of instructions inside *clean* programs. Changes made by *Hydan* may lead to another probability distribution of instructions. If these changes can be profiled and generalized, it would be possible to detect if an executable file has hidden information. The proposed steganalysis of *Hydan* is based on this approach.

The performed steganalysis has resulted in a distinguisher that is able to detect executable files with embedded information through *Hydan*. To construct this distinguisher, first a statistical model of *clean* executable files has been built 5.2.3.1. Then, a set of concealment operations has been performed in a variety of executable files. In order to build the distinguisher, the main differences between the set of *clean* executables and the set of *Hydan* modified executables have been analyzed 5.2.4.

5.2.3.1 Statistical Analysis of Clean Executable Files

The proposed distinguisher is based on the presence of unusual sets of instructions on executable files. A statistical analysis of a set of 1261 *clean* executable files has been performed. These executable files have been retrieved from */usr/bin* and */usr/sbin* of an *Ubuntu x86* distribution. Figure 5.1 shows the frequency distribution of the *functionality-equivalent instructions* sets for the set of analyzed files. This distribution specifies the probability for a random instruction to belong to a *functionality-equivalent instruction* set. Depending on this distribution, the bandwidth of the covert channel offered by an executable may differ a lot. The bigger is the proportion of instructions belonging to a big set of *functionality-equivalent instructions*, the bigger will be the information *Hydan* is able to hide.

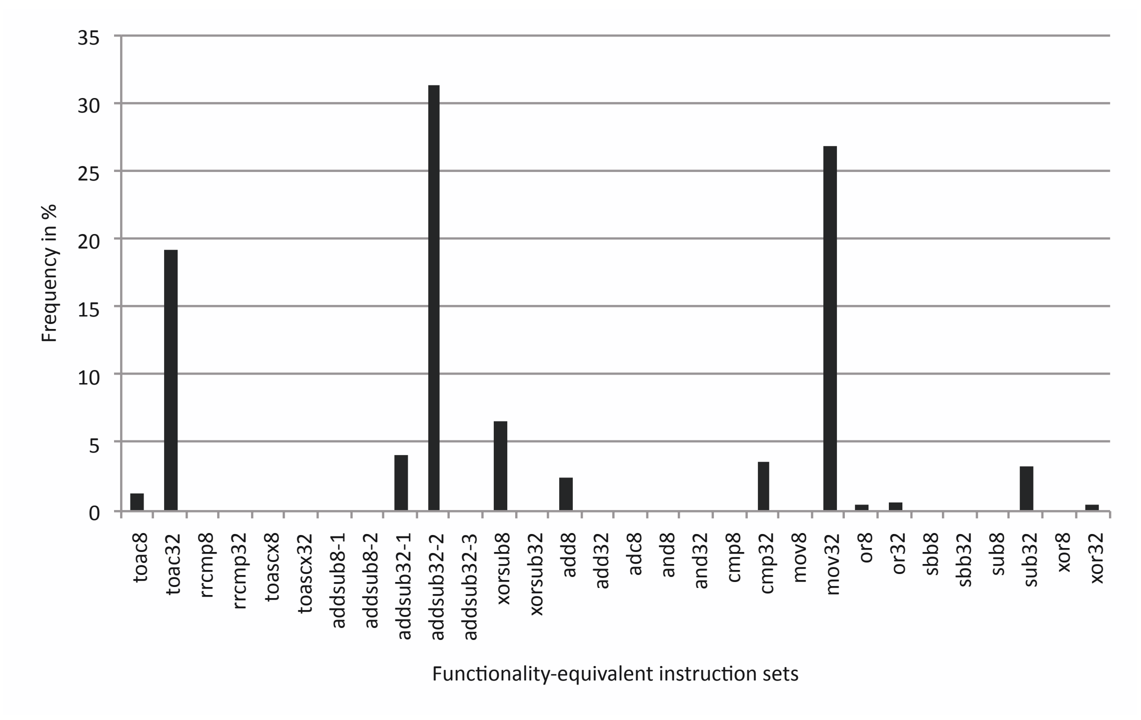


Figure 5.1: Frequency distribution of *functionality-equivalent instructions* sets

The performed analysis shows that all the *functionality-equivalent sets* of instructions are present in the analyzed test files. Nevertheless, most of the instructions found on the analyzed files belong to a small group of *functionality-equivalent instructions* sets. Therefore, the capacity of the covert channel depends on the capacity of these commonly used sets (Figure 5.1). In order to build the distinguisher statistical model, the distribution of instructions inside each of the most frequent *functionality-equivalent instructions* sets has been analyzed. In this case, all executable files are treated as a single executable in which the appearance of the different instructions belonging to each set are analyzed.

One of the most used *functionality-equivalent instructions* sets is *toac32*. This set includes five different instructions. Thus, it can encode $\lfloor \log_2(5) \rfloor = \lfloor 2,32 \rfloor = 2$ bits. Frequency distribution of instructions inside the set is shown in Figure 5.2. Results obtained in the frequency analysis of this instruction set have been gathered in Table 5.2. In all analyzed files, only one instruction of this set was used. In this case, a variation of the distribution of instructions within

Instruction	Frequency
test r/m32, r32	100,0%
or r/m32, r32	0,0%
or r32, r/m32	0,0%
and r/m32, r32	0,0%
and r32, r/m32	0,0%

Table 5.2: Frequency distribution of instructions on *toac32* set

this set would be easily detected.

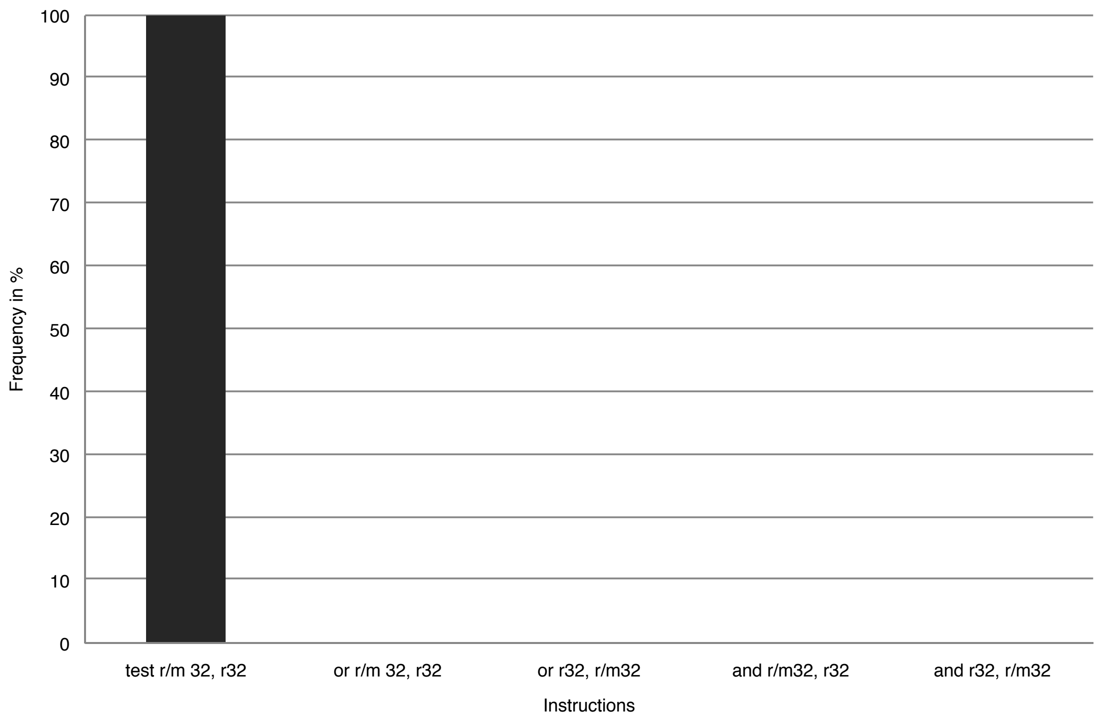


Figure 5.2: Frequency distribution of instructions on the *toac32* set

For each of the remaining sets of equivalent functions, the frequency distribution of its instructions has been computed (as in the 5.2 set). Once the frequency distribution model for each of the sets has been obtained, the proportion of instructions per set in each of the executable files has also been computed. Each of the proportions computed for each file and *functionality-equivalent instructions* set has been compared using a chi-square statistic (χ^2) against the frequency distribution of that *functionality-equivalent instructions* set calculated for all the files. For each of the *functionality-equivalent instructions* sets the average χ^2 statistic (Equation 5.1) has been calculated.

$$Average_{set_j} = \sum_{i=0}^{n} \frac{\chi^2_{file_i}}{n} \tag{5.1}$$

Where set_j is a *functionality-equivalent instructions* set, and $file_i$ is the ith file on the executable file set. Figure 5.3 shows the average χ^2 for all the *functionality-equivalent instructions* sets. For most of the equivalent instructions sets, the distribution of its instructions has remained constant in all the executable files. Thus, its average chi-square is 0. *Functionality-equivalent instructions* sets with higher average value indicate that the frequency distribution of that sets has more variability between executable files. Figure 5.3 shows six *functionality-equivalent instructions* sets that suffer lots of variability on the distribution of its instructions depending on the executable file.

Differences introduced by *Hydan* will change the frequency distribution of instructions inside each of the *functionality-equivalent instructions* sets. Comparing the new instruction distributions obtained against the reference distributions for each of the *functionality-equivalent instructions* sets will allow to determine if information has been embedded into the executable file. In fact, these will show very easily in those sets in which only one instruction was used across the set of clean executable files.

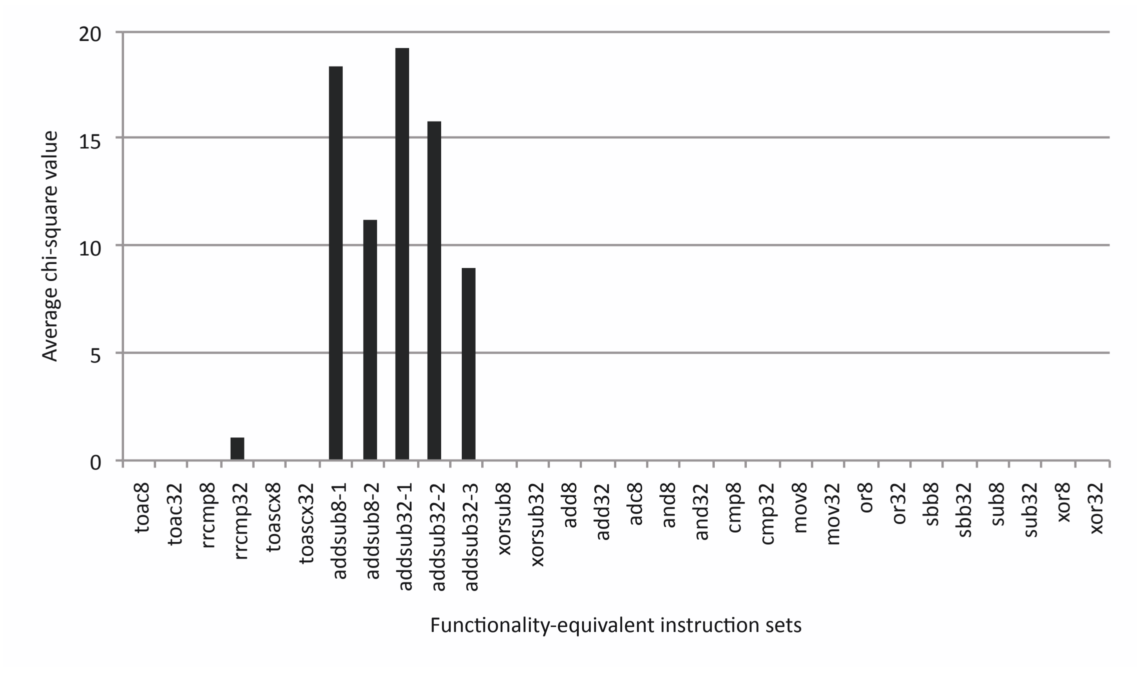


Figure 5.3: Average chi-square statistic for each of the *functionality-equivalent instructions* sets

This can be easily seen through an example. Figure 5.4 represents the differences, in terms of a χ^2 statistic, on the frequency distribution of each *functionality-equivalent instruction* set of the *apt-get* executable file with no embedded information. Differences obtained are consistent with the average shown in Figure 5.3.

Inserting information into this executable file modifies the frequency distribution of instructions inside some of the sets of equivalent instructions. Figure 5.5 represents differences, in terms of a χ^2 statistic, on the distribution of instructions inside each of the equivalent instructions sets of the *apt-get* executable with embedded information.

The frequency distribution of instructions inside the functionality equivalent instruction sets that highly variates between *clean* files has also offered high chi-square values, as in the reference (Figure 5.3) and clean file comparison (Figure 5.4). Nevertheless, distributions of

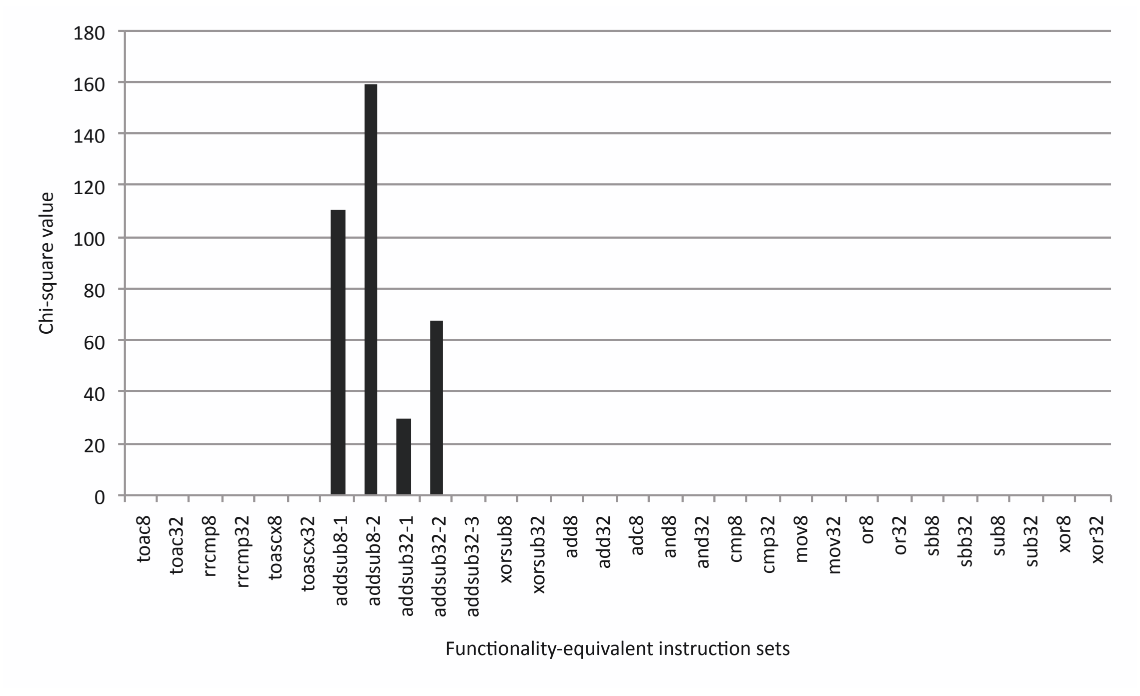


Figure 5.4: Chi-square statistics for each of the equivalent instructions sets in *apt-get*

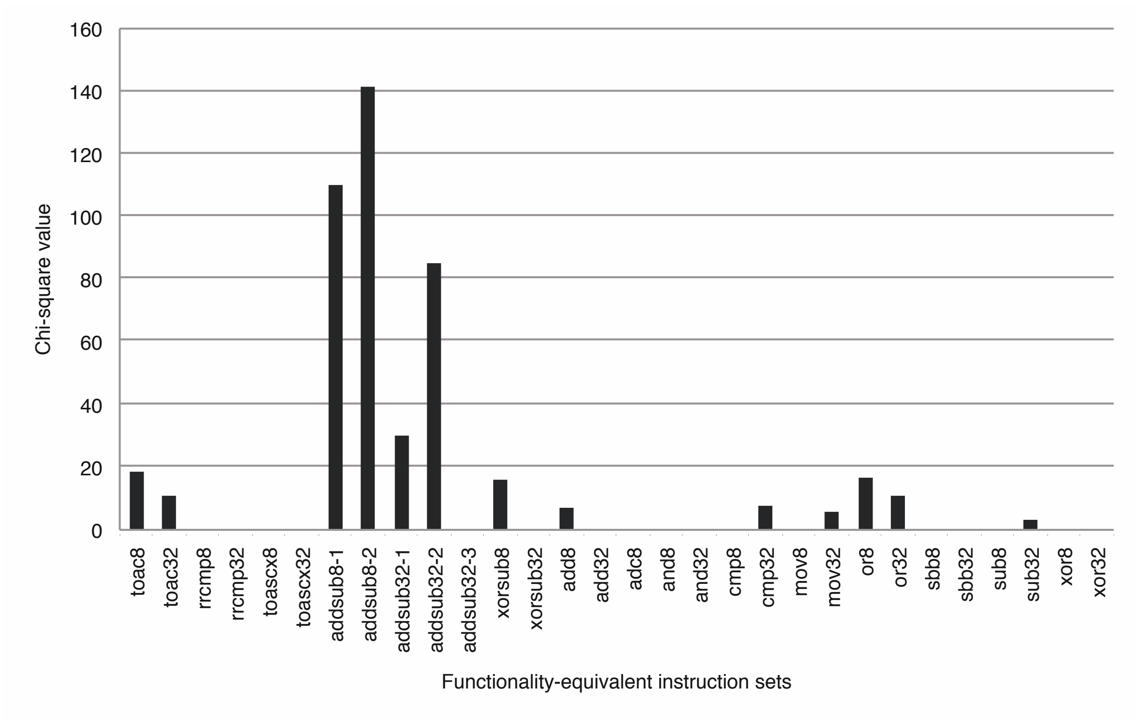


Figure 5.5: Chi-square values for each of the equivalent instruction sets in *apt-get* with hidden information

some *functionality-equivalent instructions* sets have changed and its chi-square has increased comparing it with the reference comparison (Figure 5.3) and the previous chi-square value (Figure 5.4), which was 0.

The same procedure has been performed with all the executable files, obtaining for each set a model of the frequency distribution of that set. This allows to establish which distributions of instructions inside *functionality-equivalent instruction* sets remain constant between different *clean* executable files. These results have been used to build our distinguisher which is explained in the next Section.

5.2.4 Distinguisher Design

The proposed distinguisher measures the changes on the distribution of instructions inside a selection of *functionality-equivalent instructions* sets. These measures have been made in terms of a χ^2 statistic against the reference distribution for each of the selected *functionality-equivalent instructions* sets. *Functionality-equivalent instructions* sets with high variability of instruction distribution between *clean* files have not been selected in the calculations of the distinguisher value. High variability may elevate the result offered by the distinguisher, producing false positives (marking some *clean* files as stego-objects). The distinguisher only uses the *functionality-equivalent instructions* sets which its average chi-square value is 0, as calculated in Equation 5.1. Therefore, 8 sets of *functionality-equivalent instructions* are not used: *toac8*, *rrcmp32*, *addsub8*, *addsub8-2*, *addsub32-1*, *addsub32-2*, *addsub32-3* and *xorsub8*. Mathematically, the value obtained with the proposed detector is expressed in Equation 5.2.

$$D(file) = \sum_{i=0}^{n} \chi^2_{instruction\ set_i} \tag{5.2}$$

Where n is the number of sets of *functionality-equivalent instructions* whose average chi-square value is 0. A detection threshold that maximizes the detection rate while minimizing the false positive rate has to be defined. for that purpose, the results the detector offers from three sets of files; 1) a set of clean files, 2) a set of files with embedded information using a 40 % of its capacity and 3) a set of files with embedded information using an 80 % of its capacity; has been calculated. Each set consisted on 1063 executable files. Mean and standard deviation of values obtained by the detector for the three sets have been used to stablish the detection threshold. Results obtained are shown in Table 5.3.

Distinguisher	**Clean**	**Hidden at 40%**	**Hidden at 80%**
Mean	0,000604	151,254608	299,039886
Standard Deviation	0,024571	12,298561	17,292770

Table 5.3: Distinguisher results for different sets of executable files

The threshold of the distinguisher has been established as the addition of the mean and the standard deviation of the clean files set. When a file offers a value above the expected mean and standard deviation it is marked as a stego-object. The threshold of the distinguisher is described in Equation 5.3.

$$T = Mean_{clean} + T.Deviation_{clean} = 0,000604 + 0,24571 = 0,025175 \tag{5.3}$$

5.2.5 Results

With the selected threshold a test over three sets of files, each having 1063 files has been performed. The first set of files is a selection of *clean* files from the *Ubuntu 8.10 x86* distribution. Second set of files is the set of *clean* files with embedded information up to 40% of the capacity of each file. Last set is composed by the first set of files with embedded information up to an 80% of the capacity of each file. Distinguisher values obtained for each of the files are shown in Figure 5.6.

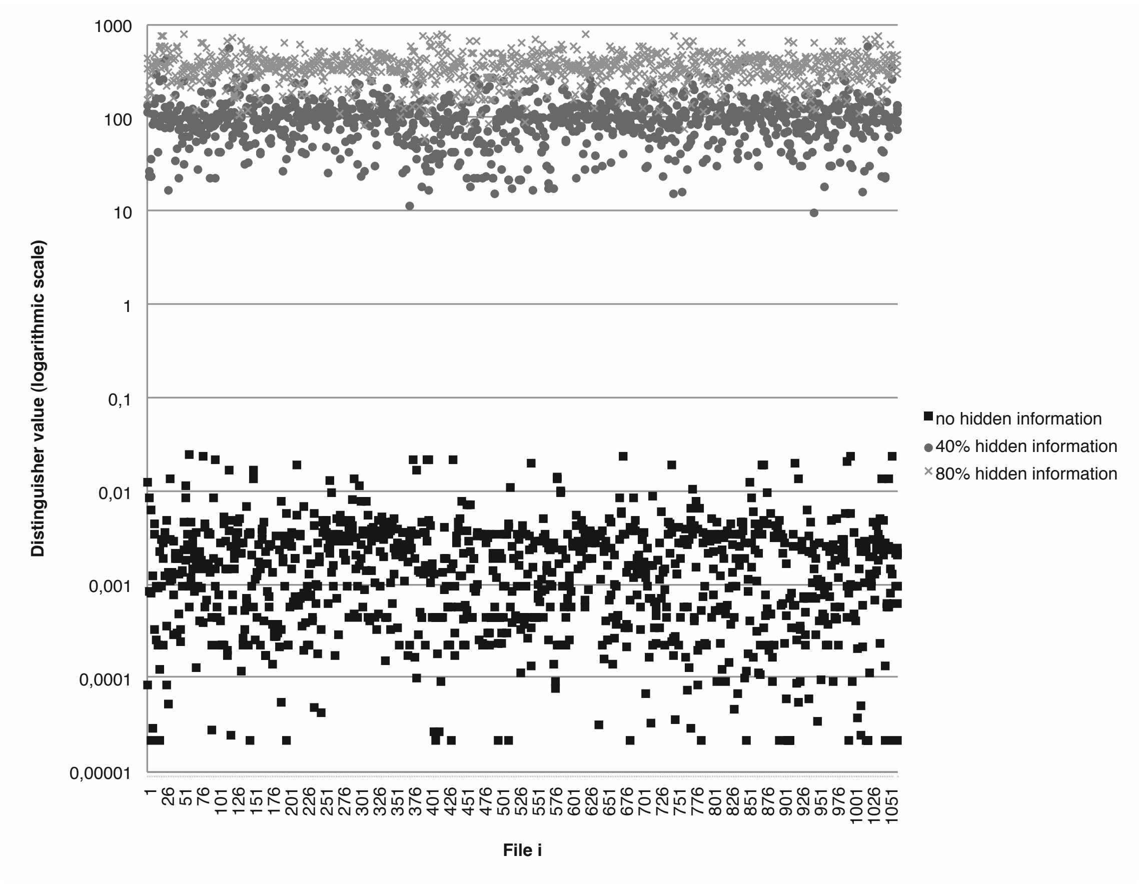


Figure 5.6: Distinguisher results for the three test sets of executable files

Values obtained by the proposed distinguisher for the clean files are separated from the ones offered by files with embedded information. Although a ROC curve could be used to select this threshold, there is no need in selecting one, as Figure 5.6 shows that with a threshold between 0,1 and 9 (logarithmic scale), the detection rate would be 1 and the false positive rate would be 0. Some results offered by embedded information files are low, but higher than the values returned by any of the clean files. In fact, the distinguisher has classified all the executables correctly (Table 5.4).

In order to produce executable files that are not detected by the proposed detector some changes should be done in *Hydan*. The performed analysis has shown that replacement of *functionality-equivalent instructions* is not secure if the frequency distribution of instructions inside a *functionality-equivalent instruction* set is constant. A first approach to secure *Hydan* would be to use only the functionality-equivalent instruction sets not used by the proposed

	Expected clean executables	**Expected embedded exec.**
Predicted clean executables	1063	0
Predicted embedded exec.	0	2126

Table 5.4: Distinguisher classification results for different sets of executable files

detector. This would reduce the capacity of hidden information up to a 35% of the original capacity. Stego-files generated this way would not be detected by the distinguiser, producing false negatives.

5.2.6 Summary

Steganalysis techniques are needed in order to ensure and improve the security of steganographic systems in the same way cryptanalysis is needed to foster the security of cryptography techniques. In this Chapter, the steganalysis of a tool that embeds information into binary executable files is described. The performed steganalysis is based on a model of files with no hidden information. In the performed tests, the proposed technique classified correctly all executable files in different proportions of concealment (0%, 40% and 80%).

The proposed detector could be implemented as part of current data leakagre protection solutions to avoid the extraction of senstive information through the usage of executable files. This system could be implemented at the network level (to inspect data in motion) or at host level (to inspect data in use and at rest). Nevertheless, improvements to *Hydan* or new steganographic tools could be used to overpass the proposed detection system. In this regard, new efforts in executable file steganalysis should be performed. For example, if the organization goals requires employees to be able to exchange and send executable files across the network, a set of policies for compiler usage could be defined by default inside the organization.

As mentioned in Section 5.2.3, compilers tend to select always the same kind of instruction from certain *functionality-equivalent instruction* sets. In this case, a model of the instruction distribution of the default compiler could be created. Any executable file that is transmitted but does not match the created compiler model should generate a security alert to the system administrator. Although some information could be embedded in executable files without raising alarms (depends on the detection threshold), the amount of information that can be embedded would has to be dramatically reduced, hindering the extraction of useful amounts of information.

5.3 Active Warden Techniques

The usage of steganalytic techniques allows to detect hidden content on transmitted objects. Literature review (Section 3.5.1) shows that proposals in this regard usually address the detection of specific algorihtms. In order to avoid information leakage through the usage of steganography, a system including several steganalytic methods (even for the same carrier) should be designed. Each piece of information leaving the organization would be passed through this detector. Depending on the quality of the steganalytic methods, several results can be given. For example, several steganalytic methods can raise an alarm for the same carrier, generating confliilcts and requiring greats amounts of time to perform forensic and audit task in order to clarify the situation.

The goal of active warden techniques is to delete the steganographic content by overwriting the possible hidden content of each transmitted object. In this way, the covert channel created by means of steganographic algorithms can be hindered or destroyed. In this regard, active warden techniques seem a more general approach to eliminate the usage of steganography, as different overwriting operations can be performed over the same object with the same result. Additionally, steganalysis techniques can be used to select only best candidates to be sanitized. In fact, if there are suspicions that an object can include information hidden by several carriers, it is possible to execute several sanitization processes over the same object. In this way, and depending on the quality of the used sanitizers, it would be possible to destroy or reduce the capacity of covert channels created by the usage of steganography

Although steganography can be used to create a covert channel through any kind of network protocol, this Sections focuses on the restriction of HTTP for several reasons:

- The restriction of HTTP traffic through firewalls 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 kinds of services can be easily used by steganography users as cover repositories and anonymous mailboxes to upload, store and download hidden information [86]. 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 unlikely that the security administrator will be able to block all of the sites.

- 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. This allows to identify the possible recipients of the hidden messages, but not the actual recipient. 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 usually involves a violation of these servers' terms of use. This kind of abuse may induce unnecessary overloads that should be actively avoided by system administrators.

This proposal enables the normal usage of HTTP connections, while hinders the creation of covert channels through these. This would allow organizations to avoid information theft through HTTP. Additionally, the proposed scheme may allow service providers to ensure clients perform an authorized usage of their services (i.e. they are not used to covertly store unauthorized material). Although this proposal is focused on HTTP, it may be easily adapted to other protocols such as SMTP, FTP, etc.

5.3.1 Related Work

Active warden techniques try to eliminate possible hidden information from communications without disrupting legitimate message exchange (See Section 3.5.4). The active warden of Simmon's scenario uses these kind of techniques to hamper covert communication between Alice and Bob [75]. 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.

Section 3.5.4 of this dissertation describes some proposals regarding the usage active wardens to avoid the creation of covert channels through the usage of steganography. The performance of the sanitization of image data has been studied in [70] with satisfactory results (175 miliseconds per image). Additionally, some proposals to avoid information leakage (see Section 2.4.3) partially use active warden techniques. In [52], HTTP headers are normalized to avoid data leakage through compromised servers.

On other contexts, commercial proxies, such as SafeSquid[1], 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 author 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 limit the capacity of steganography based covert channels, 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.

5.3.2 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 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 (actually 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

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

uses MIME types to describe the message body contents.

The composition of HTTP messages may allow the covert exchange of information between two users. Depending on the users and the intended applications, there are two main possibilities. First, 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. Second, a user establishes a covert communication between him and another user. In this case, both 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 [142].

The usage of steganography through the HTTP protocol has already been proposed to avoid censorship on the web [74]. 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 the URL of the innocuous request. The accomplice web server accesses 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 et al. [143] 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, [144] 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, a 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.

- Content: Section 3.4 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 [86] with images uploaded to Flickr as carriers.

5.3.3 A model to hinder steganography over HTTP

The proposed model, which has been named StegoProxy, aims to forbid the usage of covert channels through HTTP. In this Section, the working scenario for the proposed model is presented (Section 5.3.4) along the definitions of concepts which are used to build the model (Section 5.3.4.1). Section 5.3.4.3 describes the functional components of the model. The HTTP message sanitization process is explained in Section 5.3.4.4. Finally, the security of the proposed model is analyzed in Section 5.3.4.5.

5.3.4 Working Scenario

To illustrate the aims of the proposed system, the following scenario is considered (Figure 5.7). 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. Instead, *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 behaviour 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[2]) (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, his identity remains unknown to *Alice's* organization, as many other users may have accessed the same information (2).

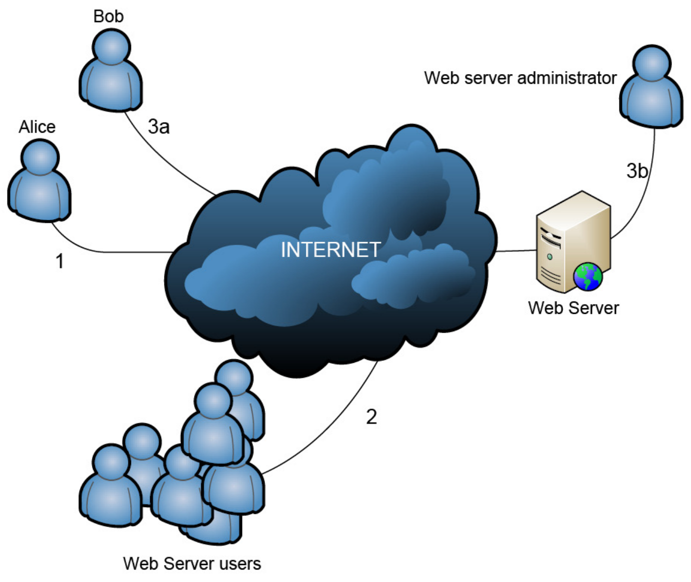


Figure 5.7: Working scenario diagram

Based on the previous scenario, the proposed system can be placed at a network infrastructure for two purposes: destroy any possible hidden communication going 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. It is necessary 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.

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

As these two networks will probably be controlled by different organizations, they both must protect against these kinds of attacks separately. The proposed framework takes this issue into account as it allows to filter both incoming and outgoing HTTP connections. Besides the previous scenario description, the following constraints are assumed:

- Systems used inside a controlled network can not be fully accessed or controlled. This may be reasonable in most scenarios 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.

5.3.4.1 Definitions

The proposed model eliminates steganographic content by sanitizing possible carriers of steganographic information. For such a purpose, the framework is based on the concepts of *Steganographic Unit* and *Minimum Request Fidelity*.

Definition 1 *A Steganographic Unit is defined as the minimum amount of semantic information that allows the transmission of hidden information, thus, the creation of a covert channel.*

Any part of a HTTP message that can be changed without modifying the semantics of the message can be considered as a steganographic unit. Steganographic units shall not be confused with the information carriers. As an example, two unstructured carriers in the form of text can refer to different steganographic units: the user agent definition and the text from a web page. Additionally, a steganographic unit may hold up more than one carrier. In fact, an image may have embedded information into their DCT coefficients (or other image characteristic) or in its textual metadata.

Definition 2 *The Minimum Request Fidelity (MRF) describes the maximum changes an object can support until it can not be considered semantically the same object [83].*

In this case, MRF is applied to steganographic units. As HTTP allows both structured and unstructured carriers, the proposed system takes into account the MRF for both kind of carriers.

5.3.4.2 Model Overview

Once deployed in an organization, the proposed model hinders the usage of steganography through HTTP. Each time a message arrives to StegoProxy, it is fragmented in several steganographic units. Each steganographic unit passes through a sanitization process that may include the execution of several sanitizers (one per possible steganographic carrier). Once all steganographic units have been sanitized, the HTTP message is built back and sent to the destination web server.

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 5.8). 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 5.9).

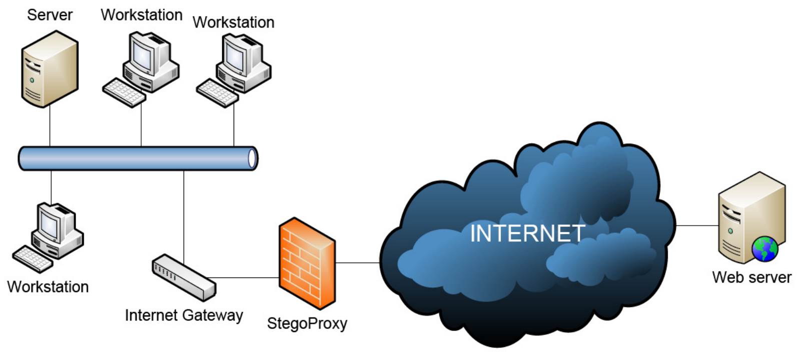


Figure 5.8: StegoProxy placement inside an organization to avoid outwards HTTP covert channels

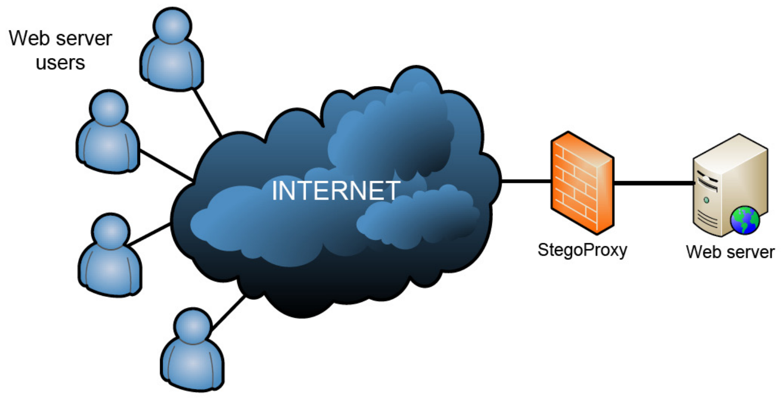


Figure 5.9: StegoProxy placement to avoid the abuse of a webserver as a steganographic mailbox

5.3.4.3 Components

As shown in Figure 5.10, the proposed system comprises four main components: the HTTP inspector, the steganographic detectors, 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 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.

Steganographic detectors

A Steganographic unit may use several carriers to covertly transmit information. A steganographic detector implements a steganalysis distinguisher for a specific carrier (it may use blind, targeted or both kinds of steganalyisis). Each steganographic unit carrier is passed through its detector to detect possible steganographic content. If hidden information is detected, the carrier is sanitized.

However, current steganalysis algorithms produce non negligible false positives and negatives [145, 146]. Thus, if a steganalysis algorithm is not accurate enough, it would be possible to transmit practical amounts hidden information. As the goal of the proposed model is to reduce the steganographic capacity of HTTP, the usage of steanographic detectors is optional. If not used for a specific carrier, all steganographic units including such carrier have to pass through a sanitization process.

Stego 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. A *targeted sanitization process* (as in targeted steganalysis) is defined 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 5.10). 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. It must be taken into account that sanitizers may be not deterministic (i.e. adding random noise to steganographic units).

HTTP Assembler

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

5.3.4.4 HTTP Message Sanitization Process

The proposed model allows to eliminate steganography from incoming and outgoing HTTP connections. The sanitization process is independent from the communication direction, as it works over HTTP messages. The following lines describe the process performed by Stego-Proxy (Figure 5.10):

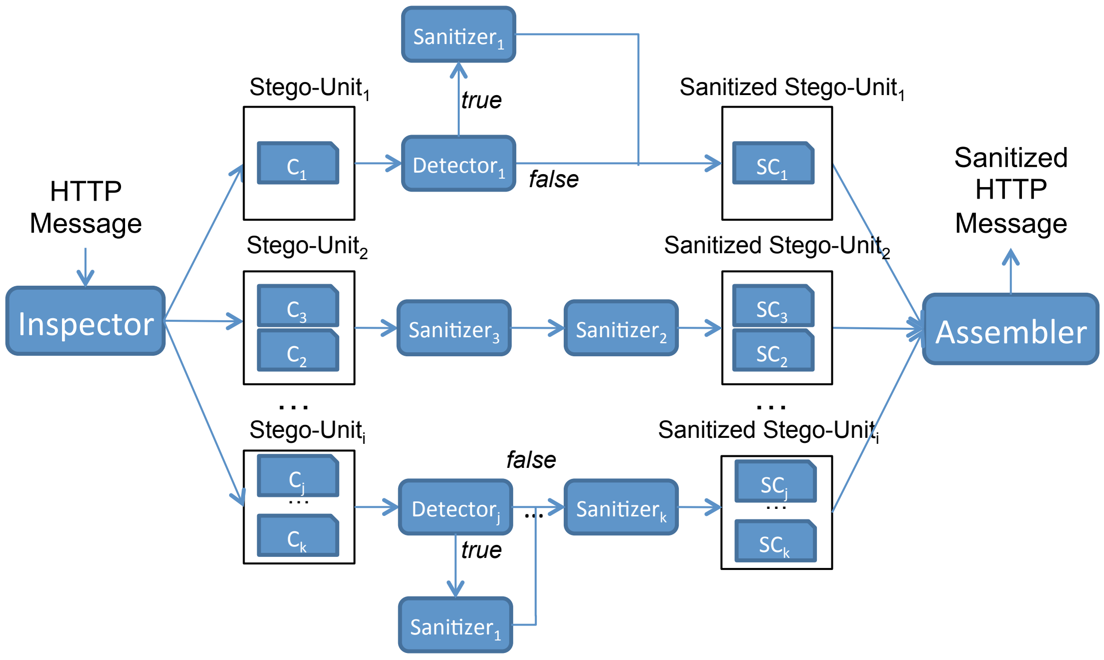


Figure 5.10: StegoProxy components and process diagram

- 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 $SU_i = \{C_1, C_2, \ldots, C_j\}$.

- For each carrier in each steganographic unit, check for hidden information with a steganographic detector. If $SD(C_j) = true$ or $SD(C_j)$ does not exist, apply sanitizer for carrier C_j.

  - Apply a sanitizer S_j to a steganographic unit to remove 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 sanitizer is composed by a set of sanitization processes (targeted or blind) which are applied sequentially $S_j = \{S_{j_1}, S_{j_2}, \ldots, S_{j_k}, \}$.

  - Once a steganographic unit SU_i has passed through all its corresponding sanitizers $S_1, S_2, \ldots, S_j$ according to their carriers, it results in a sanitized steganographic unit SSU_i.

- Once all steganographic units have been sanitized, the Assembler A ensembles the HTTP message using all steganographic sanitized units: $A(SSU_1, SSU_2, \ldots, SSU_i) = M_s$

Algorithm 2 describes the message sanitization process. This version of the algorithm includes the usage of steganographic detectors. Nevertheless, depending on the accuracy of a given steganographic detector SD_j, it may be necessary to pass all steganographic units that include that carrier C_j through the steganographic sanitizer S_j.

Algorithm 2: HTTP message sanitization algorithm

```
Data: M, HTTP message received by StegoProxy; SUDB, Steganographic unit
      databse; CDB, Carrier Database; S, Set of available sanitizers; I, HTTP message
      inspector; A, HTTP message assembler
begin
    # HTTP Message inspection
    Set $SU = \{SU_1, SU_2, \ldots, SU_i\} \leftarrow I(M)$
    Set $SU' \leftarrow SU$
    # Sanitization. For each steganographic unit
    for $SU_i$ in $SU'$ do
        # For each carrier inside the steganographic unit
        for $C_j$ in $SU_i$ do
            if $SD_j$ exists for $C_j$ then
                if $SD_j(C_j) = true$ then
                    Set $SU_i \leftarrow S_j(SU_i)$
            else
                Set $SU_i \leftarrow S_j(SU_i)$
    # Message Assembly
    Set $M' \leftarrow A(SU')$
end
```

5.3.4.5 Security Analysis

The proposed framework allows to sanitize both incoming and outgoing HTTP connections. Under the presented scenario, an attack can be considered successful if any user is able to establish a usable covert communication channel through the HTTP protocol. Three main attacks 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. [74] propose a system to evade censorship on the web. They use the Uniform Resource Locator (URL) of each request sent to an accomplice web server to hide the real URL the user wants to connect. The content of real URL is hidden in an image that is transmitted to the censored client through the HTTP response. The proposed framework is able to eliminate hidden content from the image included in the response, but can not eliminate the hidden URL transmitted by the client. Using this 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 5.3.4) we specify that the attacker does not have control over the network configuration. Nevertheless, HTTP messages could be encapsulated through other protocols such as DNS [144]. Using other protocols for such a purpose is equivalent to use them as covert channels. Applying the proposed framework to those protocols or restricting its usage through firewall rules would enforce that all HTTP messages pass through StegoProxy.

5.3.5 Evaluation

In order to evaluate the validity of this proposal, StegoProxy has been implemented 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[3].

The efficacy and efficiency of the StegoProxy implementation have been evaluated using sanitizers for several kinds of images and HTTP headers. The current implementation of StegoProxy does not use steganographic detectors. In terms of efficacy, it has been evaluated if StegoProxy is able to remove steganographic content from images sent through HTTP to web services. In terms of efficiency, the delay that the HTTP sanitization process introduces in the HTTP message exchange process has been weighed up.

5.3.5.1 Implemented Sanitizers

The current StegoProxy implementation is focused on eliminating steganographic content in images (which is the most popular steganographic carriers) and headers. Specifically it focuses on eliminating least significant bit (LSB) steganography on BMP, JPG and GIF images [147] together with steganography based on GIF shuffling [148]. Additionally, a sanitization process for different headers such as Server, User Agent, etc. has been included.

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 [149]. 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 [150].

The proposed 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 not modifying an image pixel that was used to embed information would be $p_{not\ modified} = p_{embedding} \cdot (1 - p_{sanitizing})$ where $p_{embedding}$ is the probability of embedding information into one pixel of the image and $p_{sanitizing}$ is the probability of overwriting the least 2 significant bits of a pixel (in this case $p_{sanitizing} = 1/4$). The implemented sanitizer is able to destroy hidden information of stego-images created with “Hide It In” (Section 4.3).

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

The implemented 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 $p_{not\ modified}^{pixels} = (p_{embedding} \cdot 0,75)^{pixels}$ where $pixels$ is the number of pixels in the image.

LSB JPG

This image format uses a lossy compression algorithm to reduce image size in bits. To compress an image, a discrete cosine transform (DCT) is applied to 8x8 pixel blocks. Then, the obtained amplitudes of the frequency components are quantified. Low frequency components are stored with higher resolution than high frequency components. 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 high frequency components of 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 as a consequence of 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. The proposed sanitizer avoids 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, the color index is randomly shuffled, eliminating any possible information encoded in the order of the colors.

HTTP Headers

As mentioned in Section 5.3.2, HTTP headers can also be used to transmit information. In the presented implementation, two sanitizers have been implemented for HTTP headers. The content of the following headers in each HTTP messages have been rewritten:

- *User-Agent*: is rewritten with a predefined user agent defined in StegoProxy.
- *Date*: if it does not fit the local time or a modification allowed by other time zone is randomly modified.
- *Server*: It is normalized against a set of possible servers (i.e. All servers referencing an Apache server are normalized to “Apache Server”).

- *Accept*: It is rewritten with a predefined header defined in StegoProxy.
- *Accept-Encoding*: It is rewritten with a predefined header defined in StegoProxy.
- *Accept-Language*: It is rewritten with a predefined header defined in StegoProxy.

Additionally, the order in which headers are transmitted is randomly modified to avoid the transmission of hidden information through the order of HTTP headers.

Figure 5.11 depicts the HTTP message sanitization process for the implemented StegoProxy.

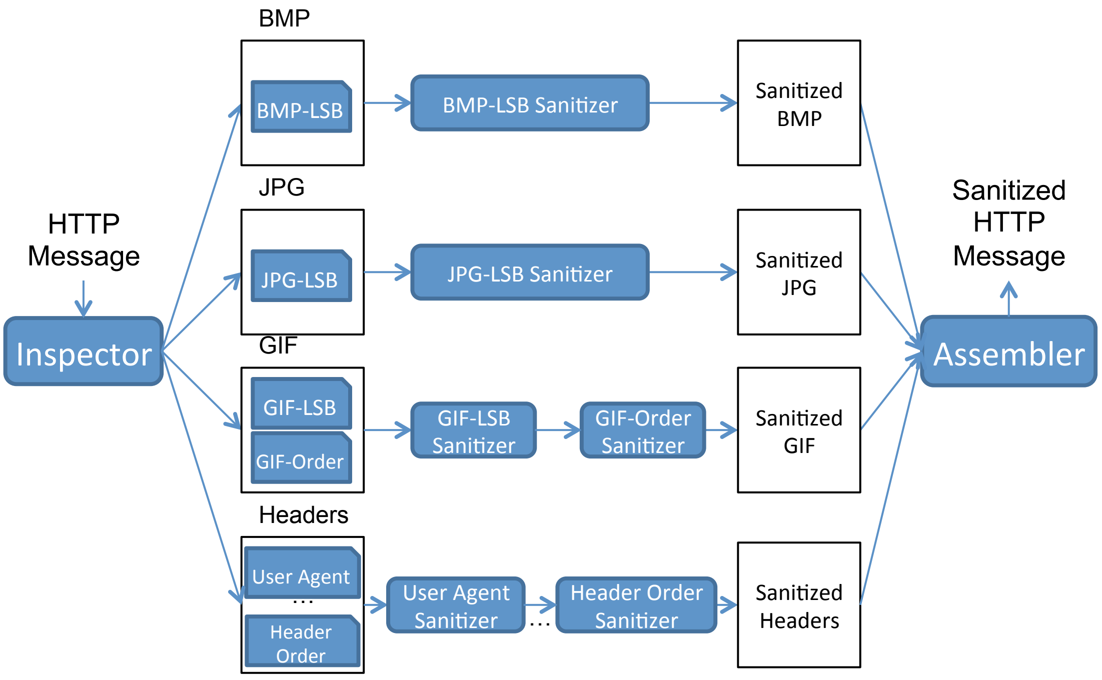


Figure 5.11: HTTP message sanitization process with implemented sanitizers

5.3.5.2 Experimental Setup

The experimental setup covers the first mentioned scenario. That is, a malicious employee that 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.

Tests have been performed using five different images. For each image, a random message using "Steghide" and "Outguess" (Table 5.5) has been embedded. Additionally, random information was embedded in the implemented HTTP headers content and order. All generated images were uploaded to Twitpic. Figure 5.12 shows the original image used in the experiments along the versions with hidden information through the aforementioned steganographic programs.

Table 5.5: 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

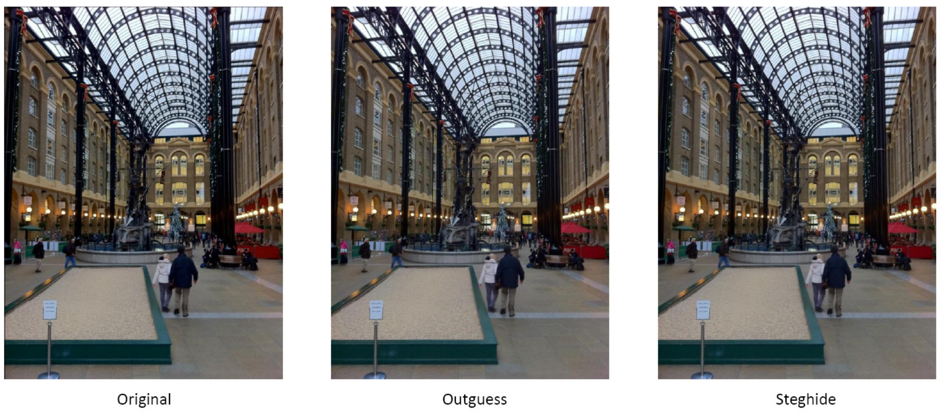


Figure 5.12: Example of image used during the experiments (Image 3) in its original form, with embedded information through Outguess and with embedded information through Steghide.

The time consumed to complete the POST request when using and not using StegoProxy has been measured. For the experiments performed with StegoProxy, it has also been calculated the time it took to dissemble the HTTP requests, to sanitize and to ensemble them again. Figure 5.13 shows graphically the aforementioned time gaps. Each image upload was repeated 5 times. In the performed experiments, StegoProxy runs on a Core 2 Duo machine at 3 Ghz while requests are generated by other computers in the same network segment (with no significant traffic at the time when experiments were performed).

5.3.5.3 Results

All images sent to Twitpic were later downloaded to check for hidden content. It was not possible to extract the hidden content from images sent through StegoProxy. Information transmitted inside the HTTP headers was lost due to the header rewriting operation performed by the HTTP header sanitizer. Results show that StegoProxy is able to eliminate steganographic content from images transferred through the network without interrupting HTTP communications or introducing any human perceptible distortion on sanitized images. However, the sanitization process increased significantly the time required to upload the images to Twitpic.

Figure 5.14 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.

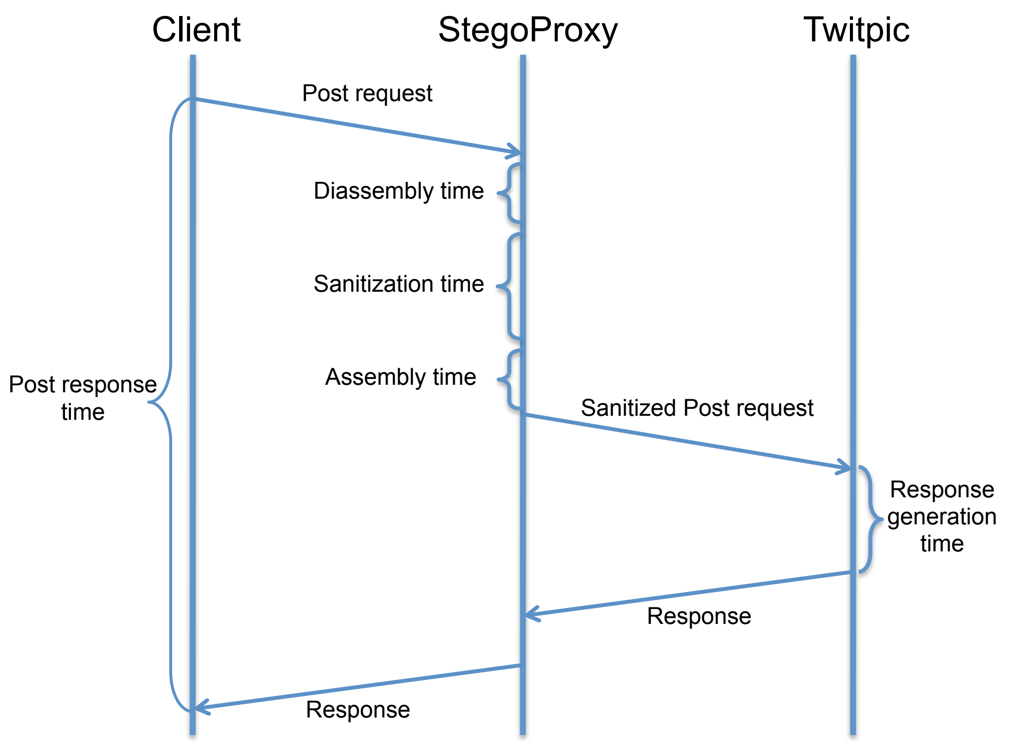


Figure 5.13: HTTP message process during the experiments and measured time gaps

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.

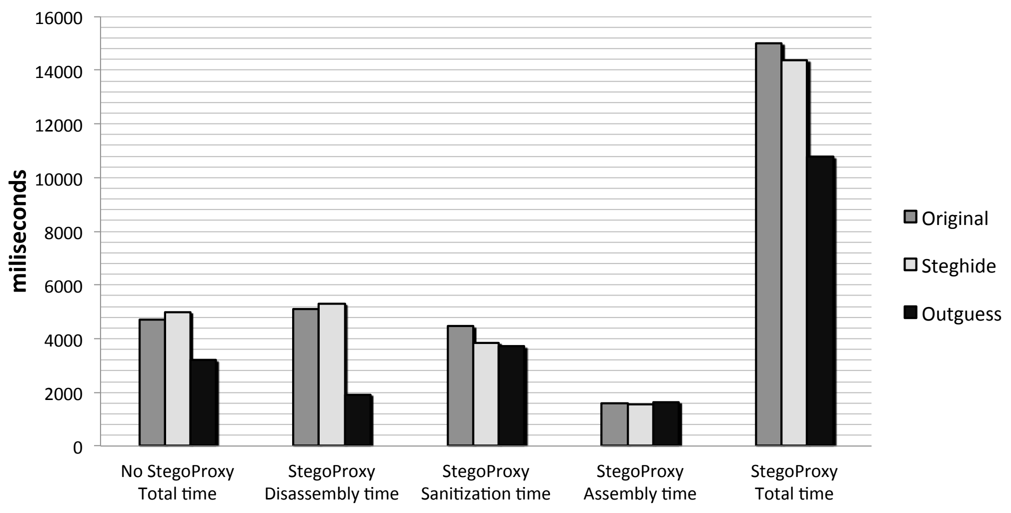


Figure 5.14: Average times measured during StegoProxy evaluation

Although this result may reduce the feasibility of the proposed approach, images used during the experiments were high resolution images 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 deployed in multiple machines to improve its performance.

5.3.6 Summary

In this Section the usage of steganography over popular network protocols such as HTTP has been addressed. A framework to limit the usage of steganography and covert channels through HTTP has been proposed. Using steganography over HTTP could allow malicious employees to steal sensitive information from their organizations. 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 cannot block this kind of covert channels effectively, as the HTTP protocol is almost essential for network communications and Internet connectivity.

A general framework that reduces the steganographic possibilities of the HTTP protocol has been proposed. This framework allows to implement different sanitizers that eliminate hidden content from any kind of information transmitted through HTTP. Although the 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. This approach allows to control both incoming and outgoing HTTP requests, being able to mitigate the malicious insider and the computer misuse risk.

The evaluation results show that the usage of StegoProxy introduces perceptible delays to users communications. Nevertheless, this implementation is far from perfect. An improved (non Java-based) implementation and a better hardware environment would significantly increase the overall performance of the current implementation. On the other hand, the main goal was achieved: it was not possible to recover any of the hidden information sent through StegoProxy. This 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.

Chapter 6

Attacks on ILP Systems Using Trusted Applications

Break me in
Teach us to cheat
And to lie, cover up
What shouldn't be shared?
All the truth unwinding
Scraping away
At my mind
Muse - Citizen Erased

6.1 Introduction

In Chapter 4, two steganographic techniques to maliciously extract information from an organization were presented. Other techniques such as cryptography could be used to steal information from an organization. Nevertheless, the goal of the presented techniques is to obtain steganographic objects that can be transmitted and extracted from the organization without being detected at the moment of the information extraction or during a post-event analysis.

As stated on Chapter 2, the high value of information assets has led organizations to implement mechanisms to prevent information leakages. One of those mechanisms are Data Leakage Prevention (DLP) solutions (Section 2.4.3.2), which are systems that are usually, installed and deployed by security vendors. To avoid the previous attacks to information confidentiality, some of these solutions implement mechanisms to restrict the applications that can be executed to a small set of applications that are strictly required by employees to perform their daily business tasks. Any other application that a user tries to execute can be stopped or raise an alarm to the security administrator of the organization.

Under this new scenario (in which a DLP solution is installed), it is much more difficult to extract information for a malicious insider. In fact, there are, to the best of the author knowledge, two available options for him to proceed.

First, the malicious insider could take advantage of a vulnerability or software misconfiguration to override the restriction imposed by these solutions or other security software (Table 6.1). Although these kind of attacks are feasible, the technical knowledge required by the attacker restricts its plausibility to scenarios where insiders have high technical knowledge.

DLP Solution	CVE	Origin	Consequence	CVSS Security
RSA Data Loss Prevention	CVE-2011-1423	XSS[1]	HTML Injection	4.3 (Medium)
Symantec Data Loss Prevention	CVE-2011-0548	Buffer overflow	Arbitrary code execution	9.3 (High)
Symantec Data Loss Prevention	CVE-2009-3037	Buffer overflow	Arbitrary code execution	9.3 (High)
Symantec Data Loss Prevention	CVE-2008-4564	Buffer overflow	Arbitrary code execution	9.3 (High)

Table 6.1: Accounted vulnerabilities of DLP software gathered from the National Vulnerability Database (`http://nvd.nist.gov/`) until April 13th 2012

Second, the malicious insider could use the set of trusted applications to help him in the information extraction process. Trusted applications used to perform daily tasks are generally executed without any restrictions. However, some of these applications offer a rich functionality and can be programmed to help an insider in carrying out an information leakage attack. In this Chapter, the usage of trusted applications to bypass current DLP systems is analyzed. A simple but effective method that could enable malicious insiders to steal information from a supposedly protected organization is shown. The presented evasion method is based on modifying the original purpose of an application using its built-in functionality. This technique does not require high technical skills or gaining administrative privileges. The presented evasion technique relies on common applications in corporative environments (i.e. Microsoft Excel) to transform sensitive information into encrypted data. Information transformed in this way can be transmitted outside the organization premises without being detected by current systems that would not allow the download or usage of any encryption software. The security and performance of this evasion method is analyzed. Additionally, the evasion technique has been tested against a well known commercial state-of-the-art DLP solution. The proposed method has been published in the Journal Computers & Security [15].

6.2 Information Leakage Protection Evasion Techniques

When DLP systems are deployed in an organization, the risk of information leakage is drastically reduced, as they are capable of properly handling most accidental information leakages. Nevertheless, the perspective of high economic benefit and the sense of entitlement to information of some employees can motivate them to steal data, even when they perceive a risk of being caught [6, 151].

Depending on the position of the malicious insider inside the organization, it may be easier for him to steal information. On one hand, system administrators and other IT-related staff usually have direct access to most systems inside the organization. This allows them to steal information more easily. As an example, during 2006 and 2007 an HSBC employee was able to steal 24.000 records from private banking clients in Switzerland [152]. Client records were delivered to French authorities to prosecute tax dodgers. On the other hand, users with limited privileges may be able to steal only the information they can access. In this case, in order to pass unnoticed to deployed protection systems they will have to use a non controlled leakage vector or an evasion technique.

The search of uncontrolled leakage vectors is a difficult task (that should be done on a test environment to avoid detection) but the complexity of current computer systems makes also difficult to control all of them. In fact, there exist some DLP solutions that do not correctly handle all the system's leakage vectors. In [153], an attack to extract information using non controlled leakage vectors is described. The author configures the DLP solution (TrendMicro

LeakProof [2]) to avoid the transmission of credit card numbers and the keywords “Boards of directors”. The DLP solution successfully stops leakages through email, but is not able to stop the transmission of the keywords and credit card numbers through the Gmail[3] and Facebook[4] chats. To the best of the author knowledge, these leakage vectors remain uncontrolled.

Regarding the usage of evasion techniques, current DLP solutions use simple pattern recognition techniques and keywords to automatically detect sensitive information (Section 2.2.2). Transforming sensitive information in such a way that patterns do not match the modified information could be used to bypass them. In fact, simple transformations (such as character or word replacing or simple mathematical operations) produce this effect. However, the security of these transformations (from the point of view of the malicious insider) is poor, as they can be easily reversed. However, simple transformations to sensitive information could be easily detected through the use of forensic hash functions (Section 2.2.2). These have been proven to correctly identify similar files that share common bit streams.

Cryptography and steganography perform transformations on information in such a way that it is necessary to know the value of a secret key to recover the original information, provided that the used algorithm is secure enough. Cryptographic suites such as GnuPG [154] or OpenSSL [155] could be used to encrypt sensitive information files. In the same way, steganographic programs such as JPHS [156] and MP3Stego [88] could hide sensitive information files into innocuously looking images or audio files. Such applications are free and easy to find on the Internet and do not require high technical skills. In this case, fingerprinting techniques and forensic hash functions (Section 2.2.2) would not detect the output of this tools as sensitive. When using cryptography an output that is completely not related to the input data is produced, and therefore not detected as sensitive. Regarding steganography, its embedding procedures embed data fragments into covers that are too small to be identified by forensic hash functions.

As previously mentioned, to avoid the execution of such applications, operating systems and DLP solutions allow system administrators to restrict application execution. These mechanisms allow system administrators to define the applications that users can run inside the organization’s computing environment. This forbids the usage of the aforementioned tools and limits the allowed ones to a very small number of trusted applications. Moreover, forensic evidence left by these tools could be used to trace back the malicious insider [109].

Nevertheless, such restrictions are not applied to programs approved for use by employees. Depending on the organization’s activity and the user role, a wide variety of programs can be approved for execution, including text processors, CAD programs, spreadsheets, etc. These applications are extremely complex systems that are constantly used for the creation and manipulation of information. DLP solutions do not consider the possibility of using these applications for malicious purposes. However, the information manipulation features included in these programs are so advanced that it is possible to transform them into cryptographic applications. Thus, users could have access to modern cryptographic algorithms that are presumably restricted. This point is proven with an Excel implementation of two widely known block ciphers: TEA and AES.

[2]TrendMicro LeakProof `http://us.trendmicro.com/us/products/enterprise/data-loss-prevention`
[3]http://www.gmail.com
[4]http://www.facebook.com

6.3 Evading ILP Systems with Spreadsheets Applications

The goal of this Section is to describe a method to bypass information leakage protection systems through the usage of common trusted applications. This method uses a commonly trusted application, Microsoft Excel, to transform sensitive information in such a way that it will no longer be recognized as sensitive. Other trusted applications could be used for the same purpose. Nevertheless, spreadsheet applications allow to implement standard ciphers as transformations, increasing the security (from the malicious insider point of view) and hindering the detection of the sensitive information. Transformations can only be reversed with the knowledge of the secret key used to encrypt the information. This enables a successful extraction of the data from the organization without detection by current DLP technologies.

6.3.1 Implementing a Spreadsheet Cipher

Spreadsheet applications allow the usage of cells to perform data manipulation, calculations through formulas and data representation for different purposes: mathematics, statistics, management, etc. This evasion method is based on the usage of spreadsheet cells to implement modern ciphers. This proposal does not rely on the use of Visual Basic Scripts, as its execution can be easily blocked by system administrators. Sensitive information can be encrypted through the usage of these “encrypting” cells, being able to export the encrypted information and use it with no restrictions. This DLP evasion technique hinders the prosecution of malicious insiders, as a secret key is needed to reveal the real nature of the extracted information.

Apart from this trivial use, we believe that this technique could be used in other environments. For example, cipher spreadsheets may be used to export certain cryptographic primitives to countries with export restrictions. As the spreadsheet cells do not constitute a program itself, export restriction laws could not be applied to it. Additionally, they could be used to encrypt information without leaving any evidence of having an actual encryption program in the computer. This could mislead computer forensic investigators and hinder criminal investigations.

The Tiny Encryption Algorithm (TEA [157]) and the Advanced Encryption Standard[5] (AES [158]) have been implemented. Both algorithms are good candidates for a proof of concept implementation because of their simplicity. Nevertheless, it must be acknowledge that TEA has been proven to be insecure [159, 160, 161]. The resulting spreadsheets are freely available, so other researchers can analyze these constructions and update them with new cryptographic algorithms.[6].

6.3.1.1 Basic operations

Spreadsheets applications such as Microsoft Excel operate with decimal numbers and do not include binary operations such as shift, rotations, bitwise XOR, etc. Standard ciphers extensively use these kind of basic operations. Thus, it is necessary to define such operations inside a spreadsheet environment. Basic bitwise transformations have been implemented using standard operations (addition, multiplication, modulo, etc.) over decimal numbers.

[5]The AES implementation is restricted to 128 bits key length.

[6]Available at `http://www.sciencedirect.com/science/article/pii/S0167404812000120`

Shifts

A shift operation is equivalent to a multiplication or division by 2^b, with b being the number of positions to be shifted. If A is the number to be shifted, a right shift operation can be described as Equation 6.1.

$$A >> b = \frac{A}{2^b} \tag{6.1}$$

This can be implemented in Excel through the following formula:

```
=QUOTIENT(A;POWER(2;b))
```

Similarly, the left shift operation is given by Equation 6.2.

$$A << b = (A * 2^b)\ mod\ 2^{32} \tag{6.2}$$

which can be implemented as:

```
=MOD(A*(POWER(2;b));POWER(2;32))
```

Rotations

Rotations are quite similar to shift operations, but bits shifted out are inserted at the other end of the number. Implementing this in decimal format requires to add the shifted bits to the result previously obtained from the shift operation. The right rotation would then look like Equation 6.3.

$$A >>> b = \frac{A}{2^b} + (A\ mod\ 2^b) * 2^{32-b} \tag{6.3}$$

This could be implemented with the following formula:

```
=QUOTIENT(A;POWER(2;b))+(MOD(A;POWER(2;b))*POWER(2;32-b))
```

Similarly, left rotation is given by Equation 6.4.

$$A <<< b = (A * 2^b)\ mod\ 2^{32} + \frac{A}{2^{32-b}} \tag{6.4}$$

And can be implemented as:

```
=MOD(A*(POWER(2;b));POWER(2;32))+QUOTIENT(A;POWER(2;32-b))
```

Exclusive OR (XOR)

Bitwise operations require to convert numbers stored in cells into vectors, so bit-to-bit comparisons can be performed. Using an auxiliary vector (P), a decimal number can be converted into a bit vector to perform the XOR operation. Equation 6.5 describes the implementation of the XOR operation in a spreadsheet environment. Let $P = \{2^1, 2^2, \ldots, 2^{31}\}$.

$$A \oplus B = \Sigma_{i=0}^{32}((([\frac{A}{P_i}] - 2 * \lceil\frac{\lceil\frac{A}{P_i}\rceil}{2}\rceil) + (\lceil\frac{B}{P_i}\rceil - 2 * \lceil\frac{\lceil\frac{B}{P_i}\rceil}{2}\rceil))\ mod\ 2) * P_i \tag{6.5}$$

This can be implemented in a spreadsheet cell by:

```
=SUMPRODUCT(MOD(INT((A/P)-2*INT(INT(A/P)/2))
 +(INT(B/P)-2*INT(INT(B/P)/2));2);P)
```

For the sake of clarity P has been defined as a reference in the formula presented in this paper. Anyway, in any spreadsheet, the vector P can obtained by replacing its reference with the following:

```
P = 2^(32-ROW(INDIRECT("A1:A32")))
```

These three operators suffice to implement both TEA and AES. Different bitwise operations present in other algorithms can be implemented accordingly. In the following the implementation process is described in more detail.

6.3.1.2 Defining Input and Constants

While TEA works with 32-bit integers, AES works with 8-bit integers. This means that in the spreadsheet implementation, a cell used to implement TEA will hold 32-bit integers and a cell used to implement AES will hold 8-bit integers. Therefore, depending on the algorithm to be implemented a different number of cells will be required to define its input and constants. In this implementation, 2 cells are required to define each plaintext block for TEA and 16 cells are required for AES.

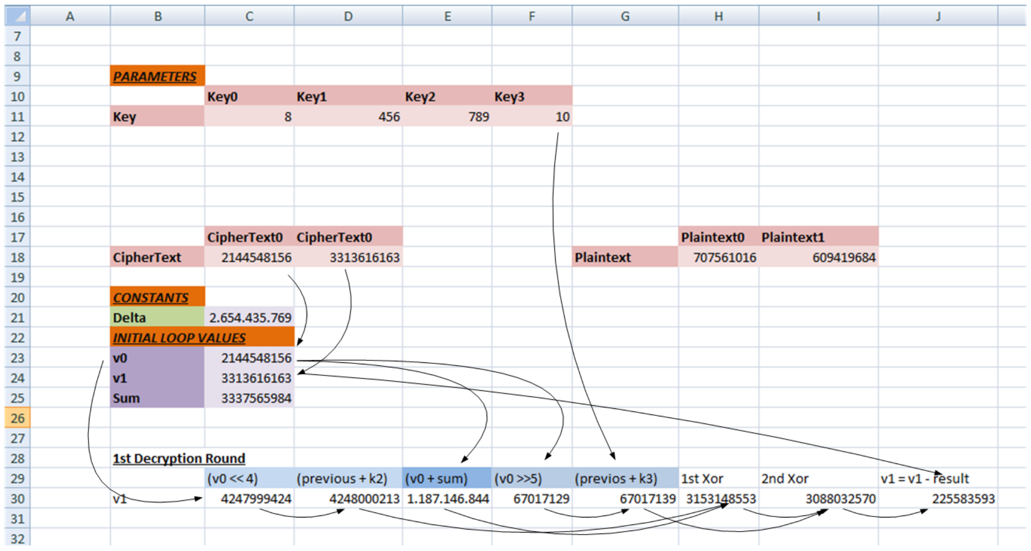
As most ciphers, the cipher implementation receives its input as integer numbers, producing also numbers as output. Depending on the data to encrypt, it may be necessary to preprocess the input before passing it to the cipher. This process can also be performed using the spreadsheet application. Text to be encrypted can be transformed to integers using the CODE() function, which converts characters to its ASCII value. To get back the original text, the CAR() function can be used. By using these functions, the spreadsheet can encrypt any sensitive information field in numeric or textual form. Furthermore, if data is not be disposed in blocks (cells) of the required length (32 bits for TEA and 8 bit for AES), the MOD() function can be used to divide the input integer in the required n bit blocks. This approach has been used to leak several fictional (but valid) social security numbers from a system protected by a modern DLP solution (see Section 6.4.3).

6.3.1.3 Building Lines of Code

In a computer program, a line of code usually performs one or more basic operation and assigns the result to a variable in memory. Similarly, in the case at hand, each basic operation can be implemented using one cell and the corresponding spreadsheet formula. However, complex operations can be also written in just one cell by replacing references to other cells with the formula included in that cell. Thus for example, Figure 6.1 shows the implementation of a line of code that calculates the value of the first 32 bit of round i (v_0) in TEA.

6.3.1.4 Rounds

Modern ciphers extensively use the concept of rounds. While rounds can be implemented using iterative clauses provided by computer languages (i.e. *for*), in the spreadsheet domain, to add a round, it is just necessary to copy and paste the cells containing the operations from the previous round. If references between cells are correctly defined, it will not be necessary to perform any changes in the copied cells. Once a full version of the cipher has been created,



	A	B	C	D	E	F	G	H	I	J
7										
8										
9		PARAMETERS								
10			Key0	Key1	Key2	Key3				
11		Key	8	456	789	10				
12										
13										
14										
15										
16										
17			CipherText0	CipherText0				Plaintext0	Plaintext1	
18		CipherText	2144548156	3313616163			Plaintext	707561016	609419684	
19										
20		CONSTANTS								
21		Delta	2.654.435.769							
22		INITIAL LOOP VALUES								
23		v0	2144548156							
24		v1	3313616163							
25		Sum	3337565984							
26										
27										
28		1st Decryption Round								
29			(v0 << 4)	(previous + k2)	(v0 + sum)	(v0 >>5)	(previos + k3)	1st Xor	2nd Xor	v1 = v1 - result
30		v1	4247999424	4248000213	1.187.146.844	67017129	67017139	3153148553	3088032570	225583593
31										
32										

Figure 6.1: A detailed view of the implementation of the first round of TEA

copying the whole algorithm (i.e. the cells) to other locations of the spreadsheet will allow to encrypt another block of information (64 bits in TEA and 128 in AES). This can be used to encrypt several blocks of information in the same spreadsheet. The amount of information that can be encrypted using a single spreadsheet is analyzed in 6.4.2.

6.3.2 Stealing Information With a Spreadsheet Cipher

If the spreadsheet application is trusted, which is an assumption, it will mean that its use is required to perform business-related activities. Then, the required knowledge of the application to carry out the attack can be taken for granted by all employees enabled to execute it. Employees that want to steal information through this method will only require temporary access to the information. As shown by [6], most employees (74% of analyzed cases) steal information they have direct access to. Therefore, that access can also be taken as granted. Algorithm 3 outlines the required steps to be performed by a malicious employee to steal information using a cipher spreadsheet.

To decrypt the stolen information, the malicious employee would have to copy the encrypted cell values to the corresponding cells in the decryption sheet, using the same password used to encrypt the information. If the information was transformed (text to number) prior to encryption, the reverse transformation should be applied on the obtained cell values.

6.4 Evaluation

This Section analyzes some aspects relative to the security and suitability of the proposed evasion technique from the point of view of the malicious insider. To obtain experimental proof of the success of the evasion technique, it has been tested against a commercial DLP solution.

Algorithm 3: Steps to steal sensitive information using a cipher spreadsheet

```
1. **Open** or **import** the file with sensitive information.
2. **Disable** the version control mechanism.
3. **Download** and **open** the cipher workbook file using the trusted application (i.e. Microsoft Excel).
4. *If* the clipboard is being controlled by the DLP solution
    (a) **Copy** the cipher cells into the sensitive information file.
    (b) **Write** the password in the corresponding cells.
    (c) **Reference** the cells whose values are to be encrypted in the cipher plaintext cells.
5. *If* the clipboard is not being monitored
    (a) **Write** the password in its corresponding cells.
    (b) **Copy** the values from the sensitive cells to the cells referencing the plaintext cells values.
6. **Copy** the ciphertext values or export them to a new file. As the information being copied is meaningless,
   no alert will be raised.
7. **Transfer** the file outside the organization through any of the available means (network, removable drive,
   etc.).
8. **Delete** the cipher workbook file and the cipher cells from the sensitive workbook (if they were copied).
```

6.4.1 Security Analysis

In order to analyze the security of the proposed DLP evasion technique, the following scenario is proposed. A disgruntled employee decides to steal sensitive information from an organization. The security policy of the organization states that sensitive files cannot leave the organization's boundaries by any means (printer, removable drives, network, etc.). To implement the security policy, the organization deploys a DLP solution that executes mechanisms to automatically detect sensitive information files and logs all attempts to extract sensitive information by any of the means defined in the security policy. Additionally, the DLP solution allows to define a set of trusted applications. Only those applications can be executed on workstations with access to sensitive information. Common applications such as the Microsoft Office suite are included in the trusted application list. For them, the installed DLP solution only controls actions that can directly produce an information leak (such as for example attempting to print a file or emailing a document).

Let $W = w_1, w_2, \ldots, w_n$ be a set of sensitive information, where w_i can be described with the regular expression exp_w. Let $C = c_1, c_2, \ldots, c_m$ be the implementation of a cipher in spreadsheet form where c_i codifies part of the cipher $Cipher(x, k)$. Both w_i and c_i are composed by a set of symbols that are accepted as content for a spreadsheet cell. Let $Sensitive_w(x)$ be a function that returns $true$ if x conforms to exp_w and $false$ otherwise. To simplify it can be assumed that $Sensitive_w(x)$ is called each time a spreadsheet cell is going to be copied into the clipboard. If $Sensitive_w(x)$ returns $true$, x can not be copied into the clipboard and the incident is logged.

As all operations are performed within a trusted application, the existence of several trusted applications being used at the same time is not considered. If a malicious insider copies C into the clipboard, $Sensitive_w(x)$ will be called m times returning $false$ for all of them. When C is pasted into the spreadsheet containing W, operations defined in C will be performed using W as input. This will result in $W' = w'_1, w'_2, \ldots, w'_n$, where $w'_i =$

$Cipher(w_i, k)$. Depending on the specific transformations performed by $Cipher(x, k)$, the result of $Sensitive_w(w_i')$ will be $false$, enabling the malicious insider to copy W out and send it outside the organization. In the case of most commercial DLP solutions, $Sensitive_w(x)$ checks x against a set of defined regular expressions and keywords. Therefore, $Cipher(x, k)$ transformations have two requirements.

1. k must be required to obtain back x from w. If $Cipher(x, k)$ can be easily reversed, the malicious insider might be able to evade the restrictions imposed by the DLP solution, but transformations could be reversed easily and new definitions of $Sensitive_w(x)$ will be able to detect the transformed information. In this case, $Cipher(x, k)$ implements both TEA and AES. The difficulty to obtain back the original information is (for both algorithms) orders of magnitude higher than any other simple transformation that could render $Sensitive_w(x)$ unusable. To the best of the author knowledge there is no publicly available attack on the full version of AES 128 that reduces the complexity of finding the encryption key to significantly less than a brute force attack [162]. In the case of TEA, [159] discovered that TEA suffers from equivalent keys. This means that each TEA key has 3 equivalent ones, reducing its complexity to 2^{126}. Other attacks exist, but still their practical implications in this setting are very limited. Nevertheless, as the key has to be introduced in the same spreadsheet, implementation attacks could be used against this proposal. However, to be able to implement such attacks, it would be necessary to detect that this kind of evasion is being used. This is addressed in Section 6.5.2.1.

2. It should generate a w that does not match with any of the defined regular expressions or keywords. This is done by adding an additional transformation at the end of the ciphering process. This transformation ensures that the encrypted data will not match any sensitive data pattern or keyword. In the current AES and TEA spreadsheet implementations, dots are introduced between each 3 digits to avoid any possible n digit pattern.

6.4.2 Performance Analysis

Implementing bitwise operations through standard decimal arithmetic functions increases the computational complexity of the cipher implementation. While in usual software implementations basic bitwise operations can be translated into a single line of assembly code, in this implementation it requires multiple operations. Spreadsheet implementations of TEA and AES have been compared with standard C implementations. Test consisted on encrypting up to 1 Megabit of information with each implementation. During tests, the machine was not executing any other task. To obtain average values, each encryption procedure was repeated 24 times. Average results show that the spreadsheet implementations are significantly (orders of magnitude) slower than C implementations (Table 6.2). In the case of AES, the Excel implementation is very inefficient, mostly due to the *MixColumns* step. Therefore, a spreadsheet application does not seem the optimal way to implement a cipher, and it is suitable only for small amounts of information or a proof of concept implementation.

In terms of memory space, the TEA implementation requires 70 cells to encrypt 64 bits of information: 2 as plaintext input, 4 as password input and 64 to implement TEA. AES requires 704 cells to encrypt 128 bits: 16 as plaintext input, 16 as password input, 512 to implement constants and 160 to implement AES operations. To encrypt more blocks, the plaintext and implementation cells must be copied. Therefore, with the current approach it is theoretically

Data	Excel AES	C AES	Excel TEA	C TEA
128 bits	0,23 sec.	$<$ 1 ms.	0,29 sec.	$<$ 1 ms.
256 bits	0,38 sec.	$<$ 1 ms.	0,33 sec.	$<$ 1 ms.
512 bits	0,66 sec.	$<$ 1 ms.	0,39 sec.	$<$ 1 ms.
1 Kb	1,35 sec.	$<$ 1 ms.	0,47 sec.	$<$ 1 ms.
2 Kb	2,70 sec.	$<$ 1 ms.	0,70 sec.	$<$ 1 ms.
4 Kb	5,28 sec.	$<$ 1 ms.	1,34 sec.	$<$ 1 ms.
8 Kb	10,53 sec.	1,6 ms.	2,68 sec.	$<$ 1 ms.
16 Kb	20,95 sec.	3 ms.	5,34 sec.	$<$ 1 ms.
32 Kb	42,23 sec.	6 ms.	10,61 sec.	$<$ 1 ms.
64 Kb	83,79 sec.	12 ms.	21,09 sec.	$<$ 1 ms.
128 Kb	171,19 sec.	25 ms.	42,22 sec.	$<$ 1 ms.
256 Kb	333,22 sec.	50 ms.	84,47 sec.	1,6 ms.
512 Kb	673,75 sec.	0,1 sec.	168,97 sec.	3,4 ms.
1 Mb	1.343,26 sec.	0,2 sec.	335,72 sec.	6,6 ms.

Table 6.2: Performance comparison of spreadsheet and C implementations to encrypt up to 1 Megabit of information (average of 24 runs)

possible to encrypt up to $2^{40} \div 66 = 16659267087$ bits for each available sheet (up to 1,93 GB) with TEA and up to $2^{40} \div 176 = 6247225157$ bits (up to 0,72 GB)[7] with AES.

6.4.3 Bypassing a Commercial DLP Solution

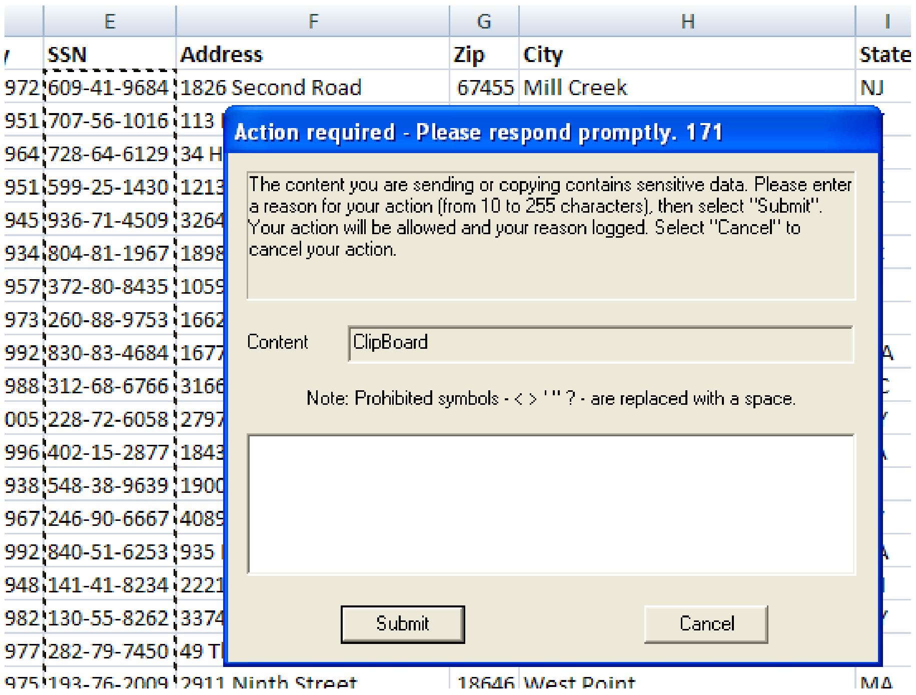
The proposed evasion method has been tested against a commercial DLP solution from TrendMicro (i.e. LeakProof v5.0[8]). LeakProof is composed by a server and a workstation agent. The LeakProof server is used by administrators to scan workstations for sensitive information, define security policies, analyze alerts sent by LeakProof agents and generate reports. The LeakProof agents crawl the system for sensitive information. Additionally they monitor and control all leakage vectors (removable drives, network, etc.). The current version of LeakProof is only able to control Windows workstations.

A simulated organization that included several workstations controlled by a LeakProof Server was built. LeakProof uses the concepts of *digital asset* and *company policy* to establish a DLP configuration. Digital assets are defined through the use of keywords and regular expressions. The tested version of LeakProof includes 21 keywords and 36 regular expressions that allow to identify possible sensitive files such as source code files, social security numbers, etc. Generative software was used to create personal data record files[9], which LeakProof correctly identified as sensitive. Company policies are built to control digital assets that are transmitted through specific channels. The server company policy was configured with compliance templates that are built in the LeakProof Server. Specifically, templates to comply with PCI-DSS (PCI Securty Standards Council, 2010), California Senate Bill 1386 of 2002 (SB 1386) and the Gramm-Leach-Bliley Act of 1999 (GLBA) were enabled. Additionally, a compliance template to avoid source code extraction and US personally identifiable information (including social security numbers) were active.

[7]Excel 2007 allows a maximum $2^{20} \times 2^{20}$ cell spreadsheets

[8]TrendMicro LeakProof v5.0 `http://us.trendmicro.com/us/products/enterprise/data-loss-prevention` - Accessed on April 15th 2012

[9]To generate fake personal records, an Excel plugin available at `http://www.codeforexcelandoutlook.com/wkbks/RandomDataGenerator_unlocked2007.xlam` was used



	E	F	G	H	I
	SSN	Address	Zip	City	State
972	609-41-9684	1826 Second Road	67455	Mill Creek	NJ
951	707-56-1016	113			
964	728-64-6129	34 H			
951	599-25-1430	1213			
945	936-71-4509	3264			
934	804-81-1967	1898			
957	372-80-8435	1059			
973	260-88-9753	1662			
992	830-83-4684	1677			A
988	312-68-6766	3166			C
005	228-72-6058	2797			Y
996	402-15-2877	1843			A
938	548-38-9639	1900			
967	246-90-6667	4089			
992	840-51-6253	935			A
948	141-41-8234	2221			
982	130-55-8262	3374			Y
977	282-79-7450	49 T			
975	193-76-2009	2911 Ninth Street	18646	West Point	MA

Action required - Please respond promptly. 171

The content you are sending or copying contains sensitive data. Please enter a reason for your action (from 10 to 255 characters), then select "Submit". Your action will be allowed and your reason logged. Select "Cancel" to cancel your action.

Content: ClipBoard

Note: Prohibited symbols - < > ' " ? - are replaced with a space.

Figure 6.2: LeakProof v5.0 alerts when trying to copy sensitive cells into the clipboard

Tests performed involved three different methods to try to leak information. First, trying to copy sensitive information from one file to another through the clipboard. Secondly, trying to mail a sensitive file using a webmail account accessed though HTTPS. Finally, using the proposed evasion technique to transform the sensitive information.

Copying sensitive information into the clipboard was detected by the LeakProof Agent, as shown in Figure 6.2. During the second test, sending a sensitive file using a webmail account, LeakProof did not allow the malicious insider to attach the sensitive file (Figure 6.3). All these operations were logged into the LeakProof server and also reported to the security administrator through his email account. Other operations such as copying to a removable drive and printing the document were also forbidden and logged.

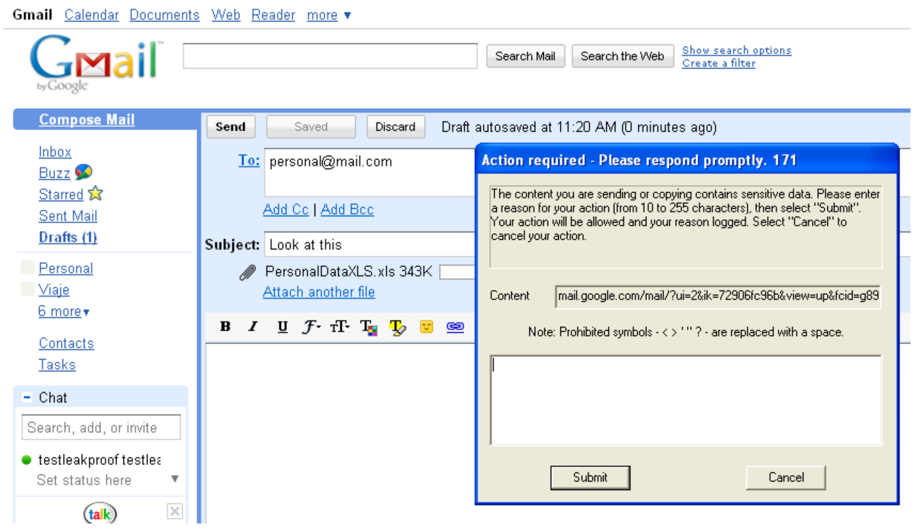
Using one of the spreadsheet ciphers and the previously defined Algorithm 3 (See Section 6.3.2), the social security numbers were successfully extracted from the simulated organization network. Figure 6.4 shows the resulting file, encrypted using the TEA spreadsheet implementation, which was transferred outside the organization using the previously forbidden leakage procedures.

In this case, as the clipboard is actively monitored by LeakProof, the cipher cells were copied into the sensitive file spreadsheet. The encrypted information did not match any of the patterns defined into LeakProof policies (see Table 6.3).

SSN Pair	LeakProof Search Pattern	ExcelTEA Result
609-41-9684, 707-56-1016	[ˆ \d-](\d{9}|\d{3}-\d{2}-\d{4})[ˆ \d-]	214.454.815,6, 331.361.616,3

Table 6.3: Example of the transformation performed on a social security number to bypass LeakProof detection mechanism

To leak other sensitive information such as names, address, etc., their values would need



Draft autosaved at 11:20 AM (0 minutes ago)

To: personal@mail.com

Subject: Look at this

PersonalDataXLS.xls 343K

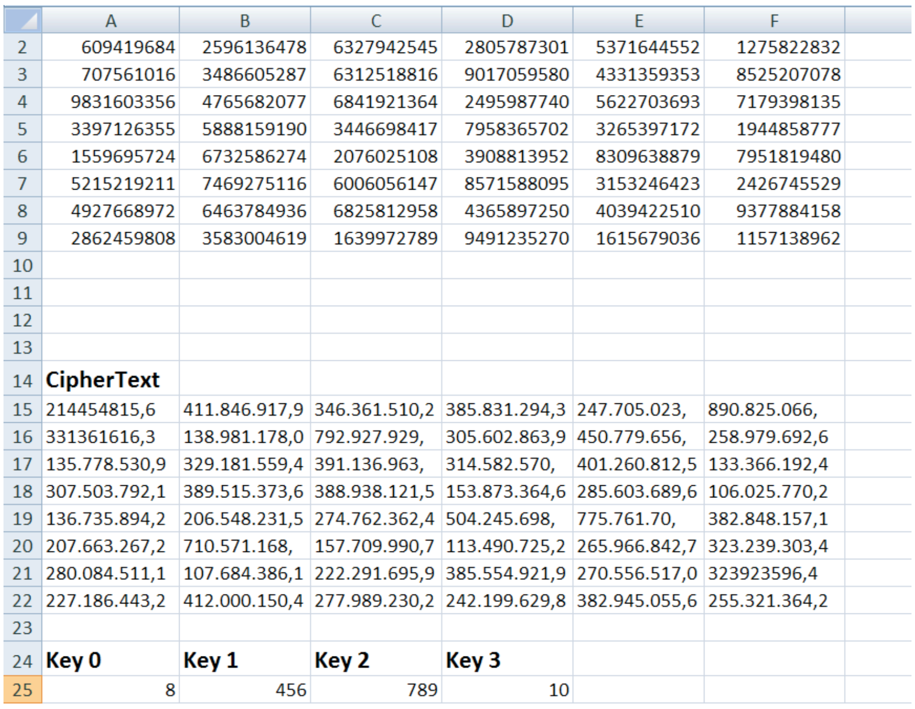
Action required - Please respond promptly. 171

The content you are sending or copying contains sensitive data. Please enter a reason for your action (from 10 to 255 characters), then select "Submit". Your action will be allowed and your reason logged. Select "Cancel" to cancel your action.

Content mail.google.com/mail/?ui=2&ik=72906fc96b&view=up&fcid=g89

Note: Prohibited symbols - < > ' " ? - are replaced with a space.

Figure 6.3: LeakProof v5.0 alerts when trying to send a sensitive file by email



	A	B	C	D	E	F
2	609419684	2596136478	6327942545	2805787301	5371644552	1275822832
3	707561016	3486605287	6312518816	9017059580	4331359353	8525207078
4	9831603356	4765682077	6841921364	2495987740	5622703693	7179398135
5	3397126355	5888159190	3446698417	7958365702	3265397172	1944858777
6	1559695724	6732586274	2076025108	3908813952	8309638879	7951819480
7	5215219211	7469275116	6006056147	8571588095	3153246423	2426745529
8	4927668972	6463784936	6825812958	4365897250	4039422510	9377884158
9	2862459808	3583004619	1639972789	9491235270	1615679036	1157138962
10						
11						
12						
13						
14	**CipherText**					
15	214454815,6	411.846.917,9	346.361.510,2	385.831.294,3	247.705.023,	890.825.066,
16	331361616,3	138.981.178,0	792.927.929,	305.602.863,9	450.779.656,	258.979.692,6
17	135.778.530,9	329.181.559,4	391.136.963,	314.582.570,	401.260.812,5	133.366.192,4
18	307.503.792,1	389.515.373,6	388.938.121,5	153.873.364,6	285.603.689,6	106.025.770,2
19	136.735.894,2	206.548.231,5	274.762.362,4	504.245.698,	775.761.70,	382.848.157,1
20	207.663.267,2	710.571.168,	157.709.990,7	113.490.725,2	265.966.842,7	323.239.303,4
21	280.084.511,1	107.684.386,1	222.291.695,9	385.554.921,9	270.556.517,0	323923596,4
22	227.186.443,2	412.000.150,4	277.989.230,2	242.199.629,8	382.945.055,6	255.321.364,2
23						
24	**Key 0**	**Key 1**	**Key 2**	**Key 3**		
25	8	456	789	10		

Figure 6.4: Social Security Numbers encrypted using the TEA spreadsheet implementation

to be translated into numeric format using the CODE() function.

6.5 Discussion

The presented attack allows malicious insiders to bypass DLP solutions and leak information from an organization. Nevertheless, a spreadsheet implementation of a cryptographic algorithm involves some restrictions that do not apply over specific purpose cryptographic applications. This section analyzes how these restrictions affect the feasibility of this method to evade information.

6.5.1 Feasibility of current DLP evasion technique

Executing encryption algorithms over a spreadsheet application enables the evasion of restrictions imposed by DLP solutions. Nevertheless, executing a cryptographic algorithm through spreadsheet cells will not execute in the same way as executing it directly using machine code. Specifically, this kind of implementation has two drawbacks that may affect its feasibility under a real scenario.

Firstly, the spreadsheet implementation is computationally inefficient. As shown in Section 6.4.2, the spreadsheet implementation of a cipher is orders of magnitude slower than a standard C implementation of the same cipher. The spreadsheet program has to first look for the order in which to executed the different cell operations. Then, it will interpret the different mathematical operations before they are executed (also over the spreadsheet program). In the case of C implementations lines of code are loaded into memory and executed directly in the system processor. Binary operations that are translated to several decimal operations have also an impact on the implementation efficiency. Additionally, as opposite to other cipher implementations, the memory requirements increase with the amount of data to encrypt: a block of cipher implementation cells is required for each block the user wants to encrypt in the same algorithm run. Encrypting large amounts of information causes an overhead on the spreadsheet application that could be easily identified as abnormal behaviour (see Section 6.5.2.3).

Secondly, the cipher spreadsheet is only able to encrypt information stored in spreadsheet cells. As these cells only allow to introduce textual information, binary information such as images or other file formats that can not be easily represented by readable characters cannot be encrypted. Thus, the presented approach only allows to extract textual information from an organization. Encrypting binary information would require the usage of additional applications to encode the binary information into cell allowed characters. Binary file editors read binary files and represent them with readable characters transforming binary information into its hexadecimal representation. This kind of programs could be used to transform binary information into spreadsheet readable information. However, its execution would be restricted in most of the workstations environments, as these applications are only used when in software development scenarios.

Information encryption restrictions together with the computational restrictions of the cipher spreadsheet, makes the presented evasion technique suitable only for small amounts of data such as personal data records or small text fragments.

6.5.2 Countermeasures to the proposed DLP evasion technique

Although the proposed attack enables bypassing current DLP solutions, there exist several techniques and mechanisms that could be used to detect the usage of the proposed technique. The following subsections discuss some of them.

6.5.2.1 Detecting the presence of cipher spreadsheets

Both the Windows registry and UNIX system logs store system status information and application messages sent to the operating system. [109, 163] proposed the use of registry entries and system logs to detect malicious insiders and suspicious applications affecting information leakage threats. Microsoft Excel generates entries on the Windows registry keys that could be used as forensic artefacts. Using Systracer[10], changes made to the Windows registry after successfully leaking sensitive information have been obtained. The only change observed was on the following key:

```
HKEY_CURRENT_USER\\Software\\Microsoft\\Office\\12.0\\Excel\\File MRU
```

This registry key, which stores the name of the recent opened files, changed to add the name of the file where the cipher was placed. That name is not enough to detect a malicious or suspicious behaviour, as it can be changed easily by the malicious insider.

Another approach to detection would be to consider the cipher spreadsheets as some sort of malware. After all, the main purpose of the cipher workbook file is arguably to steal information. Antivirus and Intrusion Detection Systems (IDS) may be adapted to detect the transmission or usage these kinds of spreadsheets on a computer system. Regarding the AES implementation, its S-Boxes could be used as a signature to detect the presence of the Excel cipher implementation. A new IDS rule could be written to detect such content into incoming network traffic.

Nevertheless, the proposed TEA and AES implementations are just an example. Virtually thousands of different files with the same purpose can be created and used easily, as well as other completely different implementations of other ciphers. This implies that it will be quite easy to develop new alternatives unknown to anti-malware software that could be employed without any risk of being detected. However, these kinds of spreadsheets share some common characteristics that could facilitate their detection, such as the usage of a short set of functions and the high amounts of nested formulas required to steal practical amounts of information.

6.5.2.2 Encrypted data detection

DLP solutions inspect contents and file metadata as well as network packets to detect sensitive information leaving the organization [69, 163]. One central property of modern ciphers is the randomness of the ciphertext they produce [164]. Randomness tests could be used to detect encrypted content being transferred out of the organization. However, spreadsheet encrypted information could be transformed to reduce its apparent redundancy or transmitted wrapped under file formats such as PDF, DOC, etc. This should be taken into account when designing encryption data detectors.

Using the software “Glavlit” [52], any file to be sent outside the organization must pass through a warden. A specific warden could be created to detect files produced with our cipher spreadsheets, usually Microsoft Excel or other format files with random contents. However, other innocent-looking information could be easily introduced into the file to confuse the warden.

6.5.2.3 Detection of abnormal behaviour

The use of the cipher spreadsheet requires the malicious insider to follow a specific procedure. All these actions are done at the application level: opening a sensitive file, performing some actions on it and storing the information in a new file. The generation of fine grained in-application log data could help to detect these kinds of malicious insider attacks. Events such

[10]Systracer 2.0 available at `http://www.blueproject.ro/systracer`

as “user opens sensitive file”, “sensitive cells are used as input for certain cells”, etc. give accurate information about the user behaviour. Data mining techniques could be used over this data to build information leakage detectors. A similar approach was proposed by Schonlau et al. to detect malicious insiders using running processes [20].

Additionally, the overhead that the cipher spreadsheet implementations produce on the system when encrypting (Section 6.4.2) could be used as a hint to detect a malicious insider. Nevertheless, this depends on the amount of information to encrypt. Depending on the organization’s activities, these variations can also be attributed to other legitimate Excel operations or other user activities, such as web navigation, etc.

6.5.2.4 Security models to avoid data leaks

Implementing application level restrictions on specific user actions concerning sensitive information may help to avoid these evasion technique (i.e. forbid to reference a cell whose value is a piece of sensitive information). In this regard, the usage of Enterprise Digital Rights Management Solutions [165] would likely defeat the proposed attack. These solutions should be correctly configured through the security policy. Additionally, authors should not have control of their own documents, as they may simply reduce the security level of the document to steal it.

Finally, the usage of multilevel security models (see Section 2.4.3) could be useful to forbid information leakage using our evasion technique. In the Bell-LaPadula model, when the malicious insider opens the sensitive information document he is no longer able to create non sensitive documents. Nevertheless, as previously mentioned, work is still needed in order to adapt these kind of models to commercial environments [66].

Other techniques such as process colouring, could be useful to trace back the origin of the leak [68]. If an application opens a sensitive information file, it propagates its color (a unique identifier) to other opened documents. DLP solutions could be configured to deny transmission of certain colors outside the organization. In this case, those techniques should be adapted to not work only over processes, but also over processes and documents.

6.6 Summary

Information leakage is one of the more rapidly growing threats that organizations face nowadays. This Chapter demonstrates that trusted applications may entail a threat to the confidentiality of sensitive information. This threat has not been addressed by current DLP systems and could allow malicious insiders to leak sensitive information by using trusted applications. Specifically, it has been shown how to implement a standard cipher in a simple and widely used spreadsheet application. As actions performed within the trusted application environment are not controlled, information can be encrypted and sent outside the organization without being detected.

This approach could be used by malicious insiders to steal information from organizations already protected by malicious insider detection or DLP systems. This evasion technique has been tested against a commercial DLP solution, demonstrating that it is able to extract sensitive information from a simulated organization without being detected.

Additionally, the main countermeasures that could be implemented to avoid this kind of evasion technique have been described. Implementing such countermeasures would enable to improve the security of DLP solutions against advanced evasion techniques.

Chapter 7

Conclusions

Weep little lion man
You're not as brave as you were at the start,
Rate yourself and rake yourself,
Take all the courage you have left
Wasted on fixing all the problems
That you made in your own head
Mumford & Sons - Little Lion Man

Malicious insiders are an increasing concern of organizations due to the damage they are able to inflict. In this regard, information theft is one of the most harmful actions that are usually committed by them. The theft of sensitive information not only produces high economical losses and competitiveness problems, but also compromises the viability of organizations [8].

In this thesis, the risk created by malicious insiders to the confidentiality of information assets has been studied. In particular, this dissertation addresses the security of documents and sensitive information the insider has already access to. Although insiders require access to these sensitive information to perform their tasks, extracting them outside the organization premises and controls should not be allowed by the security policy defined by the organization. In this case, access control policies cannot be used to prevent the leakage, as the insider has already access to the sensitive information. However, this is not the only risk that malicious insiders introduce in organizations. They may try to sabotage other organization assets, destroy them or use them for their own advantage. Although these are risk worth addressing, there has already been research that has focused on these threats [19, 18, 59, 20, 24, 23].

Addressing this problem required first to analyze the security of current proposals to avoid information leakage. A survey of current proposals regarding information leakage protection was performed (Chapter 2). During this survey, covert channels created by means of steganography (Chapter 3) and malicious usage of trusted applications were identified as issues that were not taken into account as a tool to steal information. To demonstrate the unaccounted risk introduced by steganography and trusted applications, attacks to information leakage protection mechanisms (and more specifically DLP solutions) have been developed (Chapters 4 and 6). However, mechanisms to prevent from these attacks have also been developed. In fact, active warden and steganalysis techniques have been studied as tools to hinder the usage of steganography (Chapter 5). Additionally, several recommendations to avoid information leakage through the malicious usage of trusted applications have been given (Chapter 6).

The previously stated research path was specified in Section 1.2 establishing a set of research questions. In the following and as a result of this thesis, an answer is given to each of the presented research questions. In this case, the secondary research questions are answered first.

Question 1

What kind of computer or not computer resources can be used by a malicious employee to perform a malicious activity such as information theft?

In Chapter 2, concepts and definitions related to the information leakage protection field were presented. In fact, the concept of leakage vector is defined (Section 2.2.4). A leakage vector is any mean by which some information asset can be revealed to unauthorized users. Therefore, any mechanisms belonging to a computer system that allow to transmit the information outside the organization control has to be considered as a leakage vector. Specifically, in this dissertation the following leakage vectors have been identified:

- Network
- Printers
- Portable media
- Advanced mobile devices

All these leakage vectors could be used by a malicious insider to extract sensitive information from an organization. Although these are usually monitored by DLP solutions, they do not consider the existence of covert channels.

Question 2

Is it possible to transform information in such a way that its usage and transmission passes unnoticed to information leakage prevention mechanisms?

Chapter 2 describes the techniques used by DLP solutions to track sensitive information. Due to the architecture of DLP solutions, if information going through a leakage vector is not identified as sensitive, it can be transmitted without raising any alarm. Nowadays, DLP solutions use keyword matching, pattern recognition, fingerprinting and watermarking as techniques to recognize previously defined sensitive information [69].

Techniques such as cryptography of steganography modify information in such a way, that previously mentioned techniques cannot detect the modified information as sensitive. Although other techniques could be used to detect the usage of cryptography, its detection and the differentiation from legitimate encrypted traffic can be a challenge and has not been explored in depth. Additionally, steganography techniques can produce innocuous stego-objects (images) that cannot be detected as sensitive.

In Chapter 4, two steganographic algorithms that could be used by a malicious insider to leak information are presented. First, an algorithm that transforms C source code into innocuous text is presented and evaluated [13]. The usage of this algorithm is specially suitable in software developing environments. Second, a steganographic application for the iOS operating system is described and evaluated in terms of security and performance [14]. This application is a proof of concept application that can be used to hide information into images that are later sent outside the organization infrastructure using the same mobile device.

Question 3

Is it possible to use the trusted resources and applications in a computer system to leak information?

Although steganography and cryptography can be used to leak information, restrictions to application execution can be imposed to avoid these kinds of attacks by malicious insiders. However, employees still have access to a big set of applications, that can be very complex, to complete their business tasks. As shown in this dissertation, if these applications, which use is not restricted, can be used to ease the leakage of information, malicious insiders would have a way to attack the confidentiality of sensitive information.

In Chapter 6, an scheme to leak information using trusted applications is presented [15]. The attack is built in top of spreadsheet applications, which are commonly used in lots of environments inside organizations. The spreadsheet application is used to build a cryptographic program that can encrypt sensitive information. This can be later extracted without risk of being detected. The security and performance of the proposed scheme have been evaluated. Additionally, this evasion method has been successfully evaluated against a commercial DLP solution.

Question 4

It is possible to control the required leakage vectors to hinder information leakage attempts?

Although the answers to previous questions, show that it is possible to maliciously extract information from an organization protected by a DLP solution, this dissertation has also focused on mechanisms to avoid the presented attacks.

Steganalysis and active warden techniques can hinder the usage of steganography inside organization environments [52]. Besides security evaluations of the steganographic systems proposed in Chapter 4 , Chapter 5 describes a steganalyis technique to detect hidden information inside executable files [17]. The proposal is able to detect with a 100 % accuracy stego-files produced by *Hydan*. This dissertation has also addressed the usage of active warden techniques to avoid the existence of covert channels in network communications [16] (with a specific focus on HTTP).

Additionally, in Chaper 6, a set of recommendations to improve DLP solutions is given (see Section 6.5.2. Implementing these recommendations into DLP solutions would avoid the attack presented in the same Chapter.

Main Research Question

Are the current information leakage protection mechanisms enough to protect against malicious insiders trying to steal information from an organization?

The contributions of this dissertation have shown that current ILP mechanisms do not fully address all the possible attacks that a malicious insider can commit to steal sensitive information. Previous proposals in the field of malicious insider detection usually address the malicious insider as a general problem, not analyzing the possible different behaviours that the insider can adopt. Additionally, proposals that address the leakage of sensitive information only address attacks in which a *need-to-know* violation occurs [22, 23].

Uncontrolled actions performed by insiders also increase the risk of suffering from malicious information leakages. Most Information Leakage Protection mechanisms do not take into account the malicious usage of steganographic applications to hide sensitive information into innocuous objects. To make things worse, mobile devices and trusted applications are not taken into account in any case. In this way, malicious insiders could used them to inadvertently extract information from a supposedly protected organization.

However, it is possible to hinder the extraction of sensitive information by malicious insiders. Steganalysis and active warden techniques could be used to detect and avoid the usage inside organizations. While steganalysis can be used to detect the presence of hidden information being transmitted through leakage vectors, active warden techniques allow to delete the possible hidden information being transmitted outside the organization. Additionally, several improvements could be made to DLP solutions in order to avoid information leakage through the usage of trusted applications. These include, increasing the granularity of the analysis of users actions, using a system to detect abnormal behaviour inside the group of trusted applications and improving IDS to detect malicious files that can be used to convert trusted applications into cryptographic or steganographic applications.

7.1 Future Works

Although this dissertation has presented several attacks to ILP mechanisms and some defence strategies, information leakage threads have gained the interest of the research community during the last years (See Section 1.1). However, there are still open issues that begin just in the point this dissertation finishes. The following lines describe the future work directions that are pointed out by this dissertation.

7.1.1 Improvement of Steganalysis Techniques

Steganalysis algorithms allow to detect stegranograhic content leaving the organization. The creation of new steganographic algorithms requires the design of new steganalytic approaches to detect its usage. Additionally, current steganalysis techniques have to be improved in terms of efficiency and False Positive Rate and False Negative Rate. These improvements are mandatory to provide feasible mechanisms to implement such detectors in a deployed DLP solution.

7.1.2 Improvement of Active Warden Techniques

Active Warden techniques require intercepting the communication between the covert channel communicating parties and modifying the content to erase possible steganographic information. This process introduces an overhead on the communication that must be as low as possible in order to make these techniques worth deploying. To optimize the sanitization process performed, it is necessary to optimize its software or hardware implementation of them. In terms of hardware, using dedicated devices such as FPGA [166] could increase the speed of the sanitization process. Even with an increased speed, these kind of sanitization processes have a limited bandwidth. Analysis to calculate the bandwidth limit and average delay on communications is a crucial aspect of these techniques which has not been already addressed.

7.1.3 Complementary use of Steganalysis and Active Warden Techniques

As previously mentioned, active warden techniques introduce and overhead and delay in the communication process. Depending on bandwidth and throughput of the network (or other device) in which the active warden is placed, it may be infeasible to perform the sanitization process over all messages that are transmitted.

Steganalysis techniques can be used to select the most suitable candidates in which to apply the sanitization process. A selective active warden could be implemented in high throughput scenarios, only applying the sanitization process to suspect messages. The feasibility of this depends on the efficiency of the used steganalysis techniques. In fact, the used steganalysis techniques are required to be faster than its active warden counterparts and also faster enough to be usable in high throughput scenarios.

This approach has been taken into account in StegoProxy (Section 5.3). However, it has not been analyzed nor implemented to test its feasibility.

7.1.4 Development of New Attacks Against DLP Solutions

In this dissertation, attacks to avoid information leakage detection by malicious insiders have been presented. Although previously presented future work moves in the direction of improving the detection of such attacks, it is also necessary to study new ways to attack these solutions. Advancing in this direction will allow to improve further current DLP solutions to provide a better protection against malicious insider attacks.

7.1.5 Improvements to Current DLP Solutions

Although this dissertation describes several recommendations to avoid the presented attacks, tests and evaluation of these defensive techniques have to be addressed. Additionally, proposed steganalysis and active warden techniques should be incorporated in current DLP solutions and tested. In this regard, they could be implemented as part of the workstation agents or deployed in network nodes to directly analyze traffic. Proposals in this regard have to be developed and evaluated, as they are a fundamental instrument to improve current DLP solutions against malicious insider attacks.

Appendix A

Published Works

The work performed during this PhD has resulted in several papers being published. Research work has been published in international conferences and journals or recognized prestige. Additionally, a software application has been developed, registered and published in the Apple App Store. This Section describes the published works and the impact or ranking of each of the journals or conference where works have been published. Additionally, other works where the author of this thesis has taken part are referenced. Overall, this research period has produced 10 published works, which are depicted in a timeline in Figure A.1.

A.1 Published work directly related to this PhD

Overall, 4 different research papers have been published that are directly related to this PhD dissertation. Two of them have been published in international conferences and the other two have been published in indexed international journals.

A.1.1 International Conferences

CSteg: Talking in C code [13]

This work was coauthored by Julio César Hernández, Juan M. Estévez and Arturo Ribagorda. The SECRYPT International conference is part of ICETE, the International Joint Conference on e-Business and Telecommunications. The SECRYPT is ranked as a B conference by the CORE Rankings [167].

Steganalysis of Hydan [17]

This work was coauthored by Julio César Hernández, Juan M. Estévez, Arturo Ribagorda and Miguel Ángel Orellana. The IFIP SEC conference is an international conference organized by the International Federation of Information Processing (IFIP). This conference is ranked as a B conference by the CORE Rankings.

A.1.2 International Journals

A framework for avoiding steganography usage over HTTP [16]

This work was coauthored by Julio César Hernández, José María de Fuentes and Benjamín Ramos. This paper was sent for consideration on May 24th 2011 and accepted for publication on October 2nd 2011. The Journal of Network and Computer Applications has an impact factor of 0.660.

Bypassing information leakage protection with trusted applications [15]

This work was coauthored by Julio César Hernández, Juan M. Estévez and Arturo Ribagorda. This paper was sent for consideration on September 20th 2010 and accepted for publication on May 28th 2012. Computers & Security has an impact factor of 0.889

A.1.3 Registered Software

Hide It In [14]

This work was coauthored by Julio César Hernández. It is available for free in the Apple App Store. It was last updated on 14th December 2010 and has been downloaded more than 5000 times (Consulted through Apple iTunes Connect[1] on April 13th 2012).

A.2 Research papers not directly related to this phd

Besides the previously presented publications, other research works, with not direct relation with the dissertation at hand, have been published. These are mostly related to collaboration with other authors and participation in research projects.

A.2.1 International Conferences

Automatic Rule Generation Based on Genetic Programming for Event Correlation [168]

The main author of this work is Guillermo Suárez and it is coauthored by Esther Palomar, José Mará de Fuentes and Arturo Rigagorda. It was published in CISIS, which is ranked as a B conference by the CORE Rankings. This work was partially supported by CDTI, Ministerio de Industria, Turismo y Comercio of Spain in collaboration with Telefónica I+D, Project SEGUR@ with reference CENIT-2007 2004.

Improving Network Intrusion Detection by Means of Domain-Aware Genetic Programming [169]

This work was coauthored by Agustín Orfila and Arturo Ribagorda. In was published in ARES, which is ranked as a B conference by the CORE Rankings. This work was partially supported by CDTI, Ministerio de Industria, Turismo y Comercio of Spain in collaboration with Telefónica I+D, Project SEGUR@ with reference CENIT-2007 2004.

[1]iTunes Connect is a service provided by Apple to software developers and cannot be accessed by general public - `https://itunesconnect.apple.com/`

A multi-agent scanner to detect stored-XSS vulnerabilities [170]

The main author of this work is Eduardo Galán and it is coauthored by Almudena Alcaide and Agustín Orfila. It was published in ICITST, which is not ranked by the CORE Rankings.

A.2.2 International Journals

Analysis of update delays in signature-based network intrusion detection systems [171]

The main author of this work is Hugo Gascón and it is coauthored by Agustín Orfila. This paper is published in Computers & Security, which has an impact factor of 0.889.

A.2.3 Book Chapters

Cross-Site Scripting: An Overview [172]

The main author of this work is Eduardo Galán and it was coauthored by Almudena Alcaide and Agustín Orfila. This work was partially supported by CDTI, Ministerio de Industria, Turismo y Comercio of Spain in collaboration with Telefónica I+D, Project SEGUR@ with reference CENIT-2007 2004.

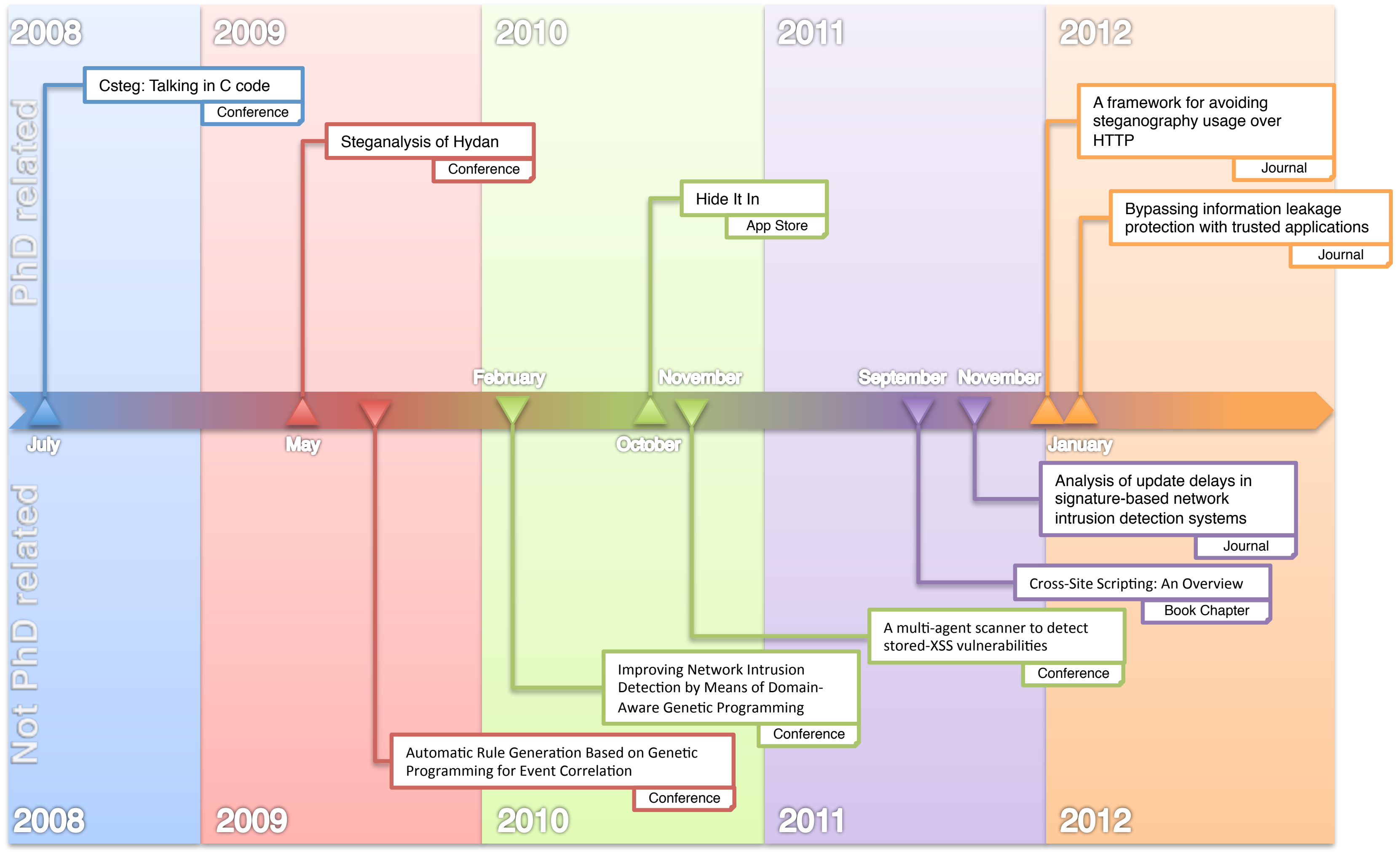

2008
2009
2010
2011
2012
PhD related
Not PhD related
Csteg: Talking in C code
Conference
Steganalysis of Hydan
Conference
Hide It In
App Store
A framework for avoiding steganography usage over HTTP
Journal
Bypassing information leakage protection with trusted applications
Journal
February
November
September
November
July
May
October
January
Analysis of update delays in signature-based network intrusion detection systems
Journal
Cross-Site Scripting: An Overview
Book Chapter
A multi-agent scanner to detect stored-XSS vulnerabilities
Conference
Improving Network Intrusion Detection by Means of Domain-Aware Genetic Programming
Conference
Automatic Rule Generation Based on Genetic Programming for Event Correlation
Conference
2008
2009
2010
2011
2012

Figure A.1: Detailed timeline of published works during the research period related to this PhD dissertation

Bibliography

[1] CSO Magazine, U.S. Secret Service, Software Engineering Institute CERT Program at Carnegie Mellon University, and Deloitte. 2011 E-Crime Watch Survey. Technical report, CSO Magazine in cooperation with U.S. Secret Service, Carnegie Mellon Univeristy Software Engineering Institute's CERT Coordination Center and Deloitte, 2011.

[2] E. Quellet and R. McMilla. Magic Quadrant for Content Monitoring and Filtering and Data Loss Prevention. Technical report, 2011.

[3] Shari Lawrence Pfleeger, Joel B. Predd, Jeffrey Hunker, and Carla Bulford. Insiders Behaving Badly: Addressing Bad Actors and Their Actions. *IEEE Transactions on Information Forensics and Security*, 5(1):169–179, 2010.

[4] T. Walker. Practical Management of Malicious Insider Threat - An Enterprise CSIRT Perspective. *Information Security Technical Report*, 13(4):225–234, November 2008.

[5] Howard Chivers, Philip Nobles, Siraj A Shaikh, John A Clark, and Hao Chen. Accumulating Evidence of Insider Attacks. In *1st International Workshop on Managing Insider Security Threats*, pages 34–50, 2009.

[6] Andrew P Moore, Dawn M Cappelli, Thomas C Caron, Eric Shaw, and Randall F Trzeciak. Insider Theft of Intellectual Property for Business Advantage: A Preliminary Model. In *1st International Workshop on Managing Insider Security Threats*, pages 1–22, West Lafayett, 2009.

[7] CSO Magazine, U.S. Secret Service, Software Engineering Institute CERT Program at Carnegie Mellon University, and Deloitte. 2010 E-Crime Watch Survey. Technical report, CSO Magazine in cooperation with U.S. Secret Service, Carnegie Mellon Univeristy Software Engineering Institute's CERT Coordination Center and Deloitte, 2010.

[8] Ramona R Rantala. Cybercrime Against Businesses. Technical report, U.S. Department of Justice, July 2004.

[9] Marisa Reddy R., Michelle Keeney, Eileen Kowalsky, Dawn Cappelli, and Andrew Moore. Insider Threat Study: Illicit Cyber Activity in the Banking and Finance Sector. Technical Report 7, Carnegie Mellon Software Engineering Institute, January 2005.

[10] Richard Wray. T-Mobile Confirms Biggest Phone Customer Data Breach, 2009.

[11] 2008 Annual Study: U.K. Cost of a Data Breach. Technical report, Ponemon Institute, 2008.

[12] M. McCormick. Data Theft: A Prototypical Insider Threat. *Insider Attack and Cyber Security*, 39:53–68, 2008.

[13] Jorge Blasco, J.C. Hernandez-Castro, J.M.E. Tapiador, and A.R. Garnacho. CSteg: Talking in C code. In *SECRYPT 2008*, pages 399–406, Porto (Portugal), 2008. INSTICC Press.

[14] Jorge Blasco and Julio Cesar Hernandez-Castro. Hide It In (Version 1.2) [Computer Software], 2010. `http://itunes.apple.com/en/app/hide-it-in/id401162613` - Accessed on April 8th 2012.

[15] Jorge Blasco, Julio Cesar Hernandez-Castro, Juan E. Tapiador, and Arturo Ribagorda. Bypassing Information Leakage Protection with Trusted Applications. *Computers & Security*, 31(4):557–568, June 2012.

[16] Jorge Blasco, Julio Cesar Hernandez-Castro, José María de Fuentes, and Benjamín Ramos. A Framework for Avoiding Steganography Usage over HTTP. *Journal of Network and Computer Applications*, 35(1):491–501, January 2012.

[17] J. Blasco, J.C. Hernandez-Castro, J.M.E. Tapiador, A. Ribagorda, and M.A. Orellana-Quiros. Steganalysis of Hydan. In *Emerging Challenges for Security, Privacy and Trust: 24th Ifip Tc 11 International Information Security Conference, SEC 2009, May 18-20, 2009, Proceedings*, page 132, Paphos Cyprus, 2009. Springer.

[18] G Magklaras and S. Furnell. Insider Threat Prediction Tool: Evaluating the probability of IT misuse. *Computers & Security*, 21(1):62–73, 2001.

[19] E.E. Schultz. A Framework for Understanding and Predicting Insider Attacks. 21(6):526–531, 2002.

[20] Matthias Schonlau, William DuMouchel, Wen-Hua Ju, Alan F. Karr, Martin Theus, and Yehuda Vardi. Computer Intrusion: Detecting Masquerades. *Statistical Science*, 16:58–74, 2001.

[21] Lance Spitzner. Honeytokens: The Other Honeypot. *Security Focus*, 21:3–7, 2003.

[22] Brian Bowen, Malek Ben Salem, Shlomo Hershkop, Angelos Keromytis, and Salvatore Stolfo. Designing Host and Network Sensors to Mitigate the Insider Threat. *IEEE Security & Privacy Magazine*, pages 1–14, 2009.

[23] DD Caputo, MA Maloof, and G Stephens. Detecting the Theft of Trade Secrets by Insiders: A Sumarry of MITRE Insider Threat Research. *IEEE Security and Privacy*, 6:14–21, 2009.

[24] BM Bowen, S Hershkop, AD Keromytis, and SJ. Baiting Inside Attackers Using Decoy Documents, 2009.

[25] Wen-Hua Ju and Yehuda Vardi. Profiling UNIX Users And Processes Based on Rarity of Occurrence Statistics with Applications to Computer Intrusion Detection. In *Fourth International Symposium on Recent Advances in Intrusion Detection*, Davis, 2001.

[26] T. Lane and C.E. Brodley. Sequence Matching and Learning in Anomaly Detection for Computer Security. In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, pages 43–49, 1997.

[27] B.D. Davison and Haym Hirsh. Predicting Sequences of User Actions. In *Notes of the AAAI/ICML 1998 Workshop on Predicting the Future: AI Approaches to Time-Series Analysis*, pages 5–12, 1998.

[28] R.A. Maxion and T.N. Townsend. Masquerade Detection Using Truncated Command Lines. In *International Conference on Dependable Systems and Networks*, pages 23–26. IEEE Computer Society, 2002.

[29] Scott Coull, Joel Branch, Boleslaw Szymanski, and Eric Breimer. Intrusion Detection: A Bioinformatics Approach. In *Proceedings of the 19th Annual Computer Security Applications Conference*, pages 24–34. IEEE Computer Society, 2003.

[30] Mizuki Oka, Yoshihiro Oyama, Hirotake Abe, and Kazuhiko Kato. Anomaly Detection Using Layered Networks Based on Eigen Co-occurrence Matrix, 2004.

[31] Kwong H. Yung. Using Self-Consistent Naive-Bayes to Detect Masquerades. *Advances in Knowledge Discovery and Data Mining*, 3506:329–340, 2004.

[32] S. Kleene. Representation of Events in Nerve Nets and Finite Automata, 1951.

[33] L.P. Cox and Peter Gilbert. RedFlag: Reducing Inadvertent Leaks by Personal Machines, 2009.

[34] Udi Manber. Finding Similar Files in a Large File System. In *USENIX Winter 1994 Technical Conference*, pages 1–10, 1994.

[35] G Salton, A Wong, and CS Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[36] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, 1999.

[37] Timothy C. Hoad and Justin Zobel. Methods for Identifying Versioned and Plagiarized Documents. *Journal of the American Society for Information Science and Technology*, 54(3):203, 2003.

[38] a Klose, a Nurnberger, R Kruse, G Hartmann, and M Richards. Interactive Text Retrieval Based on Document Similarities. *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy*, 25(8):649–654, 2000.

[39] S. Schleimer, D.S. Wilkerson, and A. Aiken. Winnowing: Local Algorithms for Document Fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 85–94. ACM, 2003.

[40] Y. Bernstein and J. Zobel. A Scalable System for Identifying Co-Derivative Documents. *Lecture Notes in Computer Science*, pages 55–67, 2004.

[41] Sergey Brin, James Davis, and Héctor García-Molina. Copy Detection Mechanisms for Digital Documents. *ACM SIGMOD Record*, 24(2):398–409, 1995.

[42] J.P. Kumar and P Govindarajulu. Duplicate and Near Duplicate Documents Detection: A Review. *European Journal of Scientific Research*, 32(4):514–527, 2009.

[43] J. Kornblum. Identifying Almost Identical Files Using Context Triggered Piecewise Hashing. *Digital Investigation*, 3:91–97, 2006.

[44] Vassil Roussev. Data Fingerprinting with Similarity Digests. In Kam-Pui Chow and Sujeet Shenoi, editors, *Advances in Digital Forensics VI*, volume 337 of *IFIP Advances in Information and Communication Technology*, pages 207–226. Springer Boston, 2010.

[45] Vassil Roussev. An evaluation of forensic similarity hashes. *Digital Investigation*, 8:S34–S41, 2011.

[46] Ingemar J. Cox, Matthew Miller, and Jeffrey Bloom. *Digital Watermarking and Steganography*. Morgan Kaufmann, 2nd edition edition, 2007.

[47] Ki-Ryong Kwon, Suk-Hwan Lee, Eung-Joo Lee, and Seong-Geun Kwon. *Watermarking for 3D CAD Drawings Based on Three Components*, volume 4109 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006.

[48] Julien Stern, Gaël Hachez, François Koeune, and Jean-Jacques Quisquater. *Robust Object Watermarking: Application to Code*, volume 1768 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000.

[49] Akito Monden, Hajimu Iida, Ken-ichi Matsumoto, Katsuro Inoue, and K. Torii. A Practical Method for Watermarking Java Programs. In *International Computer Software and Applications Conference*, pages 191–197, 2000.

[50] R. Yao, Q. Zhao, and H. Lu. A Novel Watermark Algorithm for Integrity Protection of XML Documents. *IJCSNS*, 6(2):202–207, 2006.

[51] Richard Aikman. McLaren Fire Chief Designer Coughlan, 2008.

[52] Nabil Schear, Carmelo Kintana, Qing Zhang, and Amin Vahdat. Glavlit: Preventing Exfiltration at Wire Speed. In *Proc. 5th Wksp. Hot Topics in Networks (HotNets)*, 2006.

[53] ISO27001: Information Security Management System (ISMS) standard. Technical report, International Organization for Standardization and the International Electrotechnical Commission, 2005.

[54] Edward and Humphreys. Information Security Management Standards: Compliance, Governance and Risk Management. *Information Security Technical Report*, 13(4):247–255, 2008.

[55] M Theoharidou, S Kokolakis, M Karyda, and E Kiountouzis. The Insider Threat to Information Systems and the Effectiveness of ISO17799. *Computers & Security*, 24(6):472–484, 2005.

[56] James M. Stewart, Ed Tittel, and Mike Chapple. *CISSP: Certified Information Systems Security Professional Study Guide*. Sybex, 2011.

[57] V.N.L. Franqueira and PAT Van Eck. Defense Against Insider Threat: A Framework for Gathering Goal-Based Requirements. *EMMSAD*, 7:193–202, 2007.

[58] Richard Kocsis. *Criminal Profiling*. Humana Press, 2006.

[59] N Nykodym, R Taylor, and J Vilela. Criminal Profiling and Insider Cyber Crime. *Computer Law & Security Report*, 21(5):408–414, 2005.

[60] Jude Shavlik and Mark Shavlik. Selection, Combination, and Evaluation of Effective Software Sensors for Detecting Abnormal Computer Usage. In *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '04*, pages 276–285. ACM Press, 2004.

[61] DD Caputo, G Stephens, M, and B Stephenson. An Empirical Approach to Identify Information Misuse by Insiders. *Recent Advances in Intrusion Detection*, pages 402–403, 2008.

[62] B. Aleman-Meza, Phillip Burns, Matthew Eavenson, Devanand Palaniswami, and Amit Sheth. An Ontological Approach to the Document Access Problem of Insider Threat. *Intelligence and Security Informatics*, pages 486–491, 2005.

[63] Yali Liu, Cherita Corbett, Ken Chiang, Sandia National Laboratories, Rennie Archibald, and Dipak Ghosal. SIDD: A Framework for Detecting Sensitive Data Exfiltration by an Insider Attack. In *42nd Hawaii International Conference on System Sciences HICSS '09.*, 2009.

[64] DE Bell and LJ LaPadula. Secure Computer Systems: Mathematical Foundations and Model. Technical Report ESD-TR-73-278, Mitre Corporation, 1973.

[65] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2001.

[66] H.H. Rubinovitz. Issues Associated with Porting Applications to the Compartmented Mode Workstation. *ACM SIGSAC Review*, 12(4):2–5, 1994.

[67] Xuxian Jiang, Aaron Walters, Dongyan Xu, Eugene H. Spafford, Florian Buchholz, and Yi-Min Wang. Provenance-Aware Tracing of Worm Break-in and Contaminations: A Process Coloring Approach. *International Conference on Distributed Computing Systems*, 0:38, 2006.

[68] Xuxian Jiang, Florian Buchholz, Aaron Walters, Dongyan Xu, Yi-Min Wang, and Eugene H. Spafford. Tracing Worm Break-In and Contaminations via Process Coloring: A Provenance-Preserving Approach. *IEEE Transactions on Parallel and Distributed Systems*, 19:890–902, 2008.

[69] G Lawton. New Technology Prevents Data Leakage. *Computer*, 41:14–17, 2008.

[70] A. Whitehead. Towards Eliminating Steganographic Communication. In *Proc. 3rd Annual Conference on Privacy, Security, and Trust*, 2005.

[71] 2012 Global State of Information Security Survey, 2011.

[72] N.F. Johnson and S Jajodia. Exploring Steganography: Seeing the Unseen. *IEEE computer*, 31(2):26–34, 1998.

[73] Gregory Kipper. Investigator's Guide to Steganography. *Ed. Auerbach Publications*, 2004.

[74] Nick Feamster, Magdalena Balazinska, Greg Harfst, Hari Balakrishnan, and David Karger. Infranet: Circumventing Web Censorship and Surveillance. In *Proceedings of the 11th USENIX conference on Security*, USENIX Security'02, pages 247–262. USENIX Association, 2002.

[75] G.J. Simmons. The History of Subliminal Channels. *IEEE Journal on Selected Areas in Communications*, 16(4):452–462, 1998.

[76] Butler W. Lampson. A Note on the Confinement Problem. *Commun. ACM*, 16(10):613–615, 1973.

[77] V. Gligor. A Guide To Understanding Covert Channel Analysis of Trusted Systems. Technical report, National Computer Security Center, 1993. `http://handle.dtic.mil/100.2/ADA276418` - Accessed on May 11th 2012.

[78] L Qiu, Y Zhang, F Wang, M Kyung, and HR. Trusted Computer System Evaluation Criteria, 1985.

[79] K Ahsan and D Kundur. Practical Data Hiding in TCP/IP. In *Proc. Workshop on Multimedia Security at ACM Multimedia*, volume 6, 2002.

[80] Serdar Cabuk, Carla E. Brodley, and Clay Shields. IP Covert Timing Channels. In *Proceedings of the 11th ACM conference on Computer and communications security - CCS '04*, pages 178–187. ACM Press, 2004.

[81] John Giffin, Rachel Greenstadt, Peter Litwack, and Richard Tibbetts. Covert Messaging Through TCP Timestamps. In *Privacy Enhancing Technologies*, volume 2482, pages 189–193. Springer, 2002.

[82] R. Chakinala, A. Kumarasubramanian, R. Manokaran, G. Noubir, C. Rangan, and R Sundaram. Steganographic Communication in Ordered Channels. In *Information Hiding*, pages 42–57. Springer, 2006.

[83] G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil. Eliminating Steganography in Internet Traffic with Active Wardens. *Lecture Notes in Computer Science*, pages 18–35, 2003.

[84] R.G. van Schyndel, a.Z. Tirkel, and C.F. Osborne. A Digital Watermark. In *Proceedings of 1st International Conference on Image Processing*, volume 2, pages 86–90. IEEE Comput. Soc. Press, 1994.

[85] R. Chandramouli, M. Kharrazi, and N. Memon. Image Steganography and Steganalysis: Concepts and Practice. *Lecture notes in computer science*, (2939):35–49, 2003.

[86] Sam Burnett, Nick Feamster, and Santosh Vempala. Chipping Away at Censorship Firewalls with User-Generated Content. In *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, pages 29–45. USENIX Association, 2010.

[87] W. Bender, D. Gruhl, N. Morimoto, and A. Lu. Techniques for Data Hiding. *IBM Systems Journal*, 35(3):313–336, 1996.

[88] Fabien A P Petitcolas. M{P3S}tego [Computer Software], 2006. `http://www.petitcolas.net/fabien/steganography` - Accessed on April 7th 2012.

[89] C. Grothoff, K. Grothoff, R. Stutsman, L. Alkhutova, and M. Atallah. Translation-Based Steganography. *Journal of Computer Security*, 17:269–303, 2005.

[90] M Shirali-Shahreza. Stealth Steganography in SMS. In *Proceedings of the third IEEE and IFIP International Conference on Wireless and Optical Communications Networks (WOCN 2006)*, pages 11–13, 2006.

[91] Sebastian Zander, Grenville Armitage, and Philip Branch. A Survey of Covert Channels and Countermeasures in Computer Network Protocols. *IEEE Communications Surveys & Tutorials*, 9:44–57, 2007.

[92] J Hernandez, I Blasco, J Estevez, and A. Ribagorda. Steganography in Games: A General Methodology and its Application to the Game of Go. *Computers & Security*, 25:64–71, 2006.

[93] B. Park, J. Park, and S. Lee. Data Concealment and Detection in Microsoft Office 2007 Files. *Digital Investigation*, 5:104–114, 2009.

[94] S Zhong, X Cheng, and T Chen. Data Hiding in a Kind of PDF Texts for Secret Communication. *International Journal of Network Security*, 4:17–26, 2007.

[95] I-Shi Lee and Wen-Hsiang Tsai. A New Approach to Covert Communication Via PDF Files. *Signal Processing*, 90:557–565, 2010.

[96] Christian Cachin. An Information-Theoretic Model for Steganography. In *Information Hiding*, volume 1525 of *Lecture Notes in Computer Science*, pages 306–318. Springer Berlin / Heidelberg, 1998.

[97] Rainer Böhme. Principles of Modern Steganography and Steganalysis. In *Advanced Statistical Steganalysis*, volume 0 of *Information Security and Cryptography*, pages 11–77. Springer Berlin Heidelberg, 2010.

[98] J. Fridrich and M. Goljan. Practical Steganalysis of Digital Images-State of the Art. *Proc. SPIE Photonics West*, 4675:1–13, 2002.

[99] J. Fridrich and M. Goljan. Detecting LSB Steganography in Color, and Gray-Scale Images. *IEEE Multimedia*, 8(4):22–28, 2001.

[100] S. Dumitrescu, X. Wu, and Z. Wang. Detection of LSB Steganography Via Sample Pair Analysis. In *Information Hiding*, pages 355–372. Springer, 2003.

[101] G. Bell and Yeuan-Kuen Lee. A Method for Automatic Identification of Signatures of Steganography Software. *Information Forensics and Security, IEEE Transactions on*, 5(2):354–358, june 2010.

[102] Invisible Secrets [Computer Software], 2011. `http://www.invisiblesecrets.com/` - Accessed on April 7th 2012.

[103] Neils Provos. Outguess [Computer Software], 2001. `http://www.outguess.org/` - Accessed on April 7th 2012.

[104] Andy Brown. STools [Computer Software], 2000. http://www.spychecker.com/program/stools.html - Accessed on April 7th 2012.

[105] S. Geetha, S.S.S. Sindhu, S. Gobi, and A. Kannan. Evolving GA Classifiler for Audio Steganalysis Based on Audio Quality Metrics. *IEEE Intelligent Sensing and Information Processing*, pages 105–108, 2006.

[106] J.C. Hernandez-Castro, J.M.E. Tapiador, Esther Palomar, and A. Romero-Gonzalez. Blind Steganalysis of MP3stego. *Journal of Information Science and Engineering*, To appear:1–13, 2010.

[107] Chen Zhi-li, H. Liu-Sheng, Yu Zhen-shan, Z. Xin-Xin, and Zheng Xue-ling. Effective Linguistic Steganography Detection. *CIT Workshops, Australia*, pages 2–7, 2008.

[108] M Steinebach, F A P Petitcolas, F Raynal, J Dittmann, C Fontaine, S Seibel, N Fates, and L C Ferri. StirMark Benchmark: Audio Watermarking Attacks. In *International Conference on Information Technology: Coding and Computing, 2001.*, pages 49–54, 2001.

[109] R. Zax and F. Adelstein. FAUST: Forensic Artifacts of Uninstalled Steganography Tools. *Digital Investigation*, 6:25–38, 2009.

[110] Chris McGreal. FBI Breaks Up Alleged Russian Spy Ring in Deep Cover, 2010.

[111] United States of America vs Anna Chapman and Mikhail Semenko. Technical report, Federal Bureau of Investigation, 2010.

[112] McAfee Labs and McAfee Foundstone Professional Services. Protecting Your Critical Assets: Lessons Learned from Operation Aurora. Technical report, McAfee, 2010. http://www.mcafee.com/us/resources/white-papers/wp-protecting-critical-assets.pdf - Accessed on May 17th 2012.

[113] Peter Wayner. Mimic Functions. *Cryptologia*, 16(3):193–214, 1992.

[114] M. Chapman and G. Davida. Hiding the Hidden: A Software System for Concealing Ciphertext as Innocuous Text. *Lecture Notes in Computer Science*, 1334:335–345, 1997.

[115] Peter Wayner. Spammimic [Computer Software], 2008. http://www.spammimic.com - Accessed on April 7th 2012.

[116] Leevi Marttila. Accurate language to inaccurate language (and back) translator, c2txt2c v0.2.1 [Computer Software], 2001. http://www.mavi1.org/web_security/cryptography/c2txt2c/index.html - Accessed on April 7th 2012.

[117] Bruce Scheiner. The Blowfish Encryption Algorithm. *Dr. Dobb's Journal*, 1994.

[118] Peter Wayner. *Disappearing Cryptography*. Morgan Kaufmann, 2008.

[119] Fast Lexical Analyzer [Computer Software], 2008. http://flex.sourceforge.net - Accessed on April 7th 2012.

[120] Akim Demaille Paul Eggert. Bison Parser Generator, 2008. `http://www.gnu.org/software/bison/` - Accessed on April 7th 2012.

[121] ECRYPT. eSTREAM Project, 2008. `http://www.ecrypt.eu.org/stream/` - Accessed on April 7th 2012.

[122] Bruce Schneier. *Applied Cryptography*. John Wiley and Sons, 2nd edition, 1996.

[123] Derek M. Jones. The New C Standard: A Cultural and Economic Commentary. 2003.

[124] Georges Perec. *La Disparition*. Gallimard, Paris, 1969.

[125] Antonio Calatrava. iStego (Version 1.0) [Computer Software], 2009. `http://itunes.apple.com/es/app/istego-fotos-secretas/id320990210?mt=8` - Accessed on April 8th 2012.

[126] Juicy Bits. Spy Pix (Version 1.0) [Computer Software], 2009. `http://itunes.apple.com/es/app/spy-pix/id336725065?mt=8` - Accessed on April 8th 2012.

[127] Raffaele de Lorenzo. StegoSec (Version 2.0) [Computer Software], 2012. `http://itunes.apple.com/es/app/stegosec/id339434363?mt=8` - Accessed on April 8th 2012.

[128] Rabah Rahil. CoverText (Version 1.0) [Computer Software], 2011. `http://itunes.apple.com/es/app/covertext/id460830129?mt=8` - Accessed on April 8th 2012.

[129] UnderWare LLC. Pixogram (Version 1.0) [Computer Software], 2010. `http://itunes.apple.com/es/app/pixogram/id404444977?mt=8` - Accessed on April 8th 2012.

[130] A. Ker. Quantitative Evaluation of Pairs and RS Steganalysis. *Security, Steganography, and Watermarking of Multimedia Contents VI*, 5306:83–97, 2004.

[131] O. Goldreich. *Modern Cryptography, Probabilistic Proofs, and Pseudorandomness*, volume 17. Springer Verlag, 1999.

[132] John Walker. Ent [Computer Software], 2008. `http://www.fourmilab.ch/random/` - Accessed on April 7th 2012.

[133] C.E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[134] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11, 2009.

[135] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[136] R. El-Khalil and A.D. Keromytis. Hydan: Hiding Information in Program Binaries. *Information and Communications Security*, pages 187–199, 2004.

[137] William Zhu and Clark Thomborson. Recognition in Software Watermarking. In *MCPS '06: Proceedings of the 4th ACM international workshop on Contents protection and security*, pages 29–36. ACM, 2006.

[138] Dominique Chanet Bertrand Anckaert, Bjorn De Sutter and Koen De Bosschere. Steganography for Executables and Code Transformation Signatures. *Lecture Notes in Computer Science*, 3506:425–439, 2005.

[139] DaeMin Shin, Yeog Kim, KeunDuck Byun, and SangJin Lee. Data Hiding in Windows Executable Files. In *Proceedings of the 6th Australian Digital Forensics Conference*, 2008.

[140] W.W. Peterson and D.T Brown. Cyclic Codes for Error Detection. *Proceedings of the IRE*, 49(1):228–235, 1961.

[141] M. Naor and M. Yung. Universal One-Way Hash Functions and Their Cryptographic Applications. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43, New York, NY, USA, 1989. ACM.

[142] Erno Jones. Legitimate Sites as Covert Channels: An Extension to the Concept of Reverse HTTP Tunnels, 2001. `http://gray-world.net/papers/lsacc.txt` Last accessed on June 2011.

[143] A. Dyatlov and S Castro. 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` Last accessed on February 2011.

[144] Maarten Van Horenbeeck. Deception on the Network: Thinking Differently about Covert Channels. In *Proceedings of 7th Australian Information Warfare and Security Conference*, pages 174–184, 2006.

[145] B. Li, J. Huang, and Y.Q. Shi. Steganalysis of yass. *IEEE Transactions on Information Forensics and Security*, 4(3):369–382, 2009.

[146] Q. Liu, A.H. Sung, and M. Qiao. Derivative-based audio steganalysis. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 7(3):18, 2011.

[147] Fabien A P Petitcolas, Ross J Anderson, and Markus G Kuhn. Information Hiding: A Survey. In *Proceedings of the {IEEE}*, volume 87, pages 1062–1078, 1999.

[148] Matthew Kwan. GIFShuffle, 2003. `http://www.darkside.com.au/gifshuffle//` Last accessed on August 2011.

[149] J Fridrich, M Goljan, and R Du. Reliable Detection of LSB Steganography in Grayscale and Color Images. In *Proceedings of ACM Workshop on Multimedia and Security*, pages 27–30, 2001.

[150] J Mielikainen. LSB Matching Revisited. *Signal Processing Letters, IEEE*, 13(5):285–287, 2006.

[151] Fariborz Farahmand. Insider Behavior: An Analysis of Decision under Risk. In *1st International Workshop on Managing Insider Security Threats*, pages 22–33, Purdue University, West Lafayette, USA, 2009.

[152] Haig Simonian and Sharlene Goff. Data Theft Hits 24.000 HSBC Clients. *Financial Times*, page 5, 2010. `http://www.ft.com/cms/s/0/aa0b9e2e-2ced-11df-8025-00144feabdc0.html` Accessed on January 2010.

[153] Alejandro Hernández. Trend Micro Data Loss Prevention 5.2 Data Leakage Through Certain HTTP/HTTPS Channels. Technical report, 2010.

[154] Matthew Skala, Michael Roth, Niklas Hernaeus, Rémi Guyomarch, and Werner Koch. The GNU Privacy Guard [Computer Software], 2010. `http://www.gnupg.org` - Accessed on April 15th 2012.

[155] M.J. Cox, R.S. Engelschall, S. Henson, B. Laurie, E.A. Young, and T.J. Hudson. OpenSSL [Computer Software], 2001. `http://www.openssl.org` - Accessed on April 15th 2012.

[156] Allan Latham. JPHide and JPSeek [Computer Software], 1999. `http://linux01.gwdg.de/~alatham/stego.html` - Accessed on April 15th 2012.

[157] David Wheeler and Roger Needham. TEA, a Tiny Encryption Algorithm. In *Fast Software Encryption: Second International Workshop*, pages 363–366. Springer Berlin / Heidelberg, 1995.

[158] National Institute of Standards and Technology. FIPS 197: Advanced Encryption Standard. Technical report, National Institute of Standards and Technology, 2001.

[159] J. Kelsey, B. Schneier, and D. Wagner. Related-Key Cryptanalysis of 3-Way, Biham-DES, CAST, DES-X, NEWDES, RC2, and TEA. *Information and Communications Security*, pages 233–246, 1997.

[160] S. Hong, Deukjo Hong, Youngdai Ko, Donghoon Chang, W. Lee, and Sangjin Lee. Differential Cryptanalysis of TEA and XTEA. *Information Security and Cryptology-ICISC 2003*, pages 402–417, 2004.

[161] Youngdai Ko, Seokhie Hong, W. Lee, Sangjin Lee, and J.S. Kang. Related Key Differential Attacks on 27 Rounds of XTEA and Full-round GOST. In *Fast Software Encryption*, pages 299–316. Springer, 2004.

[162] Hans Dobbertin, Lars Knudsen, and Matt Robshaw. The Cryptanalysis of the AES - A Brief Survey. In *Advanced Encryption Standard - AES*, volume 3373 of *Lecture Notes in Computer Science*, pages 571–572. Springer Berlin / Heidelberg, 2005.

[163] Eunju Baek, Yeog Kim, Jinwon Sung, and Sangjin Lee. The Design of Framework for Detecting an Insider's Leak of Confidential Information. In *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop*, e-Forensics '08, pages 14:1–14:4. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[164] Juan Soto. Randomness Testing of the Advanced Encryption Standard Candidate Algorithms. Technical report, National Institute of Standards and Technology, 1999.

[165] Yang Yu and Tzi-cker Chiueh. Enterprise Digital Rights Management: Solutions against Information Theft by Insiders. Technical report, Department of Computer Science, Stony Brook University, 2004.

[166] D. Pellerin and S. Thibault. *Practical FPGA Programming in C*. Prentice Hall Press, 2005.

[167] The Computing Research and Education Association of Australasia. Core Conference Rankings, 2010. `http://core.edu.au/cms/images/downloads/conference/08sorttitleERA2010_conference_list.pdf` - Accessed April 13th 2012.

[168] G. Suarez-Tangil, E. Palomar, J. de Fuentes, J. Blasco, and A. Ribagorda. Automatic Rule Generation Based on Genetic Programming for Event Correlation. In Álvaro Herrero, Paolo Gastaldo, Rodolfo Zunino, and Emilio Corchado, editors, *Computational Intelligence in Security for Information Systems*, volume 63 of *Advances in Intelligent and Soft Computing*, pages 127–134. Springer Berlin / Heidelberg, 2009.

[169] Jorge Blasco, Agustin Orfila, and Arturo Ribagorda. Improving Network Intrusion Detection by Means of Domain-Aware Genetic Programming. *Availability, Reliability and Security, International Conference on*, 0:327–332, 2010.

[170] Eduardo Galan, Almudena Alcaide, Agustín Orfila, and Jorge Blasco. A Multi-Agent Scanner to Detect Stored-XSS Vulnerabilities. pages 1–6. IEEE, 2010.

[171] Hugo Gascon, Agustin Orfila, and Jorge Blasco. Analysis of Update Delays in Signature-Based Network Intrusion Detection Systems. *Computers & Security*, 30(8):613–624, 2011.

[172] Almudena Alcaide, Jorge Blasco, Eduardo Galán, and Agustín Orfila. Cross-Site Scripting: An Overview. In Maria Manuela Cruz-Cunha and Joao Varajao, editors, *Innovations in SMEs and Conducting E-Business: Technologies, Trends and Solutions*, chapter Chapter 4, pages 61–75. IGI Global, 2011.

Appendix B

Resumen en Español