

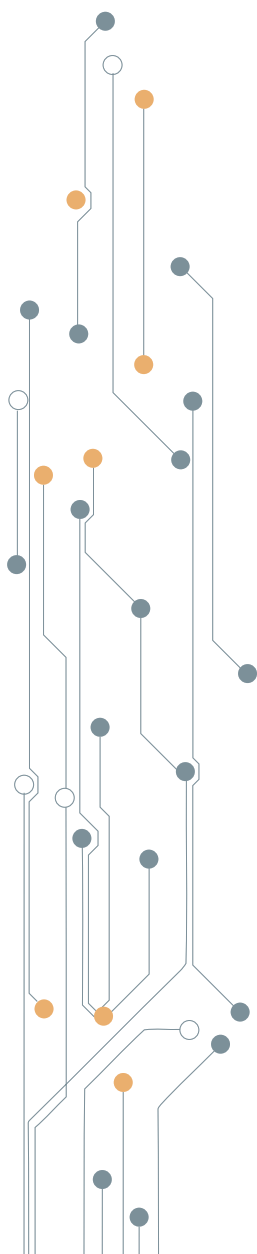


Protección de comunicaciones con esteganografía

Telefónica

EDUCACIÓN DIGITAL

Índice



1 Esteganografía en sistemas de ficheros, sistemas operativos y formatos	3	1.7 Ocultación de información en código ejecutable. Herramienta Hydan	13
1.1 Técnica de ocultación basada en la fragmentación interna de los sistemas operativos. Slack Space	4	1.8 Ocultación de información en ficheros comprimidos. Ocultación de malware	15
1.2 Técnica de ocultación mediante borrado de ficheros. Unallocated File Space	5	1.9 Ocultación de información en código interpretado	16
1.3 Técnica de ocultación ADS en sistema de ficheros NTFS	6	2 Network steganography. Fuga de información y control remoto de sistemas afe	19
1.4 Sistema de ficheros esteganografiados. El caso de StegFs	8	3 Ocultación de información en textos en lenguaje natural. Esteganografía lingüística	21
1.5 Ocultación de información en soportes de almacenamiento. Estructura lógico-física	11	4 Esteganografía en contenido multimedia	22
1.6 Ocultación de información al final de la estructura de un fichero. Técnica EoF (End of File)	12	4.1 Ocultación de información en imágenes digitales	22
		4.2 Ocultación de información en audio digital	25
		4.3 Ocultación de información en video digital	26
		5 Detección de comunicaciones ocultas. Estegoanálisis	27

1. Esteganografía en sistemas de ficheros, sistemas operativos y formatos

En los últimos años los investigadores han centrado una cantidad importante de esfuerzos en técnicas esteganográficas aplicables a un entorno más local como podrían ser los sistemas operativos y las diferentes formas de representar y almacenar la información en los diversos sistemas de ficheros y soportes físicos. Intereses hay múltiples, y aplicaciones, pero principalmente el uso de este tipo de técnicas esteganográficas ha estado muy centrado en mecanismos de ocultación de información con propósito de integridad de software propietario (copyright y licencias de instalación) y la ocultación de código ejecutable, especialmente malicioso. Recuerde el lector que una vez atacado un sistema una idea excelente (post-

explotación) es facilitar el acceso posterior ocultando y ejecutando herramientas o código en la máquina/sistema a atacar, por ejemplo, con el uso de *rootkits*¹. Todo esto hace que hoy día la ciencia de la informática forense centre cada vez más su atención en la detección de todos estos tipos de procedimientos de ocultación y la creación de herramientas lo más automatizadas posibles para detectar posibles vulneraciones de un sistema o dispositivo. Cómo se comprobará las opciones son múltiples y variadas.

Con esta filosofía en mente, es interesante conocer algunos de los mecanismos más clásicos de ocultación.

¹<https://es.wikipedia.org/wiki/Rootkit>



1.1 | Técnica de ocultación basada en la fragmentación interna de los sistemas operativos. Slack Space

Los sistemas operativos, y concretamente sus sistemas de ficheros, trabajan con estructuras lógicas (ficheros) que permiten acceder de forma conjunta a una serie de octetos guardados en un dispositivo. Desde un punto de vista físico, un fichero ocupa bloques de un elemento de almacenamiento, siendo un bloque (conjunto de octetos) la unidad mínima que maneja un sistema de ficheros concreto. El sistema de ficheros FAT16 utiliza, por ejemplo, bloques de 32 KBytes, y el sistema de ficheros ext2 utilizado en entornos GNU/Linux, bloques de 4 KBytes². Dado que el tamaño de un fichero no es un múltiplo exacto del tamaño del bloque, siempre hay parte de un bloque que no se usa, y está reservado para ese fichero. Esto es lo que se conoce como *fragmentación interna* del fichero³ o *slack space*.

Ocultar datos en la fragmentación interna tiene una serie de ventajas a destacar. Por un lado, resultan invisibles al sistema de ficheros, a las aplicaciones y a las herramientas de validación de archivos convencionales. Por otro lado, aunque tiene inconvenientes de portabilidad, ya que si un fichero con información oculta se copia con herramientas estándar la información de la zona de fragmentación interna se pierde, ya que sólo se copian los datos de

usuario, en ciertos contextos esta desventaja podría ser una ventaja interesante, precisamente si interesa que el fichero copiado no se lleve la información encubierta.

En sistemas operativos Linux/Unix una herramienta clásica que utiliza esta técnica de fragmentación interna es, por ejemplo, la herramienta Bmap. Combinando unos pocos mandatos en una consola se puede ocultar información fácilmente en un fichero. Un ejemplo podría ser el siguiente (más detallado en la referencia a pie de página⁴):

```
# echo "evil data is here" | bmap --mode putslack /etc/passwd
# bmap --mode slack /etc/passwd

getting from block 887048  file size was: 9428
slack size: 2860           block size: 4096

evil data is here
```

² En realidad el tamaño de los bloques de un sistema de ficheros ext2 se puede decidir en su creación usando el mandato mkfs

³ Estos mismos principios pueden aplicarse a diferentes formas de almacenar la información, como por ejemplo en una memoria RAM (ram slack) o en un disco duro (disk slack).

⁴ Hiding Data in Slack Space using bmap: <https://www.computersecuritystudent.com/FORENSICS/HIDING/lesson1/index.html>

1.2 | Técnica de ocultación mediante borrado de ficheros. Unallocated File Space

Habitualmente cuando un usuario utiliza las herramientas tradicionales para eliminar un fichero en un sistema operativo este no se elimina en sentido estricto. Cuando se “borra” un fichero en un sistema de ficheros se “eliminan” los enlaces a los diferentes bloques de octetos que constituyen ese fichero, marcándose esos bloques como libres. Mientras los bloques marcados como libres no se sobrescriban con una nueva información los datos previos permanecen inalterados y por tanto, en teoría, pueden ser recuperados, por ejemplo, de un disco duro de un ordenador.

Existen multitud de herramientas para recuperar archivos borrados, tanto en plataformas Windows como en GNU/Linux. Por ejemplo, en el mundo LINUX los ficheros borrados con el mandato *rm* pueden ser recuperados con herramientas como *e2undel* (<http://e2undel.sourceforge.net/>), *extundelete* (<http://extundelete.sourceforge.net/>), etc. Por tanto, si en determinadas circunstancias es posible recuperar la información borrada, este hecho se puede utilizar como un mecanismo trivial de ocultación de información borrando ficheros. Actualmente cualquier distribución seria de análisis forense, con una o más herramientas, contempla la detección de estos casos (*SIFT*⁵, *CAINE*⁶, *Autopsy*⁷, etc.). En cualquier caso la lista de herramientas gratuitas y comerciales para la recuperación de archivos es enorme⁸.

⁵ SANS Incident Forensic Toolkit (SIFT) - <http://digital-forensics.sans.org/community/downloads>

⁶ <http://www.caine-live.net/>

⁷ <http://www.sleuthkit.org/autopsy/>

⁸ File Recovery: https://en.wikipedia.org/wiki/List_of_data_recovery_software



1.3 | Técnica de ocultación ADS en sistema de ficheros NTFS

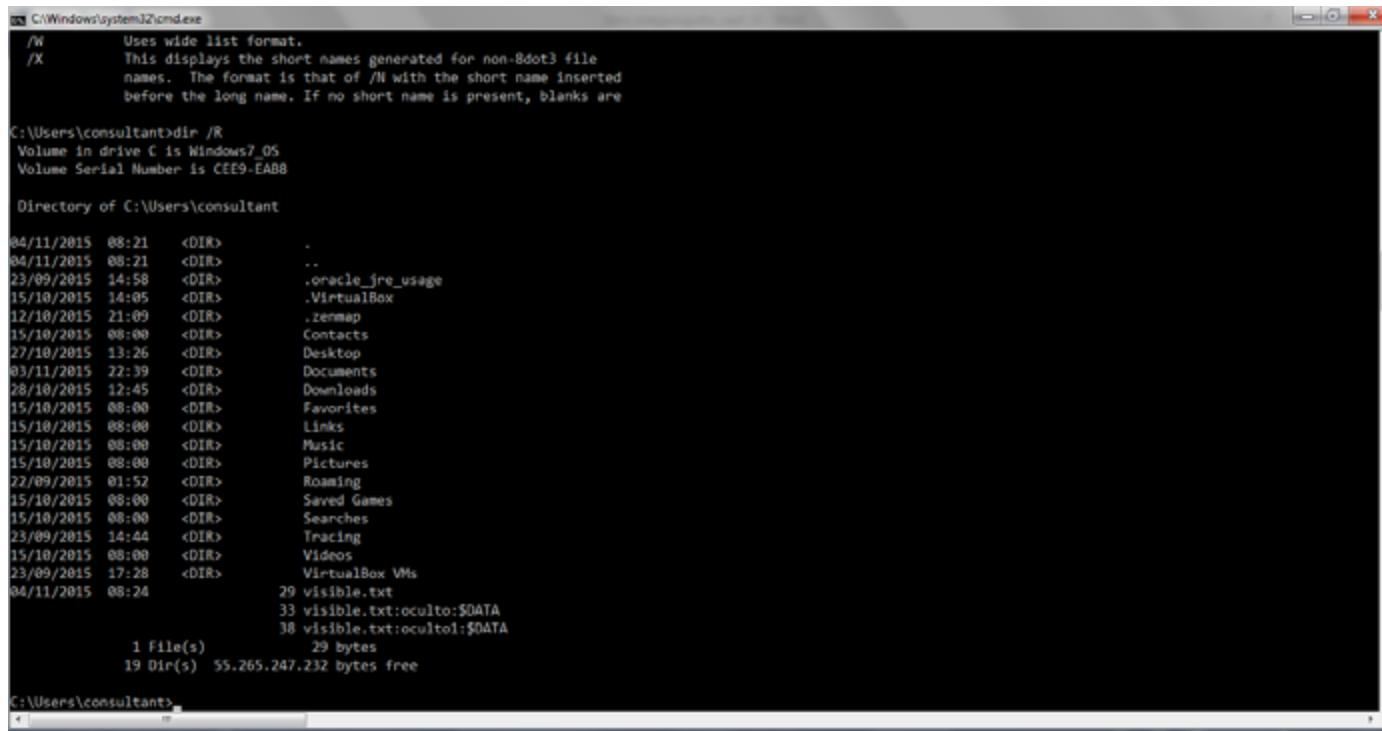
NTFS (*New Technology File System*) es un sistema de ficheros propietario, desarrollado en esencia a lo largo de la década de los 90, utilizado, en principio, para sistemas operativos Windows NT, 2000 y XP. NTFS es un sistema de alto rendimiento con soporte de seguridad con cifrado para proteger datos sensibles, compresión y auditoría. Permite gestión de discos duros de mayor tamaño y soluciones para implementar tecnologías RAID. Adicionalmente implanta NTFS ADS (*NTFS Alternate data streams*), flujos de datos alternativos, para proporcionar (al menos así lo refleja la versión oficial) compatibilidad con el sistema de ficheros de Macintosh HFS (*Hierarchical File System*).

En el sistema de ficheros NTFS un fichero se constituye de diferentes flujos de datos. Por ejemplo, un flujo almacena la información de seguridad como los permisos del fichero, otro almacena los datos reales de usuario, etc. Sin embargo, el problema surge porque es posible la creación de flujos de datos ocultos. Es decir, distintos flujos de datos en un fichero que aparentemente no pueden ser accedidos.

La creación de ADS en un sistema de ficheros NTFS permite en último lugar la creación de técnicas esteganográficas y la ocultación de información en el sistema de ficheros. Para poder aplicar esta técnica de ocultación no es necesario tener privilegios especiales de usuario, simplemente permiso de escritura en el fichero donde se desea almacenar la información a esconder. En el pasado la ocultación de ADS ha sido útil no sólo para ocultar información textual, sino sobre todo código ejecutable, debido a la facilidad para crearlos (véase ejercicio de creación de ADS).



La recuperación de la información enmascarada requiere del uso de herramientas especiales para detectar y observar esta información, o comandos presentes en el sistema operativo pero con ciertas opciones. Por ejemplo, el comando `dir /R` permite ver los flujos de datos existentes. Otra herramienta de detección muy famosa es *LADS* (List Alternate Data Streams) de *Frank Heyne*⁹. En cualquier caso cualquier herramienta forense decente en la actualidad detecta la presencia de ADS.



```
C:\Windows\system32\cmd.exe
/W      Uses wide list format.
/X      This displays the short names generated for non-8dot3 file
names.  The format is that of /W with the short name inserted
before the long name. If no short name is present, blanks are

C:\Users\consultant>dir /R
Volume in drive C is Windows7_OS
Volume Serial Number is CEE9-EAB8

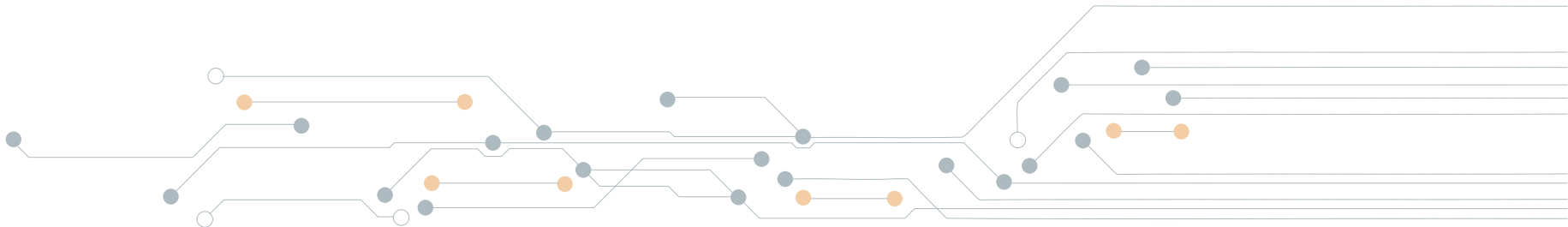
Directory of C:\Users\consultant

04/11/2015  08:21    <DIR>      .
04/11/2015  08:21    <DIR>      ..
23/09/2015  14:58    <DIR>      .oracle_jre_usage
15/10/2015  14:05    <DIR>      .VirtualBox
12/10/2015  21:09    <DIR>      .zenmap
15/10/2015  08:00    <DIR>      Contacts
27/10/2015  13:26    <DIR>      Desktop
03/11/2015  22:39    <DIR>      Documents
28/10/2015  12:45    <DIR>      Downloads
15/10/2015  08:00    <DIR>      Favorites
15/10/2015  08:00    <DIR>      Links
15/10/2015  08:00    <DIR>      Music
15/10/2015  08:00    <DIR>      Pictures
22/09/2015  01:52    <DIR>      Roaming
15/10/2015  08:00    <DIR>      Saved Games
15/10/2015  08:00    <DIR>      Searches
23/09/2015  14:44    <DIR>      Tracing
15/10/2015  08:00    <DIR>      Videos
23/09/2015  17:28    <DIR>      VirtualBox VMs
04/11/2015  08:24          29 visible.txt
                33 visible.txt:oculto:$DATA
                38 visible.txt:oculto1:$DATA

                1 File(s)                29 bytes
                19 Dir(s)  55.265.247.232 bytes free

C:\Users\consultant>
```

Detección de NTFS-ADS con comando dir



1.4 | Sistema de ficheros esteganografiados. El caso de StegFs

La idea de un sistema de ficheros esteganografiado no es para nada nueva, se ha escrito mucho acerca de ellos. En las siguientes líneas comentaremos sus características a través de diferentes artículos, y principalmente de *StegFS*¹⁰ *A Steganographic File System* de los autores *HweeHwa PANG, Kian-Lee TAN, Xuan ZHOU*, sin olvidar estudios previos, como los realizados por *A. Shamir, R. Anderson, R. Needham, A. McDonald, M. Kuhn, etc.*¹¹

La idea fundamental de esta herramienta (StegFS) surge al darse cuenta los creadores del sistema que ni la criptografía ni los controles de acceso de usuarios a determinados datos pueden al final evitar la alerta de existencia de datos, y posibles ataques para rebelarlos, especialmente si se tiene acceso físico al sistema de ficheros (a la máquina/disco duro). El control de acceso y el cifrado son mecanismos estándar de protección de datos en los sistemas de ficheros más modernos, como el clásico EFS (*Encrypting File System*) presente ya en Windows 2000 y XP, o herramientas de cifrado de

disco como el conocido *PGP Disk*¹². Sin embargo, el problema no surge solo porque alguien pueda detectar la existencia de datos e intentar atacar las protecciones para obtenerlos, sino que casos peores, pueden ser cuando un administrador intencionadamente, o no, da permiso a información en contra del deseo del propietario, por ejemplo, al añadir un usuario a una lista de control de acceso a un fichero protegido. El sistema *StegFS* intenta dificultar esta tarea ocultando información en el sistema de ficheros, de tal forma que un usuario sin la clave de acceso correspondiente a ese fichero no es capaz de deducir su existencia, aunque conozca la estructura interna del sistema de ficheros y tenga acceso total al mismo.

Sin profundizar en exceso en los sistemas de ficheros comunes en GNU/Linux, como *ext2* o *ext3*, es importante recordar algunas nociones básicas de estos sistemas de ficheros para comprender como funciona el sistema *StegFS*.

¹⁰ <https://albinoloverats.net/projects/stegfs>

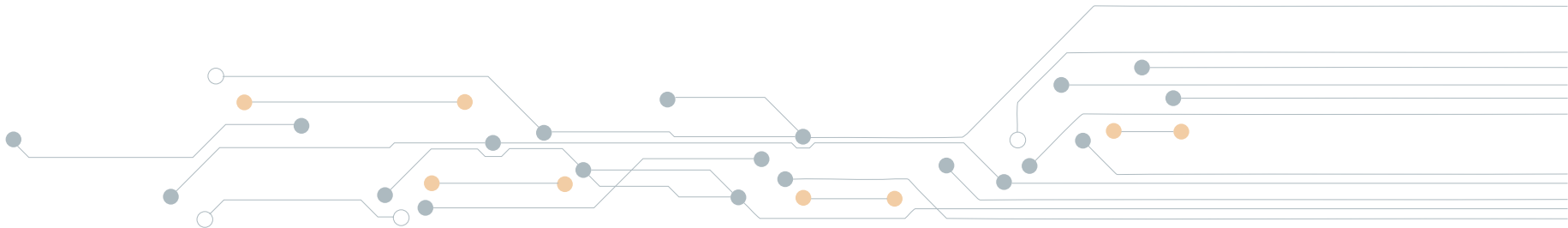
¹¹ <http://www.mcdonald.org.uk/StegFS/>

¹² Una lista completa de software de cifrado de disco duro puede verse en la siguiente url:
<http://www.cryptored.upm.es/crypt4you/temas/privacidad-proteccion/leccion2/leccion2.html>

En general, el espacio de almacenamiento se divide en bloques de tamaño estándar y una serie de vectores de bits (bitmap), que indica si un bloque está libre o reservado (ocupado). Por ejemplo, un bit a cero en una posición determinada del bitmap indica que el bloque con número esa posición está libre, mientras que un bit a 1 significa bloque usado. La gracia de esta herramienta es que todos los ficheros “normales” (no ocultos) se acceden a través del directorio raíz (en sistemas UNÍX “/”), acceso que se realiza gracias a una estructura de enlaces con información sobre los distintos ficheros, los bloques de información que ocupan, etc. Esta estructura se conoce como la tabla de i-nodos en un sistema tipo UNIX. Sin embargo, si los bloques que forman un fichero no son enlazados a la tabla de i-nodos, aparentemente desde el sistema de ficheros esta información permanece oculta, pues el sistema operativo no dispone de información para enlazar los distintos bloques. Partiendo de esta idea, lo único que debe realizar la herramienta es marcar en el bitmap los bloques de información como usados para prevenir que el espacio reservado pueda volver a ser utilizado. No obstante, todavía un usuario con conocimiento de la estructura del sistema de ficheros y acceso completo al mismo podría deducir que bloques no están enlazados y, poder inferir en que bloques existe información oculta, aunque no tendría por qué ser capaz de determinar que bloques pertenecen a qué fichero concreto.

Para dificultar esta tarea *StegFS* crea bloques falsos de dos tipos (activa en el bitmap el bit correspondiente), por un lado, reserva de forma aleatoria algunos bloques que son abandonados (no tienen uso para ocultar información) para contrarrestar un posible intento de localización de datos ocultos simplemente mirando el bitmap. Sin embargo, esto implicaría efectos negativos en uso óptimo del espacio de almacenamiento. Por esto, en la práctica este número de bloques abandonados es aconsejable que no sea grande. Además de este tipo de información, *StegFS* mantiene uno o más ficheros ocultos *dummy*. Un fichero *dummy* es un fichero falso que se actualiza periódicamente, simulando la existencia de ficheros reales que van cambiando, y que dificultaría un ataque basado en el estudio de alteración de los distintos bloques marcados en el bitmap y no enlazados. Conocido esto es importante resaltar que cada fichero oculto se accede a través de su propia cabecera, la cual contiene tres estructuras de datos:

- a. Un enlace a una tabla de i-nodos que enlaza todos los bloques de datos en un fichero.
- b. Una firma que identifica de forma única el fichero.
- c. Una lista enlazada de punteros a bloques libres reservados por el fichero.



Todos los componentes del fichero, incluidos cabecera y datos, son cifrados con una clave de acceso para hacerlos indistinguibles de los bloques abandonados y ficheros *dummy* para observadores desautorizados. Como se comentó, al no ser enlazados los ficheros en el directorio central la herramienta *StegFS* debe ser capaz de localizar la cabecera del fichero usando sólo un nombre físico del fichero y una clave de acceso. Para permitir esto en la creación de un fichero *StegFS* marca el fichero con un firma hash calculada del nombre del fichero y la clave de acceso. Utiliza esta información como semilla (entrada) a un generador pseudoaleatorio de número de bloque, es decir, calcula un número de bloque de forma pseudoaleatoria y comprueba en el bitmap si esta libre en cuyo caso procede a almacenar la cabecera del fichero. Una vez que esta está reservada, los bloques subsiguientes del fichero se asignan aleatoriamente consultando el bitmap, y almacenando el enlace entre los bloques del fichero. Para evitar situaciones de sobrescritura debido a diferentes usuarios trabajando con un

mismo nombre de fichero y clave de acceso, el nombre físico del fichero se deriva concatenándolo con el identificador de usuario en el sistema operativo (que es único por usuario y sistema) y la ruta completa del fichero. La localización una vez conocido esto resulta semejante. *StegFS* utiliza el nombre de fichero y la clave de acceso para calcular el bloque a consultar, verifica si está asignado en el bitmap, y a continuación verifica que tiene una firma de fichero igual al hash calculado (se emplean algoritmos criptográficos como SHA-1 y MD5). Extendiendo esta idea se comprende que *StegFS* tiene más efectividad en un entorno multiusuario, ya que al existir diferentes ficheros ocultos de usuarios, se le complica a un atacante determinar qué información corresponde a un usuario concreto.

De nuevo, este sistema demuestra la sofisticación que puede presentar un sistema real que mezcla criptografía y esteganografía para proteger la información¹³.

¹³ Puede descargar *StegFS* desde <http://www.mcdonald.org.uk/StegFS/download.html>



1.5 | Ocultación de información en soportes de almacenamiento. Estructura lógico-física

En las últimas décadas la estructura lógico-física de los soportes de almacenamiento (disquete, CD-ROM, disco duro, tarjeta flash, chip, etc.) se ha mostrado de gran utilidad para proteger u ocultar información.

En los clásicos disquetes se podían aplicar diferentes técnicas para marcar ciertos sectores válidos como corruptos, formatear sectores o pistas a un tamaño no estándar (con lo cual eran invisibles para el sistema operativo), etc. Todas estas técnicas antiguas de protección, pueden ser usadas para ocultar información, ya que cualquier procedimiento no estándar que se realice es habitual que pase desapercibido para el sistema operativo. Aprovechándose, precisamente de la división de un disquete en bloques lógicos llamados sectores (habitualmente de 512 bytes), la herramienta *S-Tools*¹⁴ (módulo FDD) es capaz de ocultar datos en los sectores libres de un disquete con sistema operativo MS-DOS y sistema de ficheros FAT (*File Allocation Table*). Esta asignación se realiza por medio de un generador pseudoaleatorio y en ningún caso los bloques se marcan como usados. Esto presenta la desventaja de que si la información del disquete crece, es posible que se pierda la información oculta, aunque tiene la ventaja de que un análisis de

los bloques reservados no desvela su utilización. Al final, un analista tiene que decidir si la información que se encuentra en los sectores libres pueden o no ser información real del usuario.

Hoy día, estas técnicas no se limitan a los dispositivos de almacenamiento convencionales sino en general a cualquier dispositivo electrónico que tenga algún tipo de memoria. En los últimos años ha sido significativo la ocultación de información (típicamente código malicioso) en la estructura (memoria) de la BIOS¹⁵ o en dispositivos USB. Habitualmente la BIOS en una placa base de un PC, un router u otro dispositivo se almacena en un chip de flash con suficiente espacio libre para añadir información adicional o código malicioso. La BIOS original se modificará para ejecutar el código añadido adicionalmente y continuar ejecutando su funcionamiento habitual. En los últimos años se han publicado multitud de ejemplos de implementación de malware oculto en BIOS¹⁶. Especialmente significativo han sido los múltiples proyectos de espionaje elaborados por la NSA y revelados por Edward Snowden (*IRONCHEF*, *DEITIBOUNCE*, *ANGRYMONK*, *GINSU*, *SWAP*, *IRATEMONK* y *SOUFFLETROUGH*) para infectar BIOS/firmware de equipos, cortafuegos, routers o discos duros.

¹⁴ <http://www.jjtc.com/Security/stegtools.htm>

¹⁵ El propósito fundamental del BIOS es iniciar y probar el hardware del sistema y cargar un gestor de arranque o un sistema operativo de un dispositivo de almacenamiento de datos.

¹⁶ David Barroso - Infección en BIOS, UEFI y derivados (Rootedcon2015) - <https://www.youtube.com/watch?v=Za9rwt2RDsE>

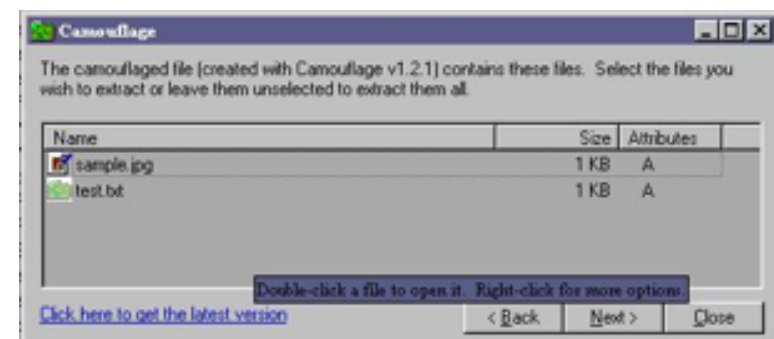
Un caso significativo en esta tendencia fue el malware *BadBIOS*. Las atribuciones a este malware daban una vuelta de tuerca a la utilidad de ocultación de código malicioso en una BIOS o firmware. Tradicionalmente una de las utilidades principales de un malware en la BIOS es generar persistencia de un malware (o funcionalidad extra) que se instalaría en un sistema operativo. Las atribuciones a *BadBIOS*¹⁷ indican que era capaz de infectar una BIOS con un malware que era capaz de comunicarse con otro malware instalado en otra BIOS mediante comunicaciones inalámbricas (por ejemplo, con sonidos imperceptibles por el ser humano). Esta idea introduce una serie de problemas interesantes de destacar. Por ejemplo, un ordenador no conectado a Internet cuya BIOS ha sido modificada

por algún procedimiento (por ejemplo, a través de la ejecución de un malware en una memoria USB) podría comunicarse con otro ordenador próximo que sí está conectado a Internet (tendría que tener la BIOS infectada también). Si esto se produce se producirían canales de comunicación fuera de la red de comunicaciones interna tradicional (evitando las medidas de seguridad clásicas) facilitando la fuga de información y control remoto de sistemas aislados. Por desgracia, existen tantas formas de ser infectado que cualquiera de estas opciones podría llegar al usuario común. Incluso con algo tan inofensivo como un cargador USB de un cigarrillo electrónico¹⁸ o múltiples accesorios USB¹⁹.

1.6 | Ocultación de información al final de la estructura de un fichero. Técnica EoF (End of File)

Una técnica muy extendida en Internet, a la vez que insegura (puede detectarse que la información existe), consiste en enmascarar datos directamente añadiéndolos al final de un fichero electrónico. En muchos formatos electrónicos, sobre todo aquellos cuyo contenido de algún modo es ejecutable (tiene estructura), añadir información al final del fichero es inocuo al comportamiento y ejecución habitual de ese formato. Esto es así porque la mayoría de los formatos de archivos no comprueban la longitud total del fichero que lo contiene. En su lugar, se va leyendo las diferentes informaciones de las estructuras que conforma el formato específico del fichero y una vez que terminan no siguen leyendo. En ningún momento verifican si hay información más allá. El lector puede comprender mejor este procedimiento de ocultación con un ejercicio que se le proporcionará.

Aunque su seguridad es realmente baja, y una inspección directa del fichero sospechoso puede delatarla, la ventaja principal que presenta esta técnica recae en permitir almacenar una enorme cantidad de datos. Con esta filosofía, entre las herramientas esteganográficas más difundidas en Internet se encuentran algunas como: Camouflage 1.2.1, Data Stash v1.5, AppendX, DataStealh v1.0, PGE v1.0, JPEGX Ver 2.1.1, etc.



¹⁷ <https://en.wikipedia.org/wiki/BadBIOS>

¹⁸ E-Cigarettes From China Spreading Malware Through USB Charger. <http://www.ibtimes.co.uk/e-cigarettes-china-spreading-malware-through-usb-charger-1476239>

¹⁹ BadUSB: <http://www.welivesecurity.com/la-es/2014/08/11/analisis-badusb-nueva-amenaza-no-es-apocalipsis/>

1.7 | Ocultación de información en código ejecutable. Herramienta Hydan

Una tendencia interesante en la ocultación de datos reside en la posibilidad de modificar código ejecutable para que enmascarar información conservando la funcionalidad lógica del ejecutable. Es decir, se modificaría las instrucciones²⁰ máquina codificadas en un fichero ejecutable y adicionalmente el fichero ejecutable funcionaría sin ningún problema. Esta idea tiene varias aplicaciones prácticas. La primera obviamente la ocultación de información y la segunda, quizás más sutil, facilita el marcado digital de un ejecutable, permitiendo introducir una marca a modo de copyright.

Uno de los mejores exponentes de este tipo de esteganografía es la herramienta *Hydan*²¹, desarrollada por Rakan El-Khalil de la Universidad de Columbia. La investigación de *Rakan* se centró en el estudio de la redundancia del conjunto de instrucciones en la familia de microprocesadores x86, por ejemplo, procesadores Intel. Su estudio, de forma concisa, concluyó que existen diferentes instrucciones que pueden realizar el mismo trabajo en un programa ejecutable, y por tanto, se puede sustituir una instrucción por otra equivalente (estas ideas, no tan estudiadas como otros procedimientos, se pueden utilizar para representar bits de datos ocultos).

²⁰ [https://es.wikipedia.org/wiki/Instrucción_\(informática\)](https://es.wikipedia.org/wiki/Instrucción_(informática))

²¹ <http://www.crazyboy.com/hydan/>

Original code	Encoding 00
83 e8 30 sub %eax, \$0x30	83 c0 d0 add %eax, \$-0x30
83 f8 36 cmp %eax, \$0x36	83 f8 36 cmp %eax, \$0x36
77 e5 ja \$-27	77 e5 ja \$-27
83 c0 08 add %eax, \$0x8	83 c0 08 add %eax, \$0x8
89 04 24 mov %eax, [%esp]	89 04 24 mov %eax, [%esp]
Encoding 01	Encoding 11
83 c0 d0 add %eax, \$-0x30	83 e8 30 sub %eax, \$0x30
83 f8 36 cmp %eax, \$0x36	83 f8 36 cmp %eax, \$0x36
77 e5 ja \$-27	77 e5 ja \$-27
83 e8 f8 sub %eax, \$-0x8	83 e8 f8 sub %eax, \$-0x8
89 04 24 mov %eax, [%esp]	89 04 24 mov %eax, [%esp]

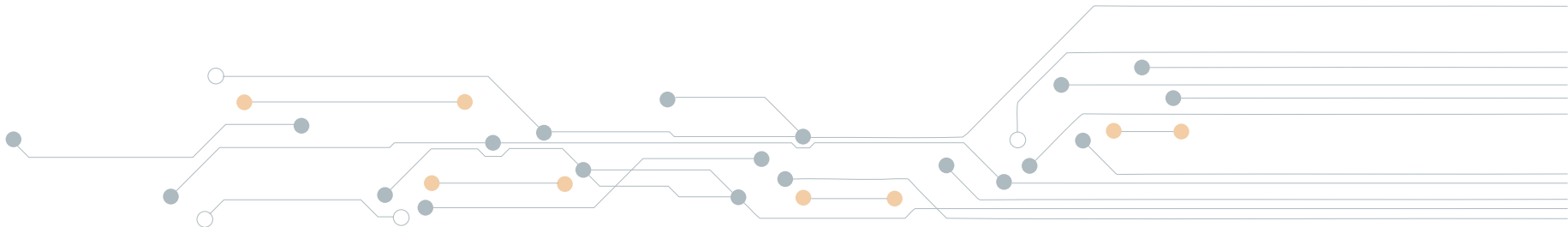
Esteganografía con Hydan en programa ejecutable

Un ejemplo sencillo de esta equivalencia se puede ver en la instrucción suma. Matemáticamente una instrucción que sume 50 a otro valor puede ser reemplazada por otra instrucción que reste -50 a ese valor. En este simple ejemplo, sustituyendo sumas por restas (y viceversa) se podría ocultar un bit por cada operación de suma o resta, **sin cambiar el funcionamiento del programa ni aumentar su tamaño**. Esta idea se puede generalizar para ocultar mayores volúmenes de información. En la siguiente figura se puede ver como ocultar la información binaria "00, 01 o 11" en un programa

ejecutable de tal forma que en el ejecutable esteganografiado la operación suma equivalga a un bit oculto a 0 y la operación resta a un bit a 1.

Es decir, en este sencillo y reducido ejemplo, si se desea ocultar un bit 0 y la operación que se encuentra en el programa original es una suma se deja como está, si es una resta se convierte en una suma. Por otro lado, si se desea ocultar un 1 y la operación que se encuentra es una suma se convertiría a su equivalente en resta. El proceso inverso consistiría en leer el ejecutable esteganografiado y extraer la información en función de las operaciones de suma y resta que se vaya encontrando.

En términos generales, *Hydan* presenta un ratio de ocultación aproximado de 1/110, es decir, permite almacenar un octeto de información oculta por cada 110 octetos de código máquina (bytes de las instrucciones). *Hydan* permite ocultar datos en un fichero ejecutable para diferentes plataformas (NetBSD y FreeBSD i386 ELF, Linux i386 ELF, Windows XP PE/COFF) sin modificar su tamaño, dando la posibilidad de cifrar dicha información con el algoritmo criptográfico *Blowfish*.



1.8 | Ocultación de información en ficheros comprimidos. Ocultación de malware

En las últimas décadas los algoritmos y formatos que se utilizan para comprimir información se han utilizado para realizar diversos ataques prácticos sobre redes y sistemas informáticos. Especialmente, es significativo su uso para la ocultación de código malicioso o malware.

Donde los antivirus tienen el trabajo un poco más complicado es con otro tipo de técnicas bastante más interesantes. Entre estas, destaca la investigación publicada en *BlackHat Europe 2010* por los investigadores *Mario Vuksan, Tomislav Pericin y Brian Karney* titulada:

Hiding in the Familiar: Steganography and Vulnerabilities in Popular Archives Formats²² - (ZIP, 7ZIP, RAR, CAB y GZIP). En esencia en este trabajo se analizan diferentes algoritmos de compresión para detectar la manera de ocultar información utilizando la estructura de los ficheros comprimidos. Es un trabajo que merece la pena leer.

Para comprender de manera sencilla como funciona este tipo de ocultación se propondrá una práctica de ocultación de malware en un fichero comprimido en formato RAR invisible a los antivirus comerciales.

0002FED0	08 12 30 41 8A 50 4B 01 02 14 00 14 00 00 00 08	..0A PK.....
0002FEE0	00 13 6B 2E 3C B7 C2 61 FA F4 92 00 00 5C 95 00	..k.<·Åaúô'..\ .
0002FEF0	00 0A 00 9 00 00 00 00 00 00 00 20 00 00 00 00	...9.....
0002FF00	00 00 00 64 6F 6E 6B 65 79 2E 6A 70 67 50 4B 01	...donkey.jpgPK.
0002FF10	02 14 00 14 00 00 00 08 00 21 74 9D 3A DD 1B FB!t.:Ÿ.û
0002FF20	A4 90 6B 02 00 B3 91 02 00 0B 00 00 00 00 00 00	¤.k...³'.....
0002FF30	00 00 00 20 00 00 00 1C 93 00 00 6D 61 6C 77 61malwa
0002FF40	72 65 2E 62 69 6E 50 4B 05 06 00 00 00 00 01 00	re.binPK.....
0002FF50	01 00 71 00 00 00 D5 FE 02 00 00 00	..q...Œp....

²⁰ BlackHat Europe 2010 - Hiding in the Familiar: Steganography and Vulnerabilities in Popular Archives Formats <https://vimeo.com/11340346>

1.9 | Ocultación de información en código interpretado

En general, cualquier lenguaje de programación, por su propia estructura, permite la posibilidad de ocultar información. Especialmente significativo han sido los esfuerzos en el desarrollo de técnicas y herramientas de ocultación aprovechándose de la estructura de los lenguajes de marcado de uso masivo, como HTML o XML. La idea de estos procedimientos esteganográficos consiste en introducir información, por ejemplo en una página web en HTML, de forma que no sea visible por un usuario que visualice esa página web. La información oculta solo podrá recuperarse utilizando una herramienta de recuperación concreta. Entre las técnicas de ocultación más famosas en HTML (las ideas son extensibles a lenguajes similares como XML²³ o al protocolo SOAP) son²⁴:

- a. **Ocultación basada en caracteres invisibles.** Técnicas triviales de ocultación en páginas web consiste en la utilización de caracteres invisibles para enmascarar información. Por ejemplo, codificar información en un texto del mismo color que el fondo de la página web, codificar la información oculta usando espacios-tabuladores entre etiquetas HTML o al final de cada línea (recuérdese que al visualizar páginas HTML se suele ignorar si existe más de un espacio entre palabras), etc. Herramientas que permiten hacer esto son: *wbStego*²⁵ o *Invisible Secret*²⁶.
- b. **Ocultación mediante codificación en caracteres.** Modificación "insensitive" de etiquetas. En multitud de lenguajes de marcado el software (navegador web) que interpreta el código fuente (páginas web) no diferencia etiquetas en mayúscula o minúscula, es decir, se dice que son *insensitive*. Por ejemplo, HTML 4.0. Esta característica permite ocultar información binaria y, por tanto, cualquier tipo de información.

²³ Office XML Steganography Tool - <http://www.irongeek.com/i.php?page=security/ms-office-stego-code>

²⁴ Sending hidden data through WWW pages: detection and prevention - http://turing.tele.pw.edu.pl/~zkotulsk/Engng_Trans_58_1_2_75_89.pdf

²⁵ <http://www.aptrio.com/Utilities/Security-Encryption/wbstego-686.html>

²⁶ <http://www.invisiblesecrets.com/invsecr4.html>

Un ejemplo sencillo de cómo funciona esta técnica se puede ver con una etiqueta concreta, por ejemplo, la etiqueta HTML `
` (salto de línea). Esta etiqueta tiene 2 letras y se interpreta exactamente igual en cualquiera de sus variantes mayúscula-minúscula: `
`, `
`, `
` o `
`. En total existen 4 combinaciones, luego se podría ocultar hasta **$\log_2(4) = 2$** bits en esta etiqueta, asignado 00, 01, 10 o 11 en binario a cada una de las combinaciones respectivamente. Aplicando este tipo de técnica para cada etiqueta específica en toda la página web se ocultaría la información deseada.

Como puede deducir el lector esta técnica de ocultación debería ser fácil de detectar dado que lo habitual es que las etiquetas se encuentren en minúsculas o mayúsculas. En cualquier caso, siempre se podrá utilizar como un procedimiento para incrustar una firma digital en el código (en lugar de información oculta). La información (la firma) será visible (al chequear el código) y su existencia permitirá verificar la autoría de una información. A la derecha puede observar un ejemplo de cómo utilizar este tipo de esteganografía en un mensaje SOAP para incluir una firma criptográfica que autentique el contenido.

```
<?xml version='1.0' ?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/Soap-
envelope">
  <env:header>
    <m:reservation
      xmlns:m="http://travelcompany.org/reservation"
      env:role="http://www.w3.org/2003/05/
SoapEnvelope/role/next"
      env:mustunderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-
pqff98fe8j7d</m:reference>
      <m:dateAndTime>2001-11-29T13:20:00.000-05:00
      </m:dateAndTime>
    </m:reservation>
    <n:passenger
      xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-
envelope/role/next"
      env:mustunderstand="true">
      <n:name>Áke Jógvan Óyvind</n:name>
    </n:passenger>
  </env:header>
  <env:body>
    <p:itinerary
      xmlns:p="http://travelcompany.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Los Angeles</p:arriving>
        <p:departureDate>2001-12-14</p:departureDate>
        <p:departureTime>late afternoon</p:departureTime>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
      <p:return>
        ...
      </p:return>
    </p:itinerary>
  </env:body>
</env:Envelope>
```

- c. Ocultación basada en el orden de los atributos de una etiqueta. En una forma parecida a la técnica anterior, es posible utilizar el orden de los atributos de un lenguaje de marcado para ocultar información, siempre y cuando su orden no influya en la adecuada visualización de la información que representa, como en el caso de HTML.

Un ejemplo de herramienta que implementa esta técnica es *Deogo*²⁷, desarrollada por Stephen Forrest. Deogo es una herramienta escrita en perl que permite la ocultación de información en ficheros html, sin cambiar el tamaño del fichero ni modificar cualquier dato que no sea una etiqueta. En general, un fichero HTML consiste de un conjunto de etiquetas que deberán ser interpretadas, con el formato:

```
<tagname attribute1=value1 attribute2=value2 ....>
```

Así, por ejemplo, las dos instrucciones siguientes se interpretan del mismo modo por un navegador:

```
<IMG SRC="picture.jpg" ALT= "A picture"> o <IMG ALT= "A picture" SRC="picture.jpg" >
```

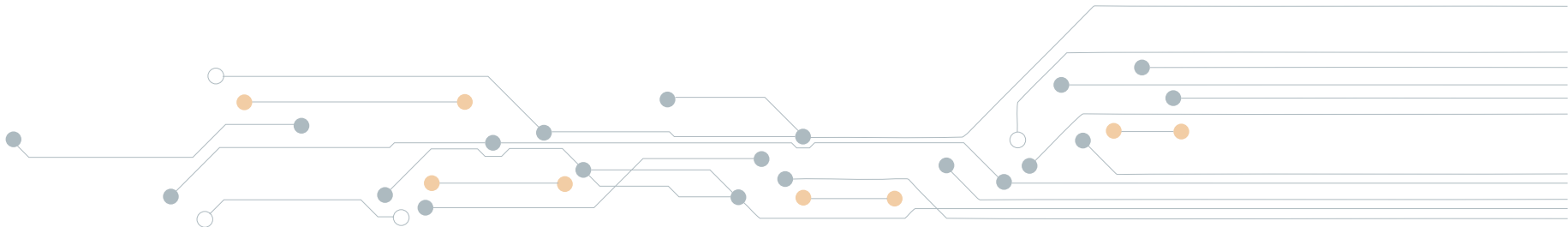
Por lo tanto, podríamos codificar con una instrucción un bit 0 y con otra un bit 1, dando la posibilidad de ocultar una información binaria. Partiendo de este principio, se puede generalizar la idea a otro tipo de etiquetas. Por ejemplo:

```
<TD NOWRAP ROWSPAN=1 COLSPAN=4 ALIGN=left  
VALIGN=top HEIGHT=40 WIDTH=40 id=col1>
```

Esta etiqueta contiene 8 atributos. Si se hacen unas pequeñas cuentas se observa que el número de posible permutaciones de estos atributos es $8!=40320$. Es decir, para esta etiqueta en particular, podemos elegir entre 40320 etiqueta equivalentes, simplemente cambiando el orden de sus atributos. Es decir, podemos ocultar **$\log_2(8!) = 15.3$** bits, casi dos octetos de información. En general, la capacidad de ocultación de esta herramienta por etiqueta dependerá del número de atributos (1 o más) y es **$\log_2(n^{\circ}\text{atributos!})$** . Ideas similares se pueden razonar en otros lenguajes de marcado como XML²⁸.

²⁷ <http://hord.ca/projects/deogol/>

²⁸ A Proposal on Information Hiding Methods using XML - http://takizawa.ne.jp/nlp_xml.pdf



2. Network steganography. Fuga de información y control remoto de sistemas afe

En general, y desde un punto de vista esteganográfico, en la mayoría de protocolos existen ciertas debilidades o consideraciones no estrictamente definidas en diseño que permiten a un individuo crear canales ocultos de información, que pasarán inadvertidos entre una conexión de datos más o menos común. Esto permite comunicarse sin molestia de un posible censor.

El concepto de canal oculto fue formalizado, entre otros, por Butler W. Lampson en 1973 y se describe como aquel canal de comunicación que viola una o más políticas de seguridad del sistema donde se aplica. En sentido práctico, consiste en procedimientos de comunicación que no forman parte del sistema original u organización, pero que se pueden utilizar para transferir información a un sistema o usuario que, en principio, no estaría autorizado a acceder a dicha información. La creación de canales ocultos, en la mayoría de los protocolos de comunicaciones, es viable ya que existen campos no utilizados en la estructura del formato de los paquetes enviados o simplemente campos a los que se les puede proporcionar una funcionalidad diferente a la establecida en la correspondiente especificación que define el protocolo en cuestión.

Un mecanismo habitual de ocultación consiste en utilizar campos reservados, para un futuro, sin utilidad actual en una determinada versión de un protocolo. No obstante, se debería verificar para casos concretos que la información ocultada no es modificada o comprobada por ningún elemento de la red informática donde se desee aplicar dicho canal, como un router o un cortafuegos. Lógicamente, como sucede en otros ámbitos de la informática, los datos, en un canal encubierto, pueden ser cifrados convenientemente para uso exclusivo de los miembros que actúan en la comunicación, también se pueden utilizar múltiples técnicas esteganográficas (múltiples protocolos telemáticos) para ocultar más información y dificultar la detección.

A día de hoy, los canales encubiertos tienen dos objetivos fundamentales: fuga de información y control remoto de equipos/software. Su objetivo fundamental es intentar evadir cualquier tipo de tecnología DLP²⁹ (*Data Loss Prevention*) o de seguridad perimetral (antivirus, cortafuegos, IDS, etc.) que pudiera detectarlos. Desde hace décadas numerosos esfuerzos³⁰ están destinados al análisis y detección de este tipo de canales encubiertos, quizás menos esquivos que otras técnicas esteganográficas descritas en este libro. De hecho, en la última década es especialmente significativo la caracterización de protocolos (comportamiento) utilizando algoritmos de aprendizaje automático (inteligencia artificial)³¹.

²⁹ https://en.wikipedia.org/wiki/Data_loss_prevention_software

³⁰ National Computer Security Center – A guide to Understanding Covert Channel Analysis of Trusted Systems – <https://fas.org/irp/nsa/rainbow/tg030.htm>

³¹ A novel comprehensive steganalysis of transmission control protocol/Internet protocol covert channels based on protocol behaviors and support vector machine – <http://onlinelibrary.wiley.com/doi/10.1002/sec.1081/abstract>

Alguno de los protocolos de comunicación típicos para crear canales encubiertos son: *HTTP, TCP, UDP, Ipv4, IPv6, ICMP, IPSEC, IGMP, FTP, DNS, los protocolos usados en redes de área local (802.2, 802.3), en redes inalámbricas³², etc.* En general, los protocolos típicos vinculados a las comunicaciones en Internet y a la torre de protocolos TCP/IP. No es posible profundizar, por tiempo, en las diferentes técnicas de ocultación para cada protocolo, no obstante se realizará un ejercicio/ ejemplo sencillo de mecanismos de ocultación en el protocolo UDP.

Existen multitud de herramientas para practicar con este tipo de esteganografía, si le interesa esta rama en concreto le recomiendo que se tome su tiempo para probar y comprender cada una de ellas. Algunas de las más famosas son:

1. **Protocolo IPv4:** StegTunnel³³ (campo identificación datagramas IP)
2. **Protocolo TCP:** Covert_tcp³⁴, StegTunnel (campo número de secuencia)

3. **Protocolo ICMP:** troyano Loki³⁵ (ICMP echo Reply/ICMP echo request), IcmpShell³⁶, Pingtunnel³⁷, Timeshifter (intervalos de tiempo)³⁸, ICMPStegano³⁹

4. **Protocolo IPv6:** VoodooNet (voodoo3t)^{40 41} (Ipv6 sobre Ipv4), IPv6-steganography⁴²

5. **Protocolo DNS:** dnssteganography⁴³, Netcross⁴⁴, [denise, dns2tcp, dnscapy, dnscat, heyoka, iodine, psudp, squeeze, TUNS]⁴⁵

6. **Protocolo HTTP:** HttpTunnel⁴⁶, Firepass⁴⁷ (transferencia de datos TCP/UDP en mensajes HTTP POST), HCovert⁴⁸, Ccvt⁴⁹.

7. **Protocolo Aplicación:** MailTunnel⁵⁰, MsnShell⁵¹ (shell remoto Linux a través del protocolo MSN), StegTorrent⁵², g00gle Crewbots⁵³.

³² WLAN steganography - <http://stegano.net/our-papers.html>

³³ <https://packetstormsecurity.com/files/31731/stegtunnel-0.4.tar.gz.html>

³⁴ <http://www.securityfocus.com/tools/1475>

³⁵ https://en.wikipedia.org/wiki/Loki_%28computer%29

³⁶ <http://icmpshell.sourceforge.net/>

³⁷ <http://www.mit.edu/afs.new/sipb/user/golem/tmp/ptunnel-0.61.orig/web/>

³⁸ Transmissions of data through time based covert channels across a network - <https://www.anfractuosity.com/projects/timeshifter/>

³⁹ <https://github.com/pshanoop/ICMPStegano>

⁴⁰ Covert channel tool hides data in IPv6 - <http://www.securityfocus.com/news/11406>

⁴¹ IPv6/ICMPv6 Covert channels - <https://www.defcon.org/images/defcon-14/dc-14-presentations/DC-14-Murphy.pdf>

⁴² <https://github.com/bpafoshizle/IPv6-Steganography>

⁴³ <https://code.google.com/p/dnssteganography/>

⁴⁴ <https://www.primianotucci.com/os/netcross-ip-over-dns-tunneling>

⁴⁵ <https://www.sans.org/reading-room/whitepapers/dns/detecting-dns-tunneling-34152>

⁴⁶ <http://http-tunnel.sourceforge.net/>

⁴⁷ http://www.gray-world.net/pr_firepass.shtml

⁴⁸ (<http://sourceforge.net/projects/hcovert/>)

⁴⁹ http://www.grap-worλd.net/προφεχτα/χχττ/ωiv32_χχττ/

⁵⁰ <http://www.securityfocus.com/tools/1309>

⁵¹ http://www.gray-world.net/es/pr_msnshell.shtml

⁵² <http://stegtorrent.sourceforge.net/>

⁵³ http://www.gray-world.net/pr_gbot.shtml

3. Ocultación de información en textos en lenguaje natural. Esteganografía lingüística

En 1997, *Friedrich L. Bauer* en *Decrypted Secrets. Methods and Maxims of Cryptology*, realizó una clasificación de los procedimientos de esteganografía textual que puede resumir muy bien las tendencias en procedimientos de ocultación en textos antes de la última década del siglo XX. Según esta clasificación, la esteganografía textual (su seguridad se basaba en la oscuridad) puede explicarse en dos grandes ramas: *códigos abiertos* y *semagramas*. Por un lado, los *códigos abiertos* son textos de apariencia inocente, que ocultan información recuperable utilizando ciertas letras, palabras, frases del texto o comunicación. Por otro lado, *los semagramas* consisten en la utilización de la estructura, formato y configuración de símbolos y objetos para establecer un medio de comunicación para ocultar una información.

En la actualidad, la esteganografía lingüística intenta mezclar principios de la ciencia de la esteganografía y la lingüística computacional (análisis automático del contenido textual, generación textual, análisis morfosintáctico, lexicografía computacional, descripciones ontológicas, etc.) para crear procedimientos públicos no triviales según los principios de Kerckhoffs. Según estos principios, existen dos grandes líneas de investigación basadas en la *generación automática de estegotextos* (evitando que un atacante pueda localizar la tapadera original) y la *modificación de textos existentes*. La mayoría de herramientas publicadas pertenecen a alguna de estas dos grandes líneas: Stelin⁵⁴, las basadas en gramática libres de contexto (spammimic⁵⁵, Nicetext⁵⁶, etc.), TEXTO⁵⁷, c2txt2c⁵⁸, Csteg⁵⁹, etc. En el curso se abordarán diferentes ejercicios mostrando el uso de las herramientas más significativas de esta modalidad de ocultación.

⁵⁴ <http://stelin.sourceforge.net>

⁵⁵ <http://www.spammimic.com/>

⁵⁶ Chapman, M. Hiding the hidden: A software system for concealing ciphertext as innocuous text. Master's thesis, University of Wisconsin-Milwaukee, May 1997.

⁵⁷ Texto tool. <ftp://ftp.funet.fi/pub/crypt/steganography>

⁵⁸ Blasco, J., et al. Csteg: Talking in C code. INSTICC, 2008. In Proceedings of SECRIPT International Conference, pp. 399–406.

⁵⁹ Zuxu, D., et al. Text Information Hiding Based on Part of Speech Grammar. IEEE Computer Society. Proceedings of the 2007 International Conference on Computational Intelligence and Security Workshops. pp. 632–635. ISBN:0-7695-3073-7.

4. Esteganografía en contenido multimedia

4.1 | Ocultación de información en imágenes digitales

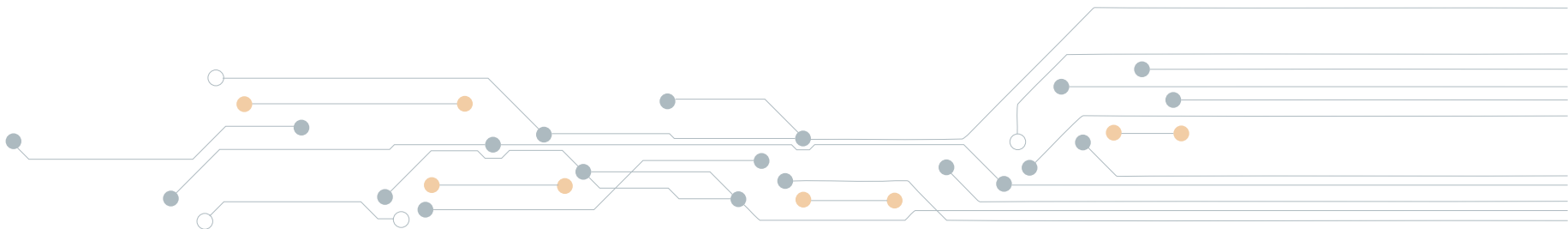
El estegomedio sobre el que más publicaciones y herramientas esteganográficas y estegoanalíticas se han publicado en la última década son las imágenes digitales. Los principios en los que se fundamentan las técnicas de ocultación sobre este tipo de estegomedio son dos: que la modificación de la imagen no introduzca un ruido visual que levante sospechas a una persona que vea la imagen y que las modificaciones introducidas no proporcionen pistas adicionales a un estegoanalista. En este sentido se han publicado una gran variedad de técnicas para diferentes formatos gráficos de fichero (BMP, GIF, JPEG, PNG, etc.).

Lastécnicasyvariantes más documentadas de estos procedimientos consisten en la modificación de los LSB (Least Significant Bit) de los píxeles de una imagen, de los índices que enlazan a la paleta de colores de un formato GIF (así como procedimientos como el reordenamiento de los colores de la paleta) o de los coeficientes

resultantes de aplicar alguna transformación matemática a una imagen, por ejemplo, los coeficientes DCT (transformada discreta del coseno) del formato gráfico JPEG o los coeficientes Wavelet del formato gráfico JPEG2000.

La variedad de técnicas de ocultación y herramientas es inmensa, por ejemplo para ocultación en imágenes GIF (*S-Tools*, *MandelSteg*, *Hide and Seek*, *EzStego*, *Gifshuffle*, *Empty-Pic*, *Git-It-Up* o *Hide4PGP*, etc.), JPEG (*Jsteg*, *Outguess 0.2*, *F5*, *MB1*, *MB2*, *steghide*, *YASS*, etc.) o PNG (*openstego*, *openPuff*).

Por cuestión de tiempo, no es posible profundizar en todas las técnicas o herramientas existentes, sin embargo es interesante detallar un poco más la técnica más sencilla y masivamente utilizada para ocultar información en imágenes digitales. La técnica de sustitución LSB.



La técnica LSB (Least Significant Bit) es el mecanismo de ocultación más clásico utilizado en esteganografía. Su utilización es muy común, debido a su facilidad de aplicación, sobre todo a imágenes y audio.

Una imagen digital se puede estudiar como un conjunto de unidades de tamaño mínimo, denominados píxeles, que tienen unos determinados valores que reflejan los diferentes colores visibles para una imagen concreta. Dependiendo de la resolución de la imagen y de la codificación empleada para un formato gráfico concreto, los píxeles se representan con 1 o más octetos en dicho formato. Es habitual encontrarse ficheros gráficos con diferentes codificaciones, por ejemplo, ficheros gráficos BMP (BitMaP) con resoluciones 8, 16, 24 o 32 bits. Por ejemplo, para una imagen con una resolución de 24 bits cada píxel se representa con 3 octetos (8 bit x 3 = 24 bit). Si estos octetos representan un modelo de color RGB (Red, Green, Blue), cada octeto, respectivamente, almacenará información, sobre el nivel de rojo, verde y azul, que está presente en cada píxel. En última instancia, esta información permite obtener el color real de un píxel.

El valor concreto que puede tomar cada octeto representa la intensidad de dicho color. Este valor oscila desde el valor 0 (el nivel más oscuro) a 255 (el nivel más luminoso). Por ejemplo, un píxel representado con los siguientes 3 octetos: 11111111 (rojo) 00000000 (verde) 00000000 (azul) da como resultado un píxel de color rojo. Siendo estrictos, la técnica LSB aplicada a imágenes es un procedimiento de sustitución que permite modificar el bit menos significativo de la codificación de cada píxel de una imagen por el bit del mensaje a ocultar.

Por ejemplo, en la siguiente imagen se muestra como insertar 8 bits (mensaje 01100011) aprovechando el bit menos significativo de 3 píxeles de ejemplo. Cada píxel está representado por 24 bits (3 octetos) y en este caso se utiliza el bit menos significativo de cada octeto para la ocultación. En este ejemplo los bits subrayados se quedan sin alterar, ya que su valor es idéntico al que se desea insertar, mientras que los otros bits se modifican para insertar los bits deseados (en este caso, en el octeto de verde del pixel0 y en el octeto de verde y azul en el pixel2).



Ejemplo: LSB en BMP (RGB-24 bits)

3 píxeles	Red	Green	Blue	
pixel0	00110110	11110010	01010101	Plano de color (bit-plane)
pixel1	01110110	11000000	00000000	
pixel2	11110001	01100010	00100000	

Mensaje a insertar (8 bits): 01100011

Píxeles modificados:	Red	Green	Blue	
pixel0	00110110	11110011	01010101	Plano de color (bit-plane)
pixel1	01110110	11000000	00000000	
pixel2	11110001	01100011	00100000	

Ejemplo de ocultación en una imagen BMP mediante técnica LSB.

La elección de los píxeles a utilizar dependerá de la variante de la técnica LSB empleada. **La elección de los píxeles suele clasificarse en tres tendencias:** a) *sustitución secuencial de píxeles en una imagen*, b) *sustitución pseudoaleatoria de píxeles de una imagen (clásicamente mediante un generador de números pseudoaleatorios)* y c) *otro criterio matemático de selección*.

Independientemente del criterio de selección de píxeles, las técnicas de ocultación basadas en **el principio LSB apoyaban su seguridad en el hecho de que los bit menos significativo de los píxeles de una imagen tenían un valor “aleatorio” y por tanto su modificación no aportaría información adicional a un analista, así como que dichas modificaciones (ruido adicional) introduciría cambios mínimos en la imagen que no serían percibidos por el ojo humano**. Ambas suposiciones son muy matizables, y a día de hoy la seguridad de las técnicas basadas en LSB deben considerar múltiples factores para que no sea detectada la información oculta.

Para entender mejor como funciona este tipo de esteganografía se realizará un ejercicio utilizando la herramienta multiplataforma *Digital Invisible Ink Toolkit*⁶⁰, que incluye diversas técnicas esteganográficas y una serie de ataques estadísticos para comprobar como de segura está oculta una información en una estegoimagen concreta. Esta herramienta es una de las más completas para experimentar con técnicas LSB, no es común encontrarse ataques estadísticos incluidos en la propia herramienta de ocultación, lo cual es sin duda algo muy interesante.

En cualquier caso, en Internet existen multitud de herramientas disponibles basadas en este tipo de técnicas esteganográficas. Una lista de herramientas interesante es, por ejemplo, la recopilada por el investigador *Neil F. Johnson* con más de 100 herramientas en su página web: <http://www.jjtc.com/Steganography/tools.html>. La mayoría de estas herramientas han sido vulneradas de forma práctica o a nivel académico. Si desea utilizarlas para proteger información sensible por favor compruebe antes los ataques publicados.

⁶⁰ <http://diit.sourceforge.net/>

4.2 | Ocultación de información en audio digital

La esteganografía, en el audio o en el video, en la mayoría de los casos se basa en aprovechar deficiencias en nuestros sistemas receptores. Al igual que sucede con las modificaciones en imágenes, que podían producir cambios visuales notorios apreciables mediante el sistema de visión humano si no se tomaban precauciones, es vital, en un entorno multimedia, audio y video, conocer también las **propiedades del sistema de audición humano**.

En términos generales, el sistema de audición humano es bastante más difícil de engañar que el sistema de visión. Presenta un rango dinámico mayor, percibiendo sonidos para rangos grandes de potencia y frecuencia. Por ejemplo, el oído presenta una sensibilidad alta a la presencia de un ruido blanco gaussiano añadido a una señal de audio. Esta detección puede ser incluso de unos 70dB por debajo del nivel de ruido ambiente. Sin embargo, a pesar de estas propiedades, existen diversas situaciones en las que el sistema puede ser engañado, es impreciso en la detección. Un ejemplo sencillo de comprender es como ante la presencia de “sonidos fuertes” y “sonidos más débiles” los primeros tienden a enmascarar

a los segundos. O, como el sistema de audición humano es poco sensible a ciertos cambios de fase en una señal de audio o, a la supresión (modificación) de ciertas frecuencias en la señal de audio, como por ejemplo, realiza el estándar MPEG-1 audio Layer III (mp3). El estudio de las limitaciones del sistema de audición humano es el punto de partida para el diseño correcto de un algoritmo esteganográfico que oculte información en señales de audio o video, pero no es el único, al igual que sucede con otros estegomédios o algoritmos comentados. La introducción de modificaciones pueden crear patrones no comunes que pueden ser detectados mediante estudios estadísticos de las propiedades de la señal empleada. El lector puede comprobar esto en herramientas típicas que se pueden descargar de Internet: *Mp3Stego*⁶¹, *Mp3Stegz*⁶², *OpenPuff*⁶³, *Silenteye*⁶⁴, *Steghide*⁶⁵, *Stegowav*⁶⁶, *S-tools*⁶⁷, *GiLBCSteg*⁶⁸, *Hide4PGP*⁶⁹ o *DeepSound*⁷⁰.

Durante el curso se expondrán diferentes ejercicios de uso de este tipo de herramientas de ocultación y su seguridad.

⁶¹ <http://www.petitcolas.net/fabien/steganography/mp3stego>

⁶² <http://sourceforge.net/projects/mp3stegz/>

⁶³ http://embeddedsw.net/OpenPuff_Steganography_Home.html

⁶⁴ <http://www.silenteye.org/>

⁶⁵ <http://steghide.sourceforge.net/>

⁶⁶ <https://packetstormsecurity.com/files/21655/stegowav.zip.html>

⁶⁷ http://bit599.netai.net/s_tools_1.htm#demo

⁶⁸ <https://github.com/Garrestotle/GiLBCSteg>

⁶⁹ <http://www.heinz-repp.onlinehome.de/Hide4PGP.htm>

⁷⁰ <http://jpinsoft.net/DeepSound/>

⁷¹ Video steganography using Flash Video (FLV) - <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5168563&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F5159258%2F5168393%2F05168563.pdf%3Farnumber%3D5168563>

⁷² http://www.compression.ru/video/stego_video/index_en.html

4.3 | Ocultación de información en video digital

Un vídeo digital puede definirse como un conjunto de *frames* individuales formado por imágenes y/o audio (y en ocasiones también texto). Por tanto, los procedimientos de ocultación de información en un vídeo se pueden aprovechar de las distintas técnicas esteganográficas sobre cada uno de esos elementos individuales (estegomédios) que configuran cada *frame* o bien combinando los frames. Esta posibilidad hace del video digital un estegomedio de gran interés por su posible gran capacidad de ocultación y la dificultad de detectar el uso de técnicas variadas: ocultación en coeficiente DCT en frames individuales (LSB,...), ocultación por regiones de

cada frame, ocultación en comunicaciones de videoconferencia en tiempo real, videostreaming por Internet⁷¹, etc. De esta forma se pueden desarrollar mecanismos de ocultación que codifiquen información a partir del cálculo de los vectores de movimiento entre una colección de frames, técnicas basadas en corrección de errores, la compresión del vídeo (*motion vector steganography*), etc. Una de las más conocidas es MSU StegoVideo⁷², de *Dimitry Vatolin* y *Oleg Petrov*, que permite la ocultación de archivos en secuencias de un vídeo, aplicando la corrección de errores adecuada para que la información pase desapercibida.

⁷¹ Video steganography using Flash Video (FLV) - <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5168563&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F5159258%2F5168393%2F05168563.pdf%3Farnumber%3D5168563>

⁷² http://www.compression.ru/video/stego_video/index_en.html

5. Detección de comunicaciones ocultas. Estegoanálisis

La detección de información oculta es un tema realmente complejo (si no se utilizan técnicas triviales de ocultación, como la técnica EoF) especialmente para los algoritmos más modernos y sofisticados. Aunque es cierto que estos últimos no son siempre fáciles de encontrar en herramientas de libre acceso que el lector pueda probar.

A grandes rasgos, independientemente de las técnicas o el medio utilizado para ocultar información, cualquier propuesta de detección estegoanalítica se basará en la utilización de procedimientos matemáticos para detectar anomalías (test estadísticos, machine learning, etc.).

En general, un atacante/analista podría intentar realizar los tres siguientes ataques para minimizar la utilidad de una comunicación enmascarada:

- a. Ataque pasivo.** El atacante puede analizar la información intercambiada, dejando que la comunicación tenga lugar si no advierte nada raro. En un entorno telemático puede que el atacante/analista sólo pueda analizar la información pero no interrumpirla.
- b. Ataque activo.** El atacante podría modificar la información, accidentalmente o a propósito, e incluso dañándola significativamente.
- c. Ataque malicioso.** El atacante puede modificar a su antojo la información oculta en una cubierta con la intención de provocar una acción determinada en las entidades receptoras de dicha información. Este ataque no es realista en la mayoría de las situaciones.

En la práctica, el ataque estegoanalítico más realista se basa en el ataque pasivo ya que se basa en que el analista disponga exclusivamente de un fichero sospechoso (una tapadera, en general) y tenga que detectar si existe o no información oculta. Para conseguir este objetivo habitualmente un atacante tendría que realizar las siguientes 3 fases: *detección de información oculta en el estegomedio sospechoso, estimación del tamaño de la información oculta y extracción y recuperación de la información enmascarada.*



Las herramientas de detección suelen centrarse especialmente en la primera y segunda fase. El motivo fundamental recae en que es habitual que la información haya sido ocultada en una tapadera seleccionando posiciones de su contenido de manera aleatoria (utilizando un generador criptográfico pseudoaleatorio) con lo que suele ser inviable recuperar la información oculta que estaría, adicionalmente, cifrada. En cualquier caso, ninguna técnica de ocultación es 100% efectiva si no se presta especial cuidado en el tamaño de la información a ocultar y en el medio que la transporta. En la última década multitud de estudios han detectado patrones (información oculta) en todas las técnicas documentadas en este curso.

En los últimos años se han publicado diversas herramientas de estegoanálisis de software libre y open source disponibles para el público en general. En la mayoría de los casos son herramientas de espectro limitado, detectando unas pocas herramientas de ocultación e implementando unos pocos algoritmos estegoanalíticos, pero muy útiles desde un punto de vista docente para comprender como funcionan las técnicas de estegoanálisis. En cualquier caso, este tipo

de herramientas son la única opción si su presupuesto es limitado o si desea implementar alguna medida ad-hoc (adaptándolas) en su sistema o red corporativa.

- Herramienta StegSpy (<http://www.spy-hunter.com/stegspydownload.htm>) publicada en la Blackhat 2004: Permite la detección de información oculta con las herramientas *Hiderman*, *JPHideandSeek*, *Masker*, *JPegX* e *Invisible Secrets*.
- Herramienta StegExpose (<http://www.darknet.org.uk/2014/09/stegexpose-steganalysis-tool-detecting-steganography-images/>). Centrada en detección de técnica LSB en imágenes digitales con formato PNG y BMP. Implementa diversos ataques estadísticos y estegoanalíticos conocidos (*Sample Pairs*, *RS*, *Chi-Square* y *Primary Sets*) para detectar la ocultación con herramientas como *OpenStego*, *OpenPuff* o *SilentEye*.
- VLS (*virtual steganographic laboratory for digital images*)⁷³, *stegolab*⁷⁴, *digital invisible ink toolkit*⁷⁵, *simple-steganalysis suite*⁷⁶, o *steganalysis*⁷⁷ de Jessica Fridrich.

⁷³ <http://vsl.sourceforge.net/>

⁷⁴ <https://github.com/daniellerch/stegolab>

⁷⁵ <http://sourceforge.net/projects/diit/files/dirt/1.0/>

⁷⁶ <https://code.google.com/p/simple-steganalysis-suite/>

⁷⁷ <http://dde.binghamton.edu/download/>

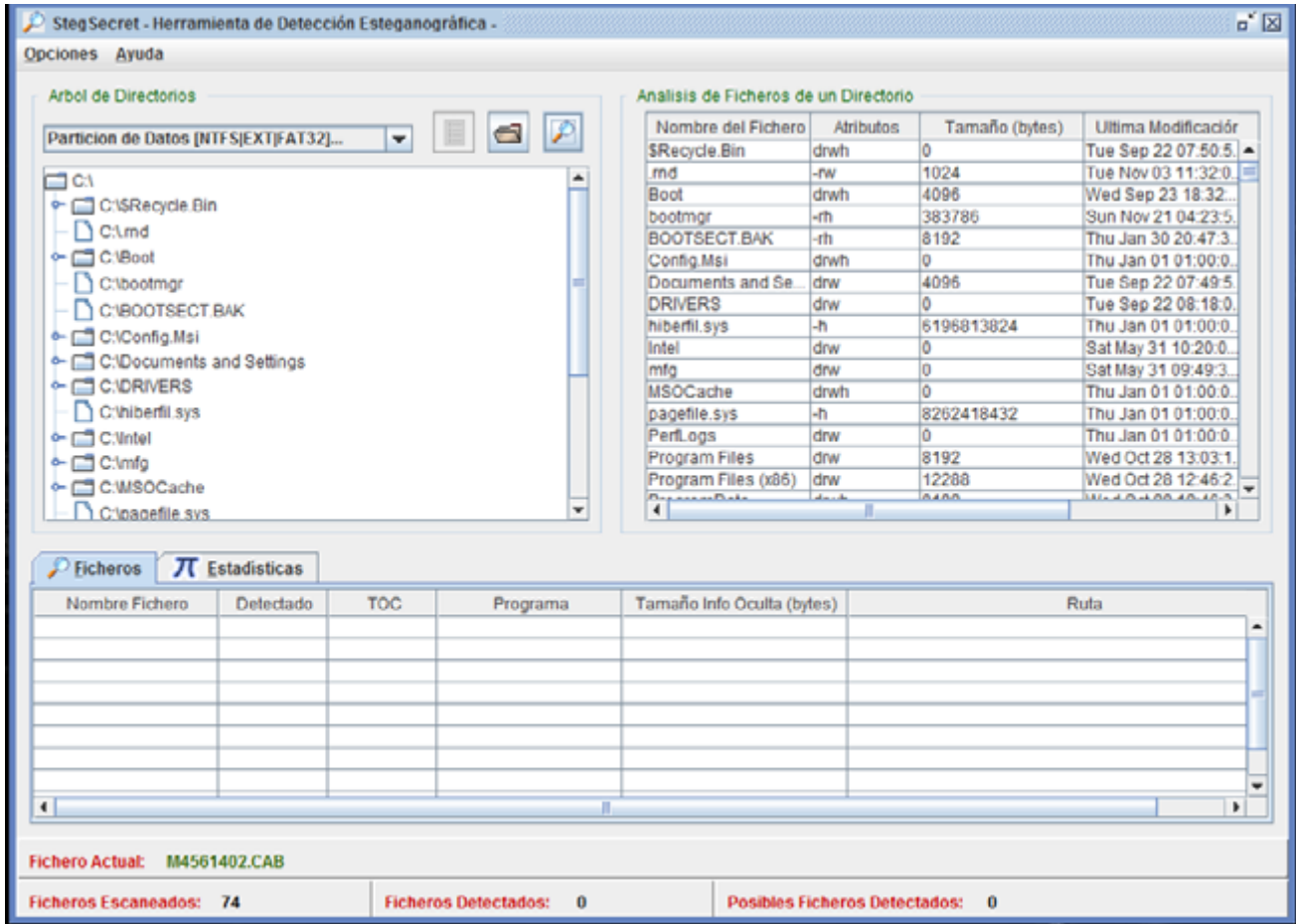


- Herramienta *Stegdetect*. *Stegdetect*⁷⁸ es una herramienta gratuita (licencia BSD) desarrollada por el investigador *Niels Provos* a partir de sus estudios, que permite la detección automática de ciertos algoritmos esteganográficos. En concreto, se centra en imágenes JPEG. Actualmente, es capaz de detectar ocultación de mensajes con *Isteg*, *Jphide*, *Outguess 01.3b*, *F5* (header analysis), *AppendX* y *Camouflage*. Junto a esta herramienta proporciona otra denominada *Stegbreak*, que permite realizar ataques de diccionario contra los mecanismos criptográficos de *Isteg-Shell*, *JPHide* y *OutGuess 0.13b*. *Stegdetect* (versión 0.6) muestra alguna idea interesante, como es el análisis mediante una función de discriminación lineal. La herramienta dado un conjunto de imágenes normales y un conjunto de imágenes que contienen información oculta con una nueva aplicación esteganográfica es capaz de determinar una función de detección lineal que se puede utilizar en un futuro para clasificar nuevas imágenes producidas por esa nueva técnica esteganográfica.
- Herramienta *StegSecret*. La herramienta de código libre *Stegsecret*⁷⁹, desarrollada por el investigador Alfonso Muñoz en 2005, permite realizar algunos ataques de manera simplificada mediante un interfaz gráfico. *StegSecret* permite detectar información oculta con los siguientes programas: *Camouflage V1.2.1*, *inThePicture v2.2*, *JPEGXv2.1.1*, *Pretty Good Envelope v1.0*, *appendX v<=4*, *steganography v1.6.5*, *inPlainView*, *DataStash v1.5* y *dataStealth v1.0*. Adicionalmente, *StegSecret* comprueba la estructura del formato gráfico de imágenes BMP, GIF y JPEG (este análisis permite detectar información oculta a final de fichero - técnica EoF) y detecta la presencia de más de 40 herramientas esteganográficas. Durante el curso se practicará con esta herramienta para comprender algunos de los ataques estegoanalísticos más clásicos.

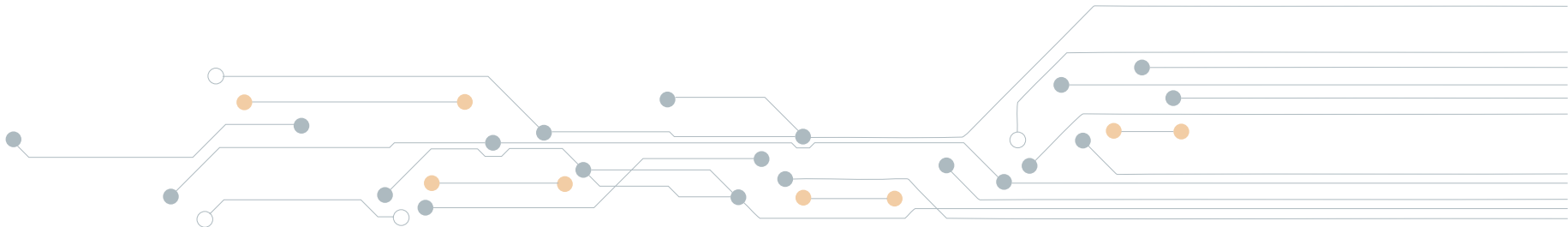
⁷⁸ <https://github.com/abeluck/stegdetect>

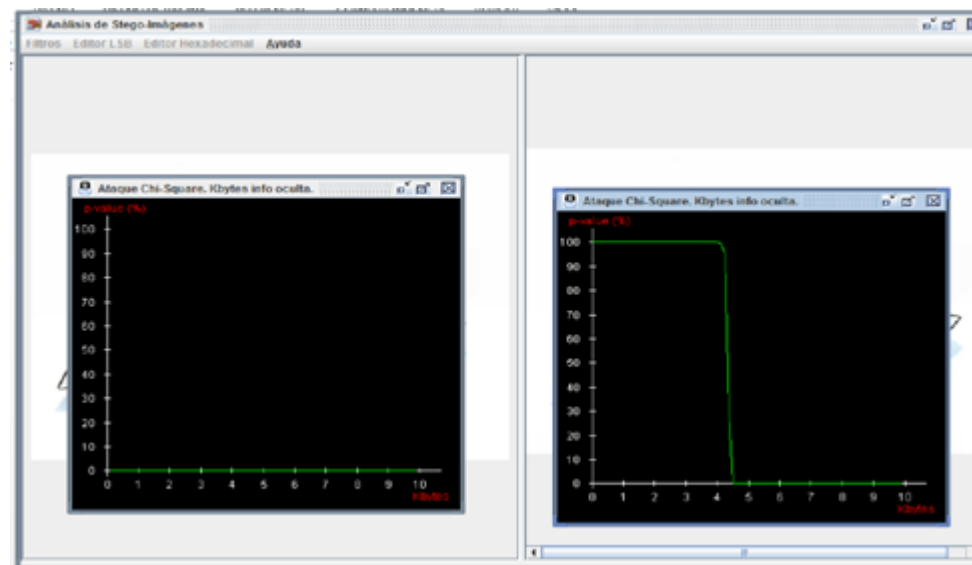
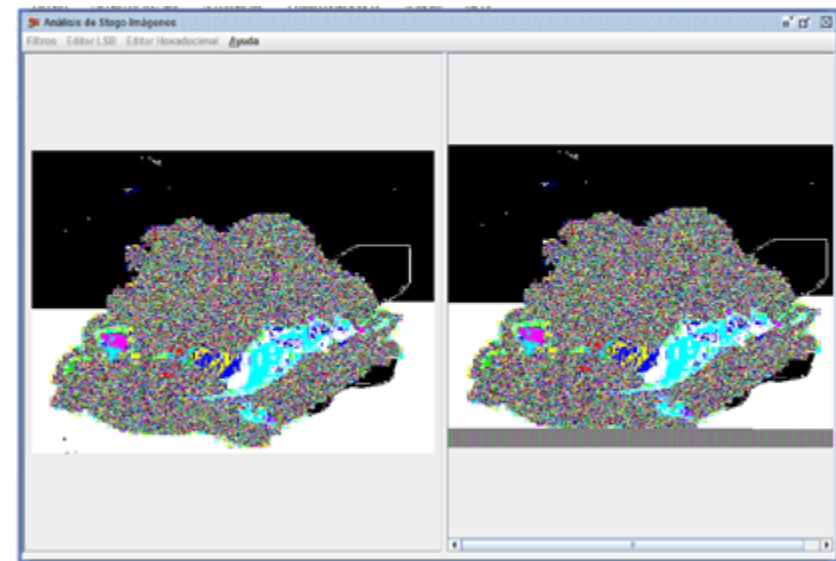
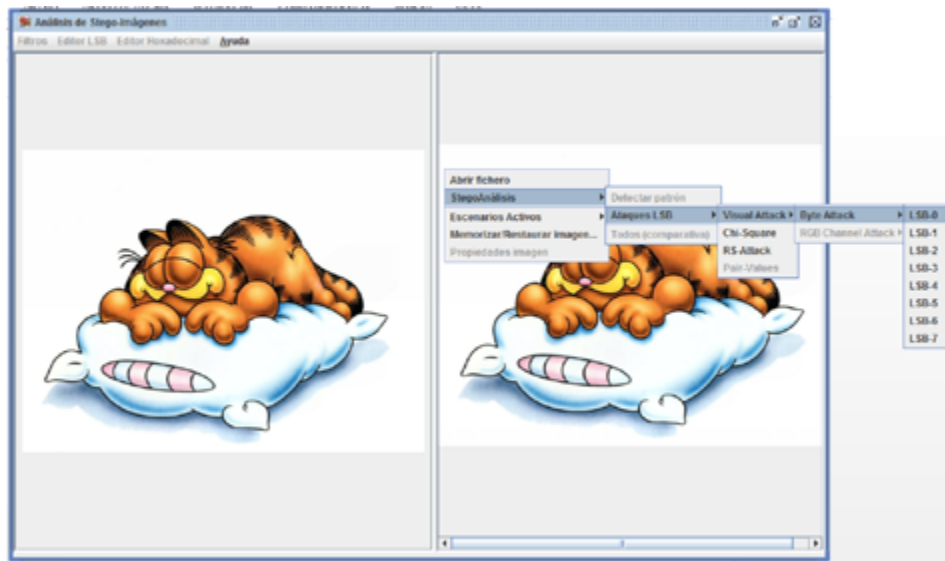
⁷⁹ <http://stegsecret.sourceforge.net>





Escaneador automático de patrones esteganográficos presente en la herramienta StegSecret





Ataques estadísticos y visuales presentes en la herramienta StegSecret

Telefonica EDUCACIÓN DIGITAL