TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica

Proyecto plan 464
Maestría en computación con énfasis en telemática – profesional

Oculto: Biblioteca de automatización de estegoanálisis en imágenes

Autor: Juan Antonio Araya Osorio
Identificador de estudiante: 200133814
Mayo 2022
Profesor: Herson Esquivel Vargas

Instituto Tecnológico de Costa Rica
Escuela de Computación
Unidad de Posgrado

ACTA DE APROBACION DE TESIS

Oculto: Biblioteca de automatización de estegoanálisis en imágenes

Presentado por el estudiante

Juan Antonio Araya Osorio

TRIBUNAL EXAMINADOR

HERSON TOBIAS ESQUIVEL VARGAS (FIRMA)
Firmado digitalmente por HERSON TOBIAS ESQUIVEL VARGAS (FIRMA)
Fecha: 2022.05.25 11:34:54 -06'00'

MSc. Herson Esquivel Vargas
Profesor Asesor

KEVIN MORAGA GARCIA (FIRMA)
PERSONA FISICA, CPF-02-0617-0362.
Fecha declarada: 26/05/2022 09:01:18 AM
Esta representación visual no es fuente de confianza. Valide siempre la firma.

MSc. Kevin Moraga García
Profesor Lector

CARLOS ENRIQUE ARAYA GUZMAN (FIRMA)
Firmado digitalmente por CARLOS ENRIQUE ARAYA GUZMAN (FIRMA)
Fecha: 2022.05.25 08:55:16 -06'00'

MSc. Carlos Araya Guzmán
Lector Externo

LILIANA SANCHO CHAVARRIA (FIRMA)
Firmado digitalmente por LILIANA SANCHO CHAVARRIA (FIRMA)
Fecha: 2022.05.30 16:12:44 -06'00'

Dra.-Ing. Lilliana Sancho Chavarría
Coordinadora
Unidad de posgrado, Escuela de Computación

Resumen

El uso malicioso de la esteganografía les permite a los grupos criminales enviar a sus objetivos archivos digitales como imágenes, videos, sonido, con *malware* escondido de manera que logre evadir sistemas de detección y protección.

También los criminales han hecho uso malicioso de la esteganografía para lograr fuga de información, al ocultar datos o archivos con información de interés, dentro de archivos digitales, y así evadir los controles de los sistemas de detección contra fuga de datos.

El presente proyecto consiste en el diseño de la Biblioteca Oculto, con el fin de automatizar el proceso de estegoanálisis para la detección de datos o archivos ocultos en imágenes a través del uso de esteganografía.

La biblioteca Oculto fue probada en una competencia internacional tipo CTF en abril 2022 con una duración de 5 horas. La competencia promocionada como StegoCTF fue compuesta por 25 retos de esteganografía aplicada en imágenes basadas en el algoritmo *Least Significant Bit*. El desafío era descubrir la mayor cantidad de mensajes secretos ocultos en el menor tiempo posible. 15 de 25 participantes que lograron resolver al menos un reto, lo hicieron usando la biblioteca Oculto. 6 de ellos se clasificaron entre los mejores 10 del evento.

Oculto seleccionó las herramientas compatibles con los archivos a analizar, ejecutó el estegoanálisis de las imágenes, y consolidó los resultados obtenidos.

Palabras claves: Esteganografía, estegoanálisis, *malware*, stegware, biblioteca, detección, *Least Significant Bit*.

Abstract

The malicious use of steganography allows threat groups to hide malware or payloads data into digital files, such as pictures, video, and sound to bypass malware detection and prevention systems.

Threat groups have been also using steganography to hide stolen data into digital files to try to bypass data leakage protection systems.

This project consists in the development of the Oculto library, which is designed to automate the steganalysis process required to detect hidden data or files using steganography.

Oculto Library was tested during a 5-hour long international CTF in April 2022. The CTF was promoted as StegoCTF and composed of 25 image steganography challenges based on the *Least Significant Bit* algorithm. The challenge was to reveal as many secret messages as possible in the shorter timeframe and to obtain the highest score. 15 out of 25 participants who were able to solve at least 1 challenge, had access to the Oculto Library. The scoreboard confirmed that 6 participants of the TOP 10 participant list used Oculto Library during the challenge.

Oculto automatically selected the steganalysis tools, analysed the possible stego files and consolidated the results.

Keywords: Steganography, steganalysis, malware, stegware, library, detection, *Least Significant Bit*.

Agradecimientos

Antes de nada, un agradecimiento a Dios por la fortaleza y apoyo durante el periodo de estudio del programa de la maestría y el desarrollo del Proyecto Final de Graduación.

Este trabajo final de graduación está dedicado a mi madre, Mayra Osorio, por su gran apoyo incondicional desde el inicio y por continuar creyendo en mis capacidades, a mi padre William Araya, quien ha sido mi gran inspiración de superación, a mi esposa Paola Arias por la paciencia, amor y apoyo, a mis hijos Natalia y Aaron por ser mi motor y motivo de lucha, y a mi profesor tutor Msc. Herson Esquivel Vargas, por su excelente guía y comprensión.

También quisiera agradecer a la Comunidad Dojo de Panamá y al grupo de *Ethical Hackers* de Costa Rica DC11506 por su gran apoyo y activa participación en la evaluación de la biblioteca de estegoanálisis Oculto.

Tabla de contenido

1. Introducción

La esteganografía es una técnica que permite el envío de mensajes o datos de forma oculta en un medio portador, la cual se ha utilizado, por ejemplo, para coordinar acciones de guerra y envío de datos confidenciales.

Hoy en día, la esteganografía tiene aplicaciones en la industria como el ocultamiento de información de propiedad intelectual en archivos de audio e imágenes [1]. También en aplicaciones militares como el intercambio de mensajes secretos. Sin embargo, la esteganografía también tiene aplicaciones maliciosas entre las que sobresalen el apoyo a las organizaciones criminales y cibercriminales en lograr ataques de fuga de información, en el envío de payloads o códigos maliciosos, hipervínculos hacía *URLs* maliciosas, entre otros tipos de ataques. Las redes terroristas también pueden utilizarlo para el envío de mensajes secretos para coordinar sus acciones y ataques.

Detectar el uso de esteganografía para ocultar datos durante ataques de fuga de información o para el envío de código malicioso oculto en archivos digitales, es un proceso laborioso y manual, dado que la gran variedad de algoritmos de esteganografía y de herramientas para ocultar datos, que dificulta la rápida detección de datos ocultos.

A continuación, se describe el proyecto Oculto: Biblioteca de estegoanálisis que se elabora en el contexto del Trabajo Final de Graduación del plan de Maestría en Computación del Instituto Tecnológico de Costa Rica. El proyecto consiste en el desarrollo de la biblioteca Oculto para automatizar el proceso de estegoanálisis en imágenes y así detectar el uso de esteganografía para ocultar datos o archivos.

1.2 Justificación

Uno de los ataques más comunes que realizan los grupos criminales y terroristas es el robo o fuga de información [2]. Los datos de interés son aquellos que contengan información sensible de empleados y clientes conocido como *Personal Identifiable Information* o PII [3], secretos comerciales, patentes, nuevas tecnologías, credenciales de usuarios, entre otros. Estos grupos también están utilizando esteganografía para el envío de *payloads*, malware, para fines de extracción de datos sensibles, entrega y distribución de *ransomware* [15].

Los datos robados son luego monetizados y pueden ser utilizados para extorsionar a las víctimas, donde se le exige alguna recompensa a cambio de la recuperación de los datos. En caso de no pago, grupos criminales, como Maze [4] o Conti [5], exponen los datos de forma libre en fórums criminales [6].

También en caso de credenciales robados se utilizan luego en ataques de secuestro de cuentas [7] dada la mala práctica de muchos usuarios de reutilizar nombres de usuario y contraseñas en múltiples cuentas sin el uso de medidas básicas de seguridad como la autenticación multifactor, ni la rotación de llaves de acceso o contraseñas, entre otras.

La esteganografía es una de las técnicas utilizadas tanto para robar información, así como para el envío de código malicioso o comandos malintencionados para su posterior ejecución durante un ataque [8].

Se pueden aplicar técnicas de esteganografía para esconder archivos, extractos de código fuente, datos sensibles, *malware*, en medios portadores como archivos de texto, audio, vídeo, imágenes, archivos de ofimática como *Word, Excel*, entre otros.

Las imágenes digitales son el medio portador más común utilizado para esconder datos y *payloads* [9]. A través de técnicas de esteganografía diseñadas para ocultar los datos y en combinación con técnicas de ingeniería social para su distribución, las imágenes convertidas en estego-objetos son un medio eficiente para el envío de datos y *payloads*. Esto debido a que se pueden ocultar datos utilizando distintos algoritmos de modo que el archivo portador sea modificado de una forma que evite generar sospechas y que dificulte su detección y mitigación.

La dificultad de detectar exfiltración de información por medio de estego-imágenes radica en que hay múltiples algoritmos y herramientas, algunas, inclusive, incompatibles entre sí.

Actualmente existen múltiples formas de aplicar de esteganografía basadas en el algoritmo de *Least Significant Bit* (LSB). Algunas de las técnicas utilizadas son la esteganografía de dominio espacial [10] o esteganografía de dominio frecuencia [11], las cuales permiten esconder archivos, texto/mensajes e incluso payloads en imágenes. También hay algoritmos para el envío de datos ocultos en archivos de audio, de video e incluso durante el *streaming* de audio y video.

Los algoritmos LSB han sido implementados en múltiples herramientas, tales como *Steghide* [12] y *OpenStego* [13], La mayoría de estas herramientas de esteganografía son de código de fuente abierta y pueden ser utilizadas por cualquier persona u organización, incluyendo los grupos criminales, quienes incluso crean sus propias versiones basadas en la lógica de estas herramientas.

La variedad de herramientas y las distintas formas de aplicar esteganografía dificulta el proceso de estegoanálisis que tiene como objetivo analizar un elemento portador para determinar si contiene algún dato oculto.

Una de las situaciones que dificulta el proceso de estegoanálisis, es el hecho que, aunque diferentes herramientas apliquen esteganografía para esconder datos en archivos, la forma en que aplican el proceso puede ser distinto [14].

En un caso de sospecha de exfiltración de datos en imágenes, un analista tendría que probar manualmente con varias herramientas. Este proceso de análisis es manual, complicado, lento y con posibilidad de obtener falsos negativos y falsos positivos. La falta de correlación o por errores humanos durante el uso de las herramientas pueden afectar la interpretación de los resultados.

Es necesaria la creación de una biblioteca de estegoanálisis que automatice el análisis de imágenes y que facilite la identificación de los algoritmos de esteganografía utilizados para ocultar información. La biblioteca Oculto tendrá como fin apoyar a los equipos de ciberseguridad en su lucha contra los ataques de *stegware*, es decir código malicioso que utiliza técnicas de esteganografía para ocultar y enviar programas malignos o para esconder datos robados en archivos digitales.

Oculto estará enfocada en detectar el uso de esteganografía en imágenes, debido a que cada vez es más común el tráfico, subida y descarga de imágenes en redes sociales, correos electrónicos, así como sitios *web*, los cuales podrían ser alteradas por grupos criminales. Por ejemplo, cibercriminales como “*The Dukes*”, también conocidos como APT29 o “*Cozy Bear*” han estado utilizando técnicas de esteganografia en imágenes para ofuscar las comunicaciones entre equipos infectados con *malware* y los sistemas de control conocidos *como Command and Control* para enviar comandos en ataques con *bots* [16].

Oculto se enfoca en la aplicación de herramientas de estegoanálisis para la detección de datos ocultos mediante el algoritmo de esteganografía conocido como *Least Significant Bit* (LSB). Esto debido a que su uso es cada vez más común por el crimen organizado [17]. Además, durante los últimos años se han estado proponiendo varias mejoras al algoritmo LSB [18] con el fin de dificultar cada vez el proceso de estegoanálisis, es decir de interpretar y extraer los datos ocultos.

Oculto es de código de fuente abierto con el fin de continuar integrando nuevos módulos y funciones de estegoanálisis, así como para motivar a la comunidad a continuar investigando el tema de esteganografía y mejorar las técnicas y herramientas de estegoanálisis disponibles.

1.3 Objetivo general del proyecto

Elaborar una biblioteca que permita la detección de datos ocultos en imágenes con técnicas de esteganografía basadas en el algoritmo de LSB.

1.4 Objetivos específicos

- Informar a la comunidad de profesionales en ciberseguridad sobre los riesgos del uso malicioso de la esteganografía mediante la creación de un sitio *web* y material educativo.
- Desarrollar una biblioteca para facilitar el análisis y detección de técnicas de esteganografía basadas en LSB de dominio espacial en archivos de imágenes.
- Evaluar la capacidad de la biblioteca Oculto de detección de datos o archivos ocultos en imágenes con técnicas de esteganografía basadas en el algoritmo LSB.

1.5 Alcance

El alcance del proyecto es desarrollar una biblioteca que permita automatizar el proceso de estegoanálisis de imágenes mediante la integración de herramientas de esteganografía basadas en LSB de dominio espacial.

2. Marco teórico

2.1 Historia de la esteganografía

El término esteganografía proviene de la palabra “steganos” que en griego significa oculto, y “grafía” que significa escrito, por lo que representa el concepto de “escritura oculta o encubierta” [19].

La esteganografía es una técnica diseñada para poder esconder información, datos e incluso archivos dentro de un medio portador. A diferencia de la criptografía, donde el objetivo es cifrar la información de modo que si es interceptado su contenido no sea legible ni entendible, en la esteganografía, la idea es poder enviar mensajes, datos ocultos sin que genere sospechas para evitar que sea interceptado, con el fin de coordinar acciones y compartir datos secretos.

En el libro “las Historias”, escrito por Heródoto de Halicarnaso unos 400 años antes de Cristo, se menciona 2 ejemplos muy interesantes sobre el uso de la esteganografía para el envío de mensajes secretos militares. En la Figura 1, se representa un caso donde el general Histieo [20] del ejército de Atenas, mandó a rasurar el cabello de uno de sus criados, y le tatuó un mensaje para coordinar una invasión a Persia. Luego esperó a que le creciera el cabello y lo envió donde Aristágoras de Mileto, quien le envió a rasurar de nuevo la cabellera con el fin de ver el mensaje oculto.

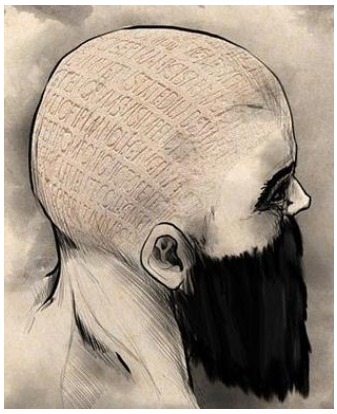


Figura 1 Mensaje oculto enviado a modo de tatuaje en la cabeza de un criado de Histeo.

El otro caso interesante del uso de la estenografía en tiempos antiguos, mencionado en el libro de Heródoto de Halicarnaso, donde el noble Hárpago decidió utilizar el cuerpo de un conejo como medio portador de un mensaje. Según el relato y tal como se ilustra en la Figura 2, un mensaje secreto fue escrito en el vientre rasurado de la liebre. Un mensajero fue enviado, vestido como un cazador, con la liebre donde el Rey Ciro. Se solicitó que el Rey Ciro degollara

personalmente la liebre, quien encontró el mensaje enviado por Hárpago donde le ofrecía ayuda para luchar contra los medos.

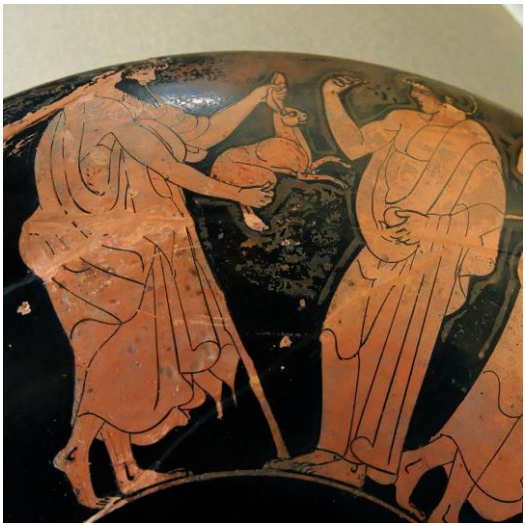


Figura 2 Imagen sólo con fines ilustrativas de la entrega de la liebre con el mensaje oculto.

También en la antigüedad se utilizaba la savia de la planta Tithymallus para escribir mensajes secretos. Lo escrito se vuelve invisible al secarse. Para poder extraer el mensaje oculto, bastaba con calentar un poco la tela, para que el texto escrito sobresalga en un color marrón [21].

Giovanni Battista della Porta, durante el siglo 15, logró escribir mensajes secretos sobre la cáscara de huevos cocidos. La tinta utilizada estaba compuesta por aluminio y vinagre. El mensaje escrito traspasaba la cáscara y quedaba impreso en la albúmina del huevo. De modo que el exterior del huevo quedaba intacto y para poder leer el mensaje bastaba con quitar la cáscara del huevo. Dicha técnica se utilizó para enviar mensajes secretos a presos.

La técnica de escritura con tinta que se vuelve invisible fue utilizada durante la guerra de la revolución de los Estados Unidos con el fin del intercambio de mensajes secretos entre espías [22].

Gaspar Schott, científico alemán, diseñó un método para enviar mensajes secretos utilizando escritura de música. En este caso, algunas notas musicales representaban una letra del mensaje a ocultar. De este modo, a simple vista la partitura no parece esconder un mensaje, e incluso si es interpretada, representa una melodía [23].

En la Figura 3 se muestra un ejemplo del uso de técnicas de esteganografía en partituras la encontramos en algunas obras del artista Johann Sebastian Bach [24], en donde las letras de su apellido se encuentran ocultas en algunas de sus obras variaciones canónicas BWV 769.

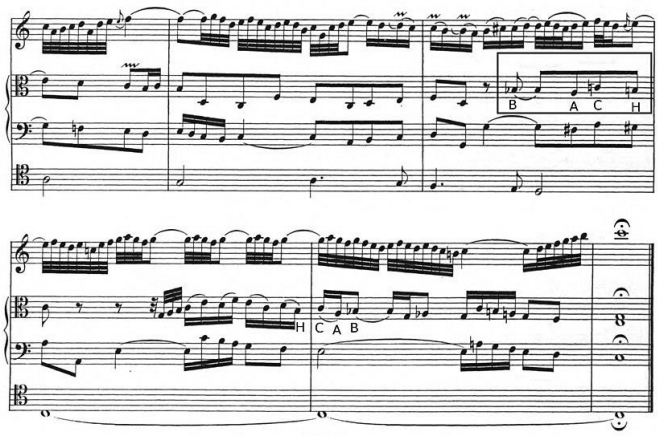


Figura 3 El motivo Bach.

Durante la segunda Guerra mundial, los espías alemanes intercambiaban mensajes secretos impresos en micropuntos de apenas 1 milímetros en fotografías que no eran legibles a primera vista [25] tal y como se muestra en la Figura 4.

Figura 4 Datos ocultos en micropuntos de una fotografía.

2.2 Aplicaciones maliciosas de la esteganografía

Hoy en día, una de las herramientas más útiles para los cibercriminales es el uso de *malware* o *software* malicioso con el fin de afectar la confidencialidad, la integridad e incluso la disponibilidad de los datos.

La esteganografía puede utilizarse para la fuga de información y afectar la confidencialidad de los datos, dado que los mismos pueden ser ocultos en archivos portadores, como imágenes, videos, archivos de ofimática y luego ser enviados hacia los cibercriminales.

También la esteganografía es utilizada por los cibercriminales para el envío de *malware*, o código malicioso oculto en archivos portadores, el cual es luego extraído y ejecutado por los criminales para afectar a las víctimas.

Los cibercriminales están utilizando técnicas de esteganografía para esquivar los sistemas de detección de *malware* al ocultarlas dentro de objetos portadores tales como fotos, videos, archivos de audio, e incluso en el metadata de archivos. Dichos objetos portadores se les conoce como *stegware,* dado que han sido alterados con técnicas de esteganografía para ser

utilizados en la cadena de ataque, ya sea para distribuir *malware*, ser parte de un proceso de armado y compilado del código malicioso, o para facilitar la fuga de información.

A continuación 2 ejemplos recientes de ataques utilizando técnicas de esteganografía.

En agosto 2018, Xiaoqing Zheng, fue arrestado en Estados Unidos, como sospechoso de robar y esconder información sensible sobre tecnologías relacionadas con turbinas desarrolladas por la empresa *General Electric*. El sospechoso utilizó técnicas de esteganografía para esconder archivos con información sensible en imágenes de paisajes, las cuales trató de enviar a su cuenta personal de correo electrónico [26].

Otro ejemplo reciente del uso de esteganografía para realizar ataques cibernéticos fue el documentado por la empresa *Cybereason* [27] quienes detectaron que durante la época de pago obligado de impuestos en Estados Unidos del periodo 2020-2021, un grupo criminal estuvo enviando campañas de phishing, es decir correos masivos con técnicas de manipulación y engaño, a ciudadanos estadounidenses. El correo incluía una serie de documentos adjuntos e hipervínculos dirigidos a archivos, que supuestamente podrían utilizarse en la declaración de impuestos a realizar a partir del 15 de abril del 2021. Dichos documentos contenían de forma oculta, macros creadas con lenguaje *Visual Basic* para Aplicaciones con instrucciones de descarga de imágenes, bibliotecas DLL y código malicioso. Además, estaban infectados con el Troyano Remcos, que aparecen en la Figura 5, el cual es ofrecido como *Malware* as a Service con el fin de tener acceso remoto no autorizado a los equipos infectados, controlarlos de forma remota y robar información sensible. Los troyanos enviados de forma oculta a través de instrucciones de funciones de *dropper* (para descarga) fueron *Netwire* y *Remcos.* Dichos troyanos son distribuidos por redes criminales como *Malware-as-a-service* (MaaS) y están diseñados para la ejecución remota de comandos maliciosos, descarga de payloads para explotar vulnerabilidades e incluso espionaje para apoyar en la fuga de información sensible.

Figura 5 Troyano Remcos ofrecido como Malware as a Service.

2.3 Esteganografía digital

Hoy en día se utilizan medios portadores, como lo son archivos de texto, imágenes, archivos de audio, archivos de ofimática, incluso archivos del sistema operativo y registros de Windows para ocultar mensajes secretos, datos y archivos.

Para seleccionar el medio portador se toman en cuenta múltiples elementos como lo son la cantidad de datos que deben ser ocultados, la probabilidad de que los datos ocultos sean descubiertos durante un estegoanálisis del archivo portador y cuantas alteraciones soporta el medio portador sin que se pierdan los datos ocultos.

En la Tabla 1, se detallan los elementos requeridos para esconder datos o archivos en imágenes utilizando esteganografía:

Elemento	**Descripción**
Secreto	Información o datos que desean ser enviados de forma oculta.
Canal de comunicación	Medio de transmisión de datos.
Portador	También conocido como objeto portador, es el objeto, por ejemplo, un audio, imagen, archivo de ofimática, que se utilizará para ocultar el archivo o datos.
Estego-clave (conocida como stego-key)	Utilizada para cifrar la información a ocultar. También se puede generar subclaves con el definir las sesiones del medio portador que serán utilizadas para esconder los datos requeridos.
Estego-Objeto (conocido como *Stegware*)	Contiene los datos a esconder. En caso de que el Portador utilizado sea una imagen y que esté utilizando para enviar *malware* o código malicioso, al estego-objeto se le conoce como *Stegware* [28].
Esteganografiado	Proceso para ocultar información con técnicas de esteganografía en un portador.
Desesteganografiado	Proceso para extraer los datos ocultos por esteganografía de un portador.
Estegoanálisis	Proceso de inspección y análisis de un portador con el fin de descubrir si contiene datos ocultos, determinar su tamaño, así como el algoritmo de esteganografía utilizada.

Tabla 1 Elementos del proceso de estegoanálisis.

A continuación, en la Figura 6 se muestran los pasos y elementos de un sistema de esteganografiado y estegoanálisis en una imagen.

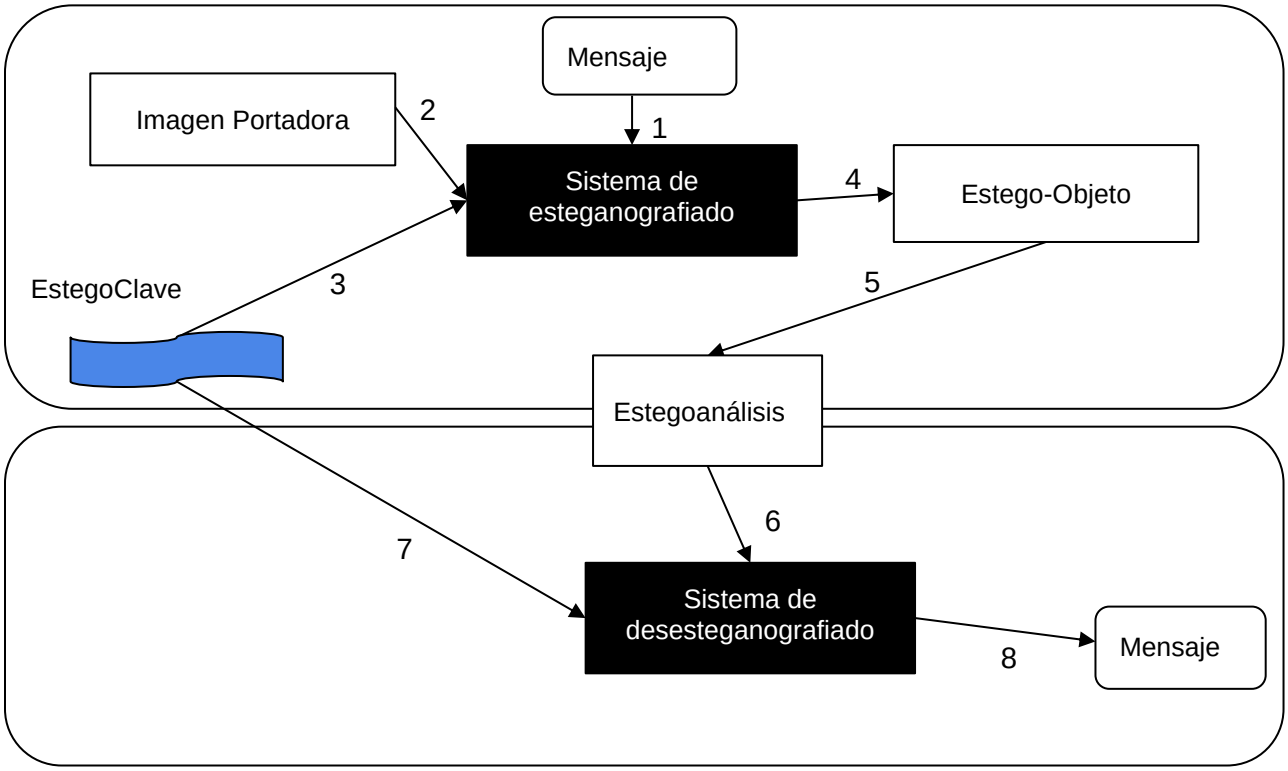


Figura 6 Proceso de estegoanálisis.

A continuación, se detallan los pasos representados en la Figura 6 sobre el proceso de aplicación de esteganografía en imágenes:

1. El mensaje secreto puede ser un archivo con datos o bien el texto con el mensaje a enviar. Este es enviado a un sistema de esteganografiado para su respectivo análisis y ocultamiento.
2. Se utiliza un archivo portador, por ejemplo, una fotografía en formato BMP o PNG, de ser posible en alta resolución y con un tamaño en bytes superior a los datos o archivo a ocultar.
3. Una estegoclave es utilizada tanto por el emisor como el receptor de la imagen o archivo con los datos ocultos (estego-objeto) con el fin de cifrar y definir donde ocultar los datos.
4. El sistema de esteganografiado, tomando en cuenta la estegoclave, integrará los datos a ocultar en el archivo portador. Durante este proceso se creará un nuevo archivo conocido como estego-objeto el cual contendrá los datos ocultos.
5. El emisor enviará el estego-objeto a través de un canal de comunicación, por ejemplo, vía correo electrónico, al receptor.
6. A través de un proceso conocido como estegoanálisis, se procederá a verificar si el estego-objeto contiene algún dato oculto.

7. El sistema de desesteganografiado utilizará la estegoclave con el fin de identificar e descifrar los datos ocultos.
8. Se procede con la extracción del mensaje o datos ocultos.

Es importante resaltar que durante el esteganografiado se evalúan otros factores tales como los siguientes:

Capacidad
La cantidad de datos que se podrían ocultar en el objeto portador. El tamaño del objeto portador debe ser mayor que el tamaño de los datos a ocultar, de modo que permita ocultar datos durante el proceso de esteganografiado sin que genere sospechas de la alteración realizada.

La robustez
Consiste en la resistencia de la información oculta y su capacidad de permanecer intacta y legible a pesar de cambios realizados en el elemento portador. Depende del método de esteganografiado realizado, algunos cambios en el elemento portador podrían afectar los datos ocultos e incluso imposibilitar la extracción de estos.

Por ejemplo, al enviar el estego-objeto que aparece en la Figura 7 a través de un sistema de mensajería de una red social, la misma es analizada, comprimida y alterada por el sistema de mensajería.

Figura 7 Ejemplo de estego-objeto.

A continuación, en la Figura 8 se muestran las propiedades de la imagen antes de ser enviada por la aplicación móvil *Whatsapp* [29].

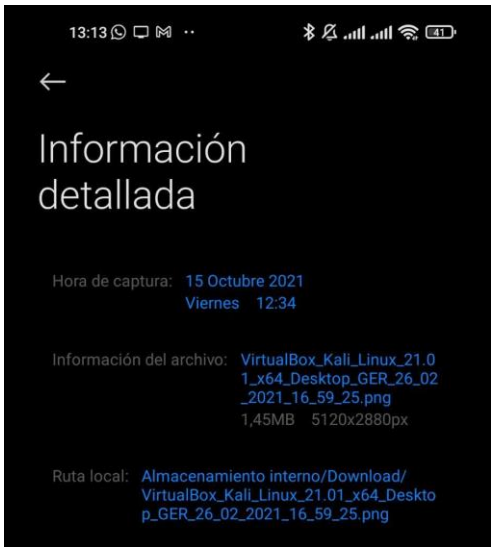


Figura 8 Propiedades de estego-objeto antes de ser enviado por una red social.

En Figura 9 se muestran las propiedades de la imagen después de ser enviada por la aplicación móvil *Whatsapp* a un contacto a través de un mensaje instantáneo.

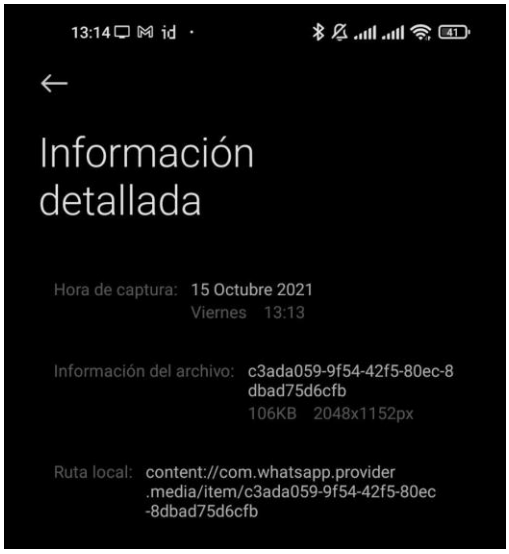


Figura 9 Propiedades de estego-objeto después de ser enviado por una red social.

Como se puede apreciar en la Figura 9, tanto el tamaño como la resolución de la imagen original fueron alterados antes de ser enviados, lo que puede impactar en la integridad y disponibilidad de los datos ocultos.

Lo mismo sucede al publicar una imagen con datos ocultos en una red social como *Facebook* o *Linkedin*, algunas características del archivo como resolución y tamaño son alterados.

Seguridad de un método de esteganografía
Es la robustez que tienen los datos ocultos ante procesos de estegoanálisis para dificultar la extracción de los datos.

Existen distintos métodos para aplicar encriptación y ofuscación durante el proceso de esteganografiar un mensaje secreto en un archivo portador. A continuación, la descripción del algoritmo *Least Significant Bit* (LSB), el cual es uno de los algoritmos más comunes de esteganografía en la era digital [30].

2.3.1 Código RGB

Los archivos de imágenes están compuestos por píxeles. El tono de color que representa cada píxel se encuentra código utilizando el sistema de RGB, los cuales utilizan los colores primarios Rojo (*Red*), Verde (*Green*) y Azul (*Blue*) con el fin de generar cada uno de los tonos a utilizar en los componentes de una imagen.

Cada color primario se representa con 8 bits, es decir en valores decimales de 0- 255. Los tonos de color se calculan con base al código conocido como RGB con un total de valor de 24 *bits* donde R representa el valor deseado en Rojo, G el valor deseado en Verde y B el valor deseado en Azul.

El valor resultante de la combinación de los colores primarios se representa con valores hexadecimales de la siguiente manera:

Por ejemplo, en la Figura 10 encontramos el tono rojizo de un píxel de una imagen:

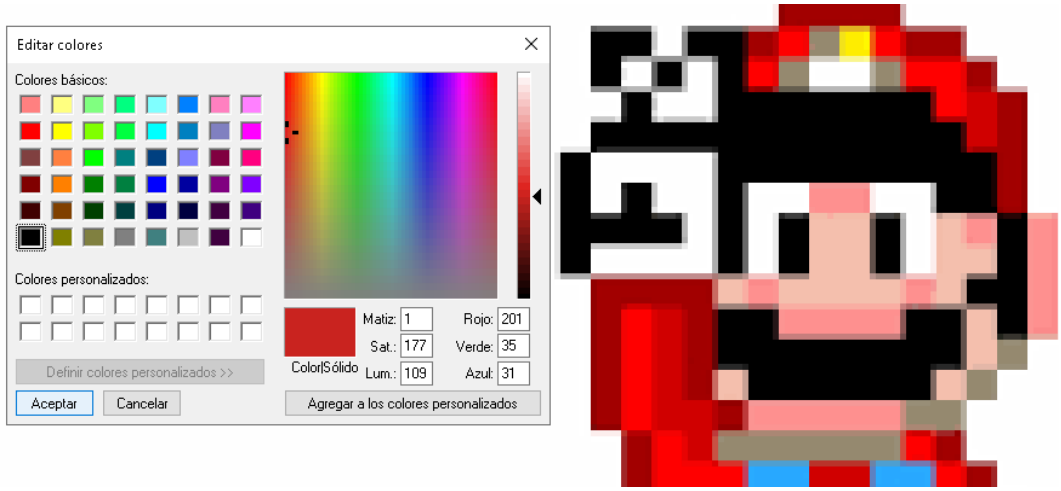


Figura 10 Imagen para representar píxeles.

En la Tabla 2 se representan los cálculos a realizar con el código RGB, si se desea obtener la tonalidad rojiza del píxel mencionado representado en la Figura 11.

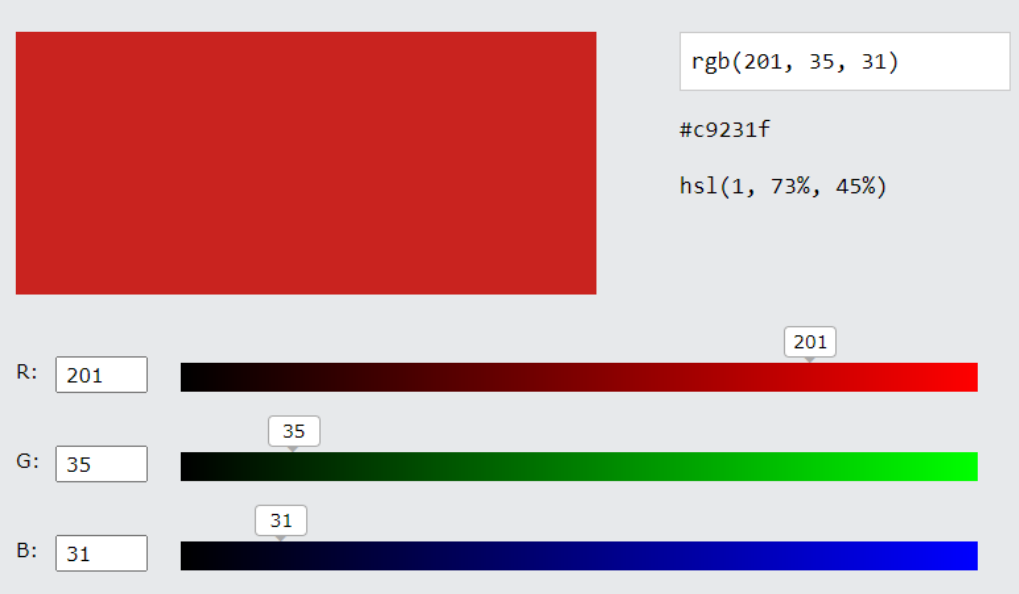


Figura 11 Uso del sistema de codificación de color RBG.

Código de color	**Decimal**	**Binario**	**Hexadecimal**
R	201	11001001	0x**c9**
G	35	00100011	0x**23**
B	31	00011111	0x**1f**

Tabla 2 Ejemplo de uso de código RGB en imágenes.

El tono de color resultante en este ejemplo sería de **c9231f en hexadecimal.**

2.3.2 Algoritmos de esteganografía en imágenes

El objetivo de aplicar técnicas de esteganografía en imágenes es lograr esconder datos o archivos dentro de imágenes, de modo que el estego-objeto resultante no genere sospechas que tiene algo oculto. El archivo de imagen resultante que contiene los datos ocultos (estego-objeto) debe representar la misma imagen que el archivo portador, de modo que a simple vista no genere sospechas.

Para esto lo que se recomienda es el uso de archivos portadores con alta resolución y con variedad de colores y que sean lo suficientemente grande, de modo que, al estar compuesto por una cantidad elevada de píxeles, al modificar una serie de píxeles para que escondan parte de la información o del archivo requerido, genere un código RGB muy parecido al código RGB del píxel original.

Entre los algoritmos más conocidos de esteganografía sobresalen:

- *Least Significant Bit* (LSB) [31]
- Ocultamiento y Cifrado (*Encrypt and Scatter*) [32]
- *Redundant Pattern encoding* [33]
- *Masking and Filtering* [34]
- *Coding and Cosine Transformation* [35]

2.3.3 Algoritmo de esteganografía digital *Least Significant Bit* (LSB)

El algoritmo *Least Significant Bit* o LSB consiste en sustituir los bits menos significativos de un píxel de una imagen con los bits de los datos a ocultar.

LSB es uno de los más utilizado dado su facilidad de uso [36] y que está diseñado para permitir esconder datos sin alterar en gran medida el medio portador, de modo que el estego-objeto resultante sea muy parecido al medio portador original y no genere sospechas al interactuar con el objeto.

Las imágenes están compuestas por píxeles, los cuales representan tonos de color basados en el sistema de codificación de color RGB mencionado anteriormente, el cual muestra el nivel de rojo, azul y verde que será mostrado en el píxel.

La selección del *bit* a sustituir se puede realizar de distintas maneras. Por ejemplo, a través de LSB secuencial [37] es posible sustituir los bits menos significativos utilizados para codificar el tono de color de un píxel a partir de una posición específica en la imagen. Un ejemplo práctico está disponible en la herramienta https://stegonline.georgeom.net/upload y se muestra en la Figura 12, en donde se definen en cuales sectores de las imágenes se realizarán los cambios de los bits menos significativos.

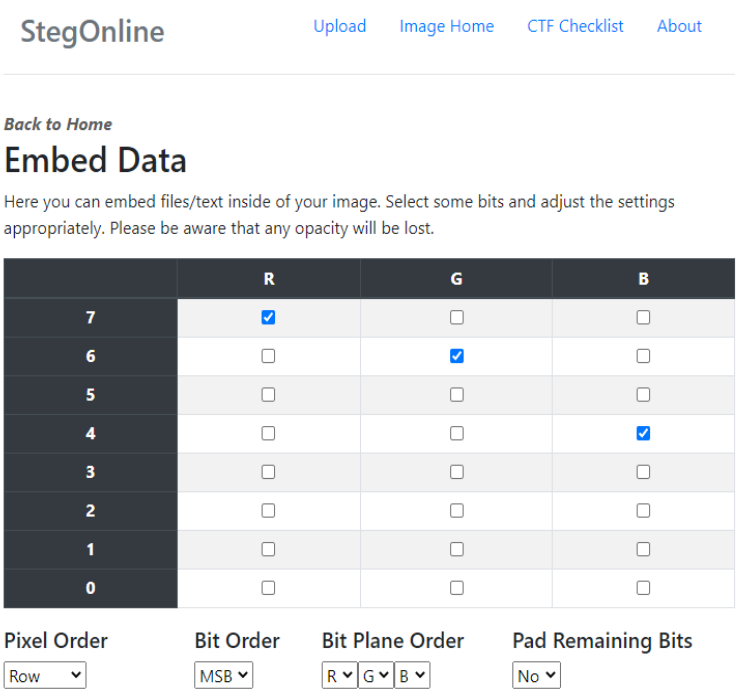


Figura 12 Uso de StegOnline para ocultar datos en imágenes.

Existen varios métodos de aplicación de esteganografía en imágenes basadas en LSB como lo son:

- Esteganografía de dominio espacial [38]
- Esteganografía de dominio de frecuencia [39]

El enfoque de este proyecto se basa en el uso de herramientas que permitan el estegoanálisis de archivos que hayan sido modificados utilizando técnicas de LSB de dominio espacial.

2.3.3.1 Algoritmo de esteganografía usando LSB de dominio espacial

A través de la técnica de LSB de dominio espacial se van cambiando los valores de los píxeles directamente, es decir, el código RGB, con el fin de alterar ciertos bits para sustituirlos por los datos que se desean ocultar.

Los siguientes algoritmos pueden ser utilizados con el fin de seleccionar los píxeles a alterar:

- Algoritmo de LSB de dominio espacial sin filtrado
- Algoritmo de LSB de dominio espacial aleatorio
- Algoritmo de LSB de dominio espacial con filtrado

2.3.3.1.1 Algoritmo de LSB de dominio espacial sin filtrado

Este es un método sencillo donde se sustituye cada *Least Significant Bit* (LSB) del píxel de una imagen por un bit del mensaje o dato a ocultar. A través de este método se analiza la imagen y al detectar el LSB de un píxel, empieza el proceso de sustitución. Luego de forma secuencial, busca el LSB del siguiente píxel y realiza la sustitución con otro bit del mensaje o dato a

ocultar. Esto provoca que los datos ocultos están en píxeles cercanos y no permite una distribución balanceada en la imagen. También facilita su detección y posible extracción de los datos dada la facilidad de detección de los píxeles alterados.

2.3.3.1.2 Algoritmo de LSB de dominio espacial aleatorio

En este método se utiliza una estegoclave y un generador de números pseudo aleatorios [40]. Esto genera una secuencia que identifica los píxeles que se usarán para sustituir los LSBs por bits de datos a ocultar. Esto permite que los píxeles alterados están distribuidos en la imagen y no agrupados como en el caso del algoritmo anterior.

2.3.3.1.3 Algoritmo de LSB de dominio espacial con filtrado

En este método se aplica un filtro en los bits menos significativos (LSB) de cada píxel y oculta los datos en el *Least Significant Bit* del píxel. En este caso los bits usados durante el proceso de filtrado no son alterados, solamente el *Least Significant Bit*, lo que garantiza que al volver a aplicar el filtro en los píxeles durante el proceso de estegoanálisis se podrá identificar en cuál de los bits menos significativos se encuentra un dato oculto.

2.4.2 Herramientas comunes de esteganografía en imágenes

A continuación, en la Tabla 3 se muestra un listado de herramientas disponibles para aplicar técnicas de esteganografía.

Herramienta	**Descripción**
Steghide	● Compatible con archivos de audio (WAV y AU) y de imágenes (BMP y JPEG). ● La versión actual es 0.5.1 y está escrita en C++. ● Diseñado para sistemas operativos Windows y distribuciones Linux como SuSE, Mandrake y Debian. ● En la Figura 13 se muestran un extracto de las opciones del menú que se pueden ejecutar desde la línea de comandos.

<table>
<tr><td></td><td>

```
┌──(kali㉿kali)-[~]
└─$ steghide --help
steghide version 0.5.1

the first argument must be one of the following:
 embed, --embed            embed data
 extract, --extract        extract data
 info, --info              display information about a cover- or stego-file
   info <filename>         display information about <filename>
 encinfo, --encinfo        display a list of supported encryption algorithms
 version, --version        display version information
 license, --license        display steghide's license
 help, --help              display this usage information

embedding options:
 -ef, --embedfile          select file to be embedded
   -ef <filename>          embed the file <filename>
 -cf, --coverfile          select cover-file
   -cf <filename>          embed into the file <filename>
 -p, --passphrase          specify passphrase
   -p <passphrase>         use <passphrase> to embed data
 -sf, --stegofile          select stego file
   -sf <filename>          write result to <filename> instead of cover-file
 -e, --encryption          select encryption parameters
```

Figura 13 Opciones de línea de comandos de steghide.
</td></tr>
<tr><td>OpenStego [13]</td><td>

- Compatible con archivos bmp, gif, jpeg, png, tif, tiff, wbmp.
- La versión más reciente es 0.8.4 y está escrita en Java.
- Requiere instalación en ambiente Windows.
- En la Figura 14 se muestra la interfaz gráfica de la herramienta diseñada con Java que permite ocultar archivos en otros archivos portadores y extraer los archivos ocultos.

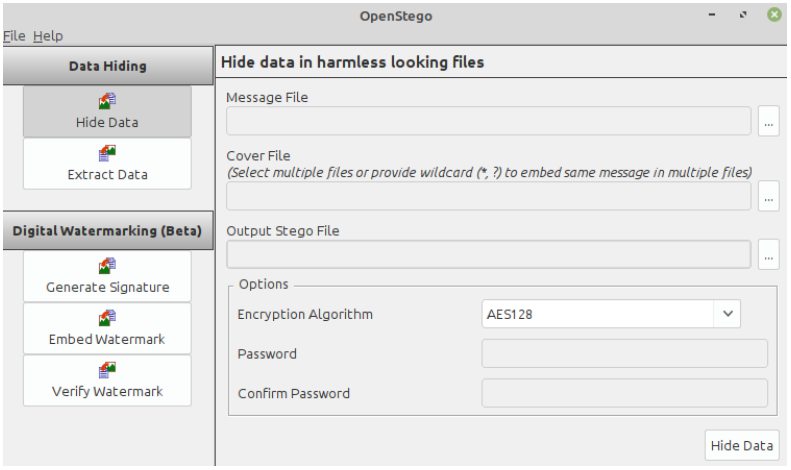
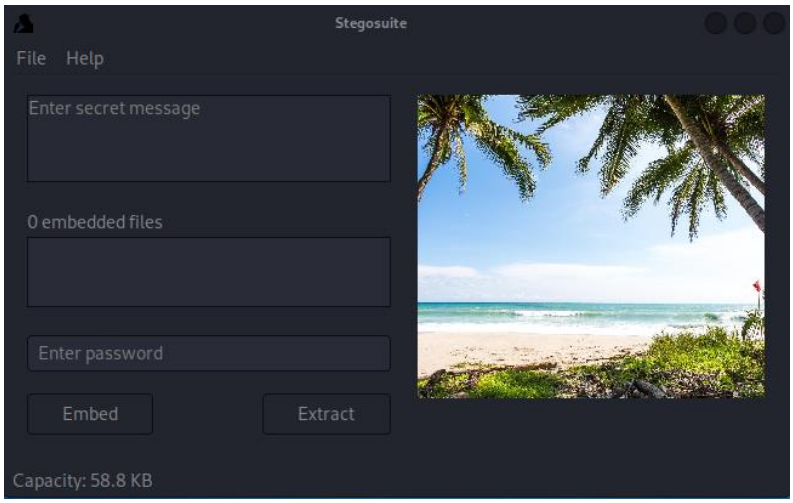
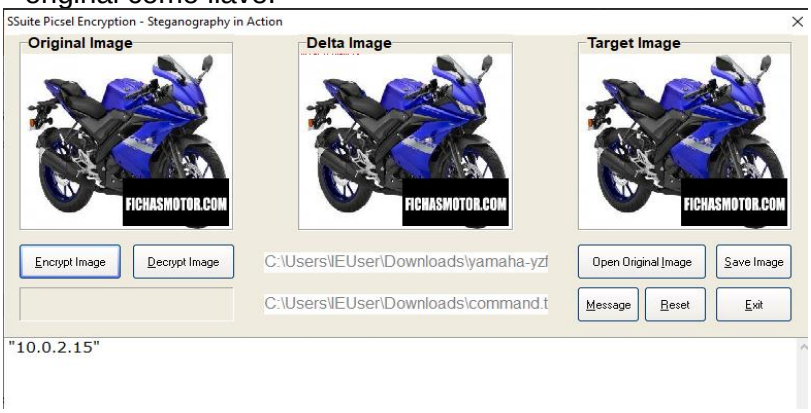


Figura 14 Interfaz gráfica de OpenStego.
</td></tr>
<tr><td>StegoSuite [41]</td><td>

- Compatible con archivos bmp, gif, jpeg, png.
- La versión más reciente es 0.8 y está escrita en Java.
- Compatible con ambientes Windows de 64 bits y Linux de 64 bits además de Debian o Ubuntu.
- En la Figura 15 se muestra la interfaz gráfica que permite ocultar y cifrar datos.
</td></tr>
</table>

	 *Figura 15 Interfaz gráfica de StegoSuite.*
SSuite Picsel [42]	• Compatible con imágenes en formato BMP, JPG, PNG y WMF. • La versión más reciente es 2.8.1 y está escrita en Java. • Diseñado para ejecutarse en ambiente Windows tanto de 32 como de 64 bits. • En la Figura 16 se muestra la interfaz gráfica de Suite Picsel, la cual es una aplicación portátil que no requiere instalación que permite esconder archivos de texto en una imagen. Para extraer el archivo de texto requiere que se utilice el estego-objeto (imagen con archivo de texto oculto) y se debe utilizar la imagen original como llave.  *Figura 16 Interfaz gráfica de SSuite Picsel.*
SteganPeg [43]	• Compatible sólo con archivos jpeg. • La versión más reciente es 1.0 y está escrita en Java. • Requiere instalación en sistema operativo Windows. • En la Figura 17 se representa la interfaz gráfica [44] de SteganPeg la cual permite esconder múltiples archivos dentro de una imagen jpeg basada en compresión DCT [45]

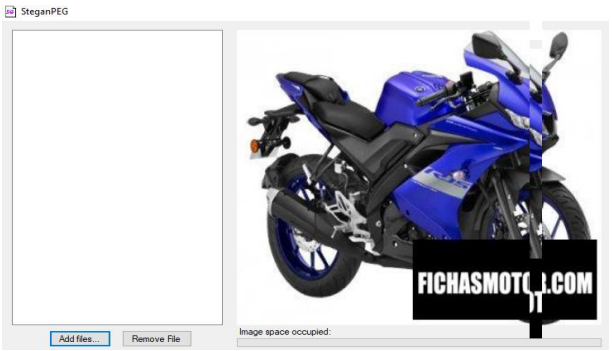
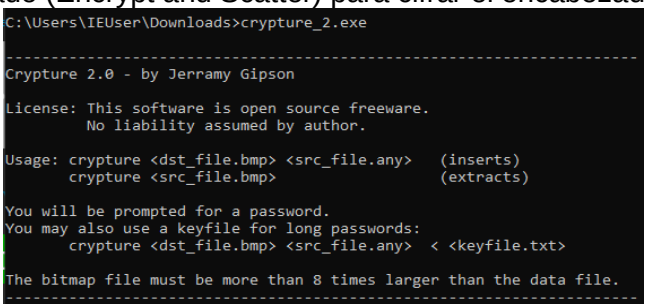
	 *Figura 17 Interfaz gráfica de SteganPEG.*
Crypture [46]	• Compatible sólo con archivos bmp. • La versión mas reciente es 2.0 y está escrito en C. • Se puede ejecutar desde equipos con ambientes Windows o y distribuciones Linux como Debian o Ubuntu. • En la Figura 18 se visualiza la interfaz en línea de comandos de Cypture, el cual es un ejecutable de apenas 6 KB que no requiere instalación y que además utiliza técnicas de Ocultamiento y Cifrado (Encrypt and Scatter) para cifrar el encabezado de datos.  *Figura 18 Interfaz de comandos Crypture.*

Tabla 3 Herramientas Comunes de esteganografía.

2.5 *Living off the land* y esteganografía en imágenes

El concepto *Living off the Land (LotL)* consiste en realizar acciones sin necesidad de descargar, instalar ni hacer cambios en la información del sistema operativo de la máquina a atacar. Se basa en realizar acciones maliciosas conocidas como *Fileless malware*, donde se pretende utilizar archivos binarios, bibliotecas y scripts existentes para afectar la confidencialidad, la integridad, y la disponibilidad de los datos.

Los sistemas operativos tienen múltiples herramientas precargadas que se pueden utilizar para aplicar esteganografía en archivos portadores de imágenes, texto, audio y video. A continuación, un listado de herramientas legítimas que podrían utilizarse con ese fin.

Comando `cat` en Linux

El comando cat [47] es muy utilizado en linux dado que permite entre otras cosas, crear

archivos, concatenar archivos, ver el contenido de archivos, hacer copias del contenido de los archivos.

También se puede utilizar para esconder datos en archivos de imágenes. Basta con aplicar la siguiente sintaxis:

cat < archivo portador> <archivo a esconder> **>** <nombre del estego-objeto>.

Por ejemplo en la Figura 19 se muestra el siguiente comando requerido para esconder el archivo listado_clientes.txt en una imagen.png

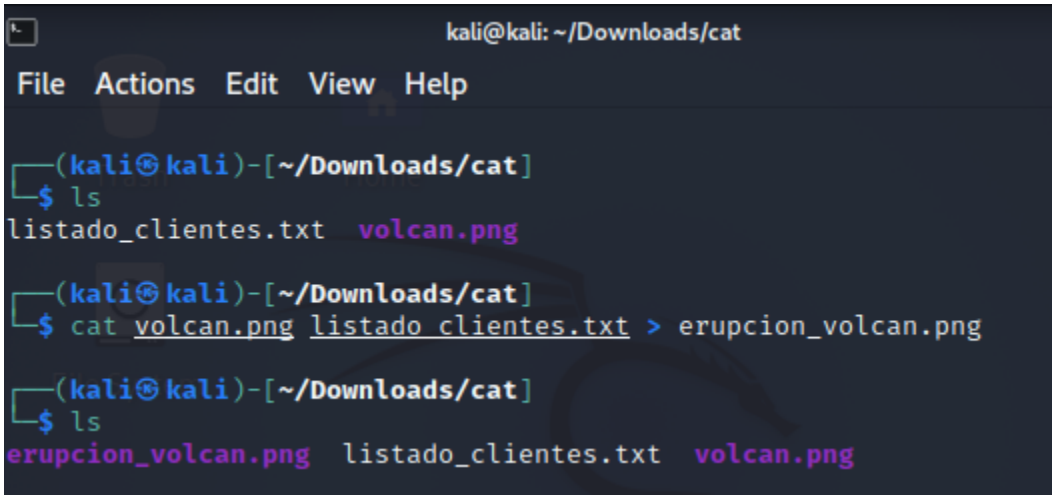


Figura 19 Uso de comando cat para aplicación de esteganografía.

En este caso, se utilizó el archivo portador (volcan.png) para crear el estego-objeto (erupcion_volcan.png) el cual tiene oculto y codificado la información del archivo con datos de interés (listado_clientes.txt).

Comando copy en Windows

El comando copy [48] viene incorporado en el sistema operativo Windows y no requiere permisos administrativos o elevados para poder ejecutarlo. Normalmente para realizar respaldos de archivos o carpetas con archivos. Sin embargo, se puede utilizar para modificar imágenes a nivel binario con el fin de escoger datos, texto u otros archivos.

Esto se logra de la siguiente manera:

Copy /b < archivo portador> <archivo a esconder> **>** <nombre del estego-objeto>.

2.6 Estegoanálisis

Es el estudio que permite detectar el uso técnicas de esteganografía en un archivo portador para esconder datos.

El proceso consiste en detectar si un archivo ha sido alterado para ocultar datos, identificar el algoritmo utilizado y de ser posible, enumerar posibles herramientas utilizadas para esconder los datos.

De esta manera, se puede utilizar la información de inteligencia obtenida en el proceso de estegoanálisis para luego tratar de extraer la información oculta.

Entre las técnicas más utilizadas para detectar el uso de esteganografía en imágenes sobresalen las siguientes:

- Detección visual de imágenes [49]
- Detección estadística basado en el análisis de cambios en patrones de píxeles a través de técnicas de LSB o a través de análisis de histogramas [50]
- Análisis de atributos de las imágenes y su contenido. Por ejemplo, análisis de metadata de las imágenes para detectar mensajes ocultos [51]

2.6.1 Herramientas de estegoanálisis en imágenes

En la Tabla 4 aparece un listado de herramientas comunes de estegoanálisis en imágenes

Herramienta	**Descripción**
zteg. [52]	● Permite analizar archivos bmp o png. ● La versión más reciente es 0.2.9 y está escrita con Ruby. ● Diseñada para ejecutarse en equipos con sistema operativo Linux. ● En la Figura 20 se muestra cómo analizar una imagen con *zsteg*.

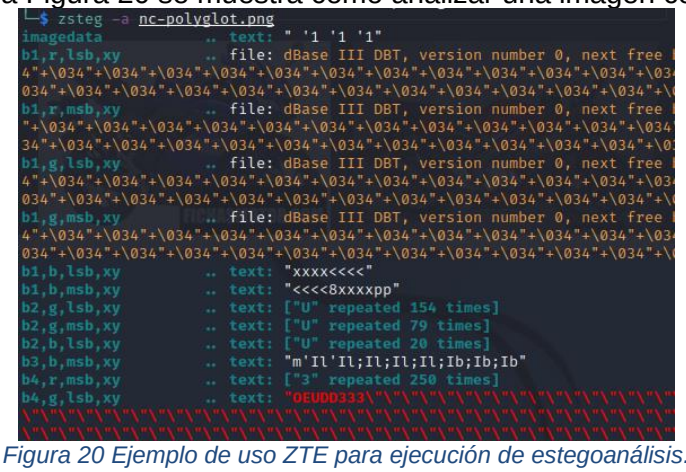
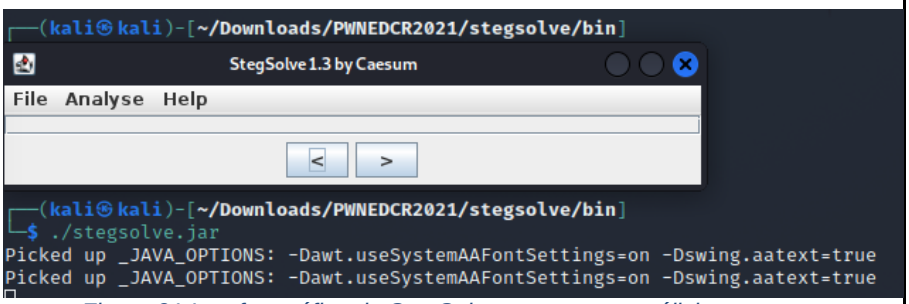
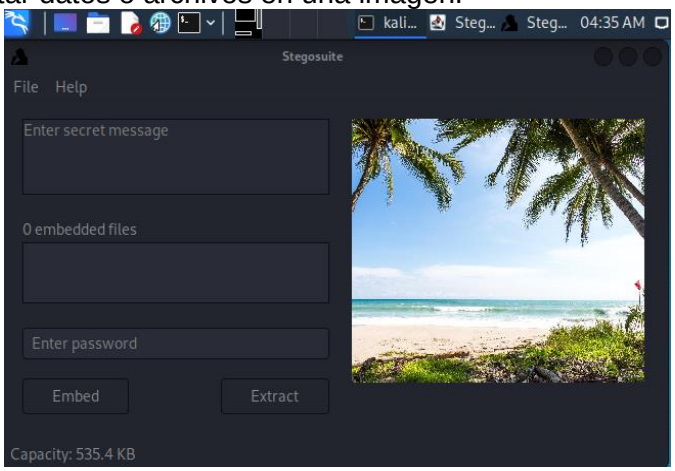
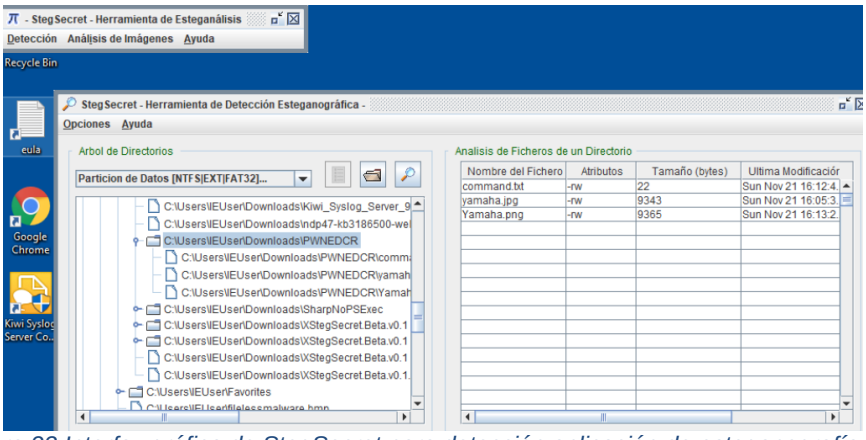
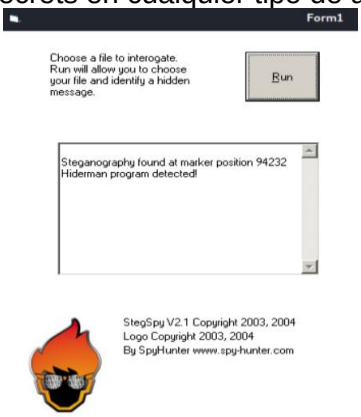


Figura 20 Ejemplo de uso ZTE para ejecución de estegoanálisis.

StegSolve [53]	• Permite analizar archivos gif, png y jpeg. • La versión más reciente es 1.3 y está escrita con Java. • Compatible en ambiente Windows y Linux. • La interfaz gráfica se muestra en la Figura 21 la cual permite el estegoanálisis visual de imágenes.  *Figura 21 Interfaz gráfica de StegSolve para estegoanálisis.*
Stegosuite	• Permite el análisis de archivos bmp, gif, jpeg, png. • La versión más reciente es 0.8 y está escrita en Java. • Compatible con ambientes Windows de 64 bits y Linux de 64 bits además de Debian o Ubuntu. • En la Figura 22 se muestra la interfaz gráfica de Stegosuite para ocultar datos o archivos en una imagen.  *Figura 22 Interfaz gráfica de StegSuite para detección de archivos ocultos en imágenes.*
StegSecret [54]	• Permite el estegoanálisis visual de imágenes jpg, png, bmp, así como archivos de datos como pdf, zlibs. • La versión más reciente es 0.1 y está escrita en Java. • Compatible con ambientes Windows de 64 bits y Linux de 64 bits además de Debian o Ubuntu. • En la Figura 23 se muestra la interfaz gráfica que permite detectar el uso de esteganografía en imágenes.

	 *Figura 23 Interfaz gráfica de StegSecret para detección aplicación de esteganografía en imágenes.*
StegSpy [55]	• Permite analizar archivos jpg y png. • La versión más reciente es 2.1 y está escrita con Visual Basic 6.0 aunque no es de código de fuente abierto. • Diseñado para su ejecución en ambiente Windows a través de un archivo ejecutable .exe. • En la Figura 24 se muestra la interfaz gráfica de StegSpy para la detección de uso de esteganografía con JPHideandSeek, Masker, JPegX, Invisible Secrets en cualquier tipo de archivo.  *Figura 24 Interfaz gráfica de StegSpy para detección aplicación de esteganografía.*
Stegano [56]	• Permite analizar imágenes jpeg y tiff para para detectar el uso de LSB en imágenes de color y en el estegonálisis estadístico. • La versión más reciente es 0.10.2 y está escrito con Python. • Diseñado para ejecutarse en ambiente Linux. • Como se muestra en la Figura 25, Stegano es un utilitario para esconder mensajes ocultos en imágenes utilizando la técnica de LSB (*Least Significant Bit*).

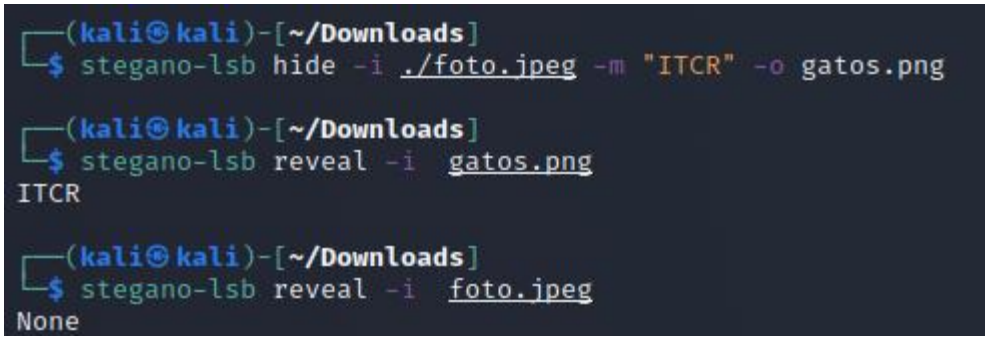
<table>
<tr><td></td><td>


Figura 25 Ejemplo de uso de Stegano para ocultar datos y su respectiva extracción.
</td></tr>
<tr><td>Hidebehind [57]</td><td>

Compatible con archivos de imagen como png y jpg, además de archivos de video y de audio.
Herramienta creada con python que permite esconder archivos en imágenes utilizando el protocolo LSB.
Compatible con equipos en ambiente Linux,
En la Figura 26 se muestra los comandos para extraer los datos o archivos ocultos de una imagen además de cifrar el estego-objeto con el uso de herramientas como GnuPG (Gnu Privacy Guard) [58]

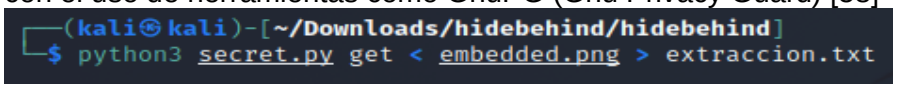


Figura 26 Ejemplo de uso de HideBehind.
</td></tr>
<tr><td>StegExpose [59]</td><td>

Compatible con imágenes png y bmp.
Herramienta diseñada con Java.
Compatible con ambientes Windows de 64 bits y Linux de 64 bits además de Debian o Ubuntu.
En la Figura 27 se presenta los pasos para detectar el uso de esteganografía en las imágenes presentes en un folder llamado testfolder.

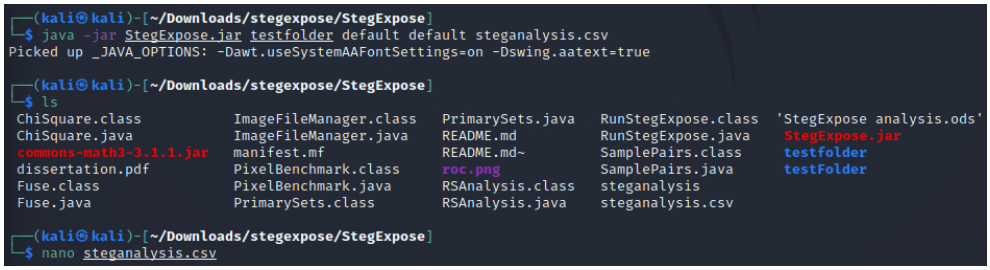


Figura 27 Ejecución de estegoanálisis con StegExpose
</td></tr>
</table>

Tabla 4 Herramientas comunes de estegoanálisis.

3. Metodología

Con el fin de mejorar la capacidad de detección y extracción de datos ocultos en imágenes con técnicas de esteganografía se realiza un estudio de las herramientas de esteganografía existentes que permiten el estegoanálisis de archivos, además de documentar sus funciones e implementación.

Los hallazgos relacionados con las herramientas de estegoanálisis y su respectiva implementación son documentados en una página *web* dedicada a este proyecto. La página web también incluye información general relacionada con la esteganografía con el fin de informar a la comunidad de profesionales en ciberseguridad del uso malicioso y riesgos de la esteganografía.

Se desarrolla la biblioteca Oculto la cual permite combinar la capacidad de estegoanálisis de múltiples herramientas en archivos de imágenes. Este proyecto es de código de fuente abierto basado en bibliotecas disponibles en *Python* y otros lenguajes de programación que permiten el análisis de imágenes en busca de datos ocultos con técnicas de esteganografía.

Se coordina una competencia tipo CTF con el fin de evaluar la capacidad de la biblioteca Oculto para la detección de datos o archivos ocultos en imágenes a través del uso de esteganografía basada en LSB.

4. Oculto: Biblioteca de estegoanálisis

4.1 Descripción

La Biblioteca Oculto tiene como objetivo principal analizar imágenes para verificar si contienen contenido oculto con técnicas de esteganografía.

Oculto integra funciones de varias herramientas con el fin de agilizar el proceso de estegoanálisis. Esta integración busca mejorar los tiempos requeridos para definir si se ha aplicado técnicas de esteganografía en un posible estego-objeto así como disminuir fallas en el proceso de estegoanálisis por errores humanos.

Oculto es modular y está enfocado en el proceso de estegoanálisis. La Biblioteca Oculto con su respectivo código fuente está disponible para descarga. Tiene la posibilidad de que se pueda extender para que sea como un servicio en la nube, donde reciba el archivo y se realice el respectivo análisis, además que permitir la inclusión de nuevas herramientas e incluso módulos de *ethical hacking* con esteganografía.

4.2 Requerimientos

- Oculto será diseñado para ejecutarse desde distribuciones Debian de Linux y derivadas.
- Instalación de las bibliotecas y programas utilizados por la biblioteca Oculto y descritos en el apéndice 9.2.

4.3 Módulos

Oculto tendrá los siguientes módulos de estegoanálisis:

- Preanálisis del archivo
- Detección de uso de esteganografía LSB en imágenes.
- Detección de uso de esteganografía en el metadata de imágenes.

4.4 Flujo de trabajo

En la Figura 28 se muestra el flujo de trabajo de Oculto para realizar el estegoanálisis en imágenes.

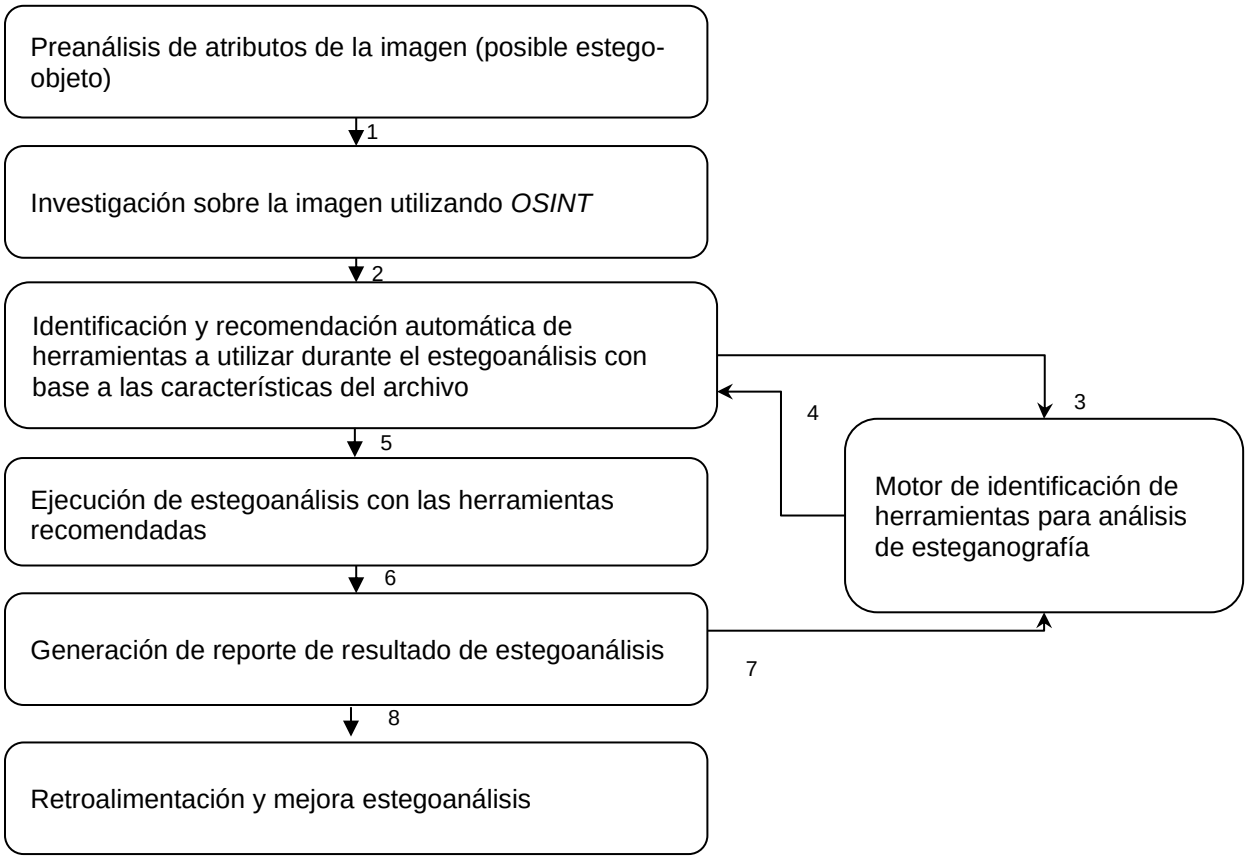


Figura 28 Flujo de estegoanálisis utilizado por Oculto.

A continuación, la descripción del flujo de trabajo mostrado en la Figura 28.

Paso 1: Oculto está diseñado para contar con un módulo de preanálisis de la imagen para documentar las características de la imagen representadas en la Tabla 5 a continuación:

Característica de la imagen	**Descripción**
Formato	Identificación del formato de la imagen (por ejemplo bmp, png…).
Dimensiones (píxeles)	Cantidad de píxeles que conforman la imagen.
Fecha de creación	Fecha o timestamp de creación del archivo de imagen.

Fecha de última modificación	Fecha de creación del archivo de imagen.
Tamaño en bytes	Tamaño de la imagen.
Color	Identificar si es una imagen a color, o escala de grises.

Tabla 5 Atributos de una imagen.

Paso 2: Utilizando herramientas de *open source intelligence (OSINT),* Oculto obtiene información sobre la imagen como el tipo de imagen, calcular el *hash*, verifica la reputación de la imagen y obtiene atributos de esta.

Paso 3: Oculto envía por parámetro las características del archivo a un módulo que permite identificar posibles herramientas de estegoanálisis que pueden ser utilizadas para la detección de uso de esteganografía. Dicho módulo recomienda el uso de ciertas herramientas con base a los siguientes criterios:
1- Extensión del archivo
2- Fecha de creación del archivo
3- Color
4- Atributos
5- Metadata

Paso 4 Motor de identificación de herramientas recomendables para estegoanálisis
Oculto utiliza un motor de identificación de herramientas para análisis de esteganografía el cual selecciona las herramientas a utilizar, dependiendo del tipo de archivo a analizar, así como las técnicas de esteganografía que pueden detectar.

Paso 5 Ejecución de estegoanálisis con las herramientas recomendadas
Oculto ejecuta el estegoanálisis del archivo con las herramientas recomendadas, mostrando en pantalla el avance del estegoanálisis.

Paso 6 Generación de reporte de resultado de estegoanálisis
Al finalizar el proceso de estegoanálisis, Oculto genera un reporte automático con el resultado del estegoanálisis. El objetivo es definir si la imagen fue alterada para ocultar información y de ser posible identificar el algoritmo utilizado.

Paso 7 Retroalimentación para motor de motor de identificación de herramientas recomendables para estegoanálisis.
Se documenta el resultado del estegoanálisis con el fin de que pueda ser utilizado para mejorar el motor de identificación de herramientas recomendables para estegoanálisis.

Paso 8 Retroalimentación para mejorar el motor de estegoanálisis.
Se documenta el resultado del estegoanálisis con el fin de que pueda ser utilizado para mejorar el motor de estegoanálisis.

4.5 Listado herramientas *OSINT* y de esteganografía incluidas en Oculto

A continuación, en la Tabla 6 se muestra el listado de herramientas de estegoanálisis incluidas en Oculto:

Tarea	**Herramientas**
Análisis de metadata	*Exiftool* y *getefix* de la librería de imágenes *Pillow.*
Detección de archivos y código oculto en imágenes	*Binwalk*.
Análisis y detección de cadenas de caracteres en archivos	*Strings, enchant.*
Herramienta de análisis de imágenes	*Gifshuffle, pngcheck, sha256sum.*
Editor Hexadecimal para análisis forense.	*Hexdump.*
Estegoanálisis	*jsteg, zsteg, stegoveritas, stegano, stegseek, openstego.*
Reputación de imágenes	Cliente API de *VirusTotal.*
Análisis de NFTs	Cliente API *NFTport.*

Tabla 6 Herramienta de OSINT incluidas en Oculto.

4.6 Opciones de estegoanálisis

Oculto ofrece al analista la opción de realizar un análisis manual de la imagen o ejecutar un módulo que seleccione las herramientas a utilizar y que coordine su ejecución de forma automática. Esta flexibilidad permite al analista decidir como desea realizar el estegoanálisis con la biblioteca Oculto según sus necesidades.

4.6.1 Estegoanálisis manual

En la Figura 29 se aprecia el menú general de Oculto, el cual, en caso de que el analista lo requiera, tiene la posibilidad de ejecutar manualmente análisis preliminar de la imagen que le permite extraer los atributos, propiedades e incluso información disponible con fuentes *OSINT*.

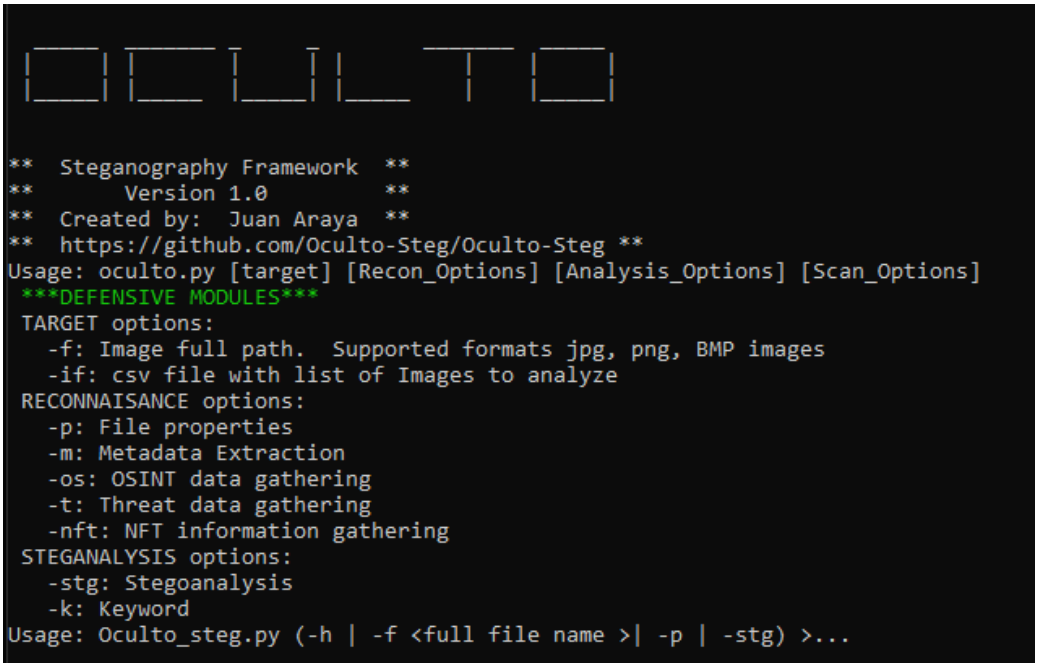


Figura 29 Menú de estegoanálisis de Oculto.

4.6.2 Estegoanálisis automático

Oculto aplicará el flujo de trabajo de la Figura 30 con el fin de facilitar el proceso de estegoanálisis:

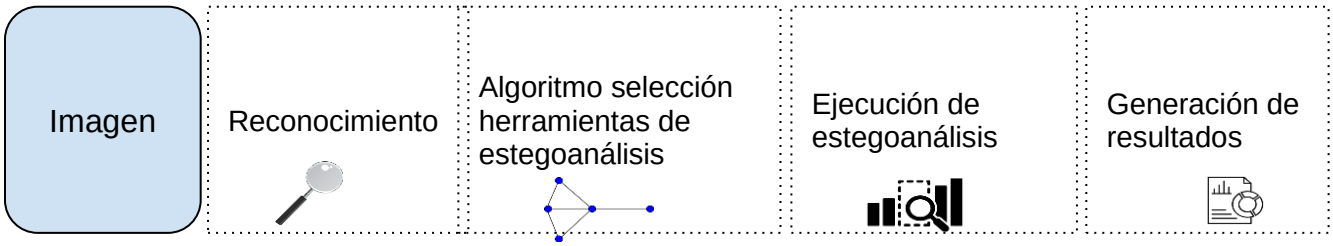


Figura 30 Flujo de trabajo de soporte para el estegoanálisis automático de Oculto.

4.6.2.1 Reconocimiento

En el proceso de reconocimiento, el objetivo es obtener información general de la imagen a analizar. En la Tabla 7 se muestren los datos que Oculto documenta de la imagen en el proceso de reconocimiento:

Dato	**Descripción**
Extensión	Definir el tipo de imagen, por ejemplo bmp, jpeg, png, gif.
Tamaño	*Bytes* del archivo.
Fecha de creación	Fecha de creación de la imagen.
Fecha de última modificación	Fecha de última modificación de la imagen.
Autor	Autor del archivo.
Reputación del archivo	Reputación de archivo en sistemas de *threat intelligence.*
Hash	*Hash* criptográfico.
Datos públicos sobre la imagen	A través del uso de herramientas *OSINT,* obtener datos sobre la imagen a analizar.
Extracción de metadata	*Exiftool.*
Análisis del archivo con editor Hexadecimal	*Hex dump.*

Tabla 7 Datos generales a obtener de una imagen durante el proceso de reconocimiento.

Con base a la información recopilada en el proceso de reconocimiento, Oculto inicia un proceso de selección de las herramientas de estegoanálisis que sean compatibles con el archivo en análisis.

4.6.2.2 Selección de herramientas de estegoanálisis

En la Figura 31 se refiere al árbol de decisión de Oculto con el fin de seleccionar las herramientas preliminares que facilite el proceso de estegoanálisis de un posible *stegware*.

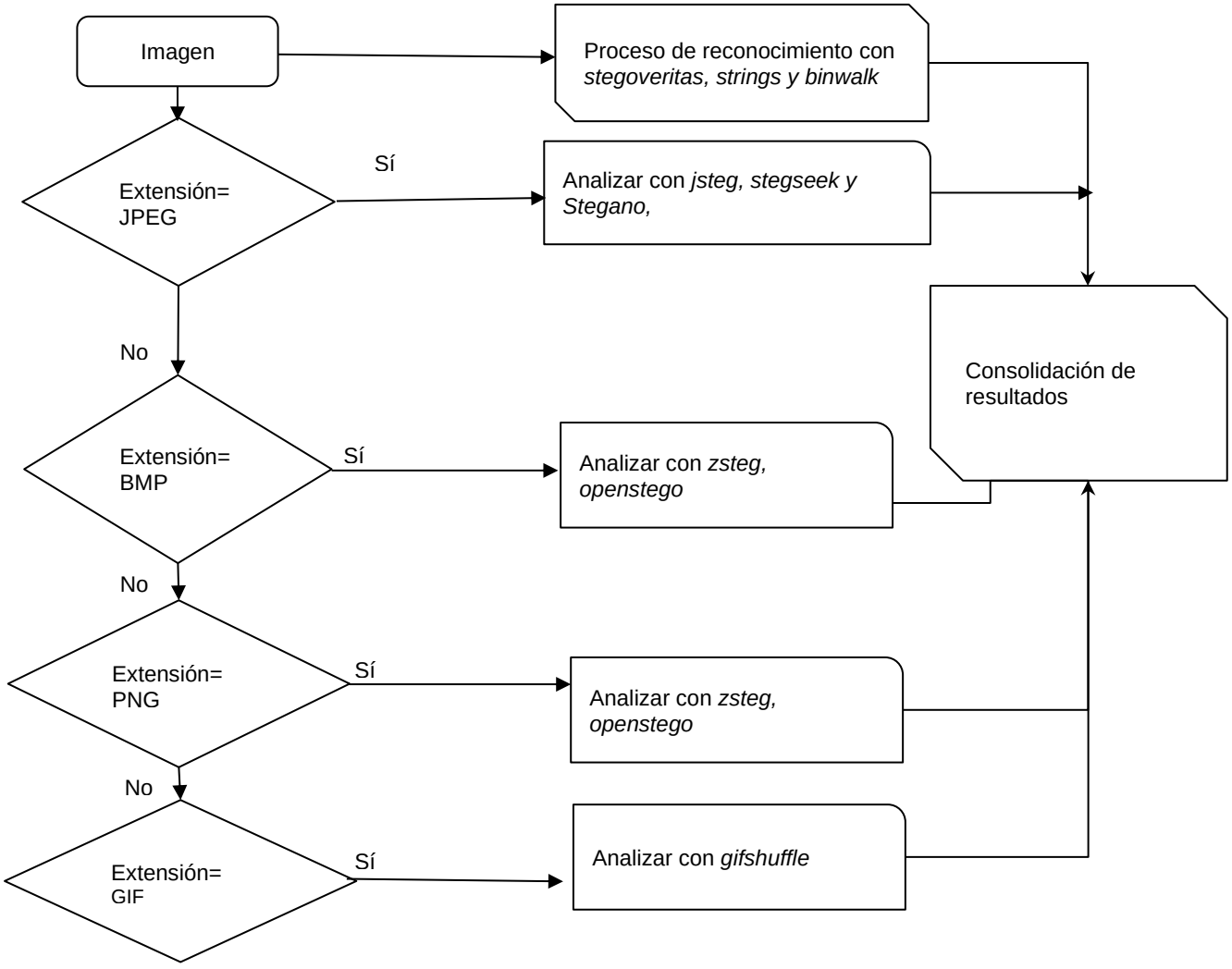


Figura 31 Árbol de decisión para selección de herramientas de estegoanálisis.

4.6.2.3 Ejecución del estegoanálisis

Con el fin de agilizar el proceso de estegoanálisis, Oculto tiene la capacidad de analizar la imagen utilizando unas herramientas que sean seleccionadas por el algoritmo de selección antes mencionado y de consolidar los resultados.

4.6.2.4 Generación de resultados

El resultado del proceso de reconocimiento, de las herramientas seleccionadas y de la ejecución de estegoanálisis se mostrará en pantalla al finalizar la ejecución.

5. Implementación de biblioteca Oculto

Oculto está compuesta por scripts hechos con Python que permiten integrar múltiples herramientas de esteganografía y estegoanálisis para facilitar la detección de uso de esteganografía en imágenes para esconder datos o *payloads*. Algunas de las herramientas integradas fueron diseñadas en otros lenguajes de programación como Ruby, Java y C. Se identificó que tipos de imágenes son capaces de analizar cada una de las herramientas integradas así sus respectivos resultados del estegoanálisis.

5.2 Desarrollo de Oculto

Oculto se desarrolló usando el lenguaje de programación Python3 en un proyecto de código de fuente abierto.

5.3 Organización de la Biblioteca Oculto

La Tabla 8 a continuación muestra una descripción general de los elementos de Oculto.

Elemento	**Descripción**
Oculto_Framework.py	Funciones para integrar múltiples herramientas de estegoanálisis, *OSINT* y de análisis de imágenes.
Oculto_Imports.py	Referencias requeridas para poder utilizar las herramientas disponibles en el Oculto_Framework.py.
Oculto_*OSINT*.py	Herramientas *OSINT* para análisis de hash de archivos así como para obtener información de imágenes de NFT (*Non fungible tokens*)
Oculto_Report.py	Generación de reporte consolidado del resultado de análisis del posible estego objeto.
Oculto_steg.py	Ejemplo de un cliente que hace uso de las herramientas disponibles en el Oculto_Framework.py.
README.md	Descripción de Oculto.
Config.ini	Contiene las credenciales requeridas para las llamadas de APIs.

Tabla 8 Elementos de Oculto.

5.4 Integración de herramientas

Las herramientas utilizadas por Oculto han sido integradas utilizando las siguientes opciones:

- Integración de herramientas a través de llamadas de API.
- Ejecución de herramientas vía comando, así como captura y análisis de salida.
- Integración de bibliotecas

5.4.1 Integración de herramientas vía llamadas de API

La Tabla 9 incluye información acerca de las herramientas integradas a Oculto a través de llamadas a los APIs:

Herramienta integrada por llamada de API	**Descripción**
VirusTotal	Análisis del hash de una imagen contra los motores de detección de *malware* disponibles en *VirusTotal*.
NFTport	Extracción de metadata de un NFT.

Tabla 9 Clientes de API integrados con Oculto

5.4.1.1 Integración de VirusTotal con Oculto

El script Oculto_*OSINT*.py contiene el código para realizas las siguientes acciones para aprovechar el API de VirusTotal:

- Cálculo del hash del posible estego-objeto
- Análisis del hash obtenido en VirusTotal

En la Figura 32 se aprecia parte de la función hash_value() programada en Oculto, la cual calcula de forma automática el hash de MD5, sha1 y sha256 del estego objeto y se guarda en un diccionario de Python

```
def hash_value(possible_stego_Object):
 hash_type= ["md5", "sha1", "sha256"]
 for hash_id in hash_type:
  global hash_values
  hash_values[hash_id] = hash_calculation(hash_id, possible_stego_Object)
 print(hash_values)
```

Figura 32 Función de cálculo múltiples valores de hash de imágenes.

La función hash_calculation() que aparece en la Figura 33 es llamada desde la función hash_value, y recibe como parámetro el *hash_id* (tipo de hash, por ejemplo SHA256) y un *possible_stego_Object* (la ruta completa y nombre de la imagen a analizar).

La función devuelve el valor del hash del estego objeto los cuales son guardados en un dictionario de python *hash_values= {}*.

```
def hash_calculation(hash_id, possible_stego_Object):
 hash_calculation = hash_id + "sum " + possible_stego_Object
 print(">>>Calculating " + hash_id + " hash of the image file")
 hash_value = subprocess.check_output(hash_calculation, shell=True)
 hash = hash_value.split()
 hash = str(hash[0])
 hash = hash.replace("'", "")
 hash = hash.replace("b", "")
 return hash
```

Figura 33 Función de cálculo de hash de imágenes.

El hash obtenido en la función hash_calculation es enviado a la función VirusTotal() la cual espera como parámetro el hash del estego objeto, el tipo de hash utilizado(por ejemplo SHA256) y la clave de *API* de VirusTotal, la cual está registrada en el archivo config.ini.

En la Figura 34 se observa como con los valores recibidos por parámetro se procede a hacer una llamada al API de VirusTotal a través del comando curl, en donde se le envía el hash de la imagen para que sea procesada por VirusTotal.

```
def VirusTotal(picture_hash, hash_type, vt_key):
 print(">>>Checking hash " + hash_type + " of the picture on
VirusTotal*********")
 steg_command = "curl --request GET --url
https://VirusTotal.com/api/v3/search?query=" + picture_hash + " --header \'x-
apikey: " + vt_key + "\'"
 print (steg_command)
 VirusTotal_output = subprocess.check_output(steg_command, shell=True)
 if ": []" in str(VirusTotal_output):
 print("Hash not found in VirusTotal")
 else:
 print(VirusTotal_output)
```

Figura 34 Función que utiliza el cliente del API de VirusTotal para verificar si tiene información de una imagen con base al hash.

El API endpoint de VirusTotal va a analizar el hash de la imagen, y en caso de que los servicios integrados de VirusTotal encuentren algún dato mostrará el resultado del análisis realizado.

Según la Figura 35, en caso de que el endpoint de VirusTotal no devuelva ningún resultado mostrará el mensaje “Hash not found in VirusTotal” es decir no se encontró el hash en el sistema de VirusTotal.

```
**********Information Gathering using OSINT**********
>>>Calculating md5 hash of the image file
>>>Calculating sha1 hash of the image file
>>>Calculating sha256 hash of the image file
{'md5': '658ed9a77d997c75da7033ec4859c4', 'sha1':
'7ecd83ee4af6cf85e7799fe65849a14a47', 'sha256':
'f0df15a872664a9e56a96dad714ecfe8526c0a67f2cd34d999632d8ee9c53'}
{'VT': 'deea7981acdff0bd73461de858efcdde18e39aca0156710ef7f2cf4474efe714', 'NFT':
nan}
>>>Checking_hash sha256 of the picture on VirusTotal**********
curl --request GET --url
https://virustotal.com/api/v3/search?query=f0df15a872664a9e56a96dad714ecfe8526c0a67f
2cd34d999632d8ee9c53 --header 'x-apikey:
deea7981acdff0bd73461de858efcdde18e39aca0156710ef7f2cf4474efe714'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   163  100   163    0     0    488      0 --:--:-- --:--:-- --:--:--   489
Hash not found in VirusTotal
```

Figura 35 Salida del endpoint de API de VirusTotal.

5.4.2 Ejecución de herramientas vía comando y análisis de salida

En la Tabla 10, aparecen las herramientas integradas a Oculto a través de la ejecución de los comandos requeridos y su respectivo análisis.

Herramienta integrada por ejecución en línea de comandos	**Comando**	**Descripción**
jsteg	`jsteg reveal " + picture + "| grep --color -i -E " + keyword`	Ejecución de *jsteg* para detectar uso de esteganografía y buscar una palabra clave en el resultado.
	`jsteg reveal " + picture`	Ejecución de *jsteg* para detectar uso de esteganografía.
exiftool	`exiftool " + picture + "| grep --color -i -E " + keyword`	Extracción del metadata de una imagen y búsqueda de palabra clave en el resultado.
	`exiftool " + picture`	Extracción del metadata de una imagen.
stegseek	*`stegseek`* `--seed -sf " + picture + " --continue"`	Análisis de fuerza bruta de presencia de uso de esteganografía. Se usa el conector –continue para que analice todo el archivo y no se detenga con sólo detectar el primer archivo oculto. Esto es útil dado que en un mismo estego objecto se pueden ocultar múltiples archivos.
Strings	`strings " + picture`	Detecta y extrae cadena de caracteres de la imagen analizada.
	`"strings " + image_file + " | grep -E \"([0-9]{1,3}[\.]){3}[0-9]{1,3}\""`	Detecta y extrae cadena de caracteres que coinciden con un regex (expresión regular) que detecta direcciones IP versión 4 en la imagen analizada.
	`command = "strings " + image_file + " | grep --color -i -E " + keyword + " -A 3 -B 3 "`	Extrae las cadenas de caracteres de una imagen, así como detecta y resalta palabra clave a buscar.
Stegoveritas	*`stegoveritas`* `" + picture + " -`*`steghide`*`"`	Detecta si se ha utilizado *steghide* para ocultar un archivo o datos en un estego-objeto.
hexdump	`"hexdump -C " + picture + " | tail -10 | grep --color -i -E \"([0-9]{1,3}[\.]){3}[0-9]{1,3}\""`	Muestra los valores en hexadecimal y en *ASCII* así como resaltar direcciones IP v4 que hayan sido ocultadas en el estego-objeto.

Binwalk	`"binwalk " + picture`	Detecta si hay algún archivo oculto dentro de un estego-objeto.
zsteg	`"zsteg -a " + picture + " | grep --color -E "+ keyword +" -B 100 -A 100"`	Busca datos ocultos en un estego objeto utilizando todos los métodos conocidos por la herramienta *zsteg* y busca una palabra clave a buscar.
pngcheck	`"pngcheck -v " + picture`	Analiza imágenes png y verifica si tiene dados ocultos al final del archivo.
gitshuffle	`"gifshuffle " + picture`	Verifica si hay datos ocultos dentro de una imagen tipo gif.

Tabla 10 Herramientas integradas con Oculto.

5.4.3 Integración de bibliotecas

La Tabla 11 contiene el listado de herramientas integradas con Oculto a través del uso de sus respectivas bibliotecas.

Herramienta	Integración	Descripción

Pillow	```def extract_file_properties(image_file):
	 extension = image_file.split(".")
	 image = Image.open(image_file)
	 image_file_properties = {
	 "Format": str(image.format),
	 "Mode" : str(image.format_description),
	 "Size": str(image.size),
	 "Mode":str(image.mode),
	 "Info": str(image.info),
	 "Im": str(image.im),
	 "Palette": str(image.palette),
	 "exif": str(image.getexif())
	 }```
	Figura 36 Uso de biblioteca para análisis de imágenes.
		La Figura 36 muestra el código de la función que permite la detección de atributos de una imagen.

stegano	```def stegano(possible_stegobject): print("*********Checking file using stegano********") clear_message = lsb.reveal(possible_stegobject) print (clear message)``` *Figura 37 Uso de biblioteca para estegoanálisis de imágenes.*	La Figura 37 contiene un extracto de la función *stegano*, la cual detecta y extrae un mensaje oculto en una imagen.

Tabla 11 Bibliotecas integradas con Oculto.

Además, en la Figura 38, se mencionan las siguientes bibliotecas que son requeridas por Oculto.

```
import enchant
import subprocess
import re
from PIL import Image
import os
import numpy
import sys
import calendar
import time
from subprocess import Popen, PIPE, STDOUT
from art import *
from bs4 import BeautifulSoup as bs
import os
from stegano import lsb
import clipboard
```

Figura 38 Listado de referencias incluidas en Oculto.

5.5 Repositorio de Oculto

El código fuente de Oculto se encuentra organizado en el repositorio privado disponible en https://github.com/Oculto-Steg/Oculto-Steg.git.

En la Figura 39 se describe la forma en que se debe ingresar el *API Key* de *VirusTotal* y de *NFTport* en el archivo config.ini

```
┌──(kali㉿LAPTOP)-
[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ cat config.ini
VT <Add your API key of VirusTotal>
NFT <Add your API key of NFT port>
```

Figura 39 Archivo config.ini con API Keys para cliente de APIs.

6. Evaluación de Oculto

Con el fin de medir la reducción de tiempo y la cantidad de mensajes ocultos con esteganografía descubiertos, se ejecutó el día 9 de abril una competencia tipo *Capture the Flag* (CTF) la cual fue anunciada como StegoCTF, donde se le presentaron retos de imágenes con datos o archivos empotrados con herramientas de esteganografía. Los participantes tuvieron que realizar el estegoanálisis correspondiente y obtener el valor secreto (flag) oculto.

A los participantes se les otorgó un tiempo máximo de 5 horas para resolver la mayor cantidad de retos y obtener el mayor puntaje posible.

El día 9 de abril del 2022 de 3:00Pm a 8:30pm hora local de España se ejecutó el CTF llamado StegoCTF. Los primeros 30 minutos se utilizaron para dar la bienvenida a los participantes, en inglés y en español, así como para brindar una introducción al CTF y aclarar dudas antes de iniciar la competencia.

El ganador del CTF tuvo que obtener el mayor puntaje posible, obtenido de forma automática cada vez que resolvía de forma correcta un reto, en el menor tiempo posible.

El CTF consistió en:

- 25 retos de esteganografía.
- Puntaje máximo de 5500.
- Retos de estegoanálisis de imágenes para detección de valor secreto(flag)
- Competencia individual.
- 5 horas máximo para completar la mayor cantidad de retos posibles y acumular la mayor cantidad de puntos en el menor tiempo posible.

El registro de participantes al CTF inició el 14 de febrero del 2022. Un total de 148 personas mostraron interés de participar en el CTF y llenaron el formulario de registro inicial.

El 22 de marzo 2022 se contactó a cada uno de los inscritos y se les solicitó la confirmación de su participación, además se les brindó los pasos requeridos para la creación de la cuenta en el servidor del CTF.

El día 1, 3 y 6 de abril se les enviaron recordatorios a los inscritos para que crearan su cuenta en el servidor del CTF a través del formulario.

De las 148 personas que se inscribieron en el formulario de registro, sólo 56 crearon su respectiva cuenta en el servidor del CTF.

Dos de esos usuarios tuvieron el papel de Administradores del CTF por lo que no compitieron (el director de la comunidad Dojo y mi persona).

Posteriormente procedió con la distribución de los participantes siguiendo el proceso descrito en la sección 6.1 de los usuarios registrados en el servidor CTF.

El día del evento 46 participantes se presentaron al evento y trataron de solucionar los retos

6.1 Grupo de control vs grupo *Experimental*

Un total de 56 participantes fueron divididos de forma aleatoria en 2 grupos: El grupo de control y el grupo *Experimental*.

Grupo de Control
Al grupo de control se le dio la oportunidad de utilizar cualquier herramienta en el proceso de estegoanálisis a excepción de Oculto, que es el nombre de una herramienta basada en la Biblioteca Oculto de estegoanálisis desarrollado en este proyecto de graduación.

Grupo *Experimental*
Los miembros de este grupo tuvieron la libertad de utilizar cualquier herramienta de estegoanálisis además de la biblioteca Oculto.

6.2 Resultados del CTF

A continuación, en la Tabla 12 aparece el resultado final al cumplirse las 5 horas del evento.

Lugar	Usuario	País	Equipo	Puntaje
1	RommGT	Ecuador	CONTROL	4300
2	hawkeye	India	CONTROL	4300
3	b1lly	Peru	CONTROL	3700
4	Viti.Beats	República Dominicana	*EXPERIMENTAL*	3450
5	ukuku	Peru	*EXPERIMENTAL*	3050
6	sh4d0wf4x	Argentina	CONTROL	2850
7	L4nc3lot	Venezuela	*EXPERIMENTAL*	1800
8	asd123	España	*EXPERIMENTAL*	1300
9	Bitbl4ck	Costa Rica	*EXPERIMENTAL*	1300
10	gudix	Panama	*EXPERIMENTAL*	1200
11	Z	Costa Rica	*EXPERIMENTAL*	1000
12	davinson	Venezuela	CONTROL	950
13	MR	Costa Rica	*EXPERIMENTAL*	800
14	alfa593	Ecuador	CONTROL	500
15	CV	Peru	CONTROL	500
16	Jorac	Costa Rica	*EXPERIMENTAL*	500
17	TuxDewan	Panama	CONTROL	400
18	VEVRO02	Panama	*EXPERIMENTAL*	400
19	lnahuel	Argentina	*EXPERIMENTAL*	300
20	konan004	República dominicana	*EXPERIMENTAL*	300
21	2ko	Costa Rica	CONTROL	200
22	dblake	Panama	*EXPERIMENTAL*	200
23	gerh4rdt	Costa Rica	CONTROL	100
24	sanstevenson	Panama	CONTROL	100
25	Goldie	Panama	*EXPERIMENTAL*	100

Tabla 12 Resultados de StegoCTF.

- De los 46 participantes del CTF, sólo 25 participantes lograron resolver al menos 1 reto en las 5 horas del evento. 14 de 25 participantes que lograron resolver al menos 1 reto, eran parte **del grupo *Experimental*.**
- 6 miembros del **grupo *Experimental*** lograron situarse en el TOP 10 del listado de participantes, logrando las siguientes posiciones: 4, 5, 7, 8, 9, 10.
- A pesar de ser la primera participación en un CTF, 4 de los usuarios (L4nc3lot, Bitbl4ck, asd123, gudix) del grupo *Experimental* lograron situarse en el TOP 10 del listado general.

6.3 Métricas obtenidas

Con base a las entradas capturadas en la bitácora */var/log/CTFd/logs/submissions.log* del servidor de CTF se obtuvieron las siguientes métricas

Como se aprecia en la Figura 40, sólo 10 miembros del grupo de control lograron resolver al menos 1 reto, mientras que 15 miembros del grupo *Experimental* lograron resolver al menos un reto durante el CTF.

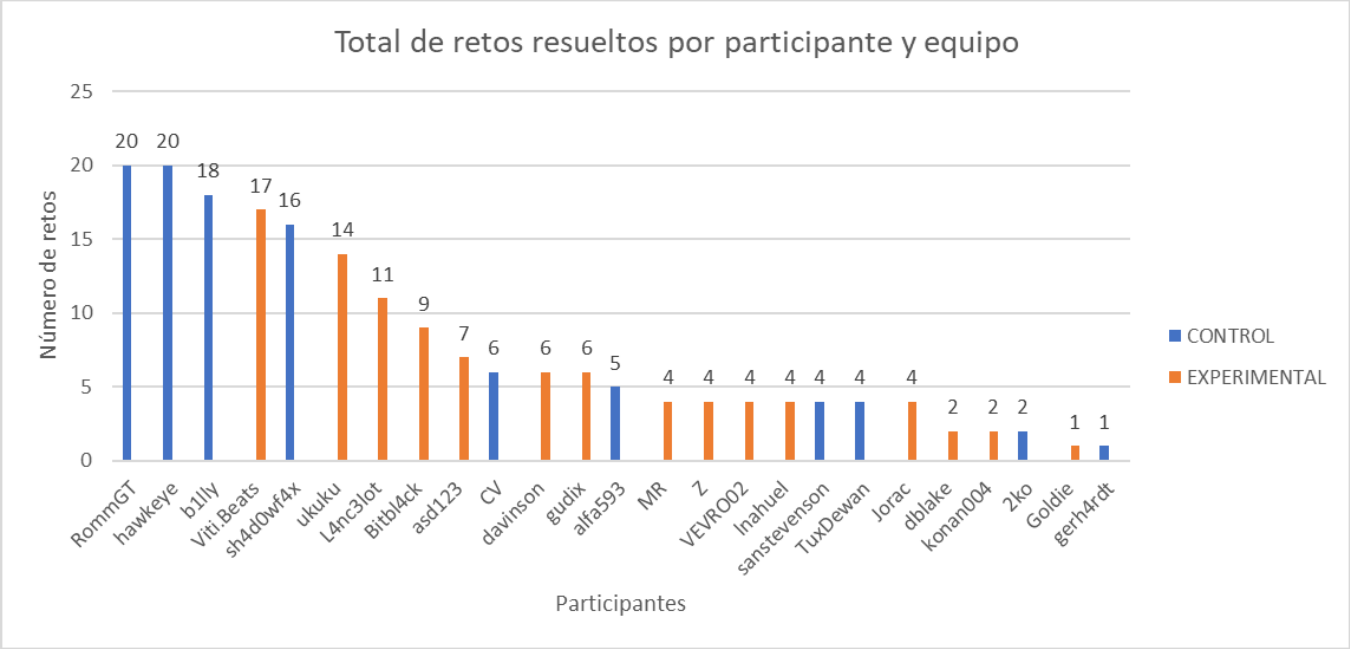


Figura 40 Total de retos resueltos por participante y equipo.

Como se muestra en la Figura 41, de los 25 retos del CTF, ninguno de los miembros del equipo de Control logró resolver 2 retos, mientras que 6 miembros del grupo *Experimental* sí lograron resolverlos. De igual forma se aprecia que un reto fue resuelto solamente por un miembro del grupo de control. También se observa que ninguno de los 2 equipos logró resolver 2 de los retos del CTF. Esto pudo ser causado por falta de tiempo, cansancio por las 5 horas de esfuerzo o falta de experiencia en estegoanálisis en imágenes.

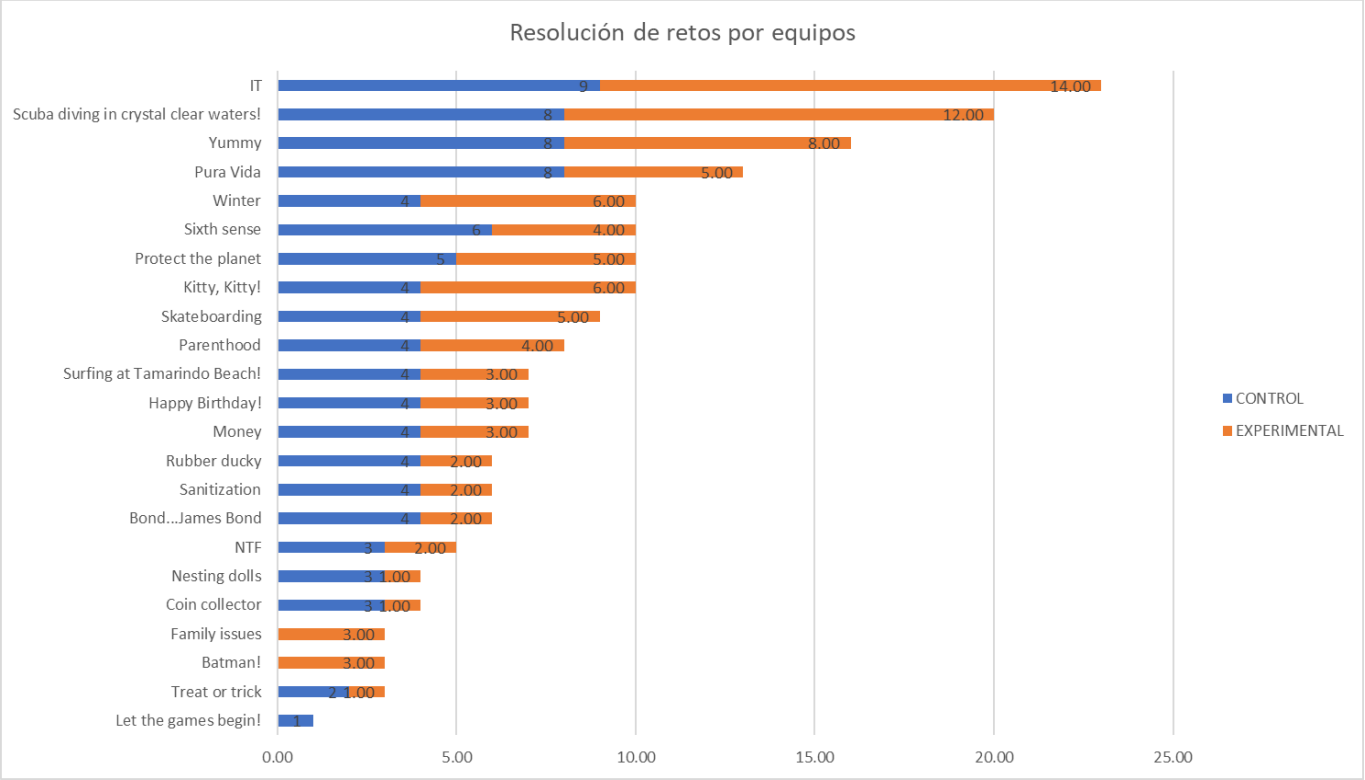


Figura 41 Resolución de retos por equipos.

En la Figura 42 se observa que el grupo *Experimental* logró resolver más retos de puntaje 100, 200 y 400 en comparación con los retos resueltos por el grupo de control. El puntaje está asociado con el nivel de dificultad de los retos. Sin embargo, los miembros del grupo de control lograron resolver más retos con puntaje individual de 300 puntos.

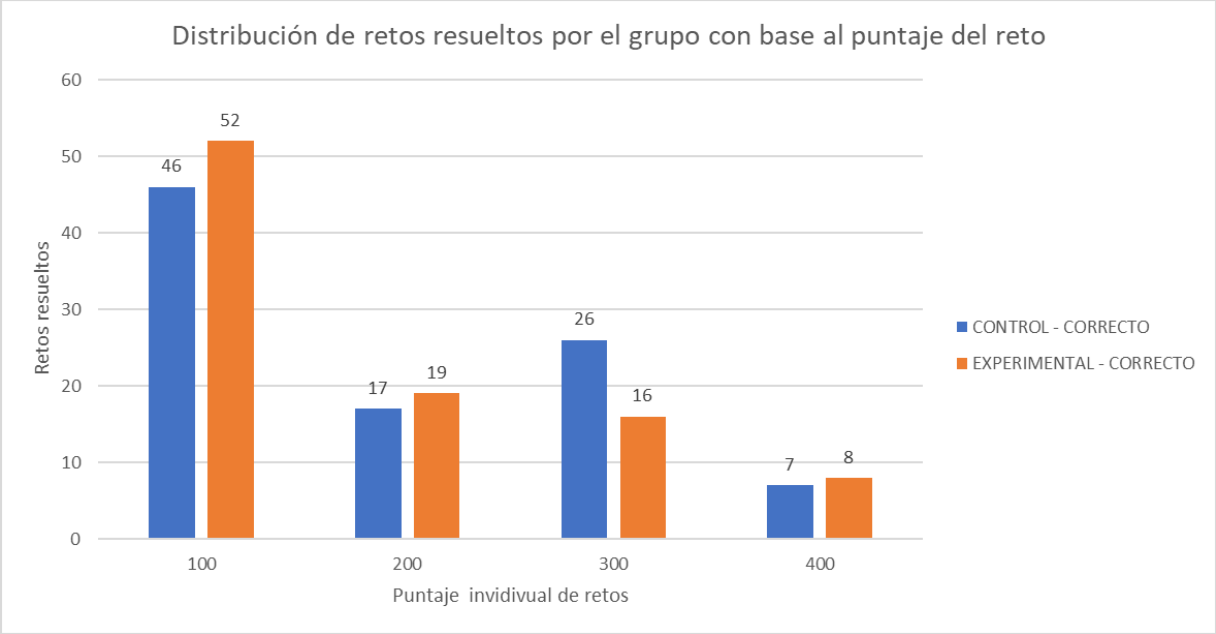


Figura 42 Distribución de retos resueltos por el grupo de control por puntaje del reto.

La Figura 43 muestra la hora en que la que miembros de los equipos de control y *Experimental* lograron resolver el primer y el último reto.

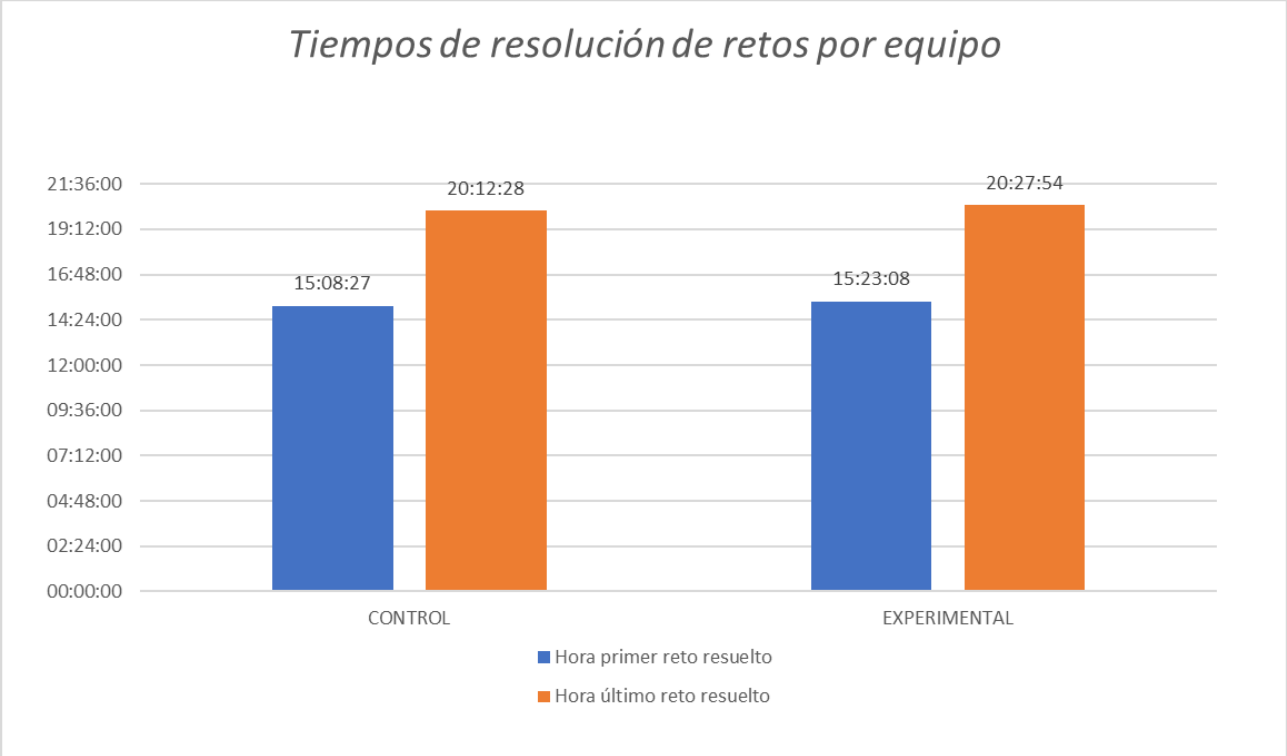


Figura 43 Tiempos de resolución de retos por equipo, hora local España 9 de abril 2022.

Además, en la Figura 278 del apéndice 9.12 se observan los tiempos requeridos por participante del grupo de control para solucionar los retos. Sobresale el corto tiempo de apenas 2 horas 52 minutos requerido por el usuario RommGT para resolver 20 de los 25 retos. También sobresalen el esfuerzo realizado por los usuarios hawkeye, quien a pesar de haber empatado con RommGT en puntaje, requirió 4 horas y 3 minutos para resolver 20 retos. El tercer lugar del grupo de control, lo obtuvo el usuario b1lly quien requirió 4 horas y 37 minutos para resolver 18 retos.

En la Figura 279 del apéndice 9.13, sobresalen el rendimiento del usuario Viti.Beats, quien en 4 horas y 9 minutos logró resolver 17 de los 25 retos. Logró resolver cada uno de los 17 retos. El usuario Ukuku logró resolver 14 retos en 4 horas y 2 minutos. El usuario L4nch3lot resolvió 11 retos en 3 horas y 46 minutos.

En la Tabla 13, se realiza una comparación general de los resultados obtenidos por el equipo de control y el equipo *Experimental* durante el CTF.

Métrica	**Grupo *Experimental***	**Grupo de control**
Total de miembros que resolvieron al menos 1 reto	15	10
Promedio de retos resueltos por miembros	6.3	9.7
Total de retos resueltos	22	21

Tabla 13 Métricas Generales obtenidas.

6.4 Retroalimentación obtenida sobre Oculto

Además, se realizó una encuesta a los participantes para obtener retroalimentación sobre el StegoCTF, así como el uso de Oculto. La Tabla 14 refleja la retroalimentación sobre Oculto brindada por los miembros del grupo *Experimental*:

- La interfaz de usuario de oculto es fácil de utilizar
- Tuvieron un ahorro de tiempo por usar Oculto durante el análisis de las imágenes.
- La documentación compartida (Guía de uso de Oculto y la Guía de instalación) fueron de utilidad.
- Algunos tuvieron errores durante la instalación o uso de Oculto, dado que Oculto fue diseñado y probado para ser compatible con Kali Linux 2022.1 y algunos usuarios utilizaron otras distribuciones de Linux como Ubuntu, así como quienes tenían una instancia de Linux en WSL (*Windows subsystem for Linux*).
- Recomendarían el uso de Oculto como apoyo en el proceso de estegoanálisis.
- Con el fin de facilitar la instalación de los componentes y dependencias de la biblioteca Oculto, los participantes sugieren la creación de instalador.

Fortalezas	Áreas de mejora
• Lo fácil de utilizar e interactuar con la herramienta. Beneficios de combinar varias herramientas de esteganografía en uno solo. • Fácil instalación. • Filtros incluidos en Oculto.	○ La instalación tiene muchas dependencias con respecto a otras herramientas, lo ideal sería incluir todo en el package de instalación para evitar algunos pasos. ○ Adicionar más opciones y herramientas.

Tabla 14 Fortalezas y áreas de mejoras de Oculto.

Funciones adicionales sugeridas por los miembros del grupo *Experimental*

- Uso de wordlist (lista con posibles valores) desde la biblioteca Oculto para el estegoanálisis con técnicas de fuerza bruta.
- Extracción de datos/archivos ocultos desde Oculto.
- Una versión con Interfaz Gráfica (GUI) de Oculto.
- Codificación y descodificación de valores desde Oculto, por ejemplo, de Base64 a ASCII.

7. Conclusión

Los objetivos propuestos en el proyecto se cumplieron de forma exitosa.

Se logró informar a la comunidad de profesionales en ciberseguridad sobre los riesgos del uso malicioso de la esteganografía mediante la creación de un sitio web y material educativo. Para ello se impartió 2 conferencias transmitidas en vivo por Internet las cuales se encuentran disponibles en el sitio del repositorio del proyecto. Se preparó material de entrenamiento sobre esteganografía, el proceso de estegoanálisis y el riesgo del uso malicioso de la esteganografía en imágenes.

Se desarrolló la biblioteca Oculto para apoyo del proceso de estegoanálisis e identificar el uso de esteganografía para ocultar datos o archivos en imágenes. Dicha biblioteca de código fuente abierto fue escrita usando el lenguaje de programación Python e integra múltiples herramientas de estegoanálisis. Oculto está basado en un algoritmo diseñado para analizar la imagen, obtener información general de la misma, incluso con herramientas *OSINT*, extraer el metadata, seleccionar las herramientas a utilizar y consolidar los resultados.

Se coordinó una competencia tipo *Capture de Flag* (CTF) con el fin de evaluar la capacidad de detección de datos o archivos del sistema Oculto de detección de datos ocultos en imágenes con técnicas de esteganografía basadas en el algoritmo de *Least Significant Bit* (LSB). Se evaluó Oculto a través de la ejecución de un CTF de 5 horas, donde se tuvo participación de personas de más de 9 países diferentes a tiempo real. De los 25 participantes que lograron resolver al menos un reto, 15 de ellos utilizaron Oculto lo que demuestra su potencial para el apoyo en la detección de uso de esteganografía en imágenes.

Oculto tiene potencial de integración con más herramientas de estegoanálisis a través del procesamiento de la salida de la ejecución de comandos, llamadas de API o de funciones. Oculto se diseñó para el análisis de uso de LSB en imágenes, sin embargo, tiene el potencial de ser extendido para el análisis de audio, video y otros archivos digitales, así como la integración de más técnicas de esteganografía. Oculto podría ser incluso parte de un servicio web que puede integrarse con herramientas de seguridad de protección contra fuga de datos.

Como trabajo futuro se consideran las siguientes mejoras a aplicar en la biblioteca Oculto:

- Extracción automática de datos ocultos.
- Generación automática de *hexdumps* y salidas de las herramientas integradas con Oculto.
- Integración con herramientas de estegoanálisis de archivos de audio y video.
- Creación de servicios web y APIs para consumir los servicios de Oculto.
- *Sandboxing* de imágenes para un análisis dinámico de imágenes para la detección de código ejecutable (*polyglots*).
- Aplicación móvil para detectar imágenes con esteganografía recibida a través de servicios de mensajería o redes sociales.

- Integración con sistemas de Prevención de Fuga de Información (DLP).
- Creación de interfaz gráfica.
- Programación Multihilos de los procesos.
- Generación de salida de datos en múltiples formatos (json, html) para su análisis.
- Integración de más funciones de estegoanálisis de fuerza bruta.

8. Referencias

[1] O. Evsutin, A. Melman y R. Meshcheryakov, «“Digital Steganography and Watermarking for Digital Images: A Review of Current Research Directions,”,» IEEE Access vol. 8, pp. 166589–166611, doi: 10.1109/access.2020.3022779., 2020. [En línea]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9187785.

[2] «Information leakage ENISA Threat Landscape,» [En línea]. Available: https://www.enisa.europa.eu/publications/information-leakage/at_download/fullReport.

[3] «PII - Glossary | CSRC,” csrc.nist.gov.,» [En línea]. Available: https://csrc.nist.gov/glossary/term/PII.

[4] «Steganography in attacks on industrial enterprises (updated) | Kaspersky ICS CERT,» 17 Junio 2020. [En línea]. Available: https://ics-cert.kaspersky.com/reports/2020/06/17/steganography-in-attacks-on-industrial-enterprises/. [Último acceso: 23 Abril 2022].

[5] J. Kennelly, K. Goody y J. Shilko, «Navigating the MAZE: Tactics, Techniques and Procedures Associated With MAZE Ransomware Incidents,» Mandiant.com, 7 Mayo 2020. [En línea]. Available: https://www.mandiant.com/resources/tactics-techniques-procedures-associated-with-maze-ransomware-incidents. [Último acceso: 1 Marzo 2022].

[6] C. Bing, «Russia-based ransomware group Conti issues warning to Kremlin foes,» Reuters.com, 25 Febrero 2022. [En línea]. Available: https://www.reuters.com/technology/russia-based-ransomware-group-conti-issues-warning-kremlin-foes-2022-02-25/. [Último acceso: 2 Marzo 2022].

[7] Sophos, «Maze ransomware: extorting victims for 1 year and counting,”,» Sophos News, 12 Mayo 2020. [En línea]. Available: https://news.sophos.com/en-us/2020/05/12/maze-ransomware-1-year-counting/.

[8] «“What Is Account Takeover (ATO)?,» Fortinet, [En línea]. Available: https://www.fortinet.com/resources/cyberglossary/account-takeover. [Último acceso: 22 Abril 2022].

[9] «Steganography in contemporary cyberattacks,» securelist.com, [En línea]. Available: https://securelist.com/steganography-in-contemporary-cyberattacks/79276/ .

[10] S. Parveen K, R. V. Ravi, B. Abd El-Rahiem y M. M. Ghonge, «Image Steganography,” Multidisciplinary Approach to Modern Digital Steganography,» nº 10.4018/978-1-7998-7160-6.ch002, p. 29–49, 2021.

[11] B. Lakshmi Sirisha y B. Chandra Mohan, «Review on spatial domain image steganography techniques,» *Journal of Discrete Mathematical Sciences and Cryptography,* vol. 24, nº 10.1080/09720529.2021.1962025, p. 1873–1883, 2021.

[12] H. Bhattacharjee y S. K. Bandyopadhyay, «Frequency Domain Approach of Image Steganography,,» *International Journal of Innovative Research in Information Security (IJIRIS),* vol. 3, nº 10.6084/M9.FIGSHARE.3484931, 2016.

[13] S. Hetzl, «*Steghide*,» 9 Octubre 2003. [En línea]. Available: http://*steghide*.sourceforge.net/. [Último acceso: 23 Abril 2022].

[14] S. Vaidya, «OpenStego,» [En línea]. Available: https://www.openstego.com/.

[15] I. A. Yari y S. Zargari,, «An Overview and Computer Forensic Challenges in Image Steganography,» *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData),* nº 10.1109/ithings-greencom-cpscom-smartdata.2017.60, 2017.

[16] mitre.org, «Data Obfuscation: Steganography,» attack.mitre.org, 15 Marzo 2020. [En línea]. Available: https://attack.mitre.org/techniques/T1001/002/. [Último acceso: 3 Marzo 2022].

[17] «Obfuscated Files or Information: Steganography, Sub-technique T1027.003 - Enterprise | MITRE ATT&CK®,» attack.mitre.org, [En línea]. Available: https://attack.mitre.org/techniques/T1027/003/. [Último acceso: 22 Abril 2022].

[18] J. R. Jayapandiyan, C. Kavitha y K. Sakthivel, «Enhanced Least Significant Bit Replacement Algorithm in Spatial Domain of Steganography Using Character Sequence Optimization,» *IEEE Access,* vol. 8, nº 10.1109/ACCESS.2020.3009234, p. 136537–136545, 2022.

[19] «Definition of STEGANOGRAPHY,» Merriam-webster.com, 2018. [En línea]. Available: https://www.merriam-webster.com/dictionary/steganography. [Último acceso: 1 Octubre 2021].

[20] T. Fernandez y E. Tamaro, «Biografia de Histieo,» En Biografías y Vidas. La enciclopedia biográfica en línea, 2004. [En línea]. Available: https://www.biografiasyvidas.com/biografia/h/histieo.htm. [Último acceso: 22 Abril 2022].

[21] A. Muñoz, «Canales subliminales. Esteganografía,» 2 Febrero 2014. [En línea]. Available: http://www.criptored.upm.es/crypt4you/temas/privacidad-proteccion/leccion7/leccion7.html#apartado2. [Último acceso: 22 Abril 2022].

[22] N. Strochlic, «The letter that won the American Revolution,» 3 Julio 2017. [En línea]. Available: https://www.nationalgeographic.com/history/article/george-washington-spy-letter. [Último acceso: 22 Abril 2022].

[23] G. Schott, Schola steganographica, Herbipoli Jobus Hertz, 1680.

[24] R. L. Marshall y W. Emery, «Johann Sebastian Bach | Biography, Music, & Facts,» Encyclopædia Britannica, 22 Enero 2019. [En línea]. Available: https://www.britannica.com/biography/Johann-Sebastian-Bach. [Último acceso: 22 Abril 2022].

[25] D. Mowry, «Cryptologic Aspects of German Intelligence Activities in South America during World War II,» Center for Cryptologic History National Security Agency, 2011. [En línea]. Available: https://www.nsa.gov/portals/75/documents/about/cryptologic-heritage/historical-figures-publications/publications/wwii/cryptologic_aspects_of_gi.pdf.

[26] U.S. Department of Justice, «New York Man Charged With Theft of Trade Secrets,» 1 Agosto 2018. [En línea]. Available: https://www.justice.gov/opa/pr/new-york-man-charged-theft-trade-secrets.

[27] D. Frank, «Cybereason Exposes Campaign Targeting US Taxpayers with NetWire and Remcos Malware,» www.cybereason.com, 18 Marzo 2021. [En línea]. Available: https://www.cybereason.com/blog/cybereason-exposes-malware-targeting-us-taxpayers.

[28] J. Jimenez, «Qué es el Stegware y cómo podemos protegernos de esta amenaza,» RedesZone,, 2 Junio 2019. [En línea]. Available: https://www.redeszone.net/2019/06/02/stegware-amenaza-protegernos-seguridad/.

[29] WhatsApp.com, «WhatsApp,» 2018. [En línea]. Available: https://www.whatsapp.com/.

[30] K. Thangadurai y G. Sudha Devi, «An analysis of LSB based image steganography techniques,» *International Conference on Computer Communication and Informatics,* nº 10.1109/iccci.2014.6921751, 2014.

[31] S. Izhar, «Hide Secret Message Inside an Image Using LSB-Steganography,» Cybrary, 7 Marzo 2018. [En línea]. Available: https://www.cybrary.it/blog/0p3n/hide-secret-message-inside-image-using-lsb-steganography/.

[32] I. S. Reddy, M. Purushotham Reddy y K. Subba Reddy, «Different Medias of Steganography- An Emerging Field of Network Security,» *(IJCSIT) International Journal of Computer Science and Information Technologies,* vol. 3, nº ISSN:0975-9646, pp. 3517-3522, 2012.

[33] A. Permana y S. Yaddarabullah, «Modification Of Least Significant Bit Method With Redundant Pattern Encoding For Protection Of Message Integration From Image Modification,» *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH,* vol. 9, nº ISSN 2277-8616, 2020.

[34] N. F. Johnson y S. Jajodia, «Exploring steganography: Seeing the unseen,» *Computer,* vol. 31, nº 10.1109/MC.1998.4655281., pp. 26-34, 1998.

[35] A. Iman, «Image Hiding Using Discrete,» J. of College of Education For Women, 2016.

[36] A. Muñoz, Privacidad y Ocultación de Información Digital Esteganografía, Paracuellos De Jarama: Ra-Ma, 2016.

[37] A. D. Ker, «A Weighted Stego Image Detector for Sequential LSB Replacement,» de *Third International Symposium on Information Assurance and Security*, 2007.

[38] M. Hussain, W. A. Wahab, Y. I. Idris, A. T. Ho y K. Jung, «Image steganography in spatial domain: A survey,» *Signal Processing: Image Communication,* vol. 65, nº 10.1016/j.image.2018.03.012, pp. 46-66, 2018.

[39] H. Caballero-Hernández, V. Muñoz-Jiménez, A. Morales-Reyes y M. Romero-Huertas, «A Review of Steganography Techniques for Digital Information Transmission for Secure Channels with Digital Images,» *IEEE LATIN AMERICA TRANSACTIONS,* vol. 17, 2019.

[40] M. Herrero, «La importancia de lo aleatorio,» INCIBE, 25 11 2014. [En línea]. Available: https://www.incibe-cert.es/blog/aleatorio.

[41] «Image Steganography using Stegosuite in Linux,» www.geeksforgeeks.org, 28 Setiembre 2021. [En línea]. Available: https://www.geeksforgeeks.org/image-steganography-using-stegosuite-in-linux/.

[42] «SSuite Picsel Steganography Encryption,» SSuite Office Software, [En línea]. Available: https://www.ssuitesoft.com/ssuitepicselsecurity.htm.

[43] V. L. Reddy, A. Subramanyam y P. C. Reddy, «SteganPEG Steganography + JPEG,» *International Conference on Ubiquitous Computing and Multimedia Applications,* nº 10.1109/UCMA.2011.17., 2011.

[44] «SteganPEG 1.0,» softpedia, [En línea]. Available: https://www.softpedia.com/get/Security/Encrypting/SteganPEG.shtml.

[45] «JPEG DCT Compression Encoding, Progressive,» www.loc.gov, 26 Enero 2012. [En línea]. Available: https://www.loc.gov/preservation/digital/formats/fdd/fdd000333.shtml.

[46] M. Aggarwal, I. Mehndiratta y D. Chahal, «Steganography for Communication of Secure Data,» vol. 6, nº IJARIIE-ISSN(O)-2395-4396, 2020.

[47] «cat(1) - Linux manual page,» man7.org, [En línea]. Available: https://man7.org/linux/man-pages/man1/cat.1.html.

[48] J. Gerend, «copy,» docs.microsoft.com, 3 Marzo 2021. [En línea]. Available: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/copy.

[49] P. A. Watters, S. H. Stripf y F. Martin, «Visual Steganalysis of LSB-Encoded Natural Images,» de *Third International Conference on Information Technology and Applications (ICITA'05)*, 2005.

[50] K. Jung, «Comparative Histogram Analysis of LSB-based Image Steganography,» *WSEAS Transactions on Systems and Control,* vol. 13, nº 2224-2856, 2018.

[51] «Hiding secret messages in images with steganography and metadata,» Drchaos, 9 Noviembre 2018. [En línea]. Available: https://www.drchaos.com/post/hiding-secret-messages-in-images-with-steganography-and-metadata.

[52] Z. Zaikin, «*zsteg*,» GitHub, [En línea]. Available: https://github.com/zed-0xff/*zsteg*. [Último acceso: 23 Abril 2022].

[53] «stegsolve,» GitHub, [En línea]. Available: https://github.com/zardus/ctf-tools/blob/master/stegsolve/install. [Último acceso: 22 Abril 2022].

[54] «StegSecret. A simple steganalysis tool,» stegsecret.sourceforge.net, [En línea]. Available: http://stegsecret.sourceforge.net/.

[55] «SpyHunter,» www.spy-hunter.com, [En línea]. Available: http://www.spy-hunter.com/stegspydownload.htm.

[56] C. Bonhomme, «stegano: A pure Python Steganography module,» PyPI, [En línea]. Available: https://pypi.org/project/stegano/.

[57] «hidebehind,» GitHub, [En línea]. Available: https://github.com/multifrench/hidebehind.

[58] «gpg(1): OpenPGP encryption/signing tool - Linux man page,» linux.die.net, [En línea]. Available: https://linux.die.net/man/1/gpg. [Último acceso: 23 Abril 2022].

[59] «StegExpose A steganalysis tool for detecting LSB steganography in images,» Github, [En línea]. Available: https://github.com/b3dk7/StegExpose. [Último acceso: 22 Abril 2022].

[60] www.dmg.tuwien.ac.at, «Stefan Hetzl - Research,» www.dmg.tuwien.ac.at, [En línea]. Available: https://www.dmg.tuwien.ac.at/hetzl/research/index.html. [Último acceso: 22 Abril 2022].

[61] «*binwalk*,» Gitub, 4 Enero 2021. [En línea]. Available: https://github.com/ReFirmLabs/*binwalk*.

[62] «VirusTotal API v3 Overview,,» VirusTotal, [En línea]. Available: https://developers.virustotal.com/reference/overview.

[63] «Pillow: Python Imaging Library (Fork),» PyPI, [En línea]. Available: https://pypi.org/project/Pillow/.

9. Apéndices

9.1 Integración de NFTPort API con Oculto

Con el fin de apoyar en el análisis de imágenes NFT, se diseñó la siguiente función que realiza llamada al API de NFTport, la cual espera como parámetro, el contract ID y el Token id del NFT que debe ser analizado. La Figura 44 muestra un extracto del código fuente de la función que permite el análisis de imágenes ofrecidas como NFTs.

```
def NTF_analysis():
 print("Token ID:")
 Token_id= input()
 print("Contract_ID:")
 contract_ID= input()
 read_config_file()
 NFTPort_API= API_KEYS["NFT"]
 print("**********Analyzing NFT**********")
 command= "curl --request GET \
 --url 'https://api.nftport.xyz/v0/nfts/" + contract_ID +"/"+ Token_id +
"?chain=ethereum&refresh_metadata=true' \
 --header 'Authorization: " + NFTPort_API +"\'"" \
--header 'Content-Type: application/json'"
 metadata = subprocess.check_output(command, shell=True)
 #metadata = os.system(command)
 json_response=str(metadata.decode("utf-8"))
 #print(str(json_response))
 y = json.loads(str(json_response))
 print("NFT details")
 if str(y["response"])== "OK":
 print("Response:" + str(y["response"]))
 print("Chain:" + str(y["nft"]["chain"]))
 print("Metadata_url:" + str(y["nft"]["metadata_url"]))
 print("Mind date:" + str(y["nft"]["mint_date"]))
 print("Updated date:" + str(y["nft"]["updated_date"]))
 extract_NFT_metadata(str(y["nft"]["metadata_url"]))
```

Figura 44 Cliente del API de NFTport para obtener información y el metadata de un NFT.

Dichos valores son ingresados manualmente por el usuario durante la ejecución del script Oculto_Steg.py tal y como se observa en la Figura 45.

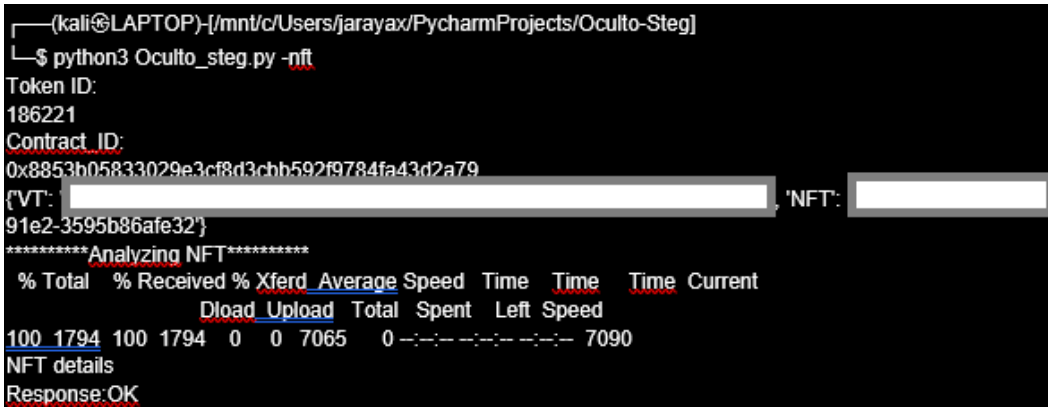


Figura 45 Salida del endpoint de API de NFTport.

Según la Figura 46, en caso de que el endpoint encuentre el NFT, el script procederá a extraer el URL de la imagen.

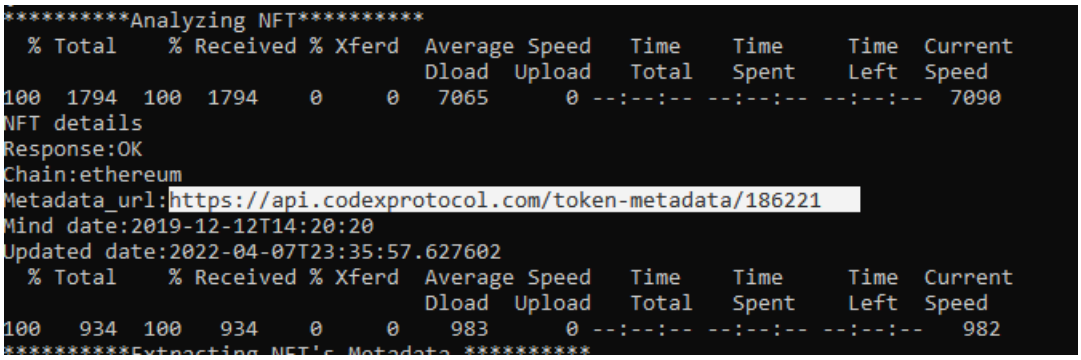


Figura 46 URL de un NFT obtenido con el endpoint de API de NFTport.

En la Figura 47 se observa que luego se lo envía por parámetro a la función extract_metadata para extraer el metadata disponible del NFT.

```
def extract_NFT_metadata(metadata_url):
 command = "curl --request GET --url " + metadata_url + ""
 NFT_metadata = subprocess.check_output(command, shell=True)
 json_response=str(NFT_metadata.decode("utf-8"))
 y = json.loads(str(json_response))
 print("**********Extracting NFT's Metadata **********")
 print("NFT Name:" + str(y["name"]))
 print("NFT Description:" + str(y["description"]))
 print("NFT image:" + str(y["image"]))
 print("NFT external URL:" + str(y["external_url"]))
 print("NFT image URL:" + str(y["image_url"]))
 print("NFT home URL:" + str(y["home_url"]))
```

Figura 47 Función para la extracción del metadata de un NFT.

Y así mostrar, según la Figura 48, el metadata del NFT incluyendo el URL desde donde se puede descargar una copia de la imagen para que el analista utilice el Oculto para realizarle el respectivo estegoanálisis

**********Extracting NFT's Metadata **********
NFT Name:Fourteen Atlas N Scale train cars in original boxes
NFT Description:Group consists of 2 3963 Wisconsin Central, 1021 60' auto parts car, 3906 Seaboard System 2 bay Centerflow, 30700 undecorated 23,500 gal tank car, 3957 ACF 4 bay centerflow Chessie System, 37071 Shell 33k gal tank car, 37072 Shell 33k gal tank car, 37073 Shell 33k gal tank car, 2348 Warren Jumbo tank, 3812 Wabash flat car, 3958 AE Staley 4 bay centerflow, 37101 Cal Gas 33k gal tank car, 3114 GATX 94' tank car. All are unrun and C8-9 in excellent to near mint original boxes.
NFT image:https://s3-us-west-2.amazonaws.com/codex.registry-production/files/1576160357488.178e07c8-b5f7-41fe-b9ef-de510968c06a.jpg
NFT external URL:https://codex-viewer.com/record/186221
NFT image URL:https://s3-us-west-2.amazonaws.com/codex.registry-production/files/1576160357488.178e07c8-b5f7-41fe-b9ef-de510968c06a.jpg
NFT home URL:https://codex-viewer.com/record/186221

Figura 48 URL de descarga del NFT para su respectivo estegoanálisis.

9.2 Instalación de Oculto

Para hacer uso de las herramientas integradas en Oculto se deben cumplir con los siguientes requisitos en un equipo corriendo Kali Linux 2022.1 o superior, ya sea virtualizado, nativo o ejecutado desde WSL(Windows Subsystem for Linux) según la Tabla 15 que aparece a continuación.

Comando	**Objetivo**
`sudo apt update`	Actualización de lista de paquetes disponibles
`sudo apt upgrade -y`	Actualización de paquetes disponibles
`sudo apt install python3-pip`	*Instalación de pip con el fin de instalar algunas bibliotecas utilizadas por Oculto como lo son art(para el ASCII art del logo de Oculto), pillow(para el análisis de imagenes), stegoveritas(para el estegoanálisis) entre otras*
`sudo apt install golang-go`	Requerido para poder utilizar la herramienta de estegoanálisis *jsteg*, la cual fue programada con el lenguaje GO.
`sudo wget -O /usr/bin/jsteg https://github.com/lukechampine/jsteg/releases/download/v0.1.0/jsteg-linux-amd64`	Descarga de la versión más reciente de *jsteg*
`sudo chmod +x /usr/bin/jsteg`	Permite dar derechos de ejecución del archivo *jsteg*
`pip install clipboard`	Lectura de valores que hayan sido copiados y almacenados temporalmente en memoria RAM. Muy útil

	para no tener que digitar los valores del *Contract ID* y del *token ID* en el momento de analizar un NFT.
`sudo apt install libimage-exiftool-perl`	Exiftool es una herramienta hecha con Perl que permite extraer el metadata de archivos
`git clone https://github.com/bannsec/stegoveritas.git`	Descarga de la última versión disponible de *stegoveritas*
`pip3 install stegoveritas`	Permite instalar la versión base de *stegoveritas*
`cd stegoveritas` `sudo python3 setup.py install`	Instala la version más reciente de *stegoveritas*
`stegoveritas_install_deps`	Instala todos los dependencias de *stegoveritas* requeridas para hacer uso sin errores de *stegoveritas*.
`sudo apt install binwalk`	Instala *binwalk* para detección y extracción de archivos ocultos
`sudo apt install steghide`	Instalación de *steghide*
`sudo apt-get install rubygems`	Instalación de Ruby. Esto dado que *zsteg* fue programada con ruby.
`sudo gem install zsteg`	Instalación de *zsteg*
`sudo apt-get install -y pngcheck`	Instalación de *pngcheck* utilizada para detectar datos ocultos al final de un archivo png
`sudo apt-get install -y gifshuffle`	Instalación de *gifshuffle* para detectar datos ocultos en imágenes gif
`sudo pip install stegano`	Instalación de stegano

`wget https://github.com/RickdeJager/stegseek/releases/download/v0.6/stegseek_0.6-1.deb`
 	
`sudo apt install./stegseek_0.6-1.deb`
	Descarga e instalación de la versión más reciente de *stegseek*

`sudo pip install pyenchant`	Biblioteca utilizada para encontrar palabras diccionario en las cadenas de caracteres extraídas con la función strings
`pip install stego-lsb`	Instalación de stegano

`wget https://github.com/syvaidya/openstego/releases/download/openstego-0.8.4/openstego_0.8.4-1_all.deb`
 	
`sudo dpkg -i openstego_0.8.4-1_all.deb`
	Descarga e instalación de openstego

Tabla 15 Comandos de instalación de componentes utilizados por Oculto.

9.3 Guía de uso de Oculto

Para conocer el uso de Oculto, se ofrecen las siguientes opciones

- Ejecución del comando Oculto_Steg.py -h o Oculto_Steg.py –help según se ilustra en la Figura 49.
- GUIA RAPIDA Oculto es un documento que explica el uso de Oculto..

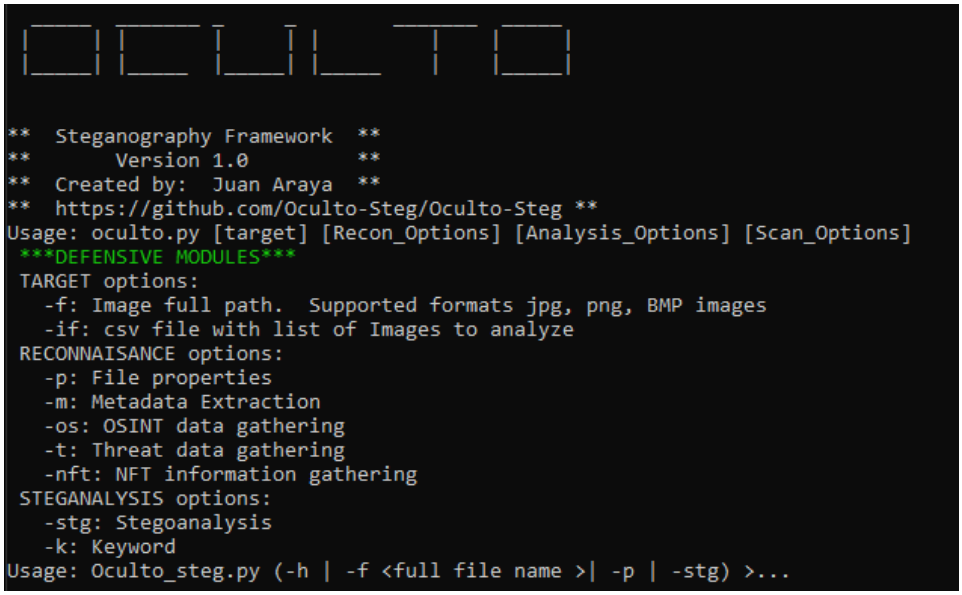


Figura 49 Salida del comando Oculto_Steg.py -h o Oculto_Steg.py –help.

9.3.1 Mostrar propiedades del archivo

Sintaxis:
python3 Oculto_steg.py -f <full path and file name with extension> -p

Ejemplo 1
python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/ejemplo2.png -p

El resultado de la ejecución del comando del ejemplo 1 se muestra en la Figura 50:

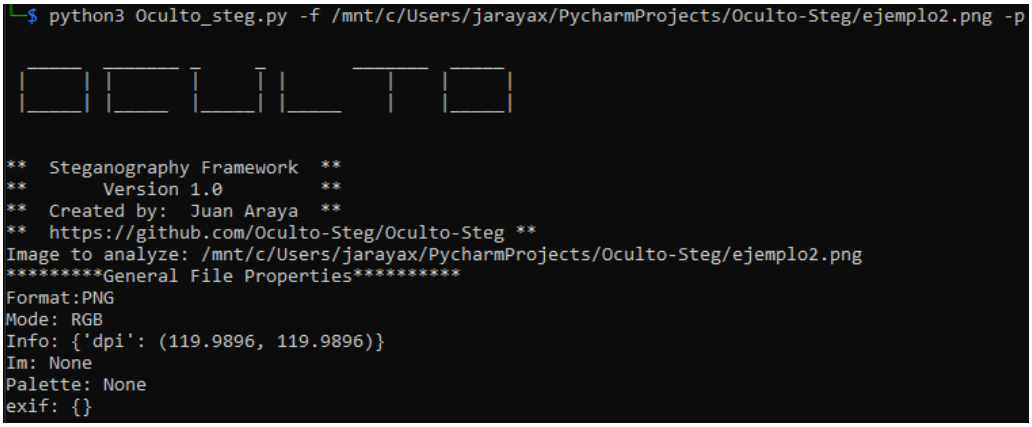


Figura 50 Salida de Oculto_Steg.py para obtener las propiedades de un archivo.

9.3.2 Mostrar propiedades y metadata del archivo

python3 Oculto_steg.py -f <full path and file name with extension> -p -m

Ejemplo 2
python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/Lambo.jpg **-p -m**

El resultado de la ejecución del comando del ejemplo 2 se muestra en la Figura 51:

```
*********File Metadata:**********
Extracting File metadata
ExifTool Version Number         : 12.40
File Name                       : Lambo.jpg
└─$
ects
File Size                       : 8.5 KiB
File Modification Date/Time     : 2022:04:08 00:53:32+02:00
File Access Date/Time           : 2022:04:08 00:53:33+02:00
File Inode Change Date/Time     : 2022:04:08 00:53:32+02:00
File Permissions                : -rwxrwxrwx
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 1
Y Resolution                    : 1
Image Width                     : 299
Image Height                    : 168
```

Figura 51 Salida de Oculto_Steg.py para obtener las propiedades y el metadata de un archivo.

9.3.3 Mostrar propiedades, metadata, datos con *OSINT*

python3 Oculto_steg.py -f <full path and file name with extension> -p -m -os

Ejemplo 3
python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/Lambo.jpg **-p -m -os**

El resultado de la ejecución del comando del ejemplo 3 se muestra en la Figura 52:

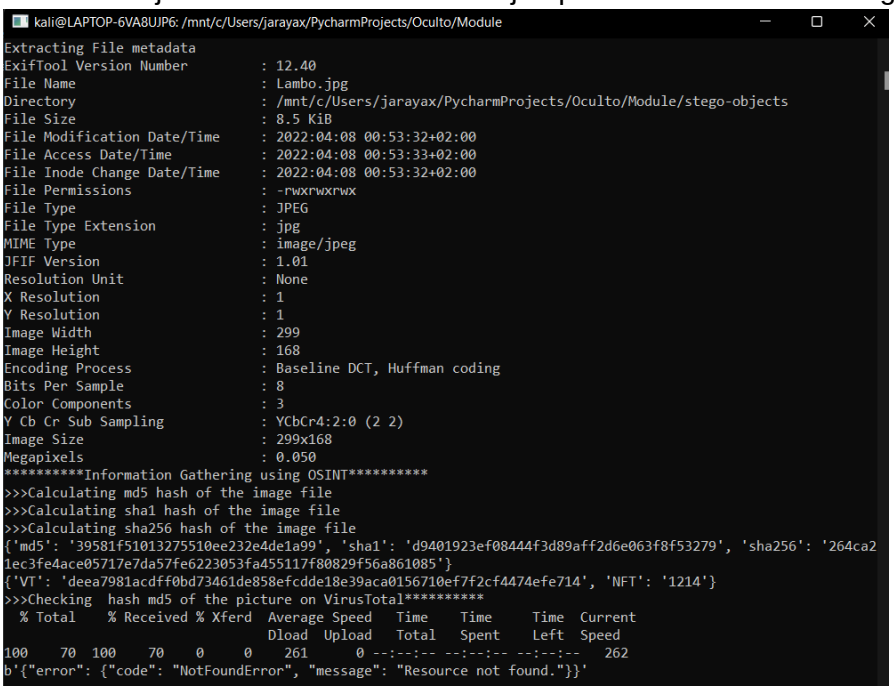


Figura 52 Salida de Oculto_Steg.py para consultar información acerca del estego objeto con OSINT.

9.3.4 Propiedades, metadata búsqueda de datos básico con *OSINT* y estegoanálisis

python3 Oculto_steg.py -f <full path and file name with extension> -p -m -os -stg

Ejemplo 4

python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/Lambo.jpg **-p -m -os -stg**

El resultado de la ejecución del comando del ejemplo 4 se muestra en la Figura 53:

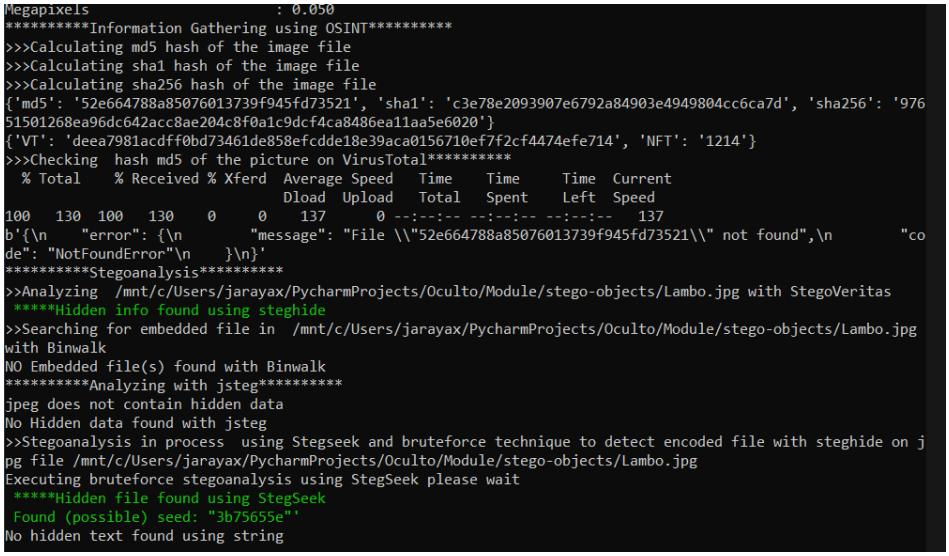


Figura 53 Salida de Oculto_Steg.py del estegoanálisis y busqueda de información con OSINT.

Ejemplo 5

python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/mydreamcar.jpg -p -m -os -stg

El resultado de la ejecución del comando del ejemplo 5 se muestra en la Figura 54:

```
**********Stegoanalysis**********
>>Analyzing  /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/mydreamcar.jpg with StegoVeritas
No hidden info found using steghide
>>Searching for embedded file in  /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/mydreamcar.jpg with Binwalk
 *****Embedded file(s) found with Binwalk
b'\nDECIMAL       HEXADECIMAL     DESCRIPTION\n--------------------------------------------------------------------------------\n0             0x0             PNG image, 920 x 690, 4-bit colormap, non-interlaced\n32840         0x8048          JPEG image data, JFIF standard 1.01\n\n'
>>Searching for embedded files or data in  /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/mydreamcar.jpg with Zsteg
NO VALUE
zsteg -a /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/mydreamcar.jpg
[?] 9141 bytes of extra data after image end (IEND), offset = 0x8048
[?] 9141 bytes of extra data after image end (IEND), offset = 0x8048
extradata:0         .. file: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, baseline, precision 8, 299x168, components 3
    00000000: ff d8 ff e0 00 10 4a 46  49 46 00 01 01 00 00 01  |......JFIF......|
    00000010: 00 01 00 00 ff db 00 43  00 09 06 07 12 12 12 15  |.......C........|
    00000020: 12 10 12 15 15 15 15 15  15 15 16 15 15 15 17 15 15  |................|
    00000030: 15 10 15 15 15 16 16 16  16 17 15 18 1d 28 20 18  |.............( .|
```

Figura 54 Salida de Oculto_Steg.py de propiedades, metadata, estegoanálisis y busqueda de información con OSINT.

9.3.5 Estegoanálisis automático y búsqueda de palabra clave(flag)

python3 Oculto_steg.py -f <full path and file name with extension> -k < clave> -p -m -os -stg

Ejemplo 6

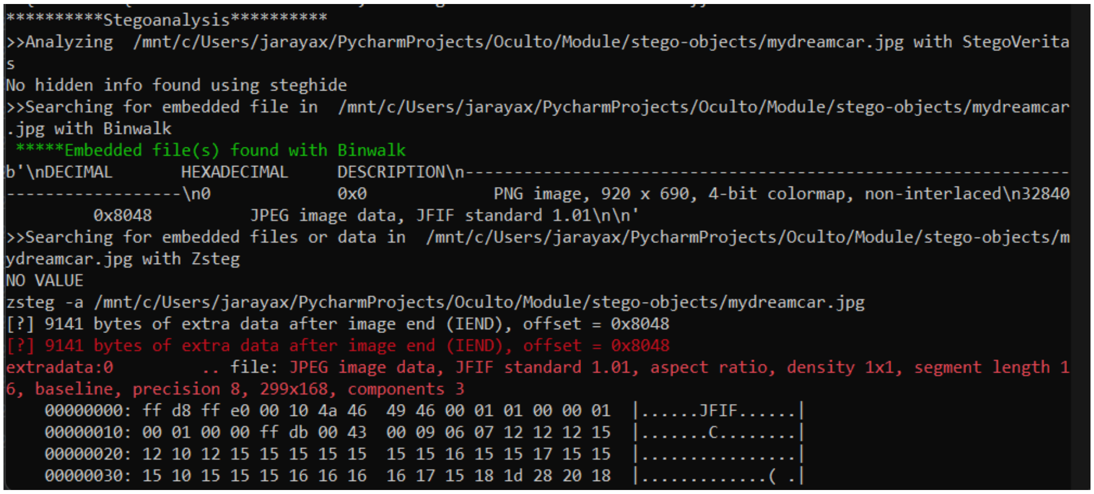
python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/toyota.png -k EJEMPLO -p -m -os -stg

El resultado de la ejecución del comando del ejemplo 6 se muestra en la Figura 55:

```
**********Stegoanalysis**********
>>Analyzing  /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/toyota.png with StegoVeritas
No hidden info found using steghide
>>Searching for embedded file in  /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/toyota.png with Binwalk
NO Embedded file(s) found with Binwalk
>>>Searching hidden keyword EJEMPLO on file using strings commands
L0I0I0I0I0
4       &       &
IEND
EJEMPLO{sample}
>>Searching for embedded files or data in  /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/toyota.png with Zsteg
EJEMPLO
zsteg -a /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/toyota.png | grep --color -E EJEMPLO -B 100 -A 100
[?] 16 bytes of extra data after image end (IEND), offset = 0x8048
[?] 16 bytes of extra data after image end (IEND), offset = 0x8048
extradata:0         .. text: "EJEMPLO{sample}\n"
imagedata           .. text: "<<<333<3<<<<"
b2,rgb,msb,xy       .. text: "WUUUUUUUUUU"
```

Figura 55 Salida de Oculto_Steg.py palabra clave encontrada con zsteg.

E incluso como se muestra en la Figura 56 Oculto es capaz de identificar con el uso de herramientas como *zsteg*, el uso de esteganografía en la imagen.

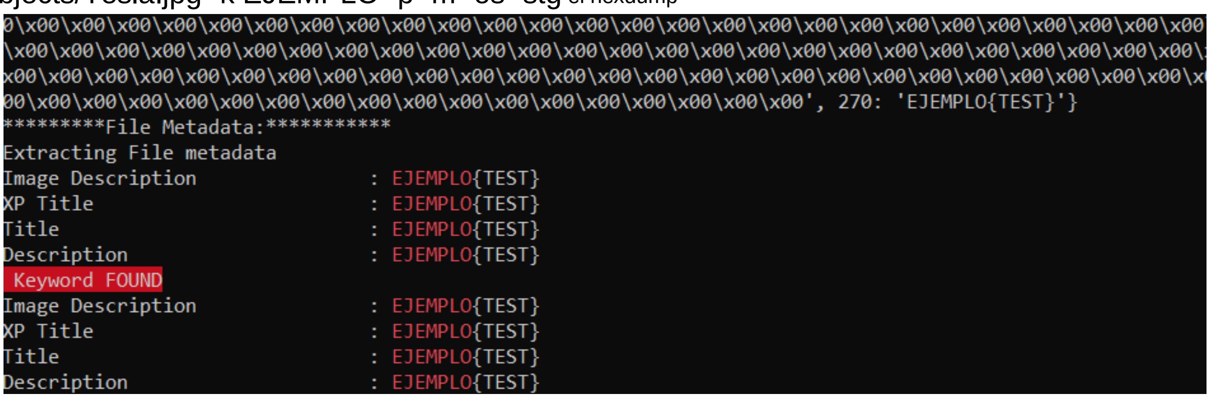


```
*********Stegoanalysis**********
>>Analyzing  /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/mydreamcar.jpg with StegoVerita
s
No hidden info found using steghide
>>Searching for embedded file in  /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/mydreamcar
.jpg with Binwalk
 *****Embedded file(s) found with Binwalk
b'\nDECIMAL       HEXADECIMAL     DESCRIPTION\n--------------------------------------------------------------
-----------------\n0             0x0             PNG image, 920 x 690, 4-bit colormap, non-interlaced\n32840
         0x8048          JPEG image data, JFIF standard 1.01\n\n'
>>Searching for embedded files or data in  /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/m
ydreamcar.jpg with Zsteg
NO VALUE
zsteg -a /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/mydreamcar.jpg
[?] 9141 bytes of extra data after image end (IEND), offset = 0x8048
[?] 9141 bytes of extra data after image end (IEND), offset = 0x8048
extradata:0         .. file: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 1
6, baseline, precision 8, 299x168, components 3
    00000000: ff d8 ff e0 00 10 4a 46  49 46 00 01 01 00 00 01  |......JFIF......|
    00000010: 00 01 00 00 ff db 00 43  00 09 06 07 12 12 12 15  |.......C........|
    00000020: 12 10 12 15 15 15 15 15  15 15 16 15 15 17 15 15  |................|
    00000030: 15 10 15 15 15 16 16 16  16 17 15 18 1d 28 20 18  |.............( .|
```

Figura 56 Salida de Oculto_Steg.py donde descubre información oculta usando binwalk.

Según se aprecia en la Figura 57, Oculto buscará la presencia de la palabra clave incluso en el metadata de la imagen

python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto/Module/stego-objects/Tesla.jpg -k EJEMPLO -p -m -os -stg el hexdump



```
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00', 270: 'EJEMPLO{TEST}'}
*********File Metadata:***********
Extracting File metadata
Image Description               : EJEMPLO{TEST}
XP Title                        : EJEMPLO{TEST}
Title                           : EJEMPLO{TEST}
Description                     : EJEMPLO{TEST}
 Keyword FOUND
Image Description               : EJEMPLO{TEST}
XP Title                        : EJEMPLO{TEST}
Title                           : EJEMPLO{TEST}
Description                     : EJEMPLO{TEST}
```

Figura 57 Salida de Oculto_Steg.py donde descubre información oculta en el metadata.

9.3.6 Analizar un NFT

Sintaxis: python3 Oculto_steg.py -nft

Ejemplo 8
Ejecuta el siguiente comando
python3 Oculto_steg.py -nft

Ingresa el Token ID y el contract ID de la imagen NFT que se desee verificar, según se ilustra en la Figura 58.

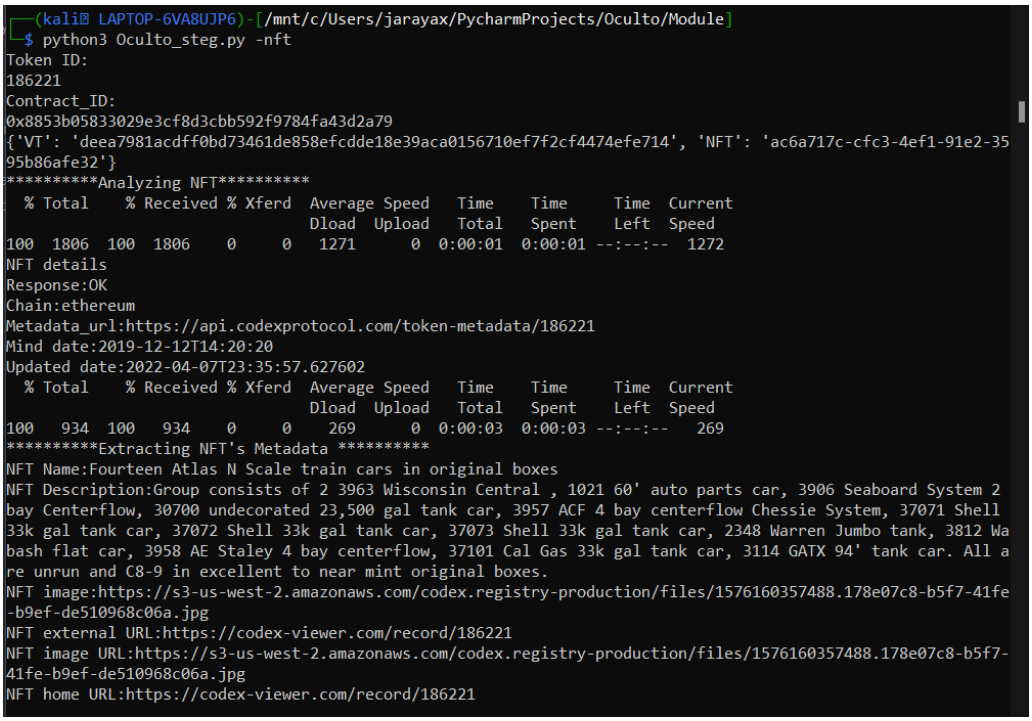


Figura 58 Salida de Oculto_Steg.py con resultados de análisis de un NFT.

9.4 Creación de Retos del CTF

Se crearon retos de esteganografía en imágenes basados en retos implementados en CTF alrededor del mundo disponibles en www.ctftime.org. Cada reto contiene un estego-objeto a analizar que se puede resolver siguiendo los mismos pasos descritos en los retos seleccionados como base. De esta manera se aseguró que tuvieran el mismo nivel de complejidad de un CTF oficial.

9.5 El criterio de selección de los retos base

Se seleccionaron de forma aleatoria 25 retos de CTFs con el fin de utilizarlos como base para crear los CTFs del proyecto de estegoanálisis.

Se aplicaron los siguientes filtros para seleccionar de forma aleatoria los 25 retos base

- Retos utilizados en CTF ejecutados en el rango de años 2015 al 2022
- El archivo para analizar debe ser imágenes o archivos zip con imágenes.
- Se haya ocultado datos o texto en la imagen utilizando LSB
- No se requiera un estegoanálisis visual para descubrir el mensaje oculto.

A continuación, en la Tabla 16 se muestra el listado de retos utilizados en CTFs anteriores y disponibles en www.ctftime.org que fueron utilizados como inspiración para la creación de retos nuevos para el evento de CTF llamado StegoCTF.

ID	CTF	Reto
1	UMDCTF 2021	https://ctftime.org/writeup/27982
2	HSCTF2021	https://github.com/BASHing-thru-challenges/HSCTF-2021-Writeups/tree/main/misc/LSBlue
3	UMDCTF2021	https://github.com/HuntClauss/Writeups/blob/main/umdctf/testuado-pizza.md
4	dvCTF2021	https://github.com/Kasimir123/CTFWriteUps/tree/main/2021-03-dvCTF/pidieff
5	MetaredCTF2021	https://www.it-sec.fail/metared-ctf-2021-stego-friends-writeup/
6	WPICTF 2021	https://github.com/yulyachert/WPICTF-writeups/blob/master/Glute.md
7	zg3r0CTF2020	https://github.com/itsecgary/CTFs/tree/master/ZH3R0CTF%202020/LSB%20Fun
8	Shakti CTF 2020	https://ctftime.org/writeup/25148
9	CyberYoddha CTF 2020	https://ctftime.org/task/13740
10	RadarCTF 2019	https://ctftime.org/writeup/14354
11	Zh3r0 CTF	https://ctftime.org/writeup/21717
12	RITSEC CTF 2019	https://ctftime.org/writeup/17314
13	DeconstruCT.F 2021	https://sutharnisarg.medium.com/deconstructf-2021-write-ups-7252dcbe894a
14	OFPPT-CTF Morocco 2022	https://ctftime.org/writeup/32716
15	UMDCTF 2021	https://ctftime.org/task/15590
16	DEADFace CTF 2021	https://ctftime.org/writeup/31024
17	Affinity CTF Lite	https://ctftime.org/writeup/25024
18	RITSEC CTF 2021	https://klefz.se/2021/04/12/ritsec-ctf-2021-write-up/#inception5
19	Tenable CTF 2021	https://ctftime.org/writeup/26458
20	RaziCTF 2020	https://ctftime.org/task/13658

21	DaVinciCTF 2021	https://ctftime.org/writeup/28941
22	UMDCTF 2020	https://ctftime.org/task/11273
23	csictf 2020	https://ctftime.org/task/12454
24	Zh3r0 CTF	https://ctftime.org/writeup/21648
25	Hacktober CTF 2020	https://ctftime.org/writeup/24241

Tabla 16 Listado de retos utilizados como base para los retos del StegoCTF.

9.6 Retos del StegoCTF

A continuación, en la Figura 59 aparece la lista de los retos nuevos diseñados desde cero para el StegoCTF, basados en los retos mencionados en la sección 11.5.

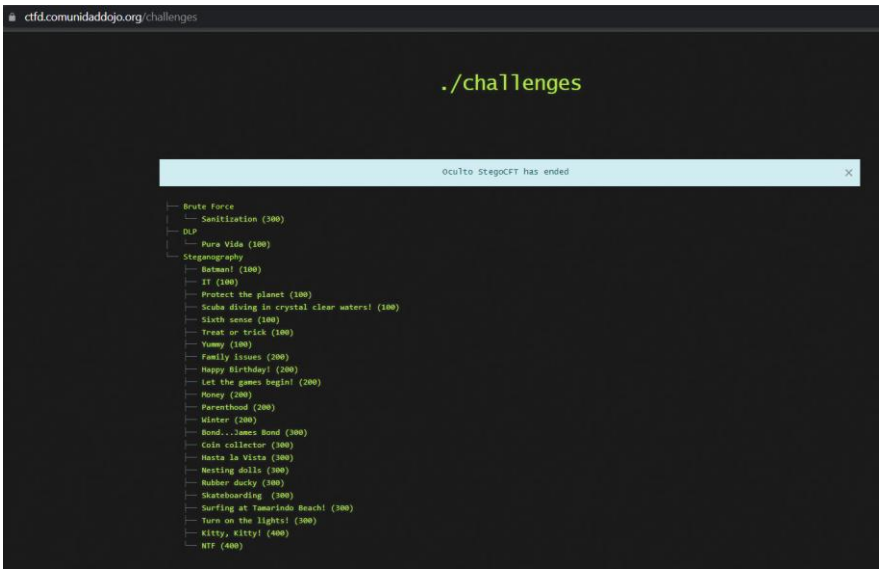


Figura 59 Listado de los retos de estegoanálisis utilizados durante el StegoCTF.

La Tabla 17 contiene cada uno de los retos fueron escritos en inglés dado que se tuvo participación de personas de la India y de África que no hablan español. También cada uno de los participantes de Latinoamérica eran bilingües y dominaban el inglés.

Nombre del Reto	**Descripción**	**Pista (Hint) ofrecido**	**Flag**	**Puntaje**
Pura Vida	Alex is a former system administrator who was fired this morning. The DLP triggered an alert that Alex tried to zip file to his personal email from the corporate	Flag has the format Oculto{} and is hidden in 1 of the pictures	`Oculto{C0st4_R1c4}`	100

	email. According to the email, the zip file contains pictures from his last trip to Costa Rica. Please analyze the pictures and verify if there is any possible data ex-filtration.			
Scuba diving in crystal clear waters!	Wow Scuba diving is fun!	NONE	Oculto{F1nd1ngN3M0}	100
Yummy	I love mediterranean food!	NONE	Oculto{Sp4nishF00disGr54t}	100
Winter	I love snow!	NONE	Oculto{y3ll0wsn0w1sn0tG00d}	200
Bond...James Bond	A spy sent a file that was intercepted by the enemy.	Try Bruteforcing it with stegcracker or *stegseek*... it might help to identify the password	Oculto{007T0REPORTHQ}	300
Skateboarding	Skateboarding is not a crime!	Extract the embedded jpg from the png and analyze it.	Oculto{T0NYH4WKR0CKS}	300
Family issues	You can choose your friends...	NONE	Oculto{14MY0URF4TH3R}	200
Rubber ducky	Got to be my lucky day, I just found a cool USB drive which only contains a zip file was found in a USB drive.	Decode to extract	Oculto{C0ntr0l1s4n1llus10n}	300
Kitty, Kitty!	Cats are mysterious animals!	*Cyberchef*	Oculto{C4tg0ty0urt0ngu3}	400
Surfing at Tamarindo Beach!	I love **s**urfing **t**he **e**dge, **g**o od **h**armony. **I**deal for having a goo**d** tim**e**! Enjoy and **B**ehave like 24 even if you are **64**	NONE	Oculto{4_b4d_d4y_surf1ng_is_b3tt3r_th4n_a_g00d_d4y_w0rk1ng}	300
Batman!	OMG! Batman!	NONE	Oculto{H0ly_rust3d_m3t4l,_B4tm4n!}	100
Treat or trick	You better feed this terrible vampire with a treat to solve this challenge!	NONE	Oculto{Dr4cuL4}	100

Turn on the lights!	Fear of the Dark!	Check the metadata for data extraction	`Oculto{H4v3_y0u_3v3r_b33n_4l0n3_4t_n1ght}`	300
Parenthood	I love my daughter, she is the apple of my eyes.	I love metadata	`Oculto{Tru3_L0V3}`	200
Sixth sense	This scary picture might contain a message from the infraworld!	NONE	`Oculto{Sh3_w4nt3d_m3_t0_t3ll_y0u...}`	100
IT	>... What if it's just some crazy guy in a clown suit?	`NONE`	`Oculto{Cr33py_Cl0wn}`	100
Let the games begin!	Play time! Check these games. One of them is addictive!	`NONE`	`Oculto{D0_n0t_B3_S0rry._B3_B3tt3r}`	200
Money	Not everything that glitters is gold	`BASE64`	`Oculto{M0N3Y_N3c3ss4ry_3v1l}`	200
NTF	Seems that one of the NFT sold by https://twitter.com/ZeddSmith4 might contains a small surprise. Use the creator name as password..it might work	NONE	`Oculto{L0s_NTFs_pueden_traer_S0RPR3S4}`	400
Hasta la Vista	The following screenshot was found on the computer of a disgruntled former employee. Analyzing it and verify if there is a hidden message	Smell what the rock is cooking	`Oculto{1_C4n_s33_Y0U}`	300
Protect the planet	It's our only home!	NONE	`Oculto{1t_1s_0uR_R3sp0ns4b1l1ty}`	100
Coin collector	Yes, I love my coin collection, but I need to tag them to remember when I purchased, where they are from and sort them from a to z. Luxembourg franc used to have one of the most beautiful designs I have ever seen. My favorites are	Smell what the *Cyberchef* is cooking! roasted BITS	`Oculto{1T-1S_L1K3_Tr4V3LL1NG}`	300

	from **H**onduras, **E**cuador and Me**x**ico			
Sanitization	A “database backup” zip file was found in a computer that was going to be donated. What is it?	Password is a famous superhero. Guess it or use brute force.	`Oculto{3ras3_4ll_d4t4_pr10r_d0n4t1ng}`	300
Nesting dolls	Matryoshka	`Check metadata`	`Oculto{1ns1d3_0uT}`	300
Happy Birthday!	Surprise!	`NONE`	`Oculto{4G3_1S_A_M1ndS3t}`	200

Tabla 17 Listado de retos del StegoCTF.

9.7 Resolución de los retos del StegoCTF

9.7.1 Reto: Pura vida

La Figura 60 muestra la descripción del reto “Pura Vida” observado por los participantes durante el CTF.

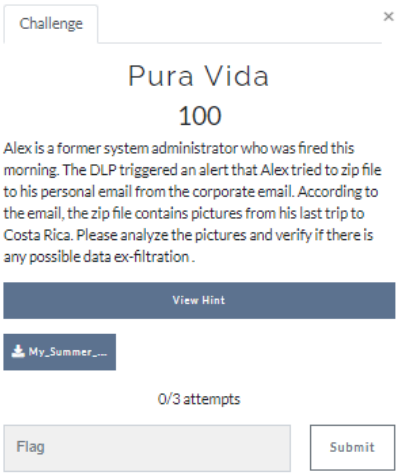


Figura 60 Descripción del reto “Pura Vida”.

Archivo: My_Summer_Vacations.zip

Reto 1: Pura Vida

1. Descargar el archivo zip y extraer los archivos. En la Figura 61 se listan los archivos que fueron extraídos del My_Summer_Vacations.zip.

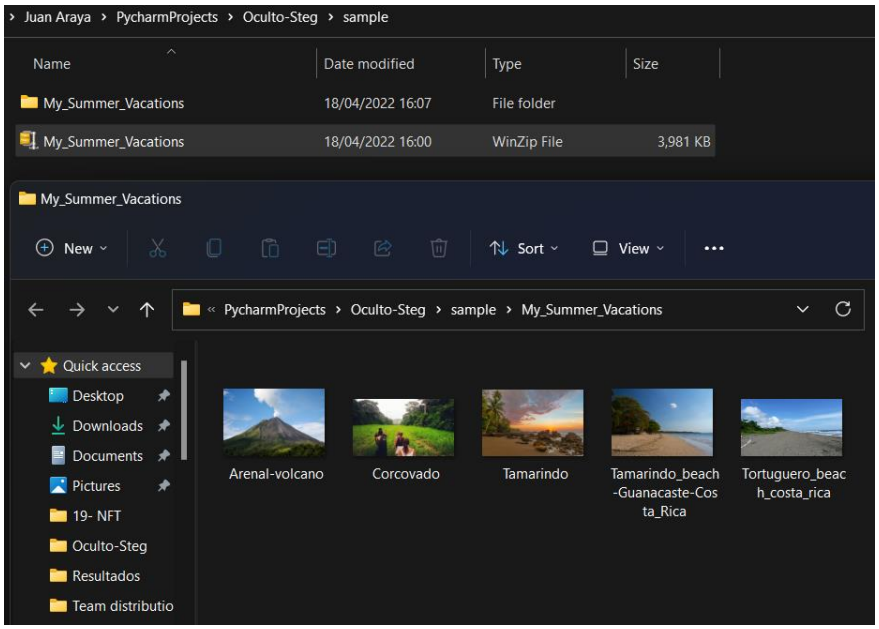


Figura 61 Extracción de archivos comprimidos dentro de My_Summer_Vacations.zip.

2. Analizar cada una de las imágenes con Oculto para descubrir el flag Oculto{VALOR} con el comando que aparece en la Figura 62.

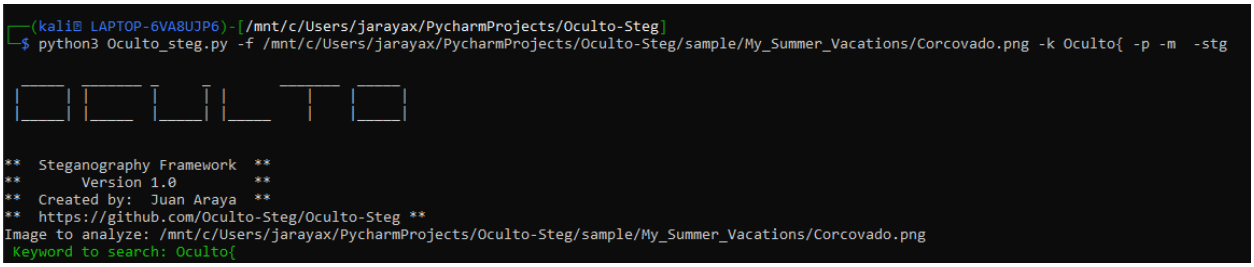


Figura 62 Análisis de la imagen corvovado.png con Oculto.

3. En la Figura 63 se observa que Oculto se descubre un archivo oculto con *Binwalk* en el archivo Corcovado.png

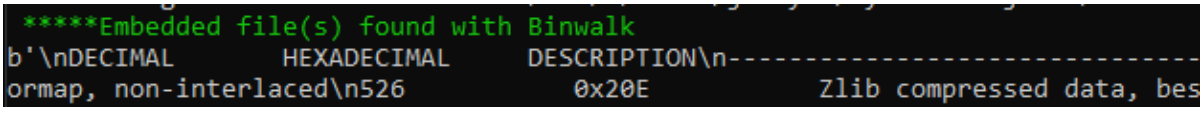


Figura 63 Dato detectado con Binwalk dentro del archivo corcovado.png.

4. En la Figura 64 se aprecia donde Oculto extrae el mensaje oculto(flag) utilizando strings

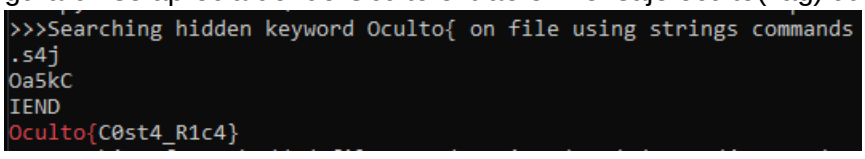


Figura 64 Valor secreto detectado dentro del archivo corcovado.png con Oculto using strings.

5. Se confirma la existencia del mensaje oculto con *zsteg* según lo demuestra la Figura 65.

```
>>Searching for embedded files or data in  /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/My_Summer_Vacations/Corcovado.png with Zsteg
Oculto{
zsteg -a /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/My_Summer_Vacations/Corcovado.png | grep --color -E Oculto{ -B 100 -A 100
[?] 19 bytes of extra data after image end (IEND), offset = 0x1ea5b
[?] 19 bytes of extra data after image end (IEND), offset = 0x1ea5b
extradata:0         .. text: "Oculto{C0st4_R1c4}\n"
```

Figura 65 Valor secreto detectado dentro del archivo corcovado.png con Oculto usando zsteg.

9.7.2 Reto: Scuba diving in crystal waters

La Figura 66 muestra la descripción del reto “Scuba diving in crystal waters” observado por los participantes durante el CTF.

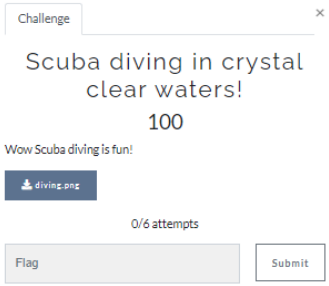


Figura 66 Descripción del reto “Scuba Diving in crystal waters”.

Archivo: diving.png. La vista previa de la imagen se aprecia en la Figura 67.

Figura 67 Vista Previa del Archivo: diving.png.

1. Descargar el archivo diving.png
2. Analizar el archivo diving.png con Oculto con el comando que aparece en la Figura 68.

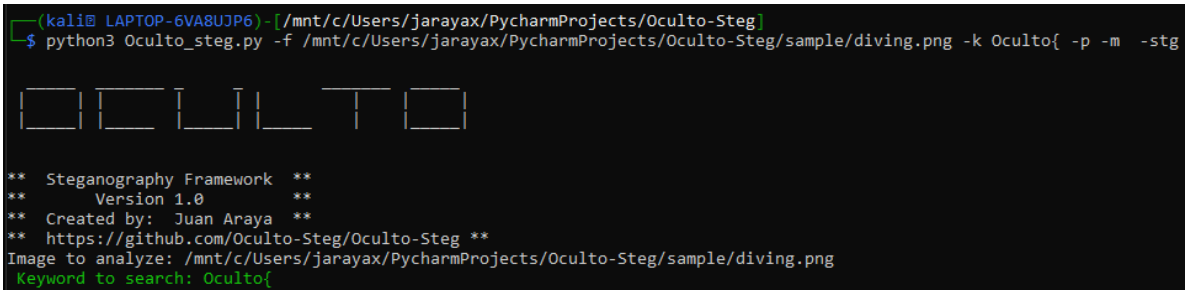


Figura 68 Comandos para el análisis del archivo diving.png con Oculto.

3. En la Figura 69 se descubre algo oculto con *Binwalk* en el archivo diving.png

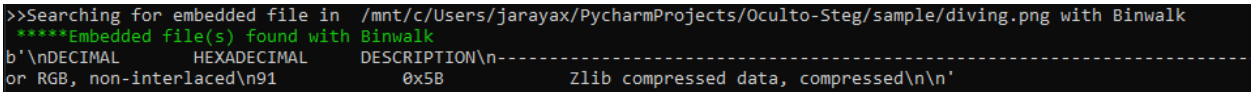


Figura 69 Detección con Oculto usando binwalk de un archivo oculto dentro de archivo diving.png.

4. Se observa en la Figura 70 que Oculto extrae el mensaje oculto(flag) utilizando strings.

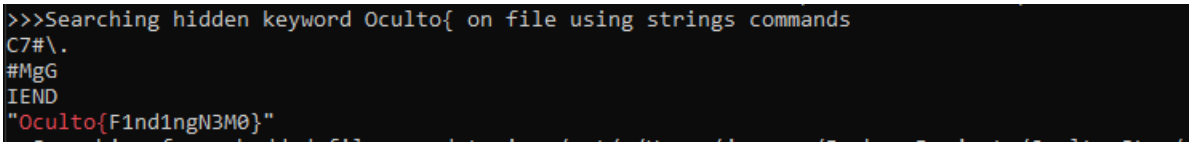


Figura 70 Detección del valor oculto usando Oculto y la herramienta strings.

5. En la Figura 71 se confirma la existencia del mensaje oculto con *zsteg*

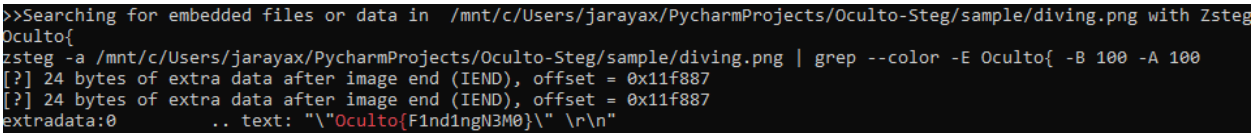


Figura 71 Detección del valor oculto usando Oculto y la herramienta zsteg.

9.7.3 Reto: Yummy

La Figura 72 muestra la descripción del reto “Scuba Diving in crystal waters” observado por los participantes durante el CTF.

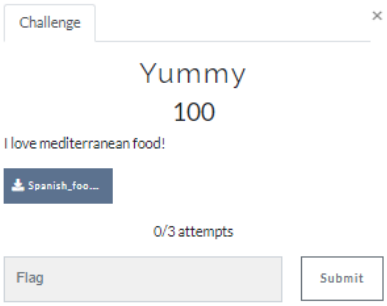


Figura 72 Descripción del reto “Yummy”.

Archivo: Spanish_food.zip.

1. Descargar el archivo Spanish_food.zip y extraer los archivos. En la Figura 73 se muestran las imágenes extraídas del archivo Spanish_food.zip.

Figura 73 Archivos extraídos del Spanish_food.zip.

2. Analizar cada una de las imágenes con Oculto para descubrir el flag Oculto{VALOR} con el comando que aparece en la imagen 74.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Spanish_food/paella.png -k Oculto{ -p -m -os -stg

 _____  _____ _     _       _____  _____
|     | |     |  \  /  |     |   | |     |
|_____| |_____|   \/   |_____|   | |_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Spanish_food/paella.png
Keyword to search: Oculto{
```

Figura 74 Comandos para el análisis del archivo paella.png con Oculto.

3. Oculto detecta y extrae el mensaje oculto(flag) utilizando strings según se observa en la imagen 75.

```
**********Stegoanalysis**********
>>Analyzing  /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Spanish_food/paella.png with StegoVeritas
No hidden info found using steghide
>>Searching for embedded file in  /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Spanish_food/paella.png with Binwalk
NO Embedded file(s) found with Binwalk
>>>Searching hidden keyword Oculto{ on file using strings commands
;<Go
9=.OL2
aN)9
Oculto{Sp4nishF00disGr54t}
```

Figura 75 Detección del valor oculto con Oculto y el comando strings.

9.7.4 Reto: Winter

La Figura 76 muestra la descripción del reto “Winter” observado por los participantes durante el CTF.

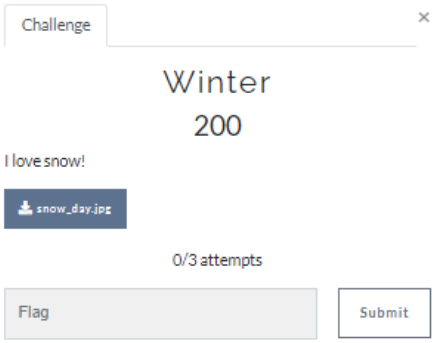


Figura 76 Descripción del reto “Winter”.

Archivo: snow_day.jpg. La Figura 77 muestra la vista previa de la imagen a analizar con Oculto.

1. Descargar el archivo snow_day.jpg

Figura 77 Vista Previa de la imagen snow_day.jpg.

1. Analizar la imagen con Oculto utilizado el comando que se muestra en la Figura 78.

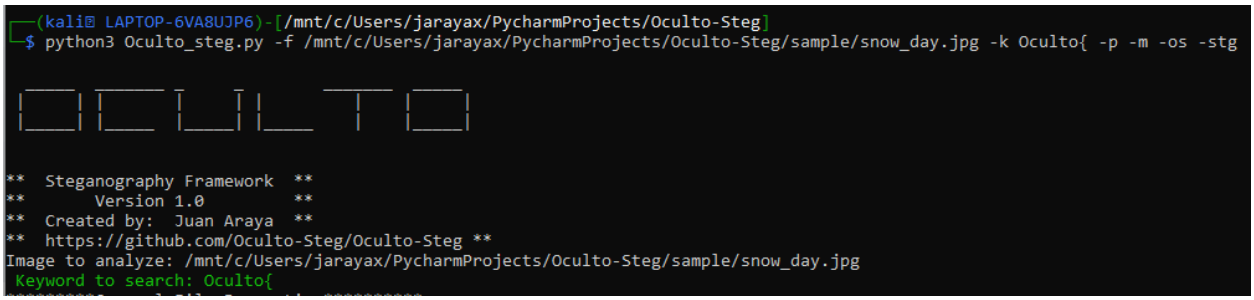


Figura 78 Comandos para el análisis del archivo snow_day.jpg con Oculto.

2. En la Figura 79 se aprecia como Oculto, a través de la herramienta *Stegoveritas* detecta la presencia de algo oculto con *Steghide*.

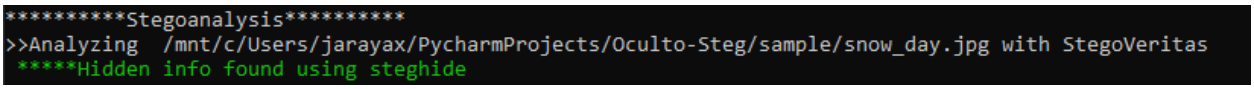


Figura 79 Detección de uso de Steghide para ocultar datos en snow_day.jpg detectado por Oculto usando Stegoveritas.

3. *Binwalk* detecta un archivo PDF oculto dentro de la imagen snow_day.jpg según se confirma en la Figura 80.

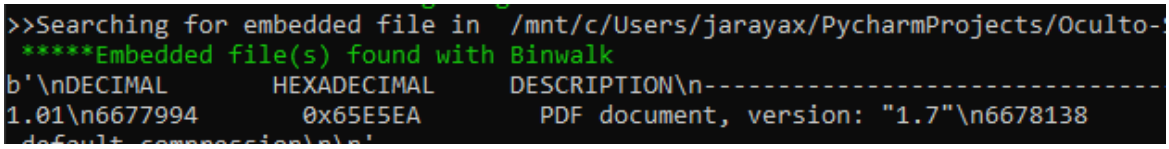


Figura 80 Detección de archivo oculto dentro de archivo snow_day.jpg detectado por Oculto usando binwalk.

4. En la imagen 81 se nota como Oculto, a través del uso de *Stegseek* confirma que hay algo oculto y cifrado con *steghide*.

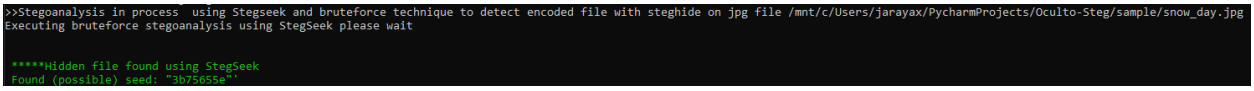


Figura 81 Detección de archivo oculto y cifrado con steghide detectado por Oculto usando Stegseek.

5. Extraiga el PDF del archivo snow_day.jpg utilizando *binwalk* tal y como se observa en la Figura 82.

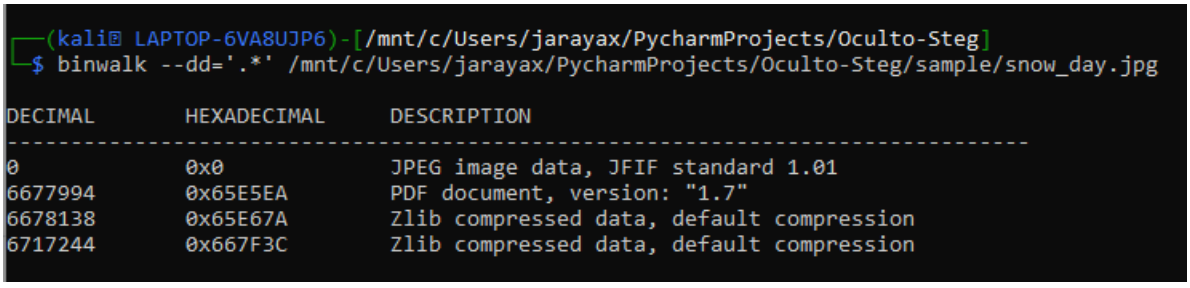


Figura 82 Extracción del archivo oculto dentro de snow_day.jpg con binwalk.

6. Revise los archivos extraídos. La imagen 83 muestra los archivos extraídos en el folder llamado “_snow_day.jpg.extracted”.

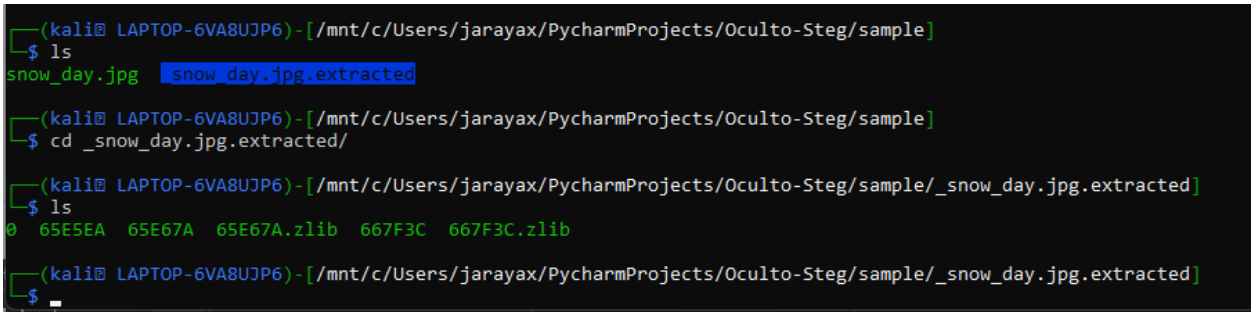


Figura 83 Revisión de archivos extraídos con binwalk.

7. Identifique el archivo pdf. En la Figura 84 se observa que el archivo 6EE5EA es un archivo PDF que estaba oculto dentro de la imagen.

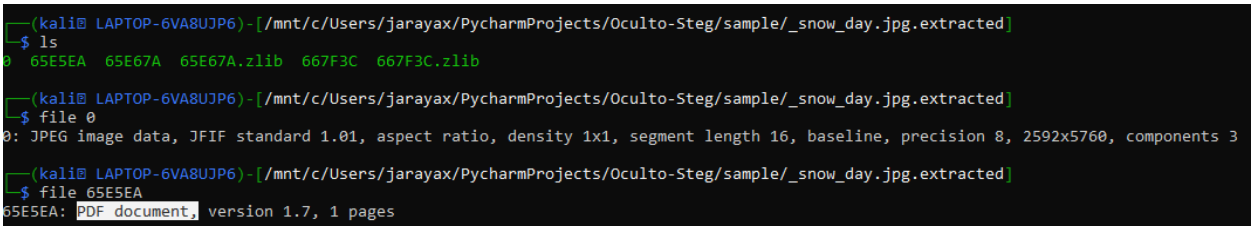


Figura 84 Identificación del tipo de archivo extraído con binwalk.

8. Cambie la extensión del archivo 65E5EA por PDF como se observa en la imagen 85.

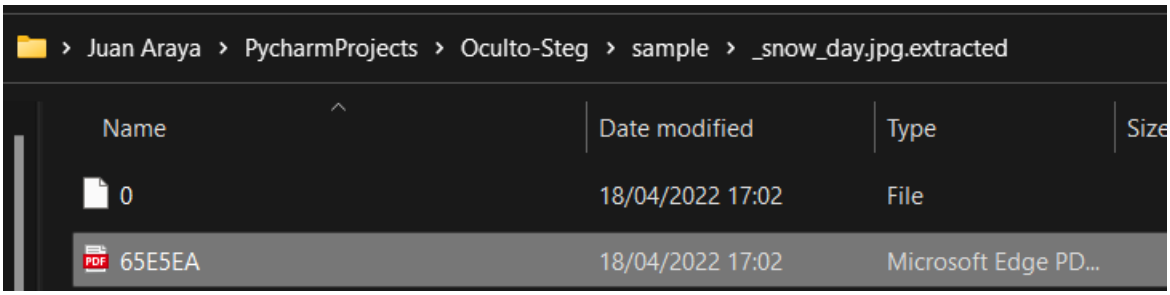


Figura 85 Cambio de extensión del archivo extraído.

9. Abra el archivo PDF y observe el mensaje oculto que se muestra en la imagen 86.

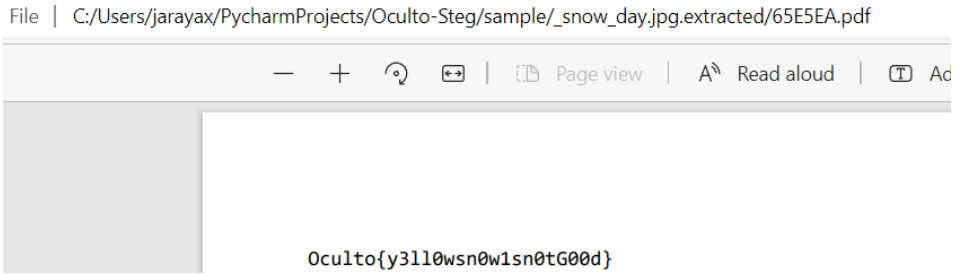


Figura 86 Flag en texto plano en el archivo 65E5A.pdf.

9.7.5 Reto: Bond...James Bond

La Figura 87 muestra la descripción del reto “Bond...James Bond” observado por los participantes durante el CTF.

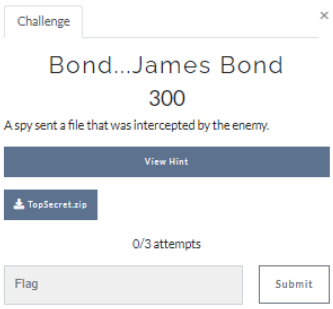


Figura 87 Descripción del reto “Bond...James Bond”.

Archivo: TopSecret.zip

1. Descargar el archive TopSecret.zip y extraer los archivos correspondientes. En la Figura 88 se observan una imagen y un archivo de texto fueron extraídos.

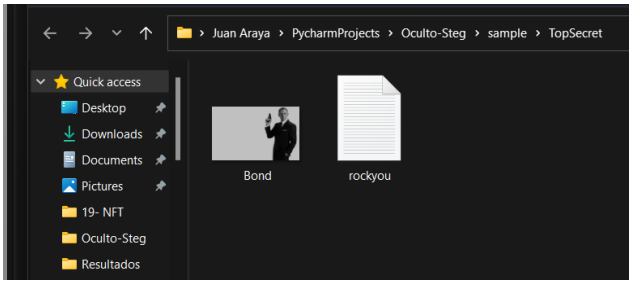


Figura 88 Archivos extraídos de TopSecret.zip.

2. Analizar el archivo con Oculto con el comando que aparece en la Figura 89.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/TopSecret/Bond.jpg -k Oculto{ -p -m -os -stg
 _____ _______ _     _       _______ _____
|     | |       |     | |     |       |     |
|_____| |_____  |_____| |_____|       |_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/TopSecret/Bond.jpg
Keyword to search: Oculto{
```

Figura 89 Comandos para el análisis del archivo Bond.jpg con Oculto.

3. Oculto, con el uso de *Stegseek* es capaz de detectar que hay algo oculto y cifrado con *Steghide* según se observa en la Figura 90.

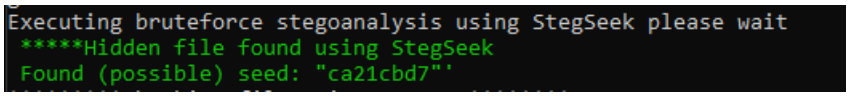


Figura 90 Detección de dato oculto con steghide y protegido con contraseña usando Oculto y stegseek.

4. Oculto, utilizando *steghide* trate de extraer los datos ocultos primero sin ingresar ninguna contraseña. En la Figura 91 se descubre que la imagen tiene algo que se podría extraer con *steghide*, pero que requiere una contraseña.

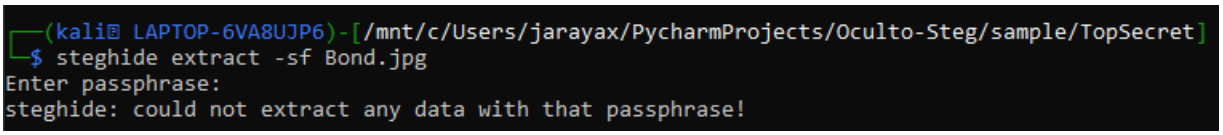


Figura 91 Confirmación de uso de steghide en el archivo Bond.jpg y protección con contraseña.

5. Esto confirma que será necesario utilizar el archivo rockyou.txt que estaba en el zip para descubrir la contraseña por fuerza bruta y así extraer el mensaje o archivo oculto. Para esto se puede utilizar *Stegseek*. En la Figura 92 se confirma que la contraseña JAMESBOND007 era requerida para poder extraer el archivo flag.txt que estaba oculto.

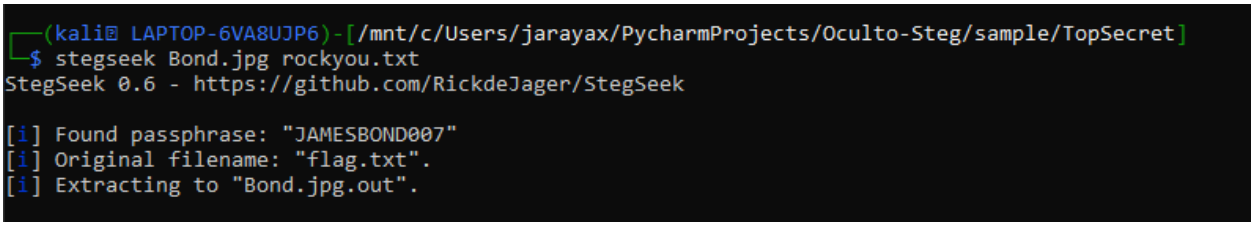


Figura 92 Extracción de archivo oculto con steghide.

6. Utilizando *steghide* y la contraseña descubierta con *Stegseek*, extraiga el mensaje oculto tal y como se muestra en la Figura 93.

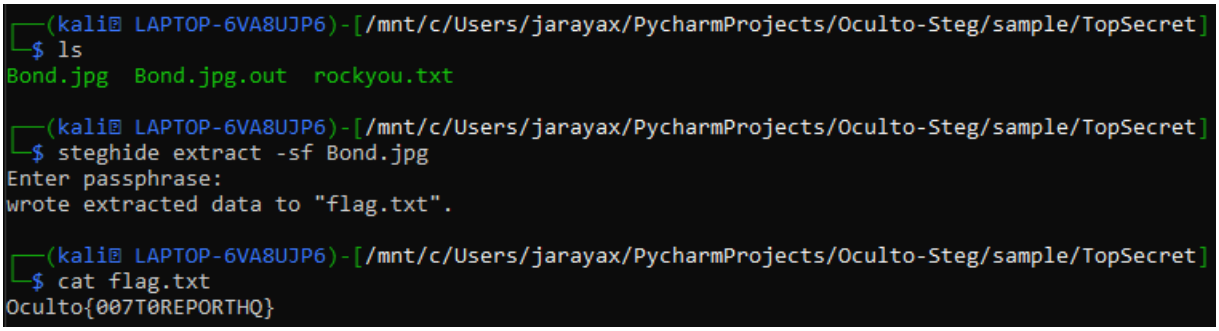


Figura 93 Extracción del mensaje oculto.

.

9.7.6 Reto: Skateboarding

La Figura 94 muestra la descripción del reto “Skateboarding” observado por los participantes durante el CTF.

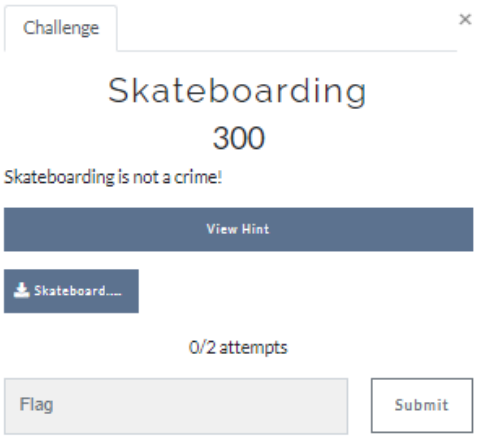


Figura 94 Descripción del reto “Skateboarding”.

Archivo: Skateboard.png. La vista previa de la imagen se muestra en la Figura 95.

Figura 95 Vista Previa de la imagen skateboard.png.

1. Descargue el archive Skateboard.png y analice la imagen con el comando que aparece en la Figura 96.

```
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Skateboard.png -k Oculto{ -p -m -os -stg

 _____ _______ _     _       _______ _____
|     | |       |     | |     |    |    |
|_____| |_____  |_____| |_____|    |    |_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Skateboard.png
 Keyword to search: Oculto{
```

Figura 96 Comandos para el análisis del archivo skateboard.png con Oculto.

2. En la Figura 97, se muestra que Oculto detecta con *Binwalk* la presencia de un archivo jpeg.

```
>>Searching for embedded file in  /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Skateboard.png
with Binwalk
 *****Embedded file(s) found with Binwalk
b'\nDECIMAL       HEXADECIMAL     DESCRIPTION\n--------------------------------------------------------------
----------------------\n0             0x0             PNG image, 726 x 435, 8-bit/color RGB, non-interlac
ed\n91            0x5B            Zlib compressed data, compressed\n406111        0x6325F         JPEG im
age data, JFIF standard 1.01\n406141        0x6327D         TIFF image data, big-endian, offset of first
```

Figura 97 Detección de archivo dentro de skateboard.png con Oculto y binwalk.

3. Oculto utilizando strings detecta el flag según la Figura 98.

```
>>>Searching hidden keyword Oculto{ on file using strings commands
IEND
JFIF
Exif
Oculto{T0NYH4WKR0CKS}
```

Figura 98 Detección de valor secreto con Oculto y el comando strings.

4. En la Figura 99 se observa que Oculto también detecta el string en el metadata.

```
+http://ns.adobe.com/xap/1.0/
<?xpacket begin='
' id='W5M0MpCehiHzreSzNTczkc9d'?>
<x:xmpmeta xmlns:x="adobe:ns:meta/"><rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"><rdf
:Description rdf:about="uuid:faf5bdd5-ba3d-11da-ad31-d33d75182f1b" xmlns:dc="http://purl.org/dc/elements/
1.1/"/><rdf:Description rdf:about="uuid:faf5bdd5-ba3d-11da-ad31-d33d75182f1b" xmlns:dc="http://purl.org/d
c/elements/1.1/"><dc:title><rdf:Alt xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"><rdf:li xml:l
ang="x-default">Oculto{T0NYH4WKR0CKS}</rdf:li></rdf:Alt>
                        </dc:title><dc:description><rdf:Alt xmlns:rdf="http://www.w3.org/1999/02/22-rdf-s
yntax-ns#"><rdf:li xml:lang="x-default">Oculto{T0NYH4WKR0CKS}</rdf:li></rdf:Alt>
                        </dc:description></rdf:Description></rdf:RDF></x:xmpmeta>
```

Figura 99 Detección de valor secreto en el metada con Oculto.

5. Zteg también detecta el flag según la Figura 100.

```
Oculto{
zsteg -a /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Skateboard.png | grep --color -E Oculto{
 -B 100 -A 100
[?] 5681573 bytes of extra data after image end (IEND), offset = 0x6325f
[?] 5681573 bytes of extra data after image end (IEND), offset = 0x6325f
extradata:0         .. file: JPEG image data, JFIF standard 1.01, resolution (DPI), density 72x72, segmen
t length 16, Exif Standard: [TIFF image data, big-endian, direntries=4, description=Oculto{T0NYH4WKR0CKS}
```

Figura 100 Detección de valor secreto en el metada con Oculto y zsteg.

9.7.7 Reto: Family issues

La Figura 101 muestra la descripción del reto “Family issues” observado por los participantes durante el CTF.

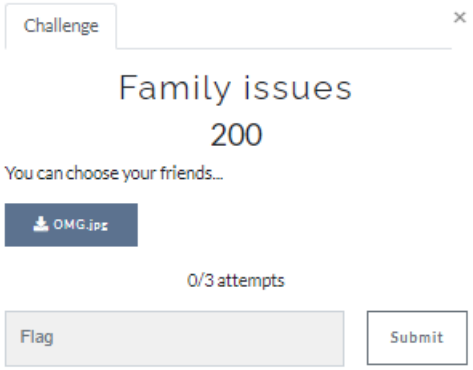


Figura 101 Descripción del reto “Family issues”.

Archivo: OMG.jpg. En la Figura 102 se observa la vista previa de la imagen a analizar en este reto.

Figura 102 Vista Previa de la imagen OMG.jpg.

1. Descargue la imagen OMG.jpg.
2. Analice la imagen con Oculto con el comando que aparece en la Figura 103.

```
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/OMG.jpg -k Oculto{
-p -m -os -stg

 _____  _____ _     _       _____  _____
 |     | |     |     | |       |   |     |
 |_____| |_____|_____| |_____  |   |_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/OMG.jpg
 Keyword to search: Oculto{
```

Figura 103 Comandos para el análisis del archivo OMG.jpg con Oculto .

3. Oculto a través de la herramienta jsteg detecta el flag, tal y como lo muestra la Figura 104.

```
**********Analyzing with jsteg**********
Oculto{14MY0URF4TH3R}
 Hidden keyword FOUND
```

Figura 104 Detección de valor oculto con Oculto y la herramienta jsteg.

9.7.8 Reto: Rubber ducky

La Figura 105 muestra la descripción del reto “Rubber ducky” observado por los participantes durante el CTF.

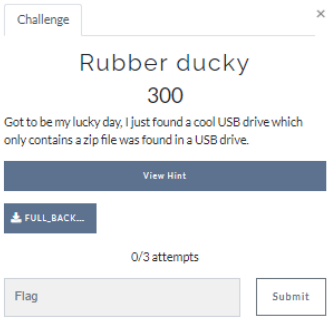


Figura 105 Descripción del reto “Rubber ducky”.

Archivo: FULL_BACKUP.zip

1. Descarga el archivo FULL_BACKUP.zip y extraiga los archivos. En la Figura 106 de observa que una imagen y un archivo de texto fueron extraídos.

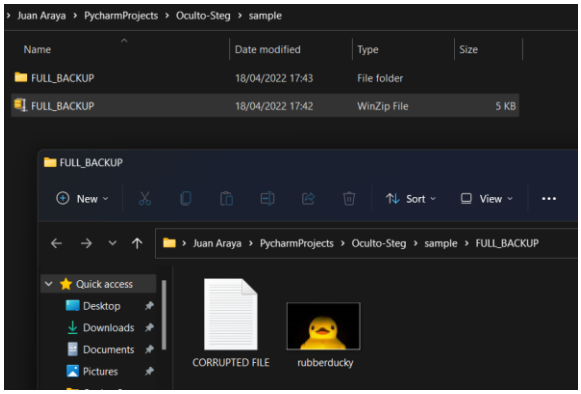


Figura 106 Archivos extraídos de FULL_BACKUP.zip.

2. Según la Figura 107, al abrir el archivo “CORRUPTED FILE.TXT” se descubre el siguiente valor RVZJTENPUlA=.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/FULL_BACKUP]
└─$ ls
'CORRUPTED FILE.txt'   rubberducky.jpg

┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/FULL_BACKUP]
└─$ cat CORRUPTED\ FILE.txt
RVZJTENPUlA=
```

Figura 107 Valor secreto codificado extraído disponible en el archivo CORRUPTED_FILE.txt.

3. La pista o HINT dice “DECODE to extract” por lo que se procede a decodificar RVZJTENPUlA= utilizando *Cyberchef* como se observa en la Figura 108.

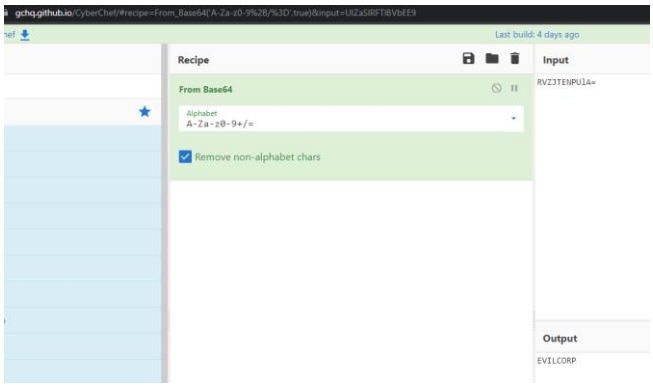


Figura 108 Valor secreto descodificado usando Cyberchef .

4. Se descubre que RVZJTENPUlA= es el valor de EVILCORP codificado en Base64.
5. Según la Figura 109, ahora se procede a analizar la imagen con Oculto

```
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/FULL_BACKUP/rubberd
ucky.jpg -k Oculto{ -p -m -os -stg

  _____  _______ _      _          _______  _____
 |     | |       |     | |        |   |    |     |
 |_____| |_____  |_____| |_____   |   |    |_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/FULL_BACKUP/rubberducky.jpg
 Keyword to search: Oculto{
```

Figura 109 Comandos para el análisis del archivo rubberducky.jpg con Oculto.

6. En la Figura 110 se observa que Oculto, con el uso de *Stegseek*, confirma que hay algo oculto y cifrado con *steghide*.

```
>>Stegoanalysis in process  using Stegseek and bruteforce technique to detect encoded file with steghide
on jpg file /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/FULL_BACKUP/rubberducky.jpg
Executing bruteforce stegoanalysis using StegSeek please wait
 *****Hidden file found using StegSeek
 Found (possible) seed: "8c824cb0"'
```

Figura 110 Detección de archivo oculto dentro de rubberducky.jpg usando Steghide y protegido con contraseña.

7. Se procede a extraer el archivo o datos ocultos con *steghide* utilizando EVILCORP como contraseña. En la Figura 111 se muestra el mensaje oculto que se encontraba en el archivo flag.txt.

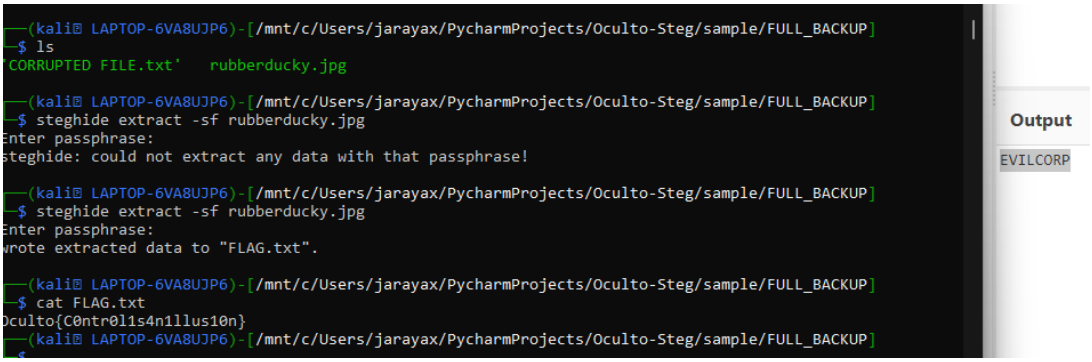


Figura 111 Extracción del archivo oculto dentro de rubberducky.jpg usando Steghide.

9.7.9 Reto: Kitty, Kitty!

La Figura 112 muestra la descripción del reto “Kitty, Kitty!” observado por los participantes durante el CTF.

Figura 112 Descripción del reto “Kitty, Kitty”.

Archivo: CAT.zip

1. Descargar el archivo CAT.zip y extraer el archivo de texto password.txt y CAT.jpg como se observa en la Figura 113.

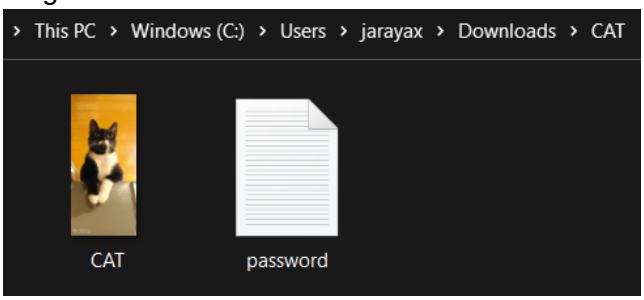


Figura 113 Archivos comprimidos dentro de CAT.zip.

2. Analice el archivo con Oculto con el comando que aparece en la Figura 114.

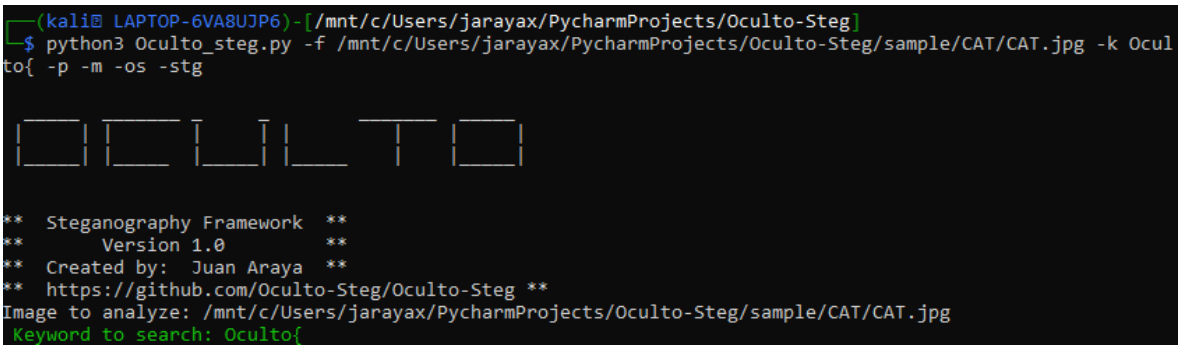


Figura 114 Comandos para el análisis del archivo CAT.jpg con Oculto.

3. Oculto, según se observa en la Figura 115, detecta con el uso de *Stegoveritas* que *steghide* se usó para ocultar algo en la imagen cat.jpg.

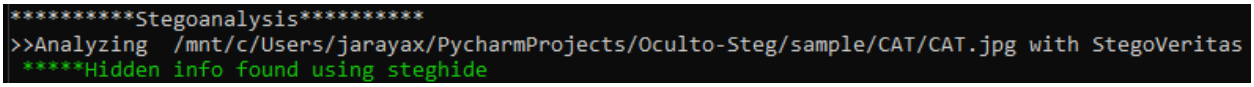


Figura 115 Detección con Oculto y Stegoveritas de uso de steghide para ocultar algo dentro del archivo CAT.jpg.

4. Oculto, con el uso de *Stegseek* confirma el uso de *steghide* según se aprecia en la Figura 116.

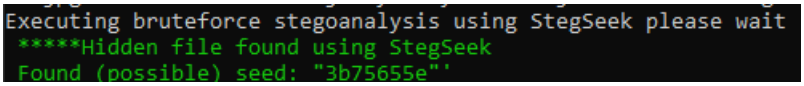


Figura 116 Detección con Oculto y Stegseek de uso de steghide.

5. Abra el archivo password.txt. Note que según la Figura 117, el archivo de texto sólo tiene la palabra “Oculto” en su interior.

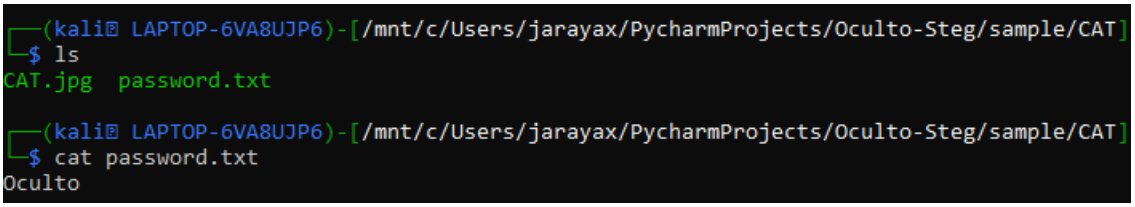


Figura 117 Detección del texto “Oculto” dentro del archivo password.txt.

Figura 128

6. Extraiga el valor oculto de la imagen cat.jpg con *steghide* sin ingresar contraseña. En la Figura 118 se demuestra que con *steghide* la extracción del archivo oculto fue exitosa.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/CAT]
└─$ steghide extract -sf CAT.jpg
Enter passphrase:
wrote extracted data to "URL.txt".

┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/CAT]
└─$ cat URL.txt
https://pastebin.com/Uy8zKmWT
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/CAT]
└─$
```

Figura 118 Extracción del archivo oculto usando steghide en CAT.jpg.

7. Se extraer un archivo URL.txt, al entrar al URL se obtiene el siguiente valor https://pastebin.com/Uy8zKmWT
8. Abra el URL. En la Figura 119 se observan los datos ocultos. Dichos datos parecen estar codificados.

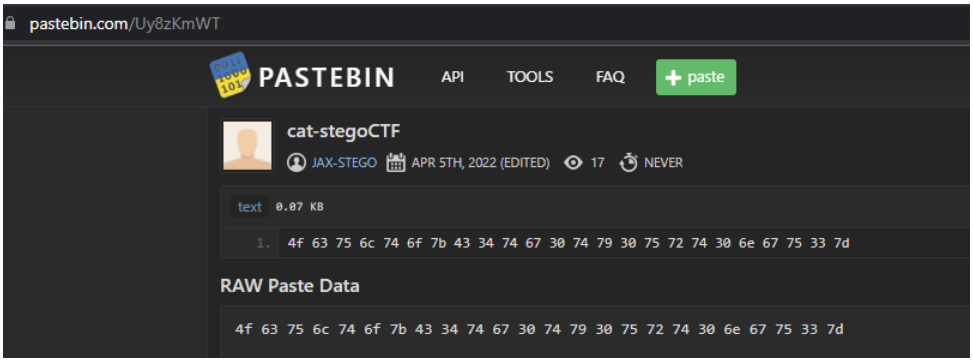


Figura 119 Datos ocultos en un el URL encontrado con steghide.

9. Decodifique el valor encontrado en el URL utilizando *Cyberchef* como se observa en la Figura 120.

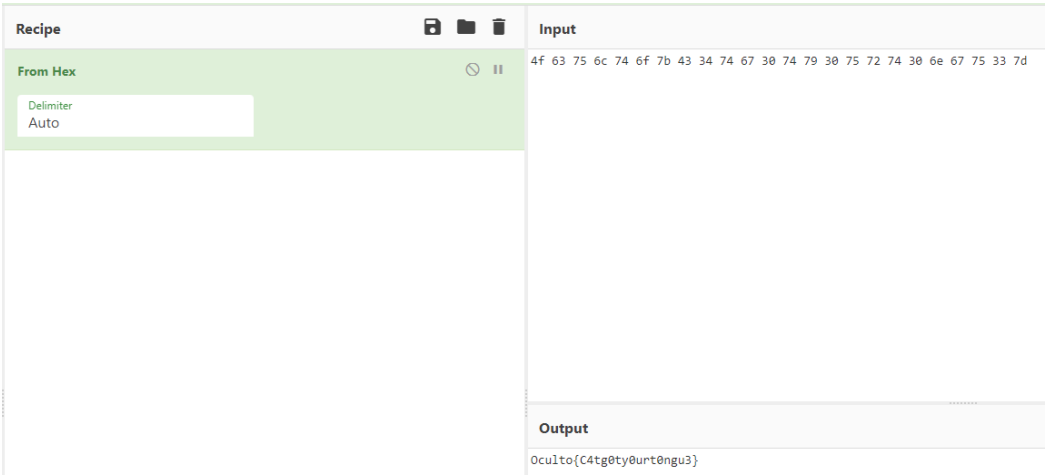


Figura 120 Datos decodificados con Cybercheg.

10. El flag es Oculto{C4tg0ty0urt0ngu3}.

9.7.10 Reto: Surfing at Tamarindo beach!

La Figura 121 muestra la descripción del reto “Surfing at Tamarindo beach!” observado por los participantes durante el CTF.

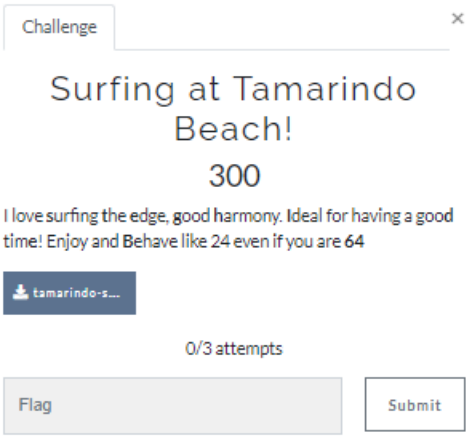


Figura 121 Descripción del reto “Surfing at Tamarindo Beach”.

Archivo: tamarindo-surf.jpg. La vista previa de la imagen se encuentra en la Figura 122.

Figura 122 Vista Previa del archivo tamarindo-surf.jpg.

1. Descargar tamarindo-surf.jpg.
2. En la descripción del reto hay una serie de caracteres en negrita que nos da pistas de que se usó *steghide* y que hay algo codificado en base64.
3. Analice la imagen con Oculto usando el comando que aparece en la Figura 123.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/tamarindo-surf.jpg
-k Oculto{ -p -m -os -stg

 _____ _______ _     _        _______  _____
|     | |       |     | |      |    |  |     |
|_____| |____   |_____| |_____ |    |  |_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/tamarindo-surf.jpg
 Keyword to search: Oculto{
```

Figura 123 Comandos para el análisis del tamarindo-surf.jpg con Oculto.

4. En la Figura 124, se muestra que Oculto detecta hay un valor personalizado en el campo comment del metadata del archivo.

```
*********File Metadata:**********
Extracting File metadata
ExifTool Version Number         : 12.40
File Name                       : tamarindo-surf.jpg
Directory                       : /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample
File Size                       : 37 KiB
File Modification Date/Time     : 2022:04:18 18:35:58+02:00
File Access Date/Time           : 2022:04:18 18:35:58+02:00
File Inode Change Date/Time     : 2022:04:18 18:38:00+02:00
File Permissions                : -rwxrwxrwx
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 72
Y Resolution                    : 72
Exif Byte Order                 : Big-endian (Motorola, MM)
XP Comment                      : cGFzc3dvcmQ=
Padding                         : (Binary data 2060 bytes, use -b option to extract)
```

Figura 124 Dato encontrado en el metadata de la imagen usando Oculto.

5. Convierta ese valor encontrado de BASE64 a ASCII con *Cyberchef* tal y como se observa en la Figura 125.

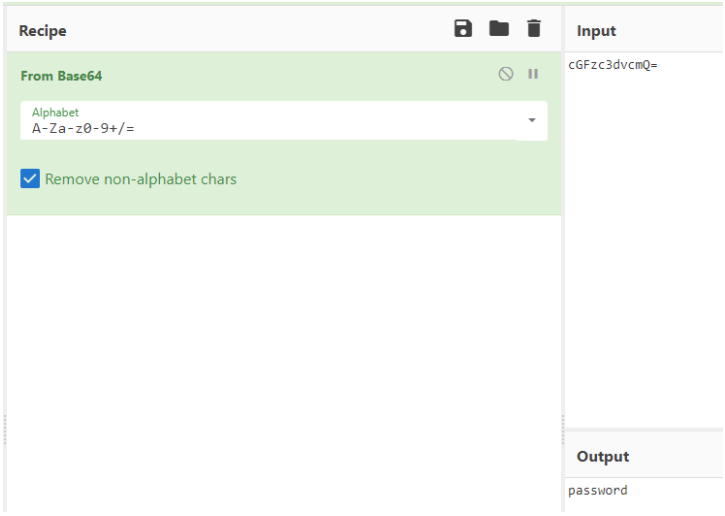


Figura 125 Dato decodificado usando Cyberchef .

6. Según la Figura 126, Oculto, con el uso de *Binwalk* detecta que hay algo oculto en el archivo tamarindo-surf.jpg.

```
>>Searching for embedded file in  /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/tamarindo-surf.
jpg with Binwalk
 *****Embedded file(s) found with Binwalk
b'\nDECIMAL       HEXADECIMAL     DESCRIPTION\n--------------------------------------------------------------
----------------------\n0             0x0             JPEG image data, JFIF standard 1.01\n30
0x1E            TIFF image data, big-endian, offset of first image directory: 8\n\n'
```

Figura 126 Archivo oculto detectado con Oculto y Binwalk.

7. La Figura 127 muestra que Oculto con el uso de *Stegseek* confirma que hay algo oculto con *Steghide*.

```
Executing bruteforce stegoanalysis using StegSeek please wait
 *****Hidden file found using StegSeek
Found (possible) seed: "aa41eba0"'
```

Figura 127 Detección hecha por Ocultoy Stegseek de uso de steghide en la imagen.

8. Extraiga el valor oculto usando *steghide* y “password” como contraseña como se observa en la Figura 128.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ steghide extract -sf tamarindo-surf.jpg -p password
wrote extracted data to "flag.txt".

┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ cat flag.txt
Oculto{4_b4d_d4y_surf1ng_is_b3tt3r_th4n_a_g00d_d4y_w0rk1ng}
```

Figura 128 Extracción de archivo oculto usando steghide.

9.7.11 Reto: Batman!

La Figura 129 muestra la descripción del reto “Batman!” observado por los participantes durante el CTF.

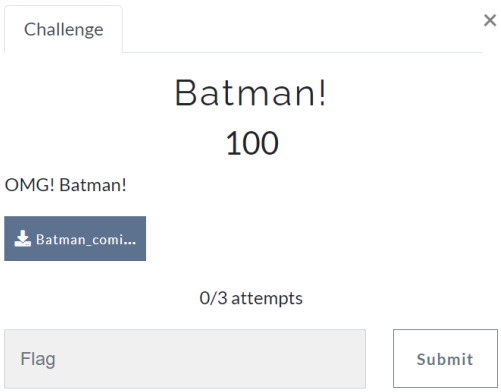


Figura 129 Descripción del reto “Batman”.

Archivo: Batman_comic.jpg. La vista previa de la imagen a analizar se encuentra en la Figura 130.

Figura 130 Vista Previa del archivo Batman_comic.jpg.

1. Descargue el archivo Batman_comic.jpg.
2. Analize el archivo Batman_comic.jpg con Oculto usando el comando que aparece en la Figura 131.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Batman_comic.jpg -k Oculto -p -m -o -stg

 _____ _____ _     _       _____ _____
|     |  ___|  |   | |     |   __|  ___|
|  |  |___  |  |___| |___  |  |  |___  |
|_____|_____|_____|_____|_|_____|_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Batman_comic.jpg
 Keyword to search: Oculto
```

Figura 131 Comandos para el análisis del archivo Batman_comic.jpg con Oculto.

3. Según la Figura 132, Oculto detecta el flag utilizando Jsteg

```
**********Analyzing with jsteg**********
Oculto{H0ly_rust3d_m3t4l,_B4tm4n!}
 Hidden keyword FOUND
```

Figura 132 Detección del valor secreto usando Ocultoy la herramienta jsteg.

9.7.12 Reto: Treat or trick

La Figura 133 muestra la descripción del reto “Treat or trick” observado por los participantes durante el CTF.

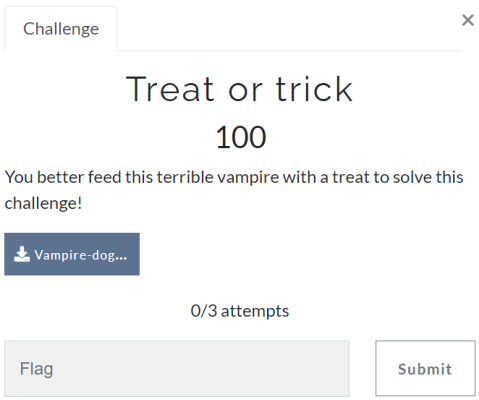


Figura 133 Descripción del reto “Treat or trick”.

Archivo: Vampire-dog.jpg. La vista previa de la imagen se encuentra en la Figura 134.

Figura 134 Vista Previa del archivo Vampire-dog.jpg.

1. Descargue la imagen Vampire-dog.jpg
2. Analize la imagen con Oculto usando el comando que aparece en la Figura 135.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Vampire-dog.jpg -k Oculto{ -p -m -o -stg

 _____ _______ _     _        _______  _____
|     | |       |     | |      |       |     |
|_____| |_____  |_____| |_____ |       |_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Vampire-dog.jpg
 Keyword to search: Oculto{
```

Figura 135 Comandos para el análisis del archivo Vampire-dog.jpg con Oculto.

3. Oculto, con el uso de *Stegseek* detecta el uso de *steghide* para esconder información en la imagen tal y como se observa en la Figura 136.

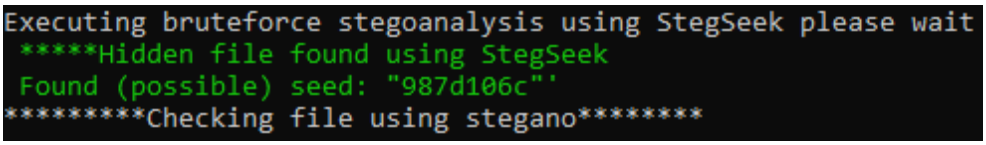


Figura 136 Detección de dato oculto con Oculto y Stegseek.

4. Extraiga el valor o archivo oculto usando *steghide* y "treat" como contraseña. En la Figura 137 se observa que se extrajo un archivo hidden.txt que estaba oculto en la imagen Vampire-dog.jpg.

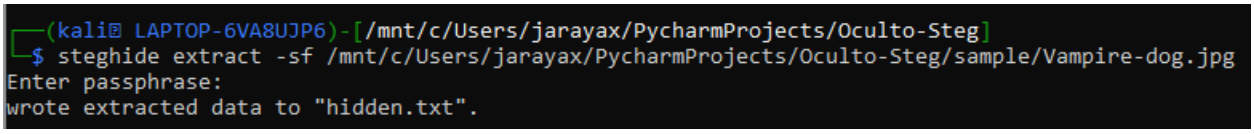


Figura 137 Extracción de dato oculto con steghide.

5. Abra el archivo extraído hidden.txt y busque el flag. En la Figura 138 se muestra en texto plano el mensaje que soluciona el reto.

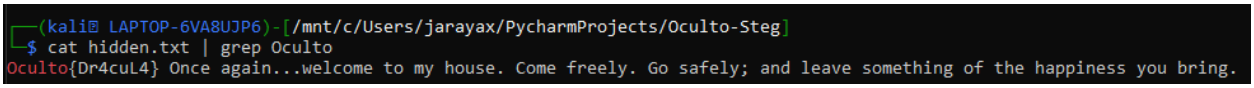


Figura 138 Extracción de valor secreto.

9.7.13 Reto: Turn on the lights!

La Figura 139 muestra la descripción del reto "Turn on the lights!" observado por los participantes durante el CTF.

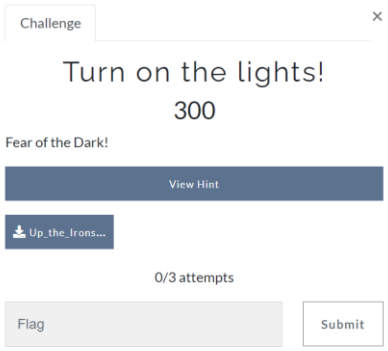


Figura 139 Descripción del reto "Turn on the lights".

Archivo: Up_the_Irons.gif. La vista previa se encuentra en la Figura 140.

Figura 140 Vista Previa del archivo Up_the_Irons.gif.

1. Descargue el archivo Up_the_Irons.gif.
2. Analize el archive con Oculto.
3. Oculto con el uso de *Binwalk* detecta un archivo jpg oculto según se muestra en la Figura 141.

```
*****Embedded file(s) found with Binwalk
b'\nDECIMAL       HEXADECIMAL     DESCRIPTION\n--------------------------------
ata, version "89a", 456 x 640\n2530420       0x269C74        JPEG image data
```

Figura 141 Detección con Oculto y Binwalk de un archivo Oculto dentro de la imagen UP_the_irons.gif.

4. En la Figura 142 se nota como Oculto, con el uso de *Gifshuffle* confirma que en el archivo gif hay algo oculto.

```
*****Embedded data found in GIF file using gifshuffle

 b'\tl\x84\x1a\xd0\x02\xabk\xf0(\x00\xad\xd42\x9e\xb4\xa3\x8c\xe2\x92\xec\xe55^\xaa\x88\x8fi\xda\x92\x93L\xbb\x1b\xbf\xd5<8\xfdz\x1d\x15\x05@\x89\x9d\xa4\xb2\x0f\xfc\xef4
5^j\x9c\xf1"\x91\x94\x8e\x8e(\xcbS\xfe\x8f.y\x17|\x05g\xa6\xdd\xefo\xabN\x85T\\(v\x8eK\xe6\xa6B\xfc\xb0\xfc\xa9\xa4I\x1fQ\xb9\t\x01\x0c\x16\xcb\xd74\x1f\r0\xd5\x11"\xda\x
1fA\x94D\x0e\n\xd3\xefL\xe3\xe4\xd1\xd8\x91\x92\xbd@\xf1\xe2\xe6\xff.m\xbb\xcf\xdc\xa4\x7f\x9e\r\xe8\xe5\x80?\xd0\x87\x86\x19\xdf\xdc\xf9*\xc3)\t\x96\xcaN\xcah\xa9\xb6\xf
5\xb1S\x8e \xc7\x9b\xc9_\xf9\x03\x8boL\xca)F\xf5\x8a\xb0\xfa\xd7\x0e\x95\xdf\xd4T\xc6\xd47\xa2\x18\rY6\xe5\xc7\xd0 Y\x04\x88Uq\xa0'
```

Figura 142 Detección con Ocultoy gifshuffle de un archivo Oculto dentro de la imagen UP_the_irons.gif.

5. Extraiga el archivo jpg de la imagen gif utilizando *binwalk* según se observa en la Figura 143.

```
└─$ binwalk --dd='.*' Up_the_Irons.gif

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             GIF image data, version "89a", 400 x 400
423330        0x675A2         Zip archive data, at least v2.0 to extract, compressed size: 16616, uncompressed size: 22927, name: Ironmaiden.jpg
440050        0x6B6F2         End of Zip archive, footer length: 22
```

Figura 143 Extracción del archivo oculto usando binwalk.

6. Identifique el archivo zip de los archivos extraídos con *binwalk* utilizando el comando file. En la Figura 144 se observa que el archivo 675A2 es un archivo zip que estaba oculto en el archivo gif.

```
└─$ file 675A2
675A2: Zip archive data, at least v2.0 to extract, compression method=deflate
```

Figura 144 Identificación del archizo zip que ha sido extraído usando binwalk.

7. Cambie la extensión del archivo por zip. En la Figura 145 se muestra el comando requerido para realizar el cambio.

Figura 145 Comando para cambio de extensión.

8. Extraiga los archivos contenidos en el archivo zip según se muestra en la Figura 146.

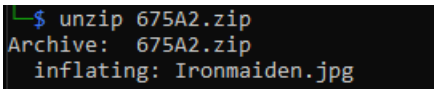


Figura 146 Extracción del archivo comprimido dentro del archivo 675A2.zip.

9. Analice la imagen IronMaiden.jpg extraída con Oculto usando el comando que aparece en la Figura 147.

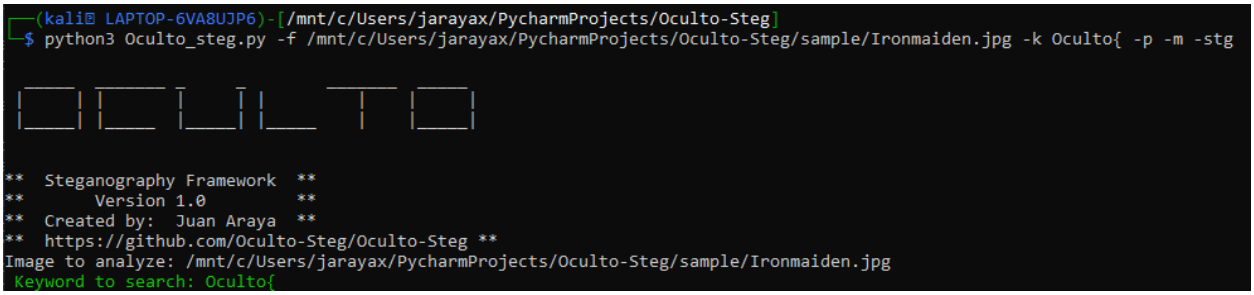


Figura 147 Comandos para el análisis del archivo Ironmaiden.jpg con Oculto.

10. En la Figura 148, se observa un valor en el metadata que podría ser útil para la resolución del reto.

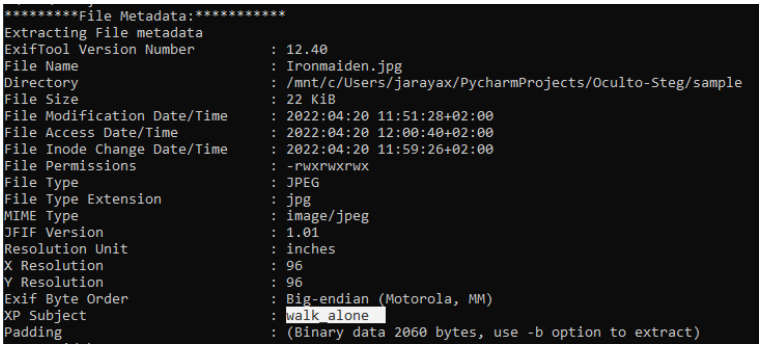


Figura 148 Valor encontrado con Oculto en el metadata.

11. Oculto con el uso de *Binwalk* detecta que hay algo dentro del archivo Ironmaide.jpg según se observa en la Figura 149.

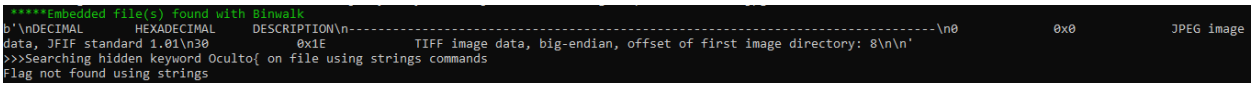


Figura 149 Archivo oculto encontrado con Oculto y Binwalk.

12. Oculto, con la ayuda de *Stegseek* detecta el uso de *steghide* para esconder información en la imagen, según de aprecia en la Figura 150.

```
Executing bruteforce stegoanalysis using StegSeek please wait
 *****Hidden file found using StegSeek
 Found (possible) seed: "14e59245"'
```

Figura 150 Detección de uso de steghide con Oculto y Stegseek.

13. Extraiga el valor oculto con *steghide* usando “walk_alone”(valor que estaba en el metadata) como contraseña. En la Figura 151 se nota la extracción exitosa de un archivo flag.txt que estaba escondido en la imagen.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ steghide extract -sf Ironmaiden.jpg -p walk_alone
wrote extracted data to "flag.txt".
```

Figura 151 Extracción de datos ocultos con steghide.

14. En la Figura 152 se observa el valor oculto en el archivo flag.txt.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ cat flag.txt
Oculto{H4v3_y0u_3v3r_b33n_4l0n3_4t_n1ght}
```

Figura 152 Extracción del valor oculto.

9.7.14 Reto: Parenthood

La Figura 153 muestra la descripción del reto “Parenthood!” observado por los participantes durante el CTF.

Figura 153 Descripción del reto “Parenthood”.

Archivo: Having_fun.png. En la Figura 154 se puede observar la vista previa de la imagen a analizar.

Figura 154 Vista Previa del archivo Having_fun.png.

1. Descargue el archivo Having_Fun.png
2. Analize el archive con Oculto usando el comando disponible en la Figura 155.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Having_fun.png -k Oculto{ -p -m -os -stg

 _____ _______ _     _        _______  _____
|     | |       |     | |      |   |   |     |
|_____| |_____  |_____| |_____ |   |   |_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Having_fun.png
 Keyword to search: Oculto{
```

Figura 155 Comandos para el análisis del archivo Having_fun.jpg con Oculto.

3. Oculto detecta un valor en el metadata según se observa en la Figura 156.

```
*********File Metadata:**********
Extracting File metadata
ExifTool Version Number         : 12.40
File Name                       : Having_fun.png
Directory                       : /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample
File Size                       : 6.4 KiB
File Modification Date/Time     : 2022:04:20 12:34:56+02:00
File Access Date/Time           : 2022:04:20 12:41:26+02:00
File Inode Change Date/Time     : 2022:04:20 12:35:44+02:00
File Permissions                : -rwxrwxrwx
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 178
Image Height                    : 283
Bit Depth                       : 8
Color Type                      : Palette
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Caption-Abstract                : Flag=T2N1bHRve1RydTNfTDBWM30=
Palette                         : (Binary data 144 bytes, use -b option to extract)
Image Size                      : 178x283
Megapixels                      : 0.050
```

Figura 156 Detección de valor secreto en el metadata con Oculto.

4. Oculto con el uso de *Binwalk* detecta algo oculto según se confirma en la Figura 157.

```
******Embedded file(s) found with Binwalk
b'\nDECIMAL       HEXADECIMAL     DESCRIPTION\n--------------------
178 x 283, 8-bit colormap, non-interlaced\n64              0x40
lt compression\n\n'
```

Figura 157 Detección de archivo dentro de la imagen con Oculto.

5. Utilizando masterchef decodifique el valor encontrado en el metadata. En la Figura 158 se confirma que el texto estaba codificado en Base64.

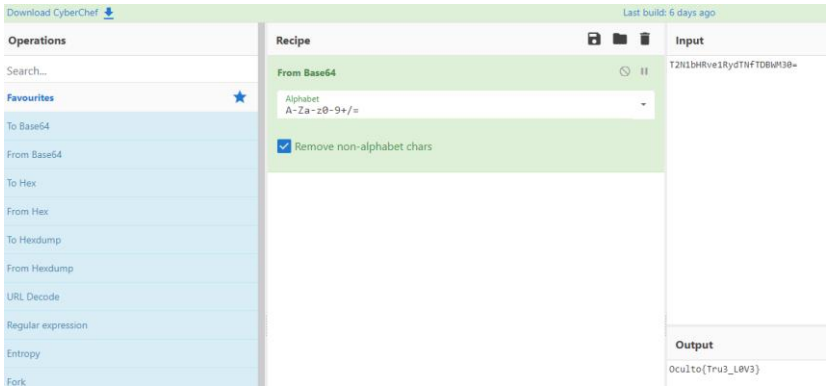


Figura 158 Decodificación de mensaje oculto.

6. La valor oculto es Oculto{Tru3_L0V3}.

9.7.15 Reto: Sixth sense

La Figura 159 muestra la descripción del reto “Sixth sense” observado por los participantes durante el CTF.

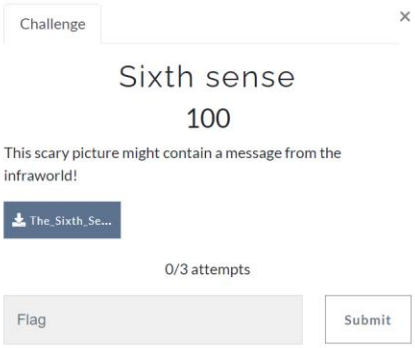


Figura 159 Descripción del reto “Sixth sense”.

Archivo: The_Sixth_Sense_movie.png. La vista previa de la imagen a analizar se muestra en la Figura 160.

Figura 160 Vista Previa del archivo The_Sixth_Sense_movie.png.

1. Descargue el archivo The_Sixth_Sense_movie.png.
2. Analize el archive con Oculto con el domando que aparece en la Figura 161.

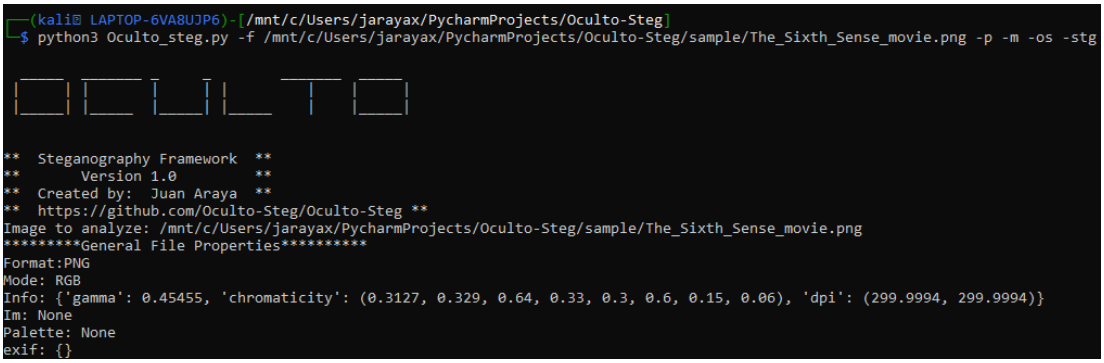


Figura 161 Comandos para el análisis de The_Sixth_Sense_movie.png con Oculto.

3. Oculto con el uso de *Binwalk* detecta un PDF oculto en la imagen según se confirma en la Figura 162.

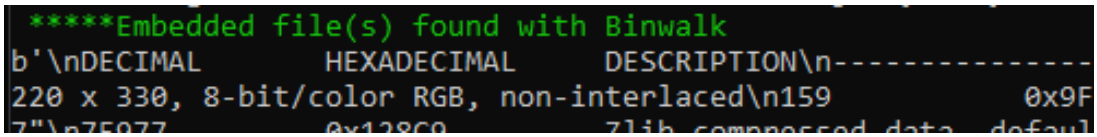


Figura 162 Detección de archivo PDF dentro de la imagen con Oculto y binwalk.

4. Oculto con el uso de *zsteg* detecta y confirma la presencia de un PDF en la imagen como se aprecia en la Figura 163.

```
>>Searching for embedded files or data in  /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/The_Sixth_Sense_movie.png with Zsteg
NO VALUE
zsteg -a /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/The_Sixth_Sense_movie.png
[?] 47367 bytes of extra data after image end (IEND), offset = 0x12839
[?] 47367 bytes of extra data after image end (IEND), offset = 0x12839
extradata:0         .. file: PDF document, version 1.7, 1 pages
    00000000: 25 50 44 46 2d 31 2e 37  0a 0a 34 20 30 20 6f 62  |%PDF-1.7..4 0 ob|
    00000010: 6a 0a 28 49 64 65 6e 74  69 74 79 29 0a 65 6e 64  |j.(Identity).end|
    00000020: 6f 62 6a 0a 35 20 30 20  6f 62 6a 0a 28 41 64 6f  |obj.5 0 obj.(Ado|
    00000030: 62 65 29 0a 65 6e 64 6f  62 6a 0a 38 20 30 20 6f  |be).endobj.8 0 o|
    00000040: 62 6a 0a 3c 3c 0a 2f 46  69 6c 74 65 72 20 2f 46  |bj.<<./Filter /F|
    00000050: 6c 61 74 65 44 65 63 6f  64 65 0a 2f 4c 65 6e 67  |lateDecode./Leng|
    00000060: 74 68 20 34 34 31 34 31  0a 2f 4c 65 6e 67 74 68  |th 44141./Length|
    00000070: 31 20 39 37 32 33 36 0a  2f 54 79 70 65 20 2f 53  |1 97236./Type /S|
    00000080: 74 72 65 61 6d 0a 3e 3e  0a 73 74 72 65 61 6d 0a  |tream.>>.stream.|
    00000090: 78 9c ec 7d 09 7c 54 45  d6 6f d5 bd b7 f7 ee 74  |x..}.|TE.o.....t|
    000000a0: 77 d2 49 77 d2 24 e9 a4  b3 77 92 4e d2 24 21 24  |w.Iw.$...w.N.$!$|
    000000b0: 21 9d 85 10 d6 2c 24 90  20 4b 02 61 93 66 51 40  |!....,$. K.a.fQ@|
    000000c0: 64 51 40 40 30 0c 22 2e  88 82 c0 38 8e 3a ea 40  |dQ@@0."....8.:.@|
    000000d0: 08 8b 01 5c 47 44 1d c5  51 41 1d 67 c6 05 67 18  |...\GD..QA.g..g.|
    000000e0: 75 14 05 47 a3 2c e9 7e  a7 aa 6e 25 9d 08 2e f3  |u..G.,.~..n%....|
```

Figura 163 Detección de archivo PDF dentro de la imagen con Oculto y zsteg.

5. Extraiga el archivo PDF de la imagen. En la Figura 164 se aprecia el listado de archivos extraídos.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ binwalk --dd='.*' The_Sixth_Sense_movie.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 220 x 330, 8-bit/color RGB, non-interlaced
159           0x9F            Zlib compressed data, best compression
75833         0x12839         PDF document, version: "1.7"
75977         0x128C9         Zlib compressed data, default compression
120663        0x1D757         Zlib compressed data, default compression


┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ ls
The_Sixth_Sense_movie.png  _The_Sixth_Sense_movie.png.extracted

┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ cd _The_Sixth_Sense_movie.png.extracted/

┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/_The_Sixth_Sense_movie.png.extracted]
└─$ ls
0  12839  128C9  128C9.zlib  1D757  1D757.zlib  9F  9F-0
```

Figura 164 Extracción de archivos ocultos con binwalk.

6. Identifique el archivo PDF utilizando el comando file. Según la Figura 165, el archivo 123839 es un archivo PDF que estaba oculto en la imagen.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/_The_Sixth_Sense_movie.png.extracted]
└─$ file 12839
12839: PDF document, version 1.7, 1 pages
```

Figura 165 Identificación del archivo PDF.

7. Cambie la extensión a pdf usando el comando que aparece en la Figura 166.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/_The_Sixth_Sense_movie.png.extracted]
└─$ mv 12839 12389.pdf
```

Figura 166 Cambio de extensión a PDF.

8. Abra el archivo pdf para ver el mensaje oculto que se refleja en la Figura 167.

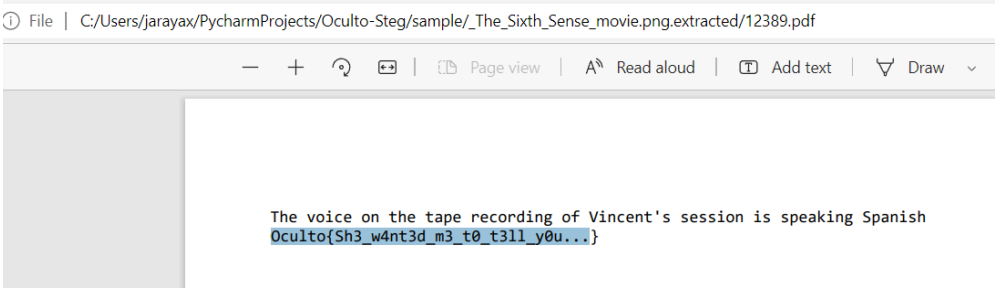


Figura 167 Valor secreto en texto plano presente en el archivo pdf oculto dentro de la imagen.

9.7.16 Reto: IT

La Figura 168 muestra la descripción del reto “IT” observado por los participantes durante el CTF.

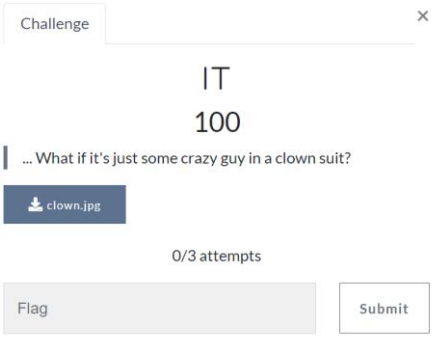


Figura 168 Descripción del reto “Sixth sense”.

Archivo: clown.jpg. La vista previa de la imagen a analizar se encuentra en la Figura 169.

Figura 169 Vista Previa del archivo clown.jpg.

1. Descargue el archivo : clown.jpg.
2. Analize el archive con Oculto con el comando que aparece en la Figura 170.

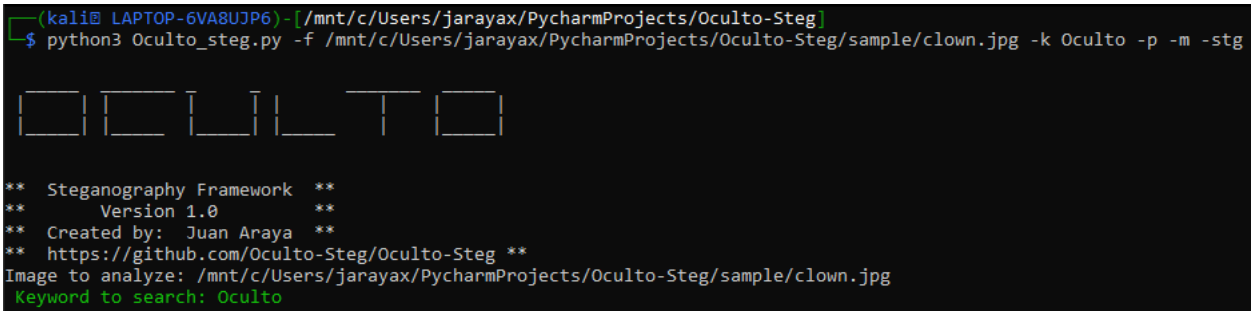


Figura 170 Comandos para el análisis del archivo clown.png con Oculto.

3. En la Figura 171 se observa que Oculto con el uso de *Stegoveritas* detecta información oculta con *steghide*

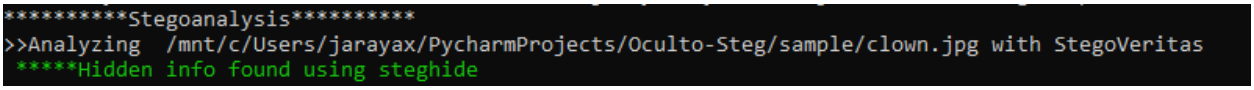


Figura 171 Detección de uso de steghide con Oculto y la herramienta stegoveritas.

4. Oculto detecta el flag utilizando el comando strings según se observa en la Figura 172.

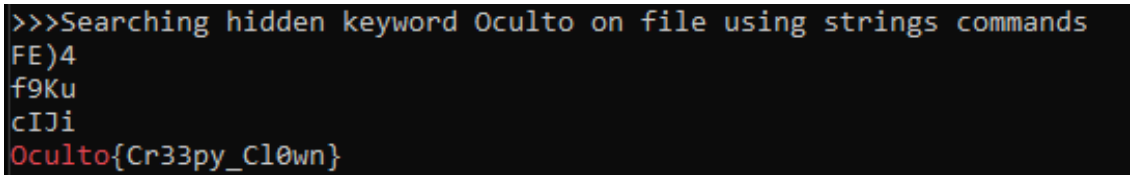


Figura 172 Detección de valor secreto con Oculto y el comando strings.

5. Oculto con el uso de *Stegseek* confirma que hay algo oculto con *steghide* en la imagen, según lo confirma la Figura 173.

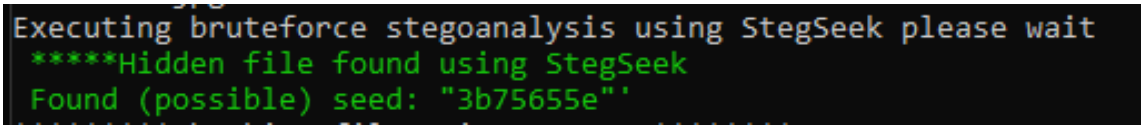


Figura 173 Detección con Oculto y stegseek de uso de steghide con contraseña y cifrado.

6. Para confirmar, extraiga el valor oculto con *steghide* sin ingresar una contraseña dado que strings pudo extraerlo en texto plano. La Figura 174 muestra el valor secreto que estaba en el archivo value.txt.

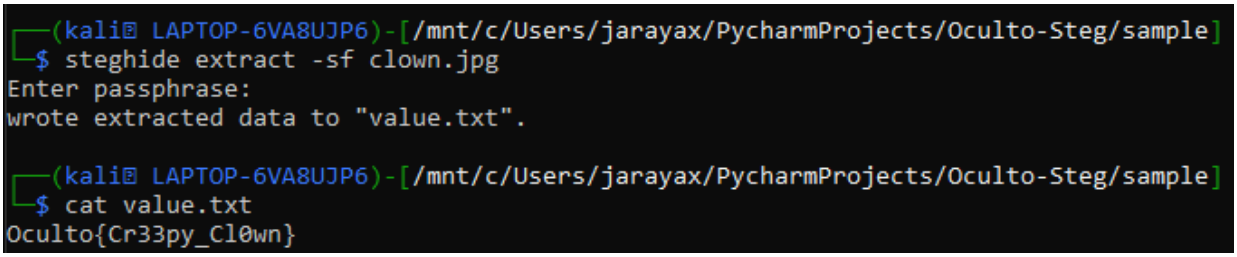


Figura 174 Extracción de archivo oculto con steghide.

9.7.17 Reto: Let the games begin!

La Figura 175 muestra la descripción del reto “Let the games begin!” observado por los participantes durante el CTF.

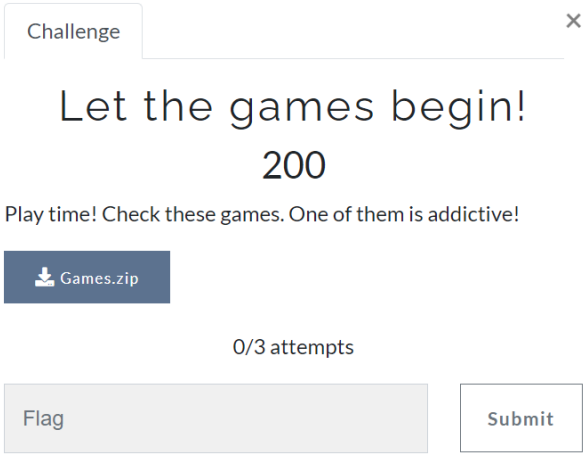


Figura 175 Descripción del reto “Let the games begin”.

Archivo: Games.zip

1. Descargue el archivo: Games.zip
2. Extraiga el contenido del archivo zip. Según la Figura 176, el archivo zip tenía 5 imágenes comprimidas.

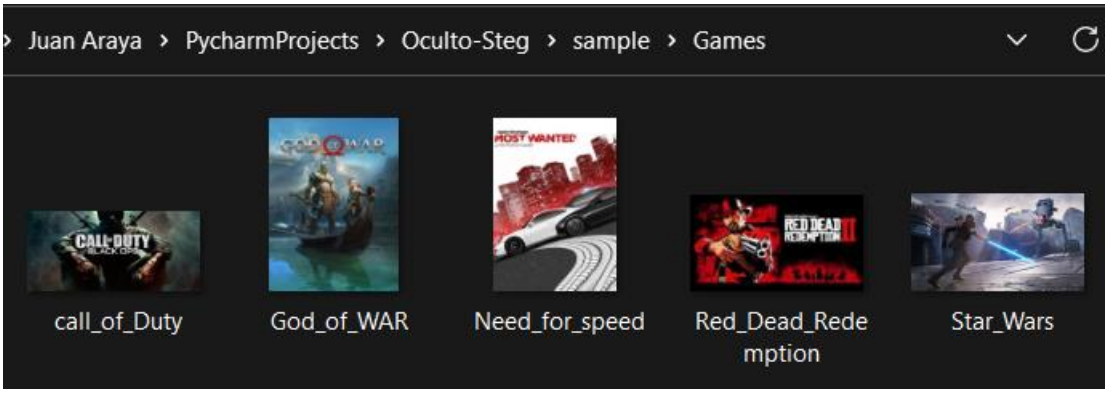


Figura 176 Archivos comprimidos dentro del archivo games.zip.

3. Analice cada imagen con Oculto usando el comando que aparece en la Figura 177.

```
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Games/God_of_WAR.jpg -k Oculto -p -m -stg

**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Games/God_of_WAR.jpg
 Keyword to search: Oculto
```

Figura 177 Comandos para el análisis del archivo clown.png con Oculto.

4. Al analizar la imagen God_of_WAF.jpg, Oculto detecta el uso de *steghide* a través de un análisis con *Stegseek* tal y como se observa en la Figura 178.

```
Executing bruteforce stegoanalysis using StegSeek please wait
 *****Hidden file found using StegSeek
 Found (possible) seed: "a373c74e"'
```

Figura 178 Detección Oultoy stegseek de uso de steghide para ocultar datos o archivos en clown.png.

5. En la Figura 179, se observa que al tratar de extraer la información con *steghide* se detecta el uso de una contraseña

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Games]
└─$ steghide extract -sf God_of_WAR.jpg
Enter passphrase:
steghide: could not extract any data with that passphrase!
```

Figura 179 Intento de extracción de datos o archivo con steghide donde se confirma que se requiere una contraseña.

6. En la imagen God_of_WAF.jpg se notan 2 personajes del juego de video. Haciendo una búsqueda en Google, encuentre el nombre de los personales. En la Figura 180 se identifica el nombre de los personales del videojuego.

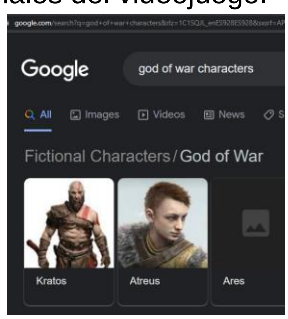


Figura 180 Búsqueda en Google sobre los personajes del juego God of War.

7. Extraiga el valor oculto utilizando Kratos como contraseña. La Figura 181 confirma que había un archivo info.txt oculto dentro de la imagen.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Games]
└─$ steghide extract -sf God_of_WAR.jpg
Enter passphrase:
wrote extracted data to "info.txt".
```

Figura 181 Extracción de archivo oculto con steghide.

8. Como se observa en la imagen 182, abra el archivo info.txt para descubrir el flag.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Games]
└─$ cat info.txt
Oculto{D0_n0t_B3_S0rry._B3_B3tt3r}
```

Figura 182 Extracción del valor secreto encontrado dentro del archivo oculto dentro de la imagen God_of_War.jpg

9.7.18 Reto: Money

La Figura 183 muestra la descripción del reto “Money” observado por los participantes durante el CTF.

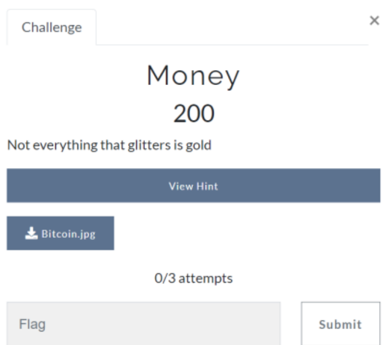


Figura 183 Descripción del reto “Money”.

Archivo: Bitcoin.jpg. La vista previa de la imagen aparece en la Figura 184.

Figura 184 Vista Previa del archivo bitcoin.jpg.

1. Descargue el archivo: Bitcoin.jpg.
2. Analice la imagen con Oculto con el comando que aparece en la Figura 185.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Bitcoin.jpg -p -m -stg

**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Bitcoin.jpg
```

Figura 185 Comandos para el análisis del archivo bitcoin.jpg con Oculto.

3. Oculto detecta un URL en el metadata según se observa en la Figura 186.

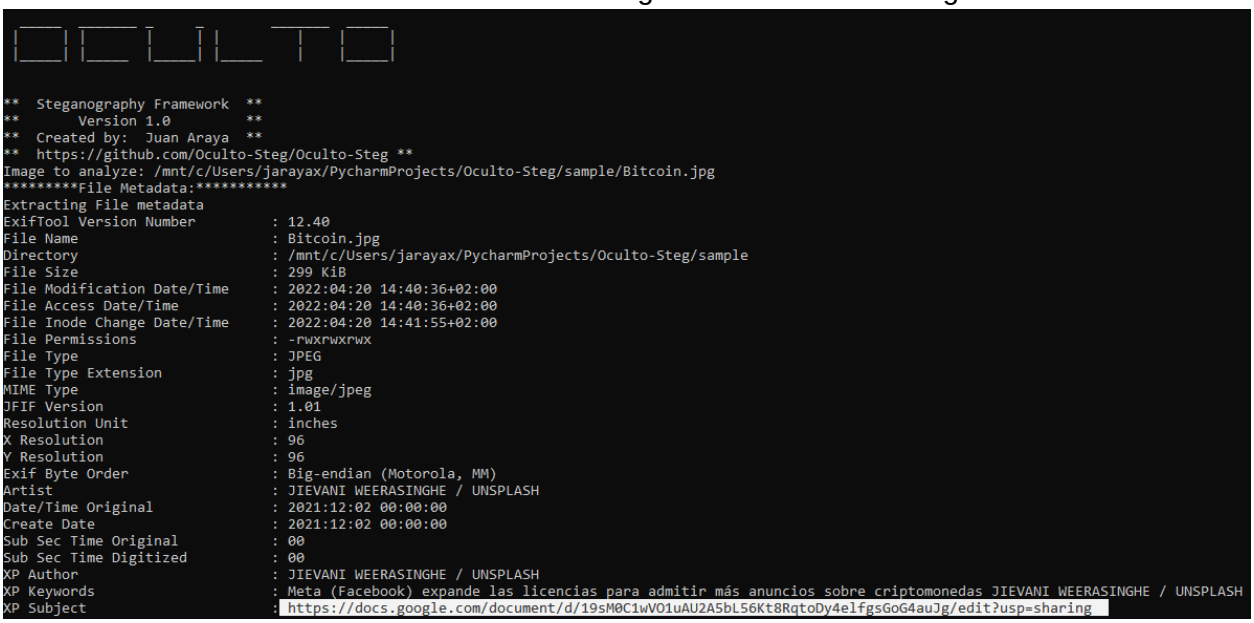


Figura 186 URL encontrada en el metadata utilizando Oculto.

4. Oculto detecta un archivo oculto utilizando *binwalk*. La Figura 187 confirma la existencia de un archivo oculto.

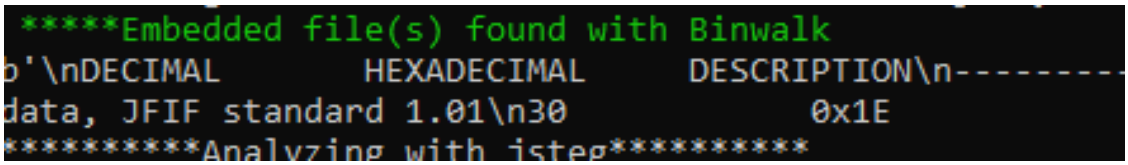


Figura 187 Detección de archivo dentro de bitcoin.jpg Oculto y binwalk.

5. Abra el URL encontrado por Oculto en el metadata. En la Figura 188 se observa un valor codificado al entrar en el URL.

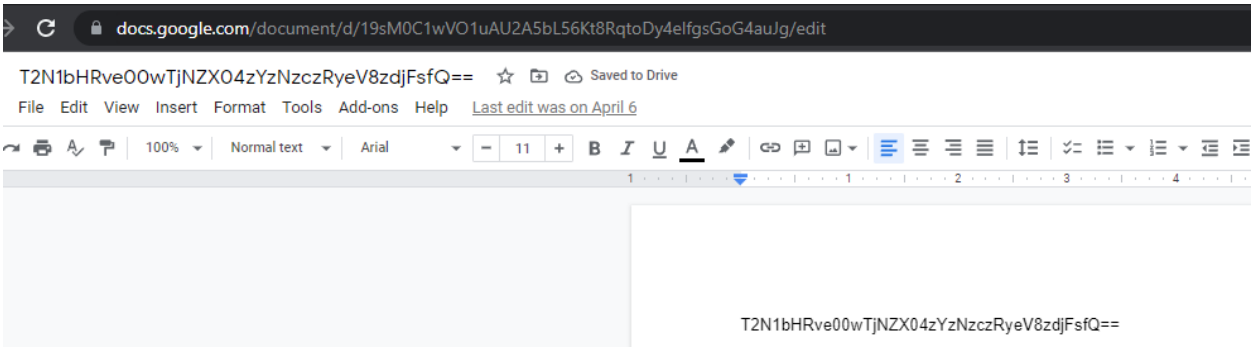


Figura 188 Valor codificado encontrado en el URL.

6. Decodifique el valor encontrado con *Cyberchef* para obtener el flag como se muestra en la Figura 189.

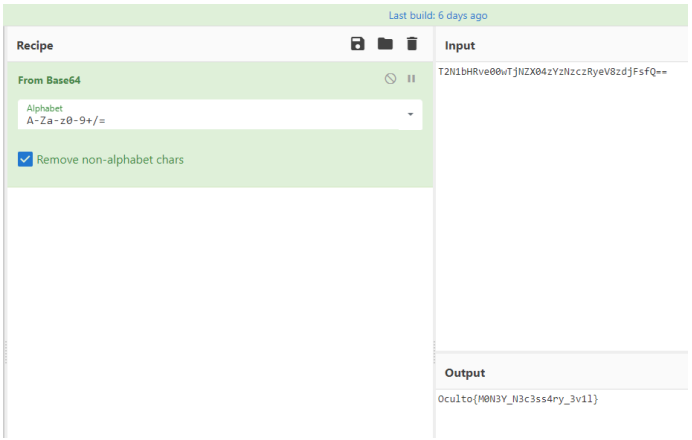


Figura 189 Valor descodificado encontrado en el URL utilizando Cyberchef .

9.7.19 Reto: NTF

La Figura 190 muestra la descripción del reto “NFT” observado por los participantes durante el CTF.

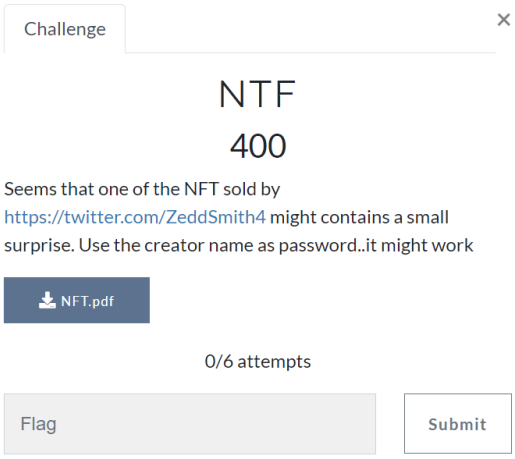


Figura 190 Descripción del reto “NFT”.

Archivo: NFT.pdf

1. Descargue el archivo NFT.pdf
2. Abra el archivo NFT.pdf. Según la Figura 191, hay 2 un contract ID y un token ID en el archivo PDF.

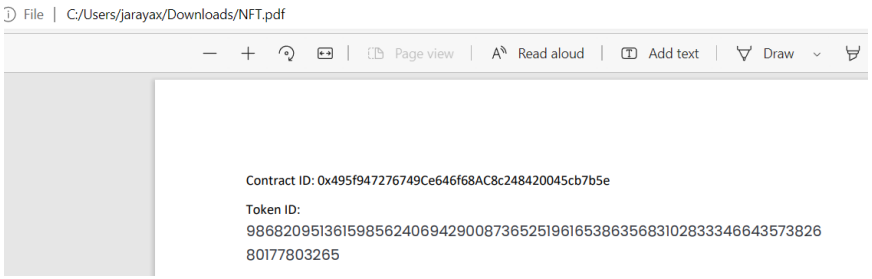


Figura 191 Contract ID y Token ID encontrados en el archivo NFT.pdf.

3. Analice el NFT con Oculto, ingresando el Contract ID y el Token ID que aparecen en el PDF con el domando que aparece en la Figura 192.

```
└─$ python3 Oculto_steg.py -nft
Token ID:
98682095136159856240694290087365251961653863568310283334664357382680177803265
Contract_ID:
0x495f947276749Ce646f68AC8c248420045cb7b5e
{'VT': 'deea7981acdff0bd73461de858efcdde18e39aca0156710ef7f2cf4474efe714', 'NFT': 'ac6a717c-cfc3-4ef1-91e2-3595b86afe32'}
**********Analyzing NFT**********
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1277  100  1277    0     0   3443      0 --:--:-- --:--:-- --:--:--  3451
NFT details
Response:OK
Chain:ethereum
Metadata_url:https://api.opensea.io/api/v1/metadata/0x495f947276749Ce646f68AC8c248420045cb7b5e/98682095136159856240694290087365251961653863568310283334664357382680177803265
Mind date:None
Updated date:2022-04-20T13:06:41.043900
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   250  100   250    0     0    641      0 --:--:-- --:--:-- --:--:--   641
**********Extracting NFT's Metadata **********
{'name': 'Stego_NFT_1', 'description': 'You have found a {Flag} ', 'external_link': None, 'image': 'https://lh3.googleusercontent.com/0rRNiwcuQtN7C9KgCD-mAQmNdfELw7wSsJKRMuOMDkJafOie1GtjPNhNAkPTVLmrj32OqQb4gCQE_Y_Q1U0CgG-vElGbehrw3lux', 'animation_url': None}
NAME: Stego_NFT_1
DESCRIPTION: You have found a {Flag}
EXTERNAL_LINK: None
IMAGE: https://lh3.googleusercontent.com/0rRNiwcuQtN7C9KgCD-mAQmNdfELw7wSsJKRMuOMDkJafOie1GtjPNhNAkPTVLmrj32OqQb4gCQE_Y_Q1U0CgG-vElGbehrw3lux
ANIMATION_URL: None
```

Figura 192 Análisis con Oculto de un NFT con base al Contract ID y Token ID encontrados en el archivo NFT.pdf.

4. Descargue la vista previa de la imagen que aparece en el metadata del NFT. En la Figura 193 se observa la imagen encontrada.

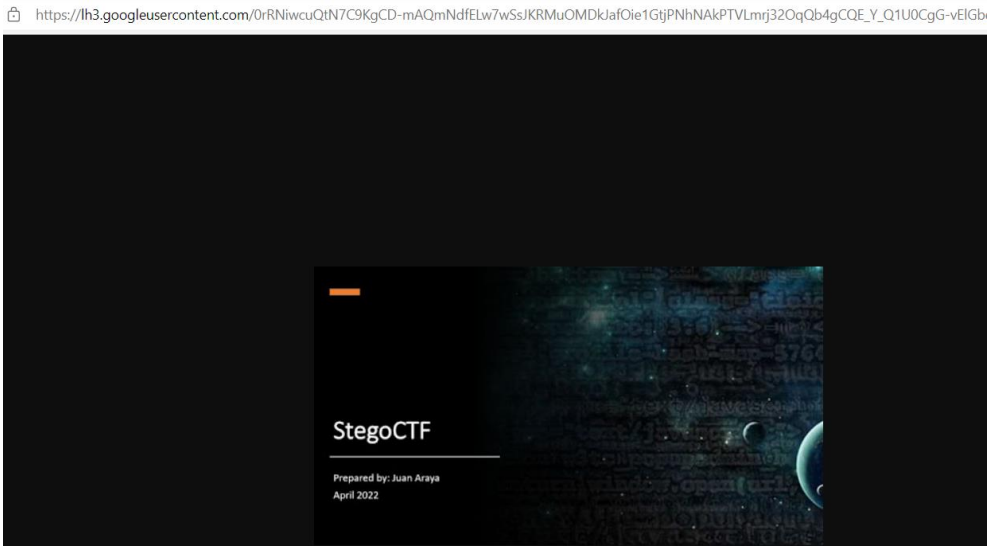


Figura 193 Vista previa de la imagen encontrada en el metadata usando Oculto.

5. Abra el hipervínculo que aparece en la descripción del reto https://twitter.com/ZeddSmith4

En la Figura 194 se observa el URL que aparece en la cuenta de twitter mencionada.

Figura 194 Perfil de twitter encontrado en la descripción del reto.

6. Abra el link de Opensea que aparece en el perfil de twitter encontrado. En la Figura 195 se observan las imágenes creadas por un usuario en particular.

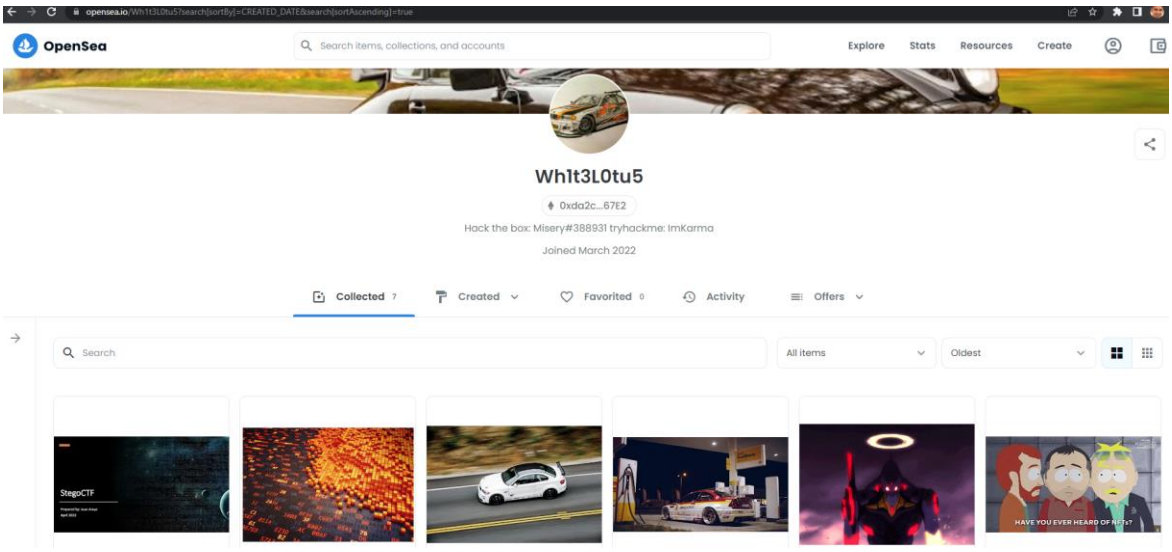


Figura 195 Imágenes a la venta del usuario Wh1t3L0tu5.

7. Abra la imagen que coincida con la que se descargó en el paso 4 y confirme que el token ID y el contract ID coincida con los valores encontrados en el PDF. En la Figura 196 se confirma que los valores del contract ID y token ID son los mismos que aparecen en el PDF que se ha analizado previamente.

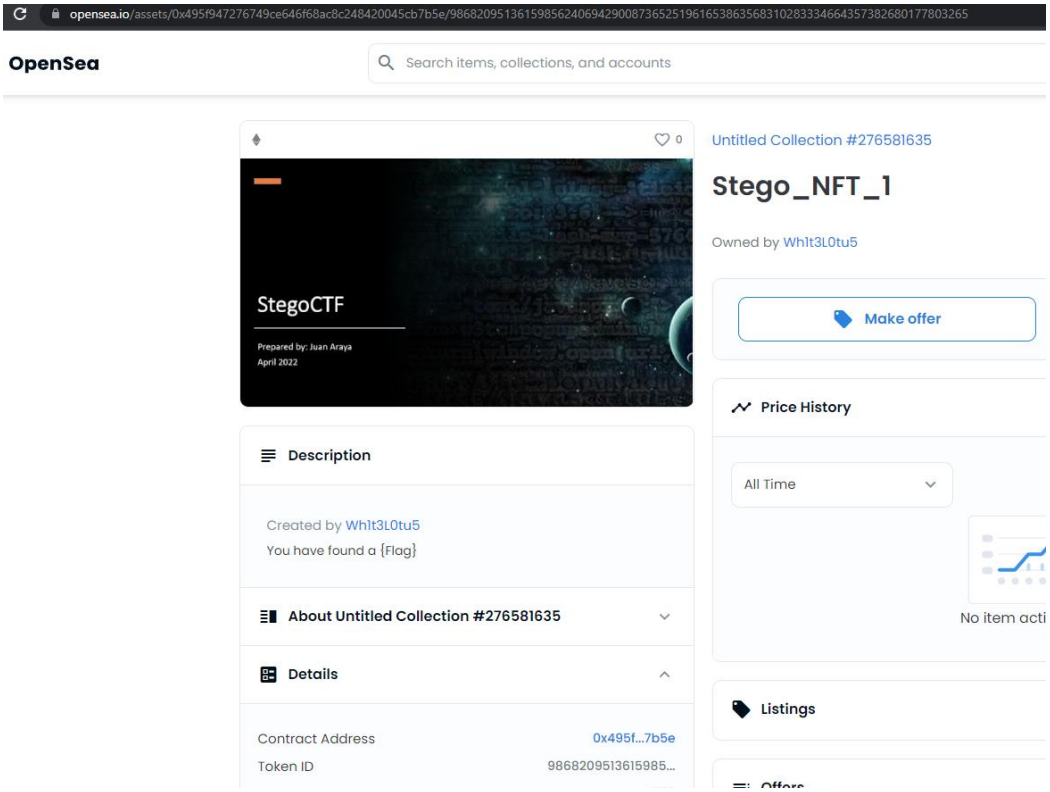


Figura 196 Imagen a la venta del usuario Wh1t3L0tu5 que coincide con el contract Id y Token ID encontrado en el PDF.

8. Haga click en la imagen y descargue la imagen como se observa en la Figura 197.

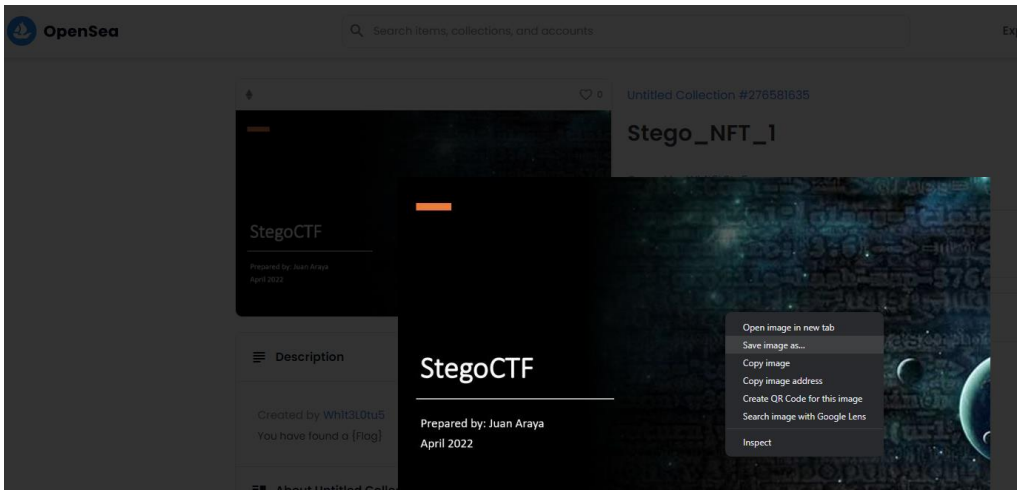


Figura 197 Descarga de imagen a la venta del usuario Wh1t3L0tu5 que coincide con el contract Id y Token ID encontrado en el PDF.

9. Analice el archivo descargado con Oculto usando el comando que aparece en la Figura 198.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/NFT_encontrado.jpg -p -m -stg

 _____  _______ _     _       _______  _____
|     | |       |     | |         |    |     |
|_____| |_____  |_____| |_____    |    |_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/NFT_encontrado.jpg
```

Figura 198 Comandos para el análisis del archivo NFT_encontrado.jpg con Oculto.

10. Oculto detecta que la imagen tiene algo oculto y cifrado con *steghide* según se aprecia en la Figura 199.

```
Executing bruteforce stegoanalysis using StegSeek please wait
 *****Hidden file found using StegSeek
 Found (possible) seed: "75cbd7e3"'
```

Figura 199 Detección con Oculto y stegseek de uso de steghide.

11. Según la descripción del reto, para extraer el mensaje use el nombre del creador del NFT como contraseña. En este caso es Wh1t3L0tu5 según se nota en la Figura 200.

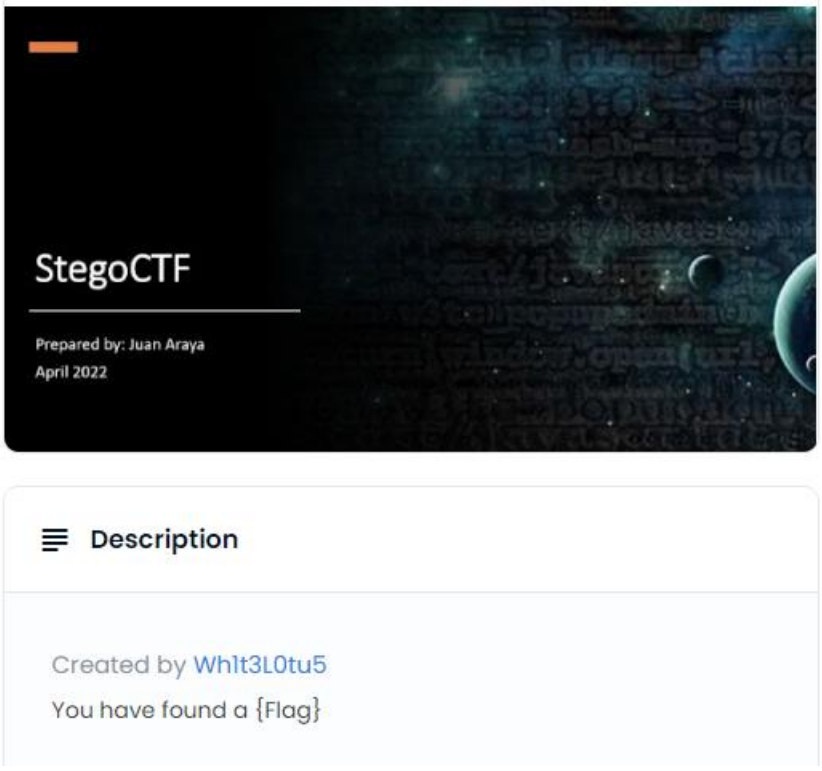


Figura 200 Detección del nombre del creador de la imagen disponible a la venta.

12. Extraiga el mensaje con *steghide* y con la clave Wh1t3L0tu5 como aparece en la Figura 201.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ steghide extract -sf NFT_encontrado.jpg -p Wh1t3L0tu5
wrote extracted data to "flag.txt".

┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ cat flag.txt
Oculto{L0s_NTFs_pueden_traer_S0RPR3S4}
```

Figura 201 Extracción del archivo con steghide.

9.7.20 Reto: Hasta la vista

La Figura 202 muestra la descripción del reto “Hasta la vista” observado por los participantes durante el CTF.

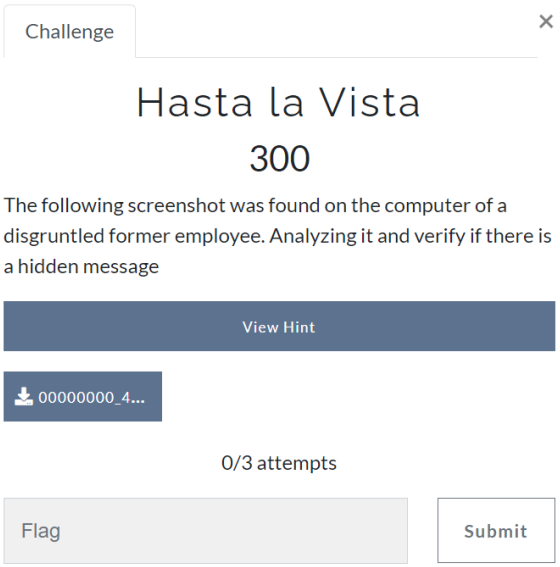


Figura 202 Descripción del reto “Hasta la Vista”.

Archivo: 00000000_4f_63_75_6c_74_6f_43_54_46.zip

1. Descargue y extraiga los archivos comprimidos dentro de 00000000_4f_63_75_6c_74_6f_43_54_46.zip
2. El archivo zip está protegido con contraseña como se observa en la Figura 203.

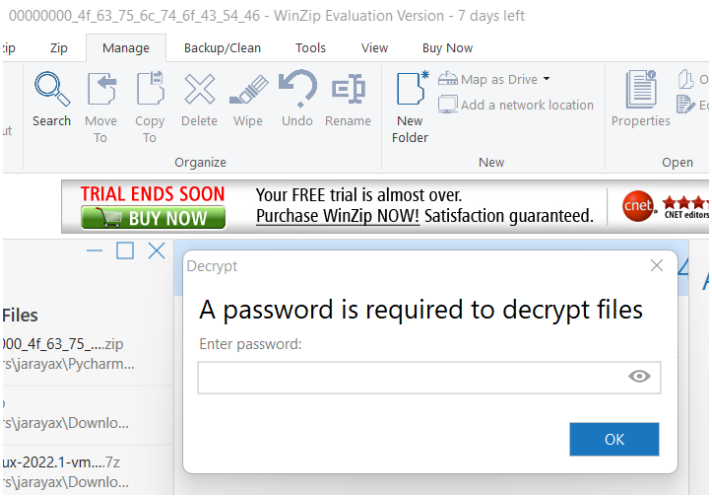


Figura 203 Detección de contraseña requerida para extraer los archivos comprimidos.

3. Copie el nombre del archivo zip y decodifíquelo con *Cyberchef* con base a la Figura 204.

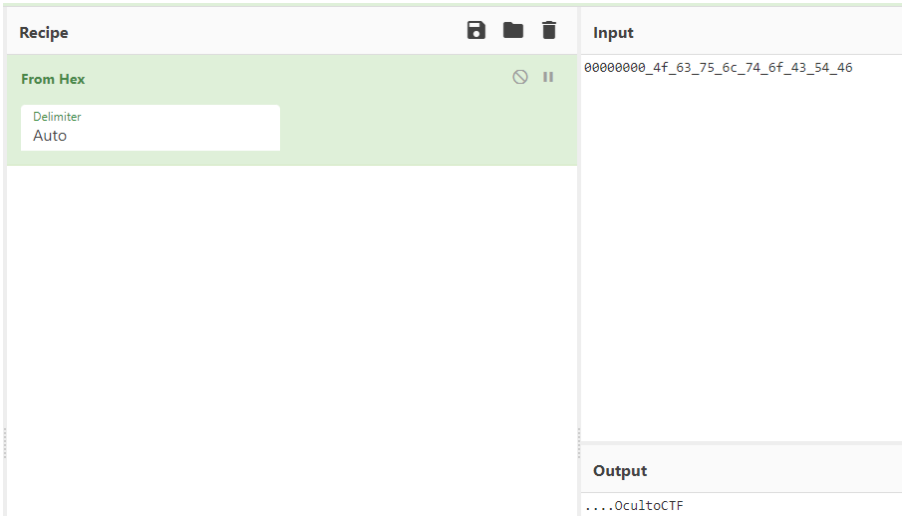


Figura 204 Descodificación del nombre del archivo de Hex a ASCII.

4. Extraiga los archivos comprimidos usando OcultoCTF como contraseña.
5. Este contiene una imagen Screenshot.png según se confirma en la Figura 205.

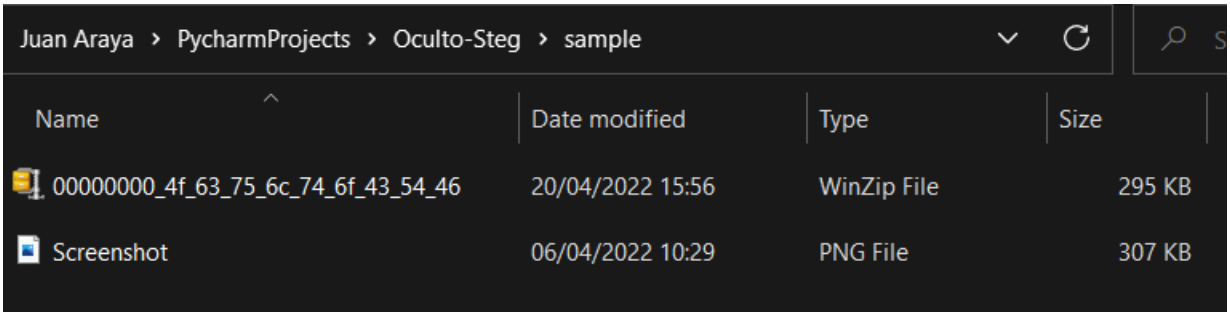


Figura 205 Extracción de archivos de 00000000_4f_63_75_6c_74_6f_43_54_46.zip.

La Figura 206 muestra una vista previa de la Screenshot.png.

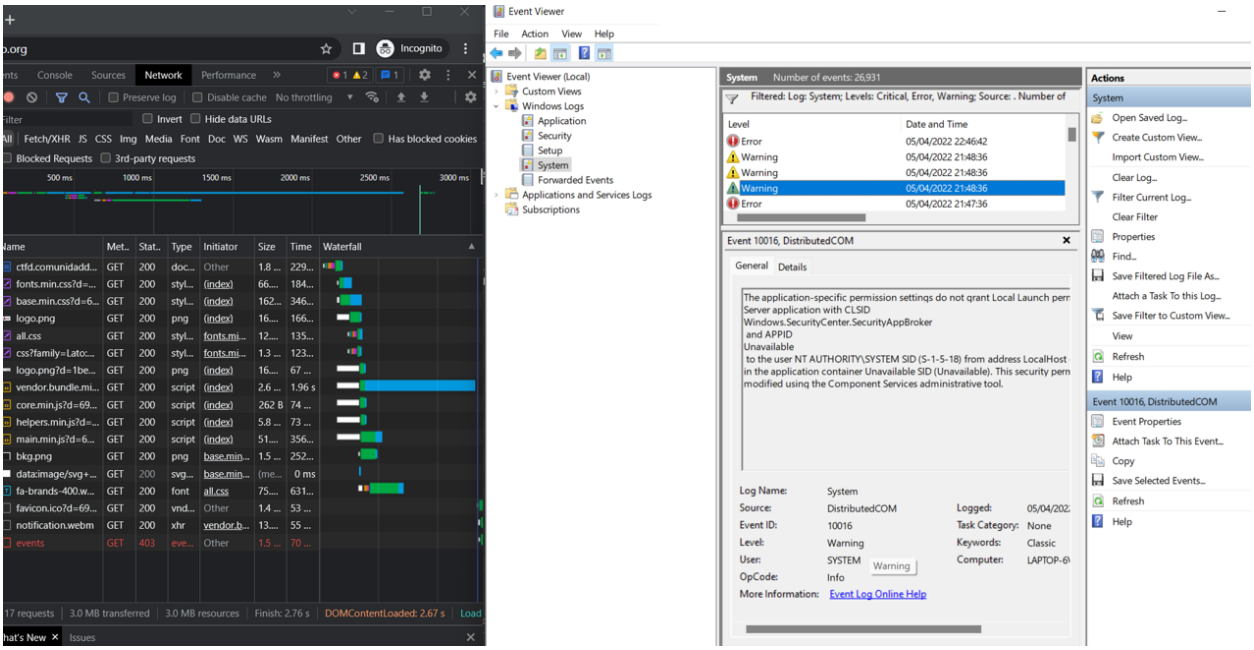


Figura 206 Vista previa del archivo Screenshot.png.

6. Analice el archivo screenshot.png con Oculto usando el comando que aparece en la Figura 207.

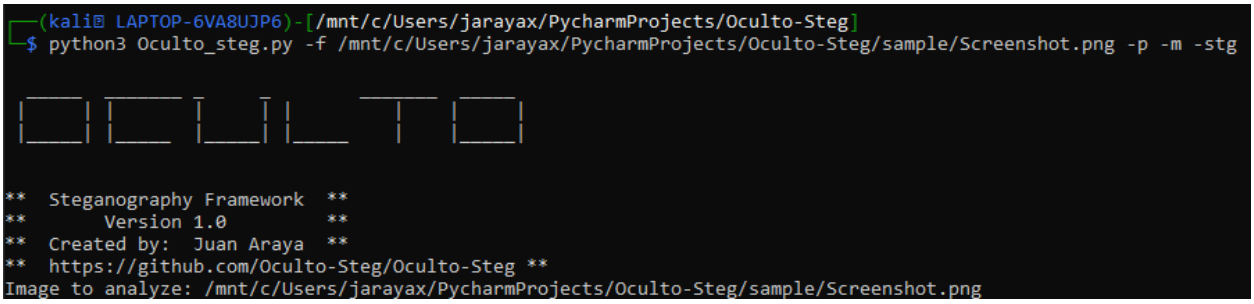


Figura 207 Comandos para el análisis del archivo Screenshot.png con Oculto.

7. Oculto , con el uso de *Binwalk*, detecta algo escondido como se observa en la Figura 208.

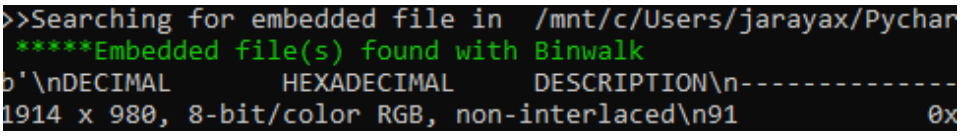


Figura 208 Detección con Oculto y Binwalk de archivo dentro de Screenshot.png.

8. Y Oculto con el uso de Openstego extrae el archivo oculto, el cual contiene el valor buscado según la Figura 209.

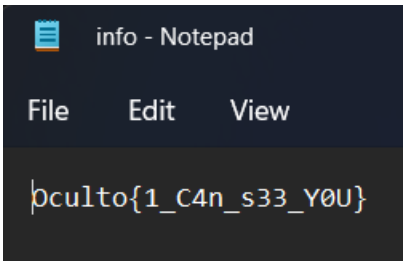


Figura 209 Extracción de valor secreto con Oculto y Openstego.

Figura 212

9.7.21 Reto: Protect the planet

La Figura 210 muestra la descripción del reto “Protect the planet” observado por los participantes durante el CTF.

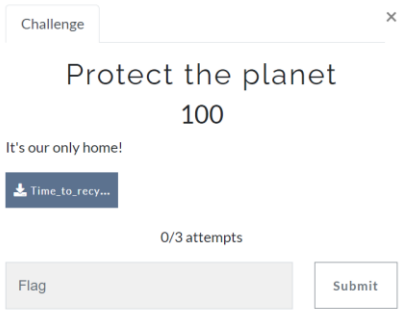


Figura 210 Descripción del reto “Protect the planet”.

Archivo: Time_to_recycle.png. La vista previa de la imagen a analizar aparece en la Figura 211.

Figura 211 Vista Previa de la imagen Time_to_recycle.png

1. Descargue la imagen Time_to_recycle.png.
2. Analice la imagen con Oculto con el comando que aparece en la Figura 212.

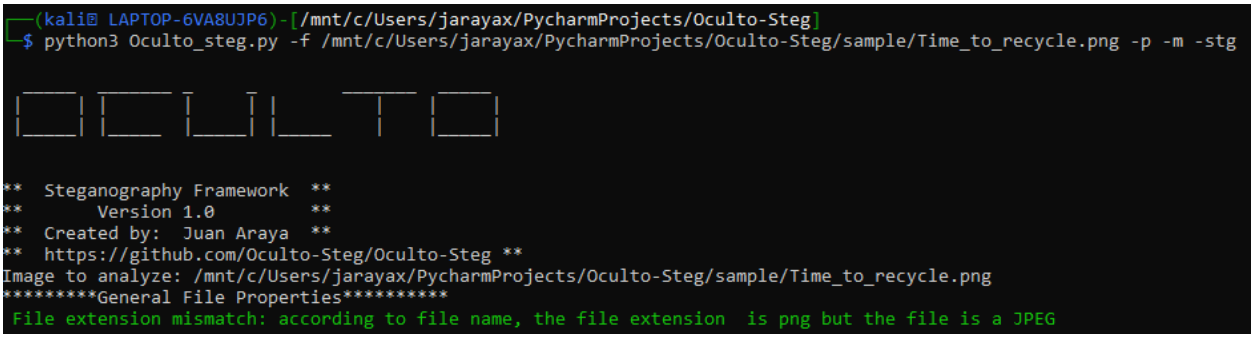


Figura 212 Comandos para el análisis del archivo Time_to_recycle.png con Oculto.

3. Oculto detecta que el archivo no es un png sino un JPEG.
4. Según la Figura 213, Oculto detecta la presencia de un archivo PDF dentro de la imagen.

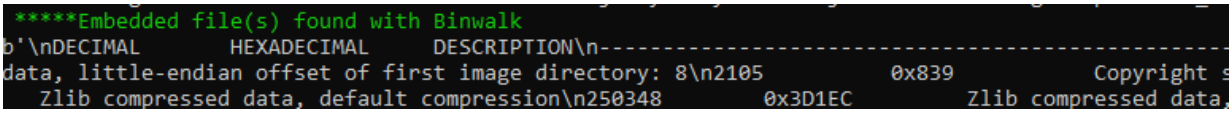


Figura 213 Detección con Oculto y Binwalk de archivo dentro de Time_to_recycle.png.

5. Extraiga la imagen utilizando *binwalk* como se muestra en la Figura 214.

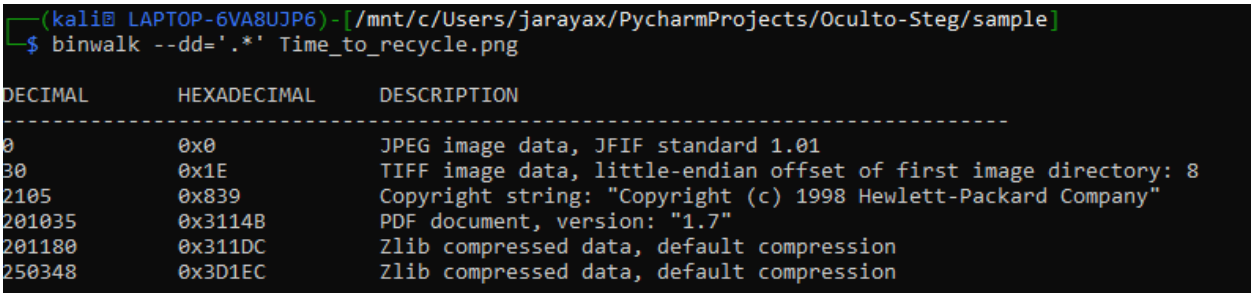


Figura 214 Extracción de datos ocultos con binwalk.

6. Abra el folder _Time_to_recycle.png.extracted.
7. Identifique el archivo PDF. Según la Figura 215, el archivo PDF es el 3114B.

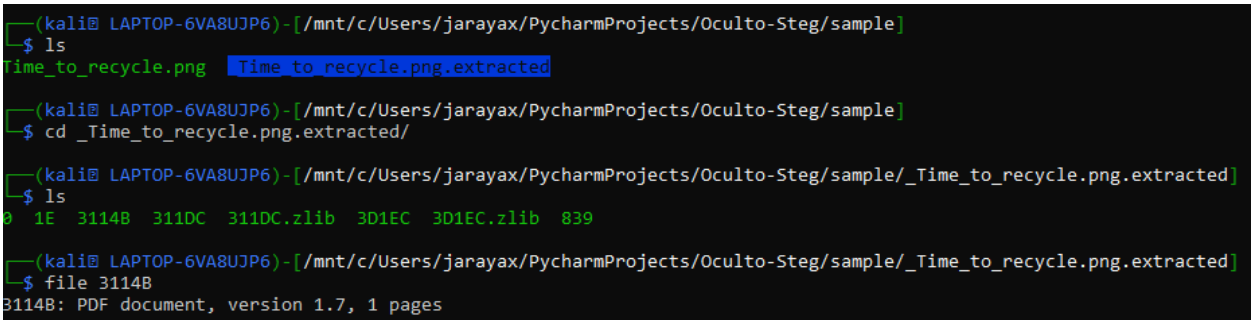


Figura 215 Identificación del archivo PDF oculto dentro de la imagen.

8. Cambie la extensión a PDF al archivo 3114B con el comando mv que aparece en la Figura 216.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/_Time_to_recycle.png.extracted]
└─$ mv 3114B 3114B.pdf
```

Figura 216 Cambio de extensión del archivo extraído por pdf.

9. Abra el archivo PDF. La Figura 217 muestra el valor secreto para resolver el reto.

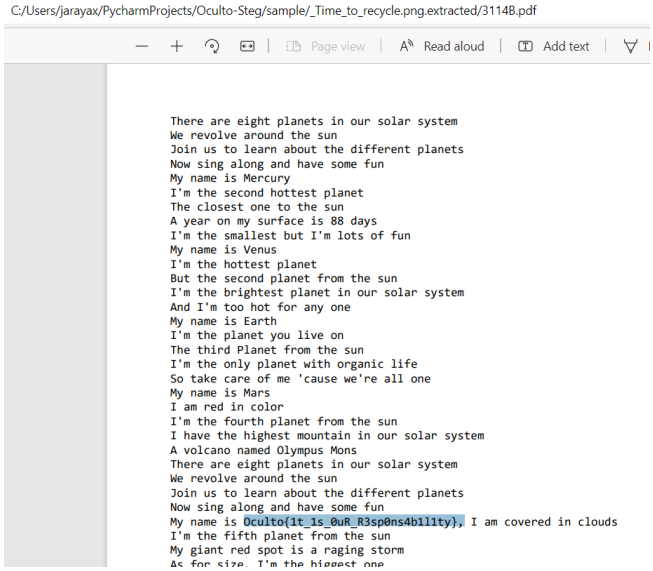


Figura 217 Valor secreto encontrado en el PDF.

9.7.22 Reto: Coin collector

La Figura 218 muestra la descripción del reto “Coin collector” observado por los participantes durante el CTF.

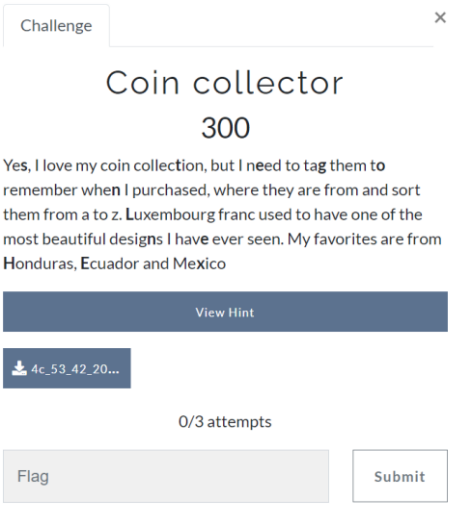


Figura 218 Descripción del reto “Coin Collector”.

Archivo: 4c_53_42_20_52_33_20_47_35_20_42_31.png. La vista previa de la imagen a analizar se muestra en la Figura 219.

Figura 219 Vista previa de la imagen 4c_53_42_20_52_33_20_47_35_20_42_31.png.

1. Descargue la imagen 4c_53_42_20_52_33_20_47_35_20_42_31.png
2. Analice la imagen con Oculto con el comando que aparece en la Figura 220.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/4c_53_42_20_52_33_20_47_35_20_42_31.png -p -m -stg

 ____  ____  _     _      ____  ____
|  _ \|  _ \| |   | |    |  _ \|  _ \
| |_) | |_) | |___| |___ | |_) | |_) |
|____/|____/|_____|_____||____/|____/


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/4c_53_42_20_52_33_20_47_35_20_42_31.png
```

Figura 220 Comandos para el análisis del archivo 4c_53_42_20_52_33_20_47_35_20_42_31.png con Oculto.

3. Oculto detecta la presencia de un archivo dentro de la imagen como se observa en la Figura 221.

```
*****Embedded file(s) found with Binwalk
b'\nDECIMAL       HEXADECIMAL     DESCRIPTION\n----------------
        Zlib compressed data, compressed\n\n'
>>Searching for embedded files or data in  /mnt/c/Users/jarayax/
```

Figura 221 Detección con Oculto y Binwalk de archivo dentro de la imagen.

4. Sin embargo, el mismo no se puede extraer con *binwalk* por la herramienta utilizada para ocultar la información.
5. Como se observa en la Figura 222, la descripción del reto tiene una serie de caracteres en negrita que nos da una pista

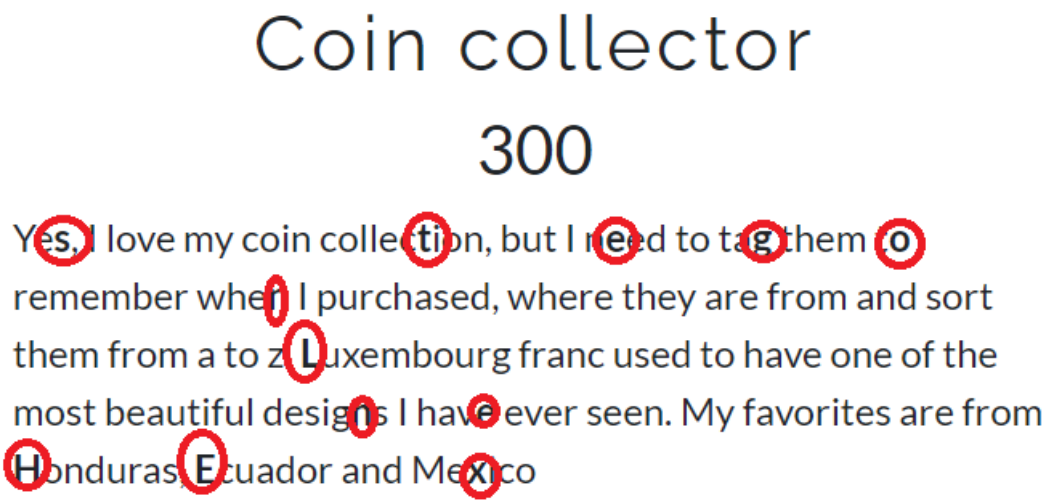


Figura 222 Pistas encontradas en la descripción del reto

6. Del cual se extrae la frase “stegonline HEX”
7. Esto nos indica que posiblemente con stegonline se pueda abrir y que hay algún valor en Hexadecimal que se debe convertir
8. Se procede a abrir la imagen con stegonline (https://stegonline.georgeom.net/). La Figura 223, muestra el menú para extraer información de stegonline.

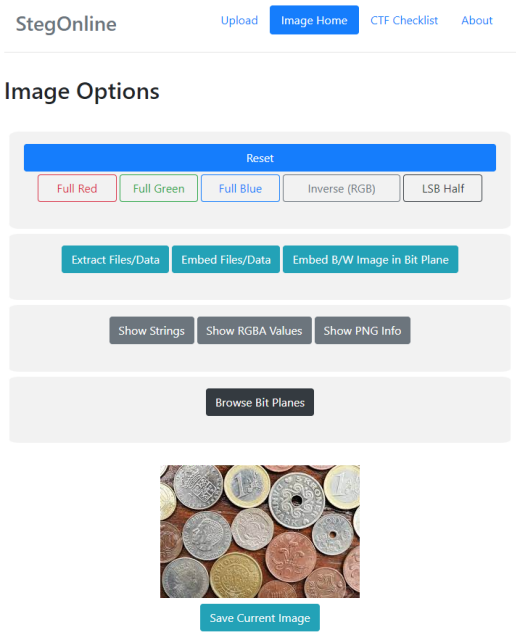


Figura 223 Análisis del imagen con stegonline.

9. Se convierte el nombre del archivo de Hex a ASCII utilizando *Cyberchef* como aparece en la Figura 224.

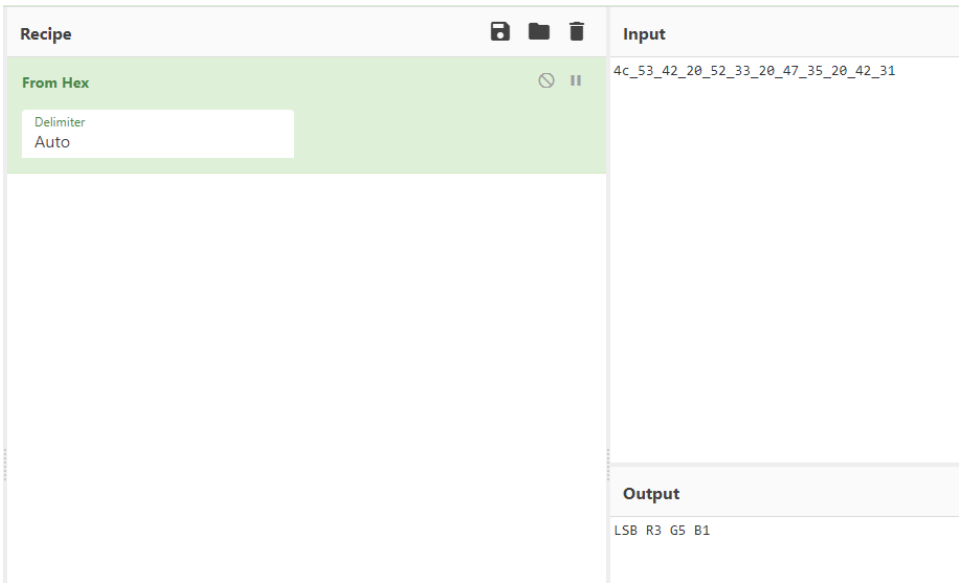


Figura 224 Decodificación del valor encontrado a ASCII.

10. En stegonline, según la Figura 225, se selecciona la opción de Extract/Data y con base a información obtenida en *Cyberchef se* cambian los siguientes valores
    a. BIT order: LSB
    b. Bit Rojo(R) = 3
    c. Bit Verde(G)= 5
    d. Bit Azul(B)= 1

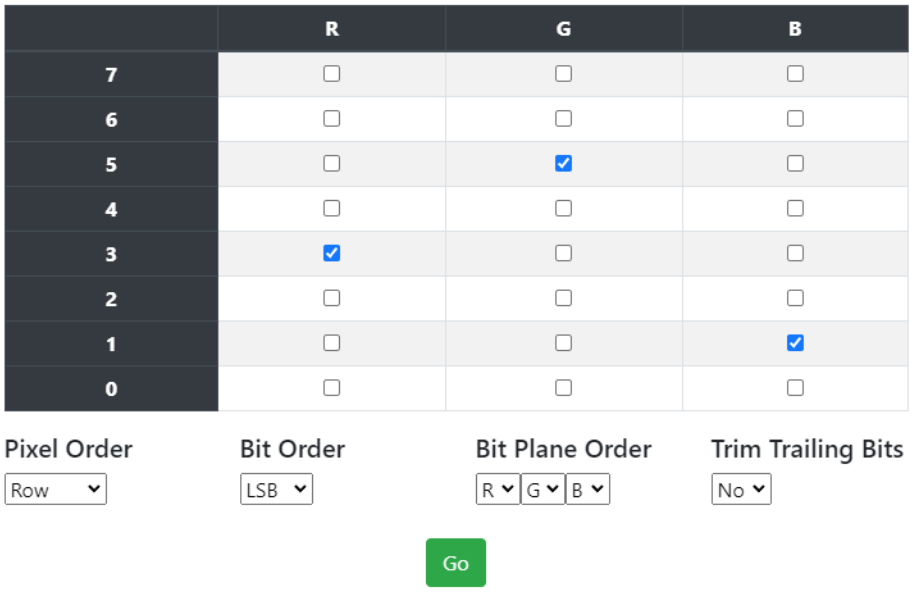


Figura 225 Configuración de valores en stegonline.

11. Presiona el botón GO
12. El valor oculto aparece en texto plano según la Figura 226.

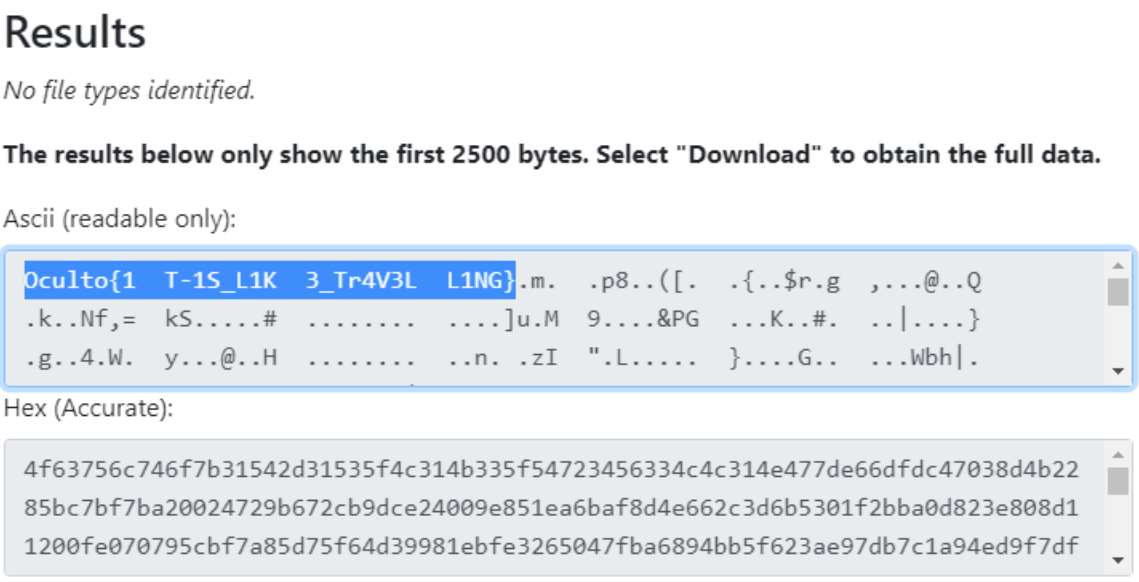


Figura 226 Detección de valor secreto con stegonline.

9.7.23 Reto: Sanitization

La Figura 227 muestra la descripción del reto “Sanitization” observado por los participantes durante el CTF.

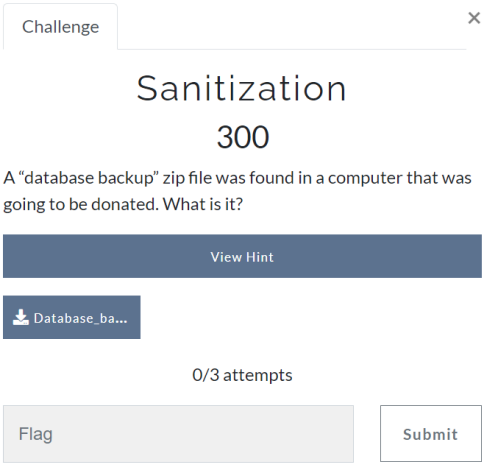


Figura 227 Descripción del reto “Sanitization”.

Archivo: Database_backup.zip

1. Descargue y extraiga los archivos comprimidos dentro de Database_backup.zip. La Figura 228 muestra los archivos extraídos.

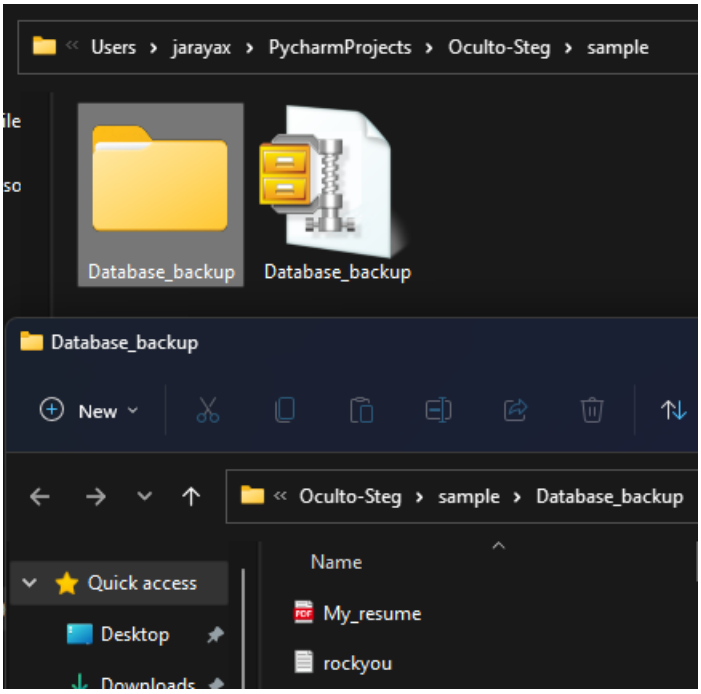


Figura 228 Extracción de archivos comprimidos dentro del Database_backup.zip.

2. El archivo zip tiene un PDF y un archivo de texto que se puede utilizar como archivo de diccionario para descubrir contraseñas con fuerza bruta.
3. Analice el archivo PDF con el comando *binwalk* según aparece en la Figura 229.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Database_backup]
└─$ binwalk --dd='.*' My_resume.pdf

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PDF document, version: "1.6"
2765          0xACD           Zlib compressed data, default compression
5232          0x1470          Zlib compressed data, default compression
16226         0x3F62          Zlib compressed data, default compression
25675         0x644B          Zlib compressed data, default compression
44428         0xAD8C          Zlib compressed data, default compression
44838         0xAF26          Zlib compressed data, default compression
61409         0xEFE1          Zlib compressed data, default compression
61933         0xF1ED          Zlib compressed data, default compression
66270         0x102DE         Zlib compressed data, default compression
72486         0x11B26         Zlib compressed data, default compression
76654         0x12B6E         Zlib compressed data, default compression
80264         0x13988         Zlib compressed data, default compression
83665         0x146D1         Zlib compressed data, default compression
87572         0x15614         Zlib compressed data, default compression
91877         0x166E5         Zlib compressed data, default compression
94508         0x1712C         Zlib compressed data, default compression
98631         0x18147         Zlib compressed data, default compression
101427        0x18C33         Zlib compressed data, default compression
105323        0x19B6B         Zlib compressed data, default compression
109070        0x1AA0E         Zlib compressed data, default compression
173401        0x2A559         Zlib compressed data, default compression
174561        0x2A9E1         Zlib compressed data, default compression
178506        0x2B94A         Zlib compressed data, default compression
184334        0x2D00E         Zlib compressed data, default compression
184572        0x2D0FC         Zlib compressed data, default compression
184790        0x2D1D6         Zlib compressed data, default compression
185224        0x2D388         Zlib compressed data, default compression
197514        0x3038A         Zlib compressed data, default compression
197826        0x304C2         Zlib compressed data, default compression
273483        0x42C4B         Zlib compressed data, default compression
286805        0x46055         Zlib compressed data, default compression
301808        0x49AF0         JPEG image data, JFIF standard 1.01
```

Figura 229 Análisis del archivo My_resume.pdf con binwalk.

4. Identifique la imagen jpg de los archivos extraídos. Según la Figura 230, el archivo extraído 49AF0 es una imagen.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Database_backup/_My_resume.pdf.extracted]
└─$ file 49AF0
49AF0: JPEG image data, JFIF standard 1.01, resolution (DPI), density 120x120, segment length 16, baseline, precision 8, 466x271, components 3
```

Figura 230 Identificación de tipo de archivo.

5. Cambie la extensión del archivo 49AF0 por jpg
6. Abra la imagen 49AF0.jpg. La vista previa de la imagen está disponible en la Figura 231.

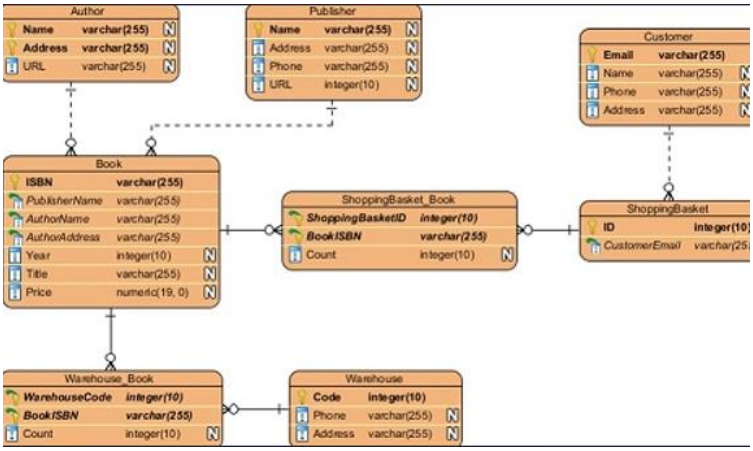


Figura 231 Vista Previa de la imagen 49AF0.jpg.

7. Analice la imagen con Oculto usando el comando que aparece en la Figura 232.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/49AF0.jpg -p -m -stg

 _____  _______ _      _         _______  _____
|     | |       |     | |       |   |   |     |
|_____| |_____  |_____| |_____  |   |   |_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/49AF0.jpg
```

Figura 232 Comandos para el análisis del archivo 49AF0.jpg con Oculto.

8. Oculto detecta la presencia de un archivo o dato oculto con *steghide* tal y como aparece en la Figura 233.

```
>>Stegoanalysis in process  using Stegseek and bruteforce technique to detect encoded file with steghide
Executing bruteforce stegoanalysis using StegSeek please wait
 *****Hidden file found using StegSeek
 Found (possible) seed: "a8cdc123"'
```

Figura 233 Detección con Oculto y stegseek de uso de steghide en la imagen.

9. Analice el archivo con *stegseek* para identificar la clave utilizada para ocultar el archivo en la imagen 49AF0.jpg. En la Figura 234 se demuestra la extracción de un archivo info.txt

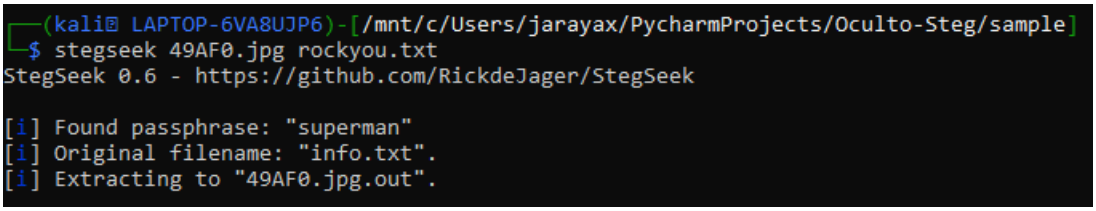


Figura 234 Detección de clave usada para cifrar archivo con steghide.

10. Extraiga el archivo oculto usando *steghide* y la clave superman. La Figura 235 muestra que la extracción fue exitosa.

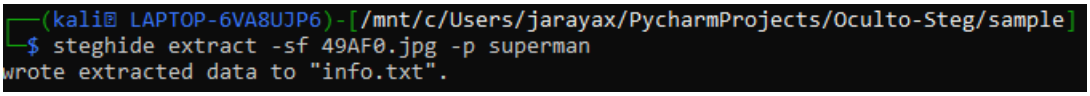


Figura 235 Extracción de archivo oculto.

11. Abra el archivo info.txt para descubrir el mensaje oculto en texto plano según la Figura 236.

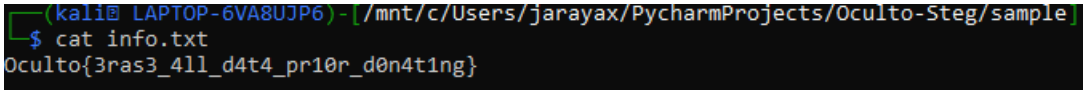


Figura 236 Extracción de valor secreto.

9.7.24 Reto: Nesting dolls

La Figura 237 muestra la descripción del reto “Nesting dolls” observado por los participantes durante el CTF.

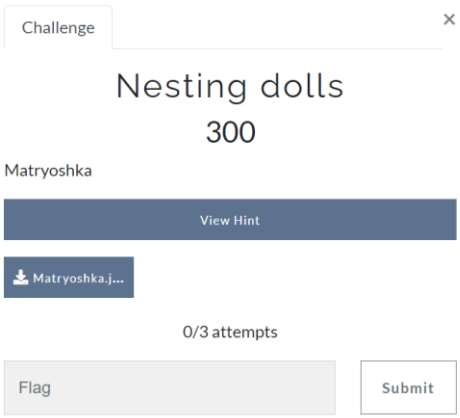


Figura 237 Descripción del reto "Nesting dolls".

Archivo: Matryoshka.jpg. La vista previa de la imagen a analizar se encuentra en la Figura 238.

Figura 238 Vista previa de la imagen Matryoshka.jpg.

1. Descargue la imagen Matryoshka.jpg
2. Analice la imagen con Oculto usando el comando que aparece en la Figura 239.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg]
└─$ python3 Oculto_steg.py -f /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Matryoshka.jpg -p -m -stg

 _____  _______ _     _        _______  _____
 |     | |       |     | |      |       |     |
 |_____| |_____  |_____| |_____ |       |_____|


**  Steganography Framework  **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://github.com/Oculto-Steg/Oculto-Steg **
Image to analyze: /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Matryoshka.jpg
```

Figura 239 Comandos para el análisis del archivo Matryoshka.jpg con Oculto.

3. Como se observa en la Figura 240, Oculto detecta en el metadata de la imagen un URL y un valor en Base62.

```
*********File Metadata:***********
Extracting File metadata
ExifTool Version Number         : 12.40
File Name                       : Matryoshka.jpg
Directory                       : /mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample
File Size                       : 673 KiB
File Modification Date/Time     : 2022:04:21 15:49:53+02:00
File Access Date/Time           : 2022:04:21 15:51:18+02:00
File Inode Change Date/Time     : 2022:04:21 15:51:18+02:00
File Permissions                : -rwxrwxrwx
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 96
Y Resolution                    : 96
Exif Byte Order                 : Big-endian (Motorola, MM)
XP Comment                      : https://www.better-converter.com/Encoders-Decoders/Base62-Decode
XP Keywords                     : 6NATgArbND8tbpB
```

Figura 240 Detección con Oculto de URL y valor secreto oculto en el metadata de la imagen.

4. Oculto detecta una imagen oculta en imagen Matryoshka.jpg como se observa en la Figura 241.

```
 *****Embedded file(s) found with Binwalk
b'\nDECIMAL       HEXADECIMAL     DESCRIPTION\n--------------------------------------------------------------------------------
data, big-endian, offset of first image directory: 8\n654124        0x9FB2C         JPEG
**********Analyzing with jsteg**********
```

Figura 241 Detección con Oculto y Binwalk de archivo oculto en la imagen.

5. Decodifica el valor encontrado en el metadata. Según la figira 242, el valor decodificado de Base62 a ASCII es “Abracadabra”.

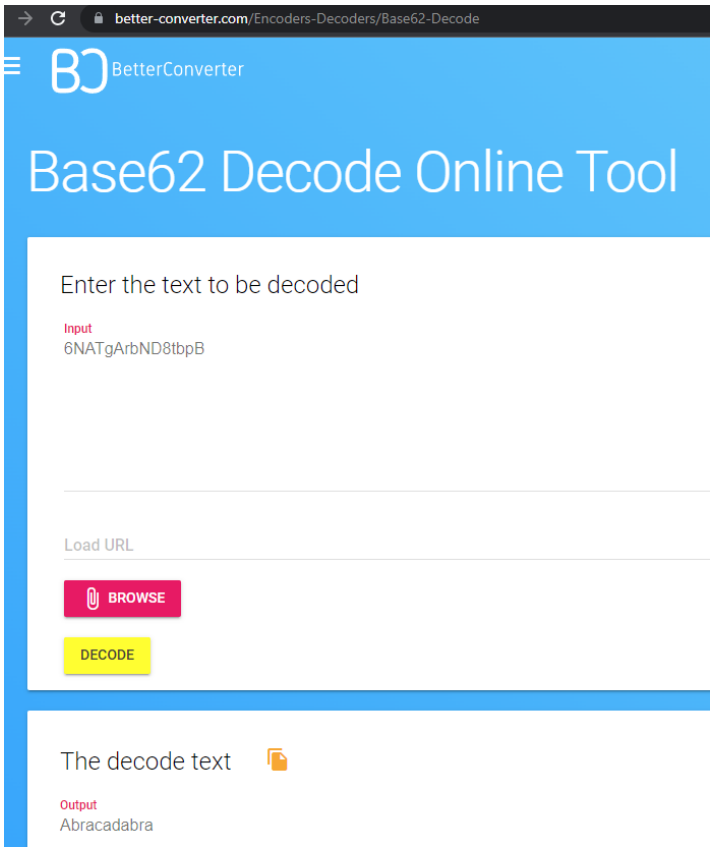


Figura 242 Decodificación de valor secreto usando el URL obtenido del metadata.

6. Extraiga la imagen oculta del archivo Matryoshka.jpg según se observa en la Figura 243.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ binwalk --dd='.*' Matryoshka.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
30            0x1E            TIFF image data, big-endian, offset of first image directory: 8
654124        0x9FB2C         JPEG image data, JFIF standard 1.01
```

Figura 243 Extracción de archivos ocultos con Binwalk.

7. Identifique la imagen oculta. Según la Figura 244, el archivo extraído 9FB2C es una imagen que estaba oculta.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/_Matryoshka.jpg.extracted]
└─$ file 9FB2C
9FB2C: JPEG image data, JFIF standard 1.01, resolution (DPI), density 120x120, segment length 16, baseline, precision 8, 231x403, components 3
```

Figura 244 Identificación de la imagen oculta.

8. Cambie la extensión por jpg como se muestra en la Figura 245.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/_Matryoshka.jpg.extracted]
└─$ mv 9FB2C 9FB2C.jpg
```

Figura 245 Cambio de extensión del archivo oculto.

9. Analice la imagen con Oculto. La Figura 246 muestra una vista previa de la imagen extraída.

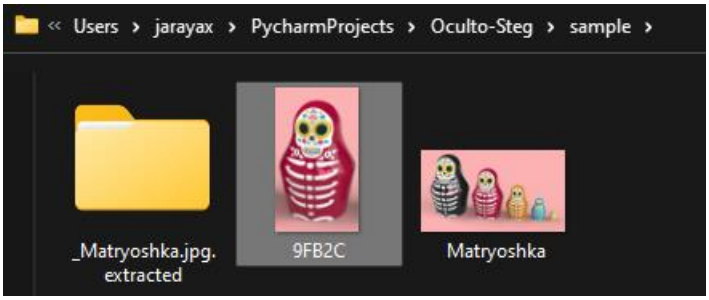


Figura 246 Vista previa de la imagen extraída.

10. Oculto detecta un archivo o data oculto con *steghide* y protegido con contraseña según se muestra en la Figura 247.

```
Executing bruteforce stegoanalysis using StegSeek please wait
 *****Hidden file found using StegSeek
Found (possible) seed: "0abc422a"'
```

Figura 247 Detección con Oculto y Stegseek de archivo oculto en la imagen.

11. Extraiga el archivo oculto usando *steghide* y “Abracadabra” como contraseña. En la Figura 248 se muestra que se logró extraer un archivo B58.txt

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ steghide extract -sf 9FB2C.jpg -p Abracadabra
wrote extracted data to "B58.txt".
```

Figura 248 Extracción de archivo oculto usando steghide.

12. Lea el contenido del archivo B58.txt. según la Figura 249, el valor presente en B58.txt parece estar codificado.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample]
└─$ cat B58.txt
4HyybUmoGa3ZpGP25dpBeKRgg
```

Figura 249 Extracción de valor codificado en base 58.

13. El nombre del archivo da la pista que el valor está codificado en Base58
14. Descodifique el valor usando un decodificador de Base58 como se observa en la Figura 250.

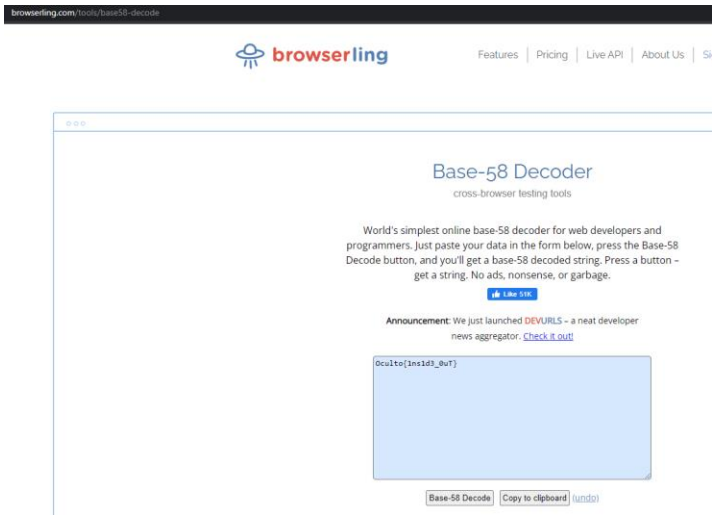


Figura 250 Decodificado de valor secreto.

9.7.25 Reto: Happy birthday!

La Figura 251 muestra la descripción del reto “Happy birthday!” observado por los participantes durante el CTF.

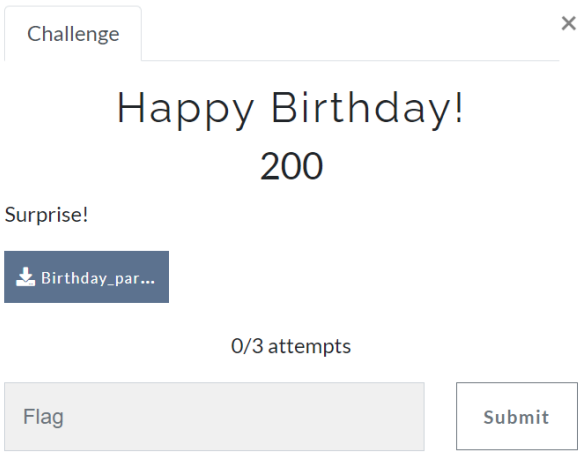


Figura 251 Descripción del reto “Happy Birthday”.

Archivo: Birthday_party.zip

1. Descargue y extraiga los archivos contenidos en Birthday_party.zip. Según se aprecia en la Figura 252, el archivo zip tiene 6 imágenes comprimidas.

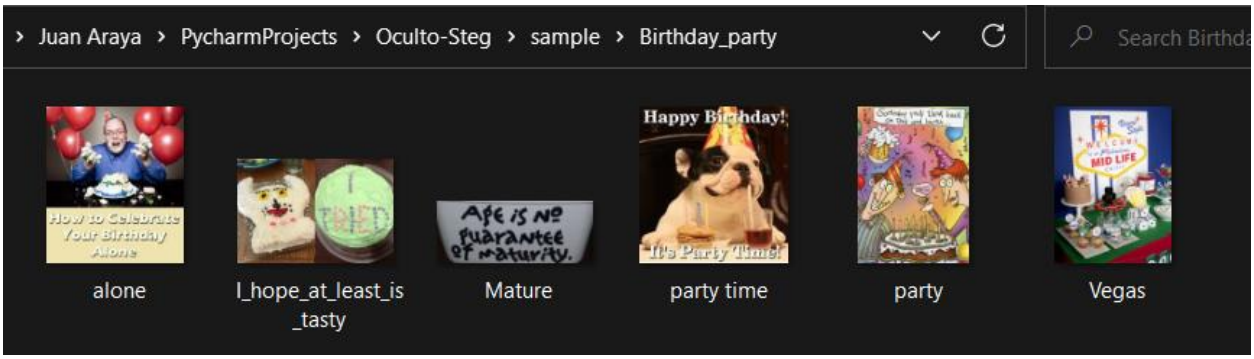


Figura 252 Archivos extraídos Birthday_party.zip.

2. Analice cada foto con Oculto
3. Ocultodetecta algo oculto que en los archivos I_hope_at_least_is_tasty.png, Mature.png y en Vegas.jpg. En la Figura 253 se observa el mensaje de confirmación que Oculto ha detectado un archivo oculto.

```
*****Embedded file(s) found with Binwalk
b'\nDECIMAL       HEXADECIMAL     DESCRIPTION\n--------------------------------------------------------------------------------
tring: "Copyright (c) 1998 Hewlett-Packard Company"\n14779         0x39BB          TIFF image data, big-endian, of
```

Figura 253 Detección con Oculto y binwalk de archivo oculto en la imagen.

4. En la Figura 254 se confirma que Oculto detecta un PDF en el archivo I_hope_at_least_is_tasty.png con *binwalk*

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Birthday_party]
└─$ binwalk --dd='.*' I_hope_at_least_is_tasty.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
382           0x17E           Copyright string: "Copyright (c) 1998 Hewlett-Packard Company"
144901        0x23605         PDF document, version: "1.7"
145046        0x23696         Zlib compressed data, default compression
237425        0x39F71         Zlib compressed data, default compression
416296        0x65A28         Zlib compressed data, default compression
```

Figura 254 Extracción y detección con PDF dentro de la imagen I_hope_at_least_is_tasty.png.

5. Identifique el archivo PDF entre los archivos extraídos. Según la Figura 255, el archivo 23605 es un PDF.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Birthday_party/_I_hope_at_least_is_tasty.png.extracted]
└─$ ls
0  17E  23605  23696  23696.zlib  39F71  39F71.zlib  65A28  65A28.zlib

┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Birthday_party/_I_hope_at_least_is_tasty.png.extracted]
└─$ file 0
0: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, progressive, precision 8, 1500x1000, components 3

┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Birthday_party/_I_hope_at_least_is_tasty.png.extracted]
└─$ file 17E
17E: data

┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto-Steg/sample/Birthday_party/_I_hope_at_least_is_tasty.png.extracted]
└─$ file 23605
23605: PDF document, version 1.7
```

Figura 255 Identificación del PDF extraído con binwalk.

6. Cambie la extensión al archivo usando el comando que aparece en la Figura 256.

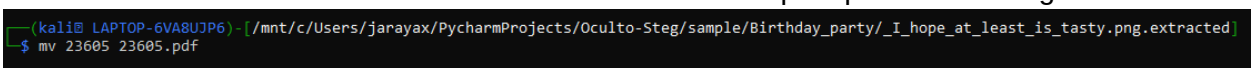


Figura 256 Cambio de extensión a pdf del archivo extraído con binwalk.

7. Abra el archivo PDF. Según la Figura 257 aparece un hipervínculo en el PDF.

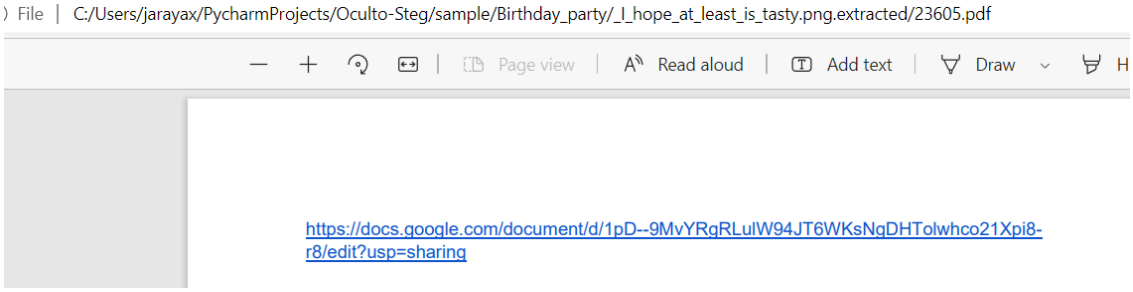


Figura 257 URL encontrada en el PDF extraído.

8. Abra el hipervínculo que se encuentra en el PDF. En la Figura 258 se observa el valor secreto para resolver este reto.

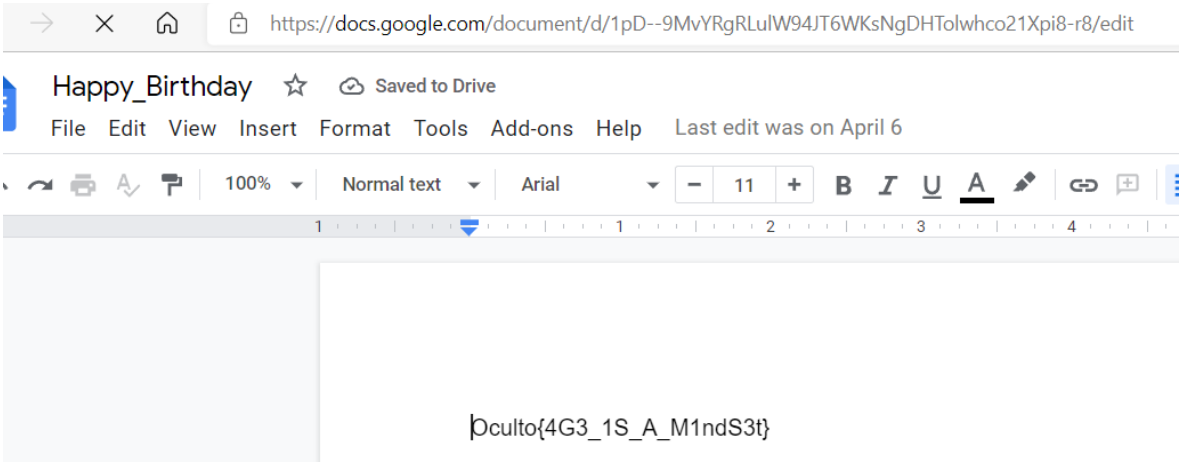


Figura 258 Valor secreto encontrado en el URL.

9.8 Coordinación del CTF

Se hicieron varios banners y anuncios publicitarios en Redes sociales, como Linkedin y Twitter, con el fin de informar y motivar a la participación del CTF

Por ejemplo, a continuación, la Figura 259 representa el anuncio publicado en mi perfil de Linkedin:
https://www.linkedin.com/posts/juaraya_stegoctf-cibersecurity-activity-6904765220997128192-R2KU?utm_source=linkedin_share&utm_medium=member_desktop_web

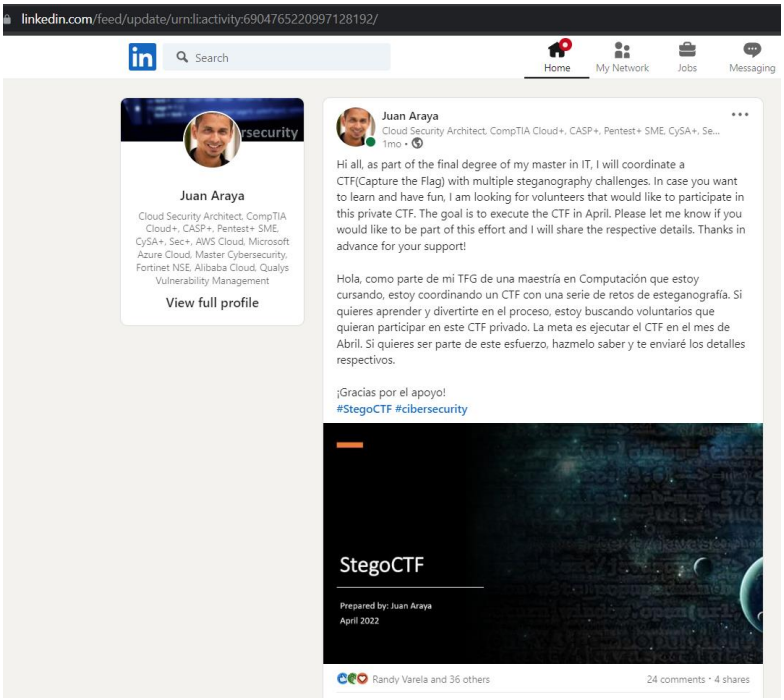


Figura 259 Publicidad en red social de Linkedin sobre el CTF.

En la Figura 260 se muestra el apoyo publicitario realizado por la Comunidad Dojo de Panamá(grupo de formación en temas de tecnologías de información, Cloud computing y ciberseguridad).

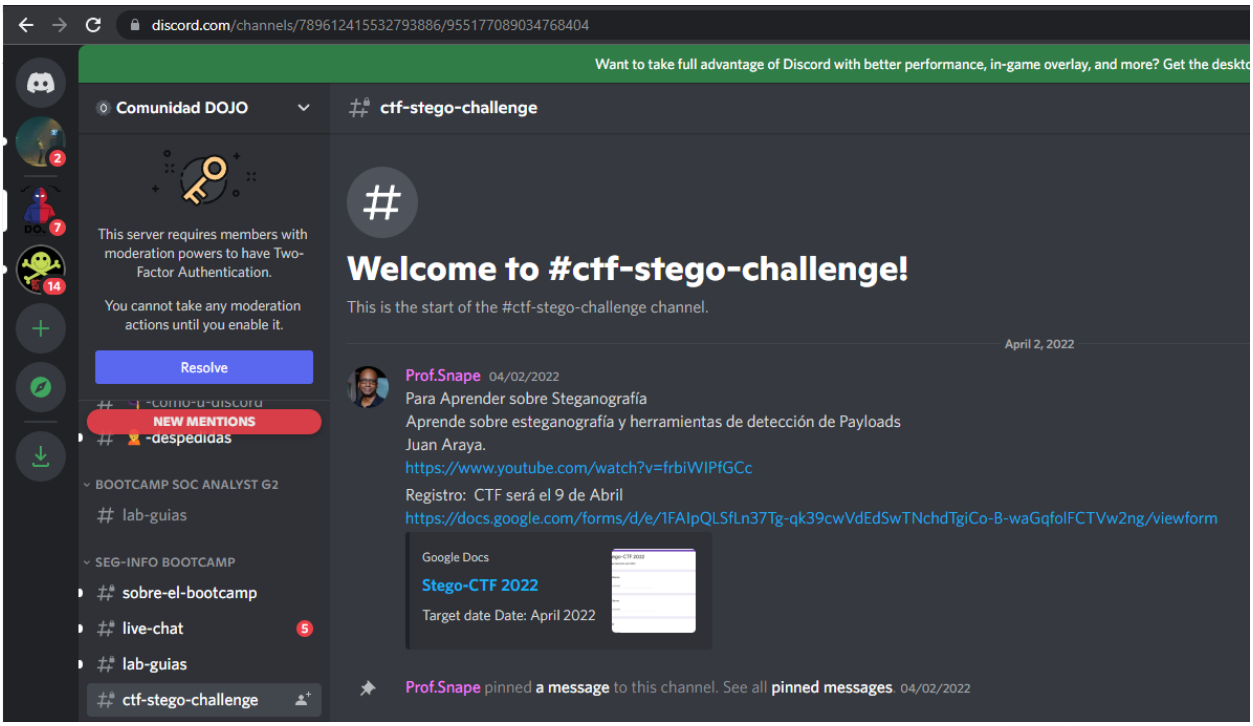


Figura 260 Publicidad en servidores de Discord sobre el StegoCTF.

En la Figura 261 se demuestra el apoyo de la comunidad de Ethical Hackers DC11506 de Costa Rica, quienes brindaron un fuerte impulso al CTF.

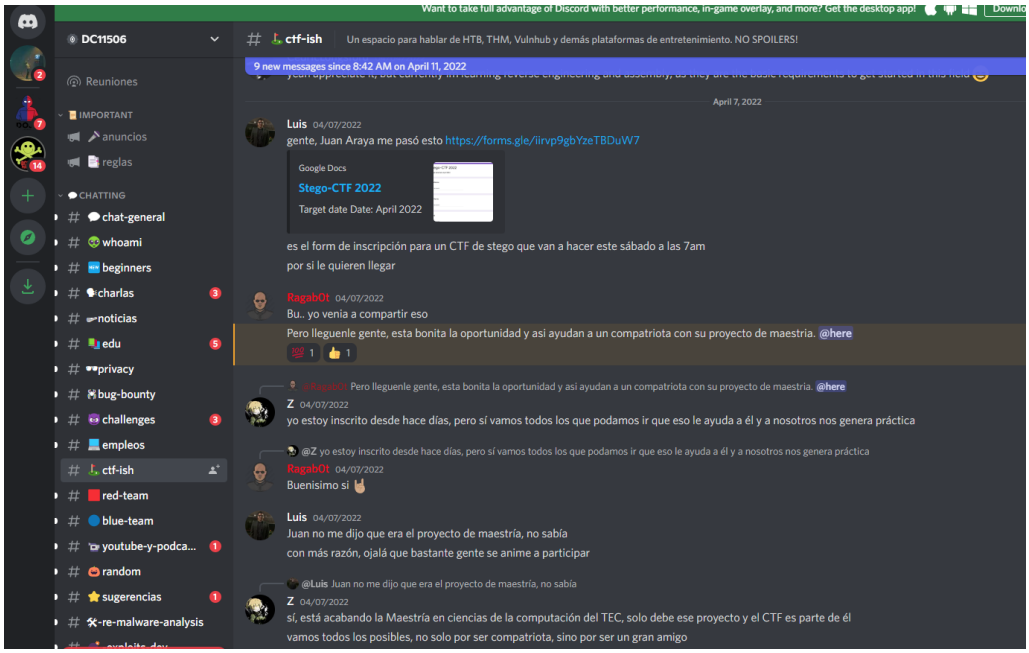


Figura 261 Coordinación y motivación realizada por el grupo DC11506 para participar en el CTF.

Con el fin de motivar y aclarar dudas sobre el proceso de estegoanálisis impartí 2 conferencias gratuitas:

- La Figura 262 muestra evidencia de la charla de Introducción al estegoanálisis. 5 de diciembre 2021. Organizado por DC11506 Ethical Hackers de Costa Rica.

Figura 262 Charla de Introducción al estegoanálisis disponible en https://www.youtube.com/watch?v=07bNNTeJyjY.

- La Figura 263 representa la publicidad de la charla “Aprende sobre Esteganografía y herramientas de detección de payloads” impartida el 26 de Marzo 2022. Organizado por Comunidad Dojo.

Figura 263 Charla de Introducción al estegoanálisis y herramientas de detección de payloads disponible en https://www.youtube.com/watch?v=frbiWIPfGCc.

Con el fin de motivar a los participantes se ofreció como premio un Voucher para el examen de certificación CompTIA CySA+ valorado en 350 dólares.

9.9 Servidor de CTF

En la Figura 264 se muestran las opciones de conFiguración usadas en el servidor de la plataforma de CTFD.

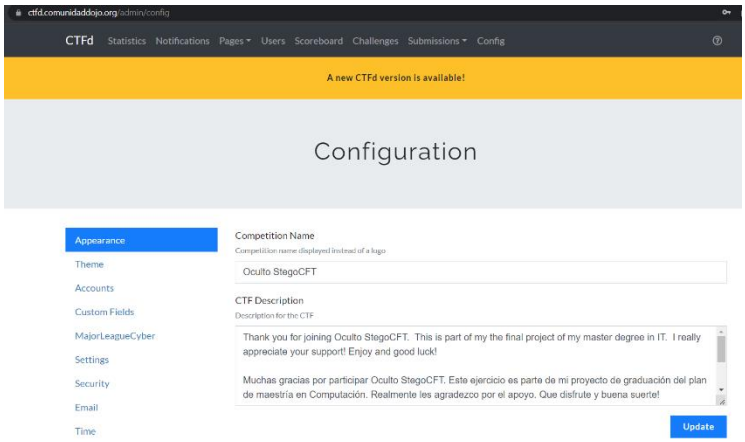


Figura 264 ConFiguración del servidor de CTF.

Dicho servidor a nombre de la comunidad Dojo está hospedado en la nube en la región de Alemania.

Según se muestra en la Figura 265 y 266, se configuró el servidor para que el CTF estuviera disponible el 9 de Abril de 7:00am a 12:30pm hora local Costa Rica.

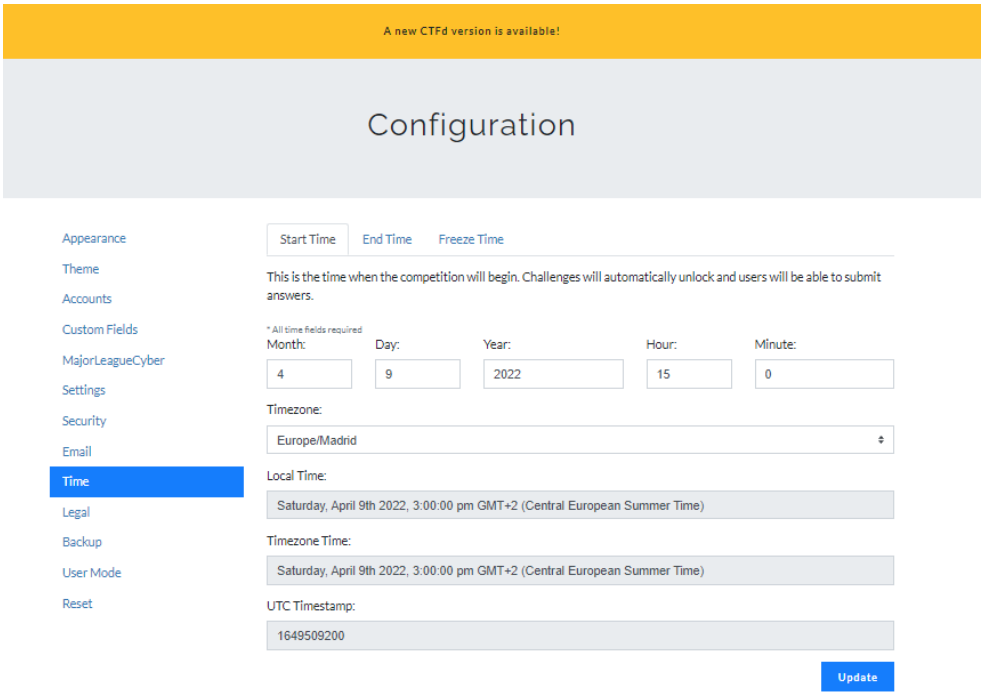


Figura 265 Fecha y Hora de inicio del CTF.

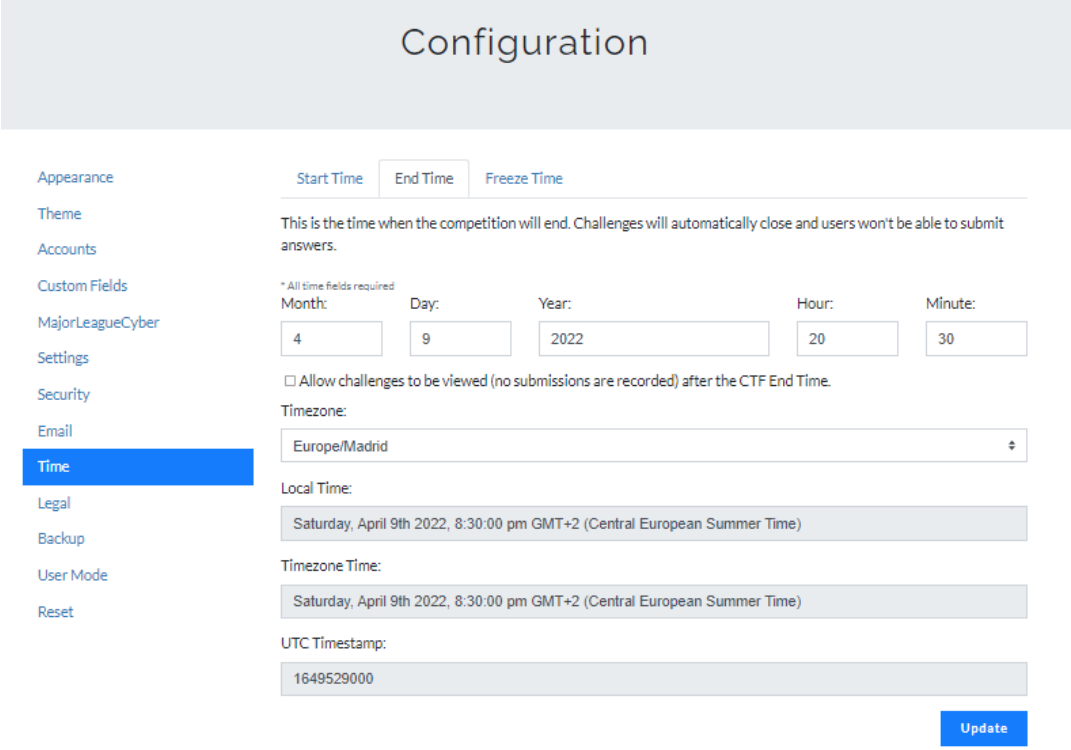


Figura 266 Fecha y Hora de fnalización del CTF.

9.10 Retroalimentación de la ejecución del CTF

A continuación, los datos obtenidos de las 21 personas que llenaron la encuesta. Según la Figura 267 un 33% de los participantes tenían más de 5 años de experiencia en IT.

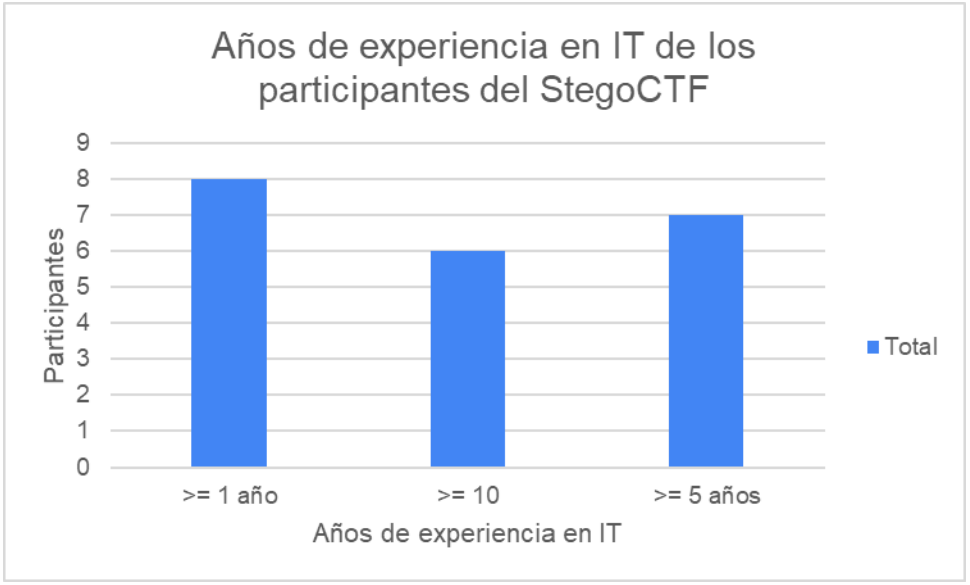


Figura 267 Años de experiencia en IT.

Como se observa en la Figura 268, para 11 de los 21 participantes, StegoCTF fue su primera participación en una competencia de este tipo.

Figura 268 Gráfico de nivel de experiencia en CTFs de los participantes que llenaron encuesta.

Según la Figura 269, 16 de 21 participantes calificó como excelente la coordinación del evento.

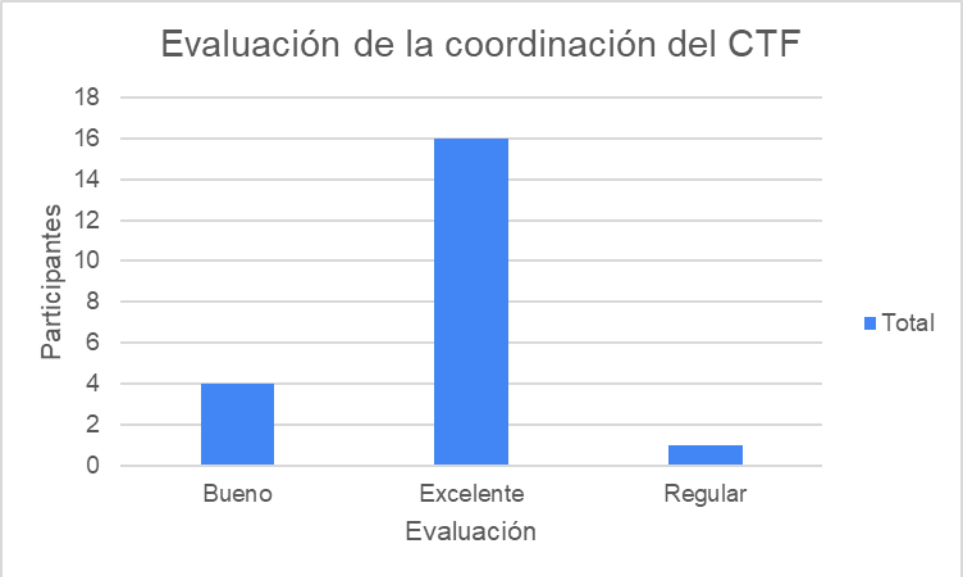


Figura 269 Gráfico de nivel de satisfacción con la coordinación del CTF.

En la Figura 270 se nota que la mayoría confirmó que eran suficientes y acorde a este tipo de competencias.

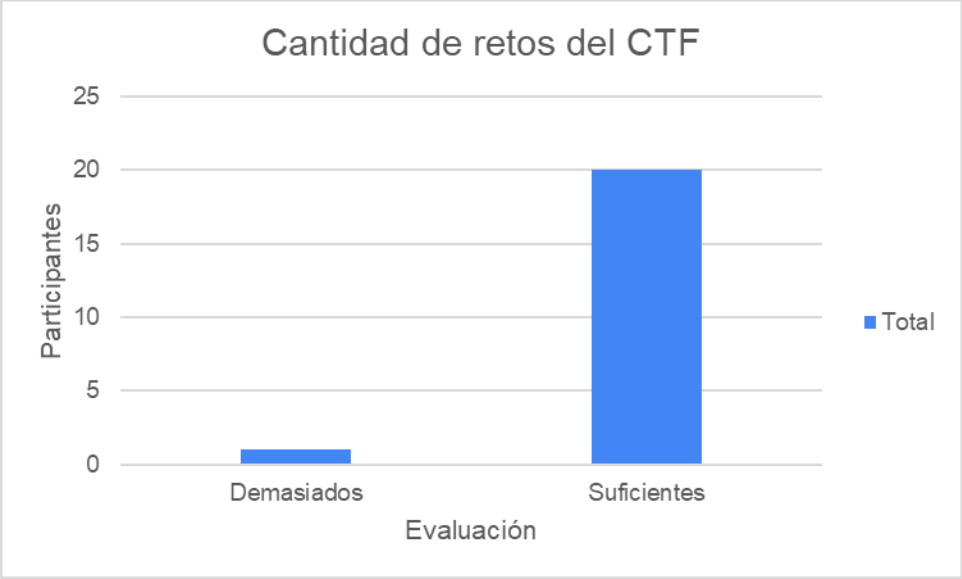


Figura 270 Gráfico de evaluación de la cantidad de retos del CTFs.

En la Figura 271 se confirma que los retos estaban claros y bien redactados según los participantes.

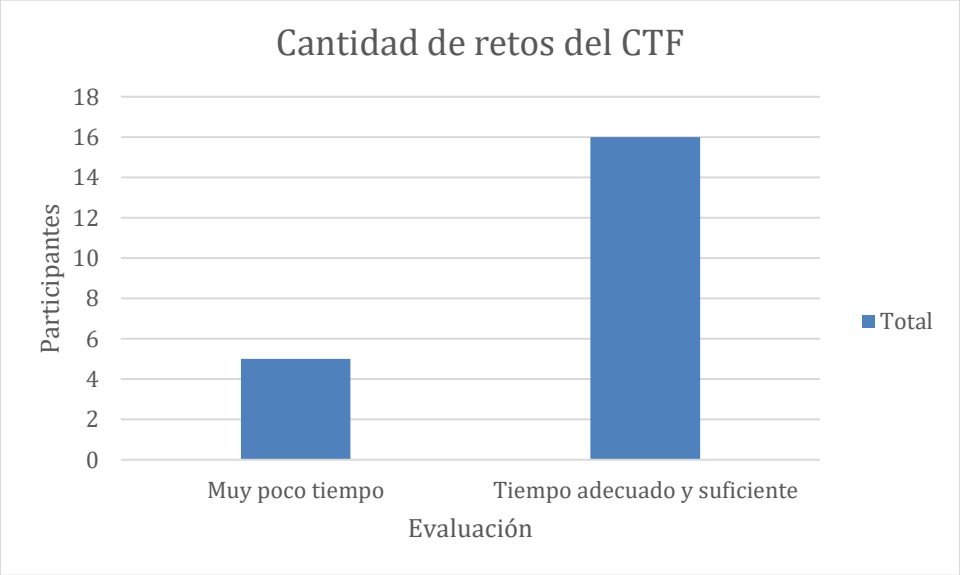


Figura 271 Gráfico de evaluación de la calidad de retos del CTFs.

En la Figura 272 se confirma que los retos no eran fáciles, sino que requerían un buen nivel técnico para su comprensión y resolución.

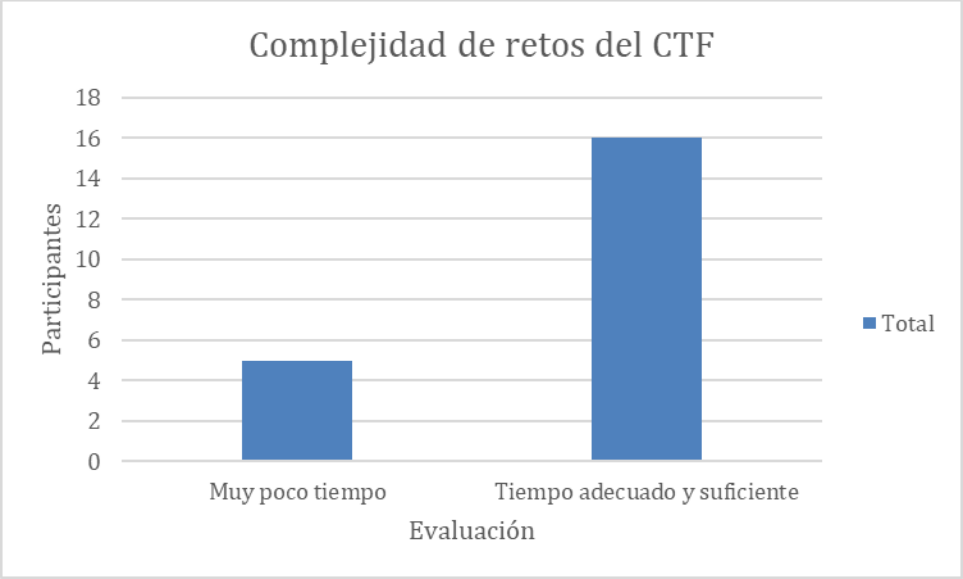


Figura 272 Gráfico de evaluación de la complejidad de retos del CTFs.

Según Figura 273 un 76.2% de los participantes consideraron que 5 horas eran suficientes para el reto.

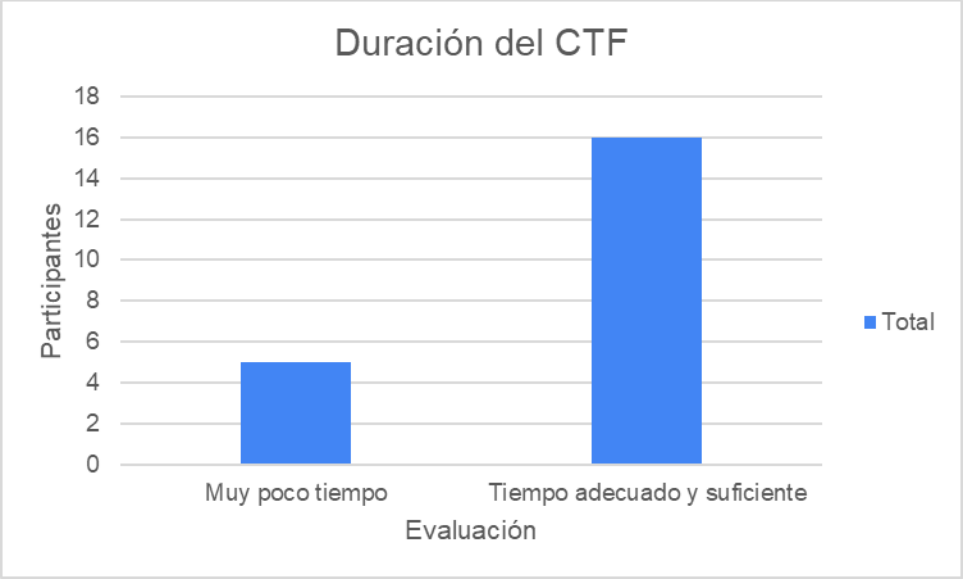


Figura 273 Gráfico de evaluación de la duración del CTF.

Áreas de mejora del CTF según las 21 personas que llenaron la encuesta:

- “Ninguna”
- “Ningún comentario negativo”
- “Podrían dar hints de como resolver los ejercicios planteados”
- “Menos pistas.”
- “Nada”
- “Mejorar yo mis habilidades”

- “pistas Free”
- “Mas premios para incentivar las primeras 3 posiciones.”
- “Los hints no me sirvieron demasiado en los desafios que no pude resolver. (Por ej: CAT)”
- “N/A”
- “Los avisos/notificaciones en la modificación de challenges”
- “No tengo mucha experiencia en CTF como para poder anotar alguna observación. Me hubiese gustado tener más experiencia en el área para poder desarrollarme mejor a la hora de efectuar los desafíos.”
- “Más tiempo para jugar, pero entiendo que todos tenemos labores por realizar y se necesita personal y dinero para alargarlo.”
- “s/n”
- “Opción de visualizar los first blood”
- “Felicidades por el desarrollo de este CTF y los feedback que nos diste sobre los videos que presentaste en youtube por lo menos capture 2 jajaja quede con esa espina de saber la solución de los otros challenge por que utilice la herramienta donde estaba el video que dice Aprende sobre Stego”
- “Quizás alguna forma en la que todos los participantes puedan interactuar como en eventos presenciales”
- “Aprendizaje antes de, en el caso de estego fue excelente ya que el instructor enseño el tema a participar.”

Áreas positivas del CTF según las 21 personas que llenaron la encuesta:

- “Los retos, logística, etc”.
- “Los retos fueron muy bien pensados”.
- “Pude investigar y aprender más sobre mi primer CTF”.
- “La coordinación y el uso de la plataforma. Excelente disponibilidad del organizador y puntualidad”.
- “Todo”.
- “El nivel del CTF muy bueno”.
- “Absolutamente todo”.
- “Pude poner en práctica lo aprendido”.
- “The challenges were very well-written”.
- “Entretenido”.
- “Fue entretenido”.
- “Me obligó a investigar sobre las distintas herramientas utilizadas para stego, y estuvo entretenido”.
- “Este tipo de eventos forza tus habilidades al límite, y aunque se genera cierta ansiedad, es placentero cuando logras obtener un flag, agregando así valor y experiencia como profesionales de tecnología”.
- “La temática”.

- “Los desafíos estaban bastante interesantes, a pesar de que no tengo mucha experiencia en esto me ayudó a interesarme y descubrir una rama más”.
- “Si me gustó mucho, es una temática interesante”.
- “Todos los ejercicios de stego fueron ideales, es un área que se puede potenciar y seguir capacitando”.
- “A nivel general estuvo muy bueno, aprendí nuevos conceptos como base62, NFT durante el desarrollo del evento”.
- “Fue muy bueno porque me resultó. Me desanime un poco porque no encontraba los flag jaja es la primera vez que participo activo en un CTF y con esto volveré a participar. Pero me gustaría saber si se puede conocer la solución de algunos o una pista de cómo se resuelve. Me gustó mucho Felicidades por tu maestría”.
- “Principalmente el nivel de los retos y el grado de conocimiento que me vi obligado a buscar al tener una dificultad mayor a los retos básicos presentados en plataformas como THM o HTB, pero con un muy buen nivel para los objetivos buscados”.
- “Quiero aprender más este tipo de temas y mejorar mi rendimiento en los mismos”.

9.11 Definición de participantes del grupo de control y del grupo *Experimental*

Para ello, se programó un script hecho con Python cuyo código está disponible en https://github.com/jarayax/Oculto_CTF

El script realiza las siguientes acciones

1. Como se observa en la Figura 274, al ejecutar el script de distribución de participantes, solicita al usuario identificar el archivo csv que tiene los identificadores (nicknames) de los participantes y la cantidad de equipos en los que deben ser separados.

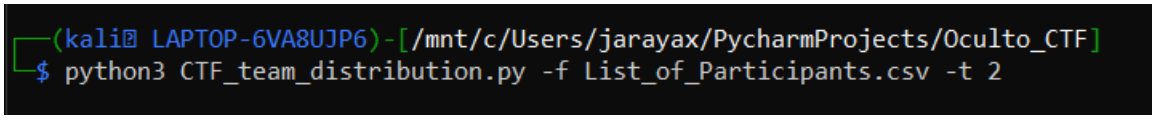


Figura 274 Script para la distribución aleratoria de participantes, unos en un grupo de control y otros en un grupo Experimental.

2. Lee la lista con el identificados (nickname) de participantes del archivo csv,
3. Almacena el usuario en una estructura de dato tipo lista
4. En la Figura 275 se demuestra que el script recorre la lista y muestra la lista de participantes.

```
┌──(kali㉿ LAPTOP-6VA8UJP6)-[/mnt/c/Users/jarayax/PycharmProjects/Oculto_CTF]
└─$ python3 CTF_team_distribution.py -f List_of_Participants.csv -t 2

 _____  _______ _     _        _______  _____
|     | |       |     | |        |    |     |
|_____| |_____  |_____| |_____   |    |_____|


**  Oculto CTF Team distributor **
**       Version 1.0         **
**  Created by:  Juan Araya  **
**  https://www.github.com/jax **
File with list of participants: List_of_Participants.csv
[['\ufeffAce'], ['Krishnan'], ['Z'], ['Hackingmess'], ['Massiel20'], ['Ricky'], ['Hack666'],
['gitHBadger'], ['Del'], ['svegar'], ['ukuku'], ['Deadpoolxcr'], ['Valefavaron'], ['kooxz'],
['Bitbl4ck'], ['Xatruch04'], ['Jorac'], ['SeniorWarden']]
```

Figura 275 Ejemplo de carga de lista de participantes en el script

5. Ordena de forma aleatoria los datos de la lista, recorre la lista y muestra la lista de participantes según lo muestra la Figura 276.

```
 >>>Sorting list randomly
[['Z'], ['kooxz'], ['Krishnan'], ['SeniorWarden'], ['Xatruch04'], ['Hack666'], ['Bitbl4ck'],
['Ricky'], ['Jorac'], ['svegar'], ['Hackingmess'], ['gitHBadger'], ['Valefavaron'], ['\ufeffA
ce'], ['Deadpoolxcr'], ['Del'], ['Massiel20'], ['ukuku']]
```

Figura 276 Ejemplo de ordenamiento aleatorio de la lista de participantes.

6. Divide la lista en la cantidad de equipos seleccionados. Por ejemplo en la Figura 277 se observa el resultado de la acción mencionada.

```
kali@LAPTOP-6VA8UJP6: /mnt/c/Users/jarayax/PycharmProjects/Oculto_CTF

>>>Splitting the team


Team 1
['kooxz']
['Z']
['kooxz']
['Krishnan']
['SeniorWarden']
['Xatruch04']
['Hack666']
['Bitbl4ck']
['Ricky']
['Jorac']


Team 2
['Hackingmess']
['svegar']
['Hackingmess']
['gitHBadger']
['Valefavaron']
['\ufeffAce']
['Deadpoolxcr']
['Del']
['Massiel20']
['ukuku']
```

Figura 277 Ejemplo de distribución de participantes.

El Grupo 1 es el Grupo de Control y el Grupo 2 es el Grupo *Experimental*.

9.12 Tiempos de resolución de retos por miembros del equipo de control

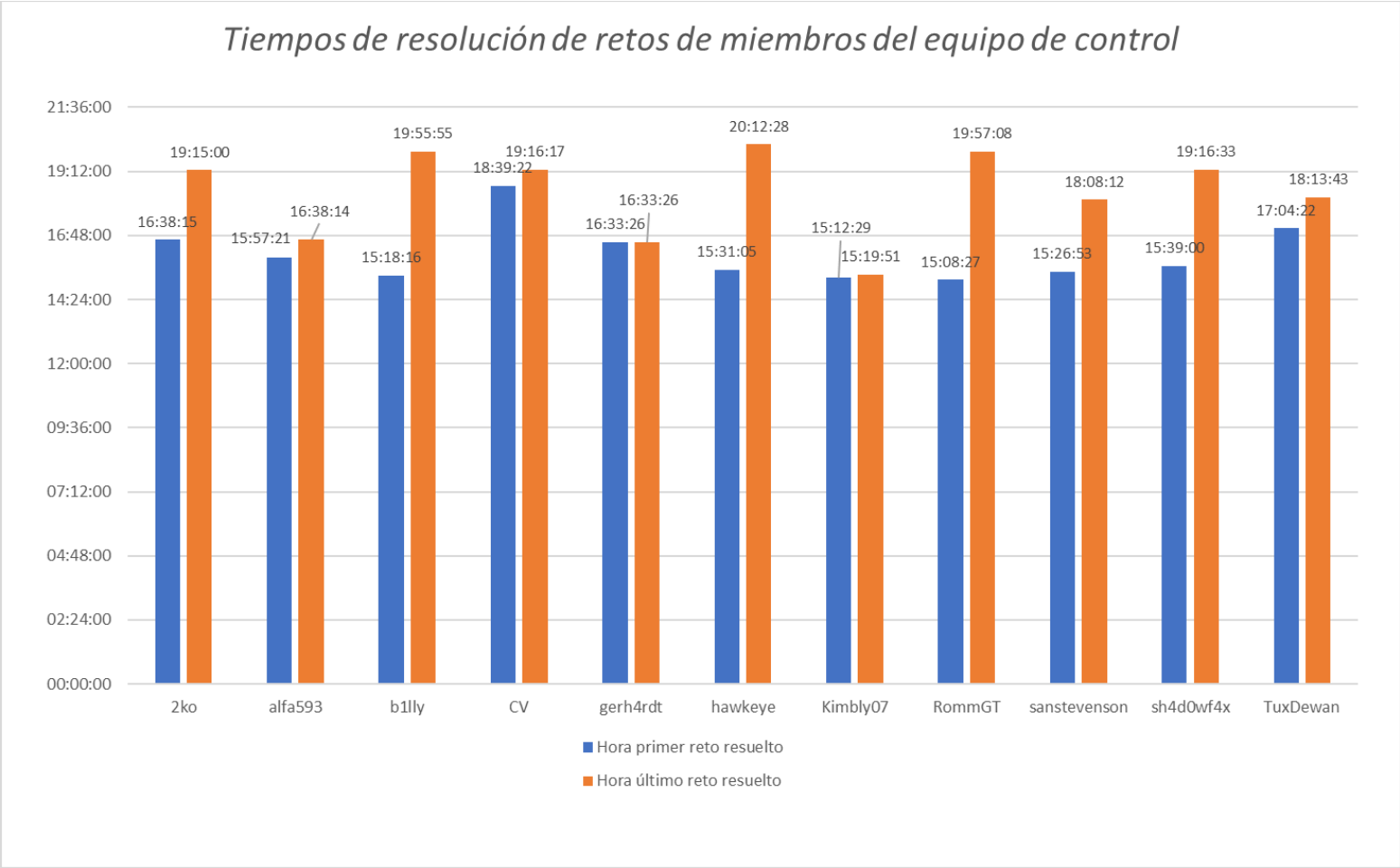


Figura 278 Tiempos de resolución de retos de miembros del equipo de control.

9.13 Tiempos de resolución de retos por miembros del grupo *Experimental*

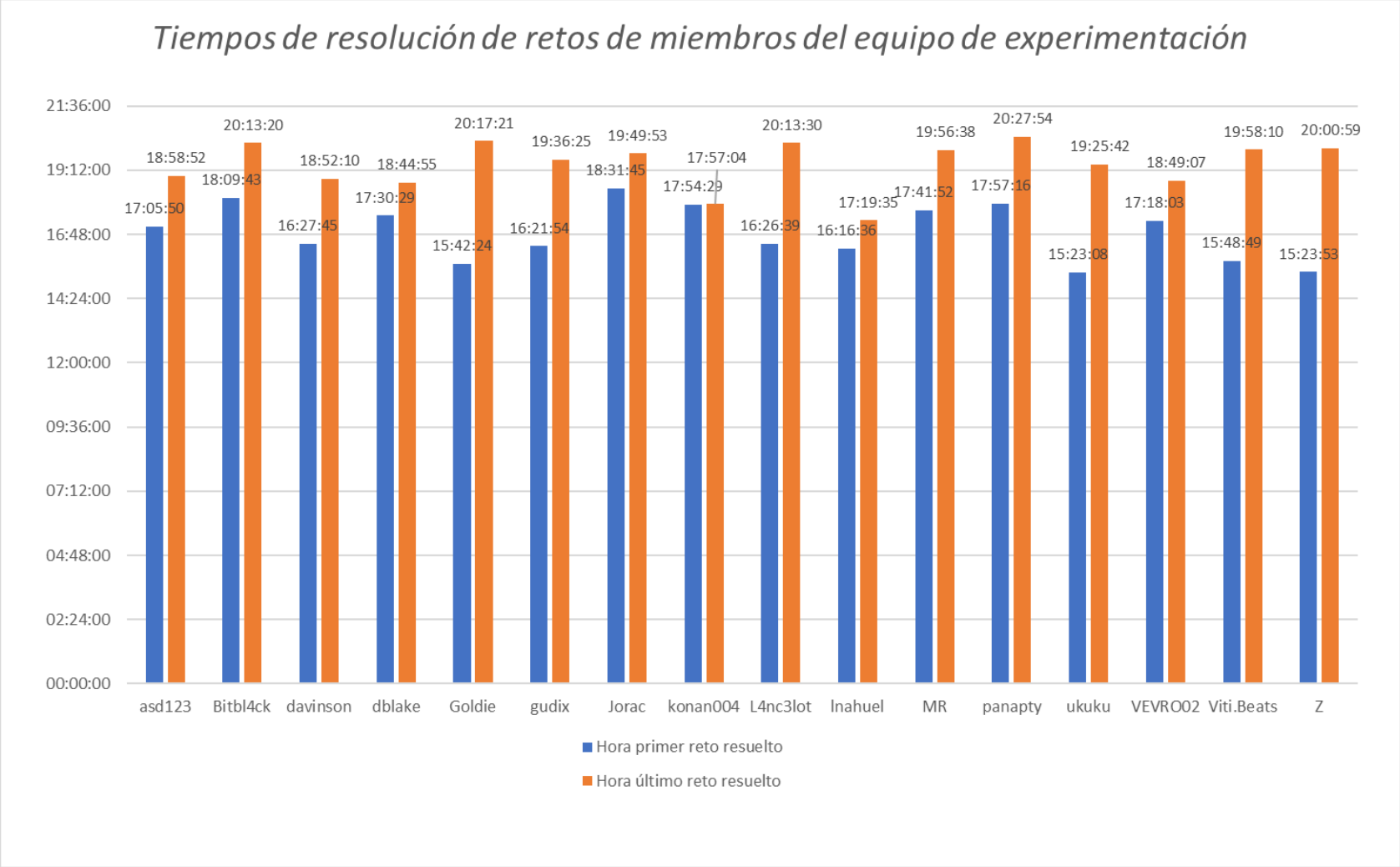


Figura 279 Tiempos de resolución por reto de miembros del equipo Experimental.