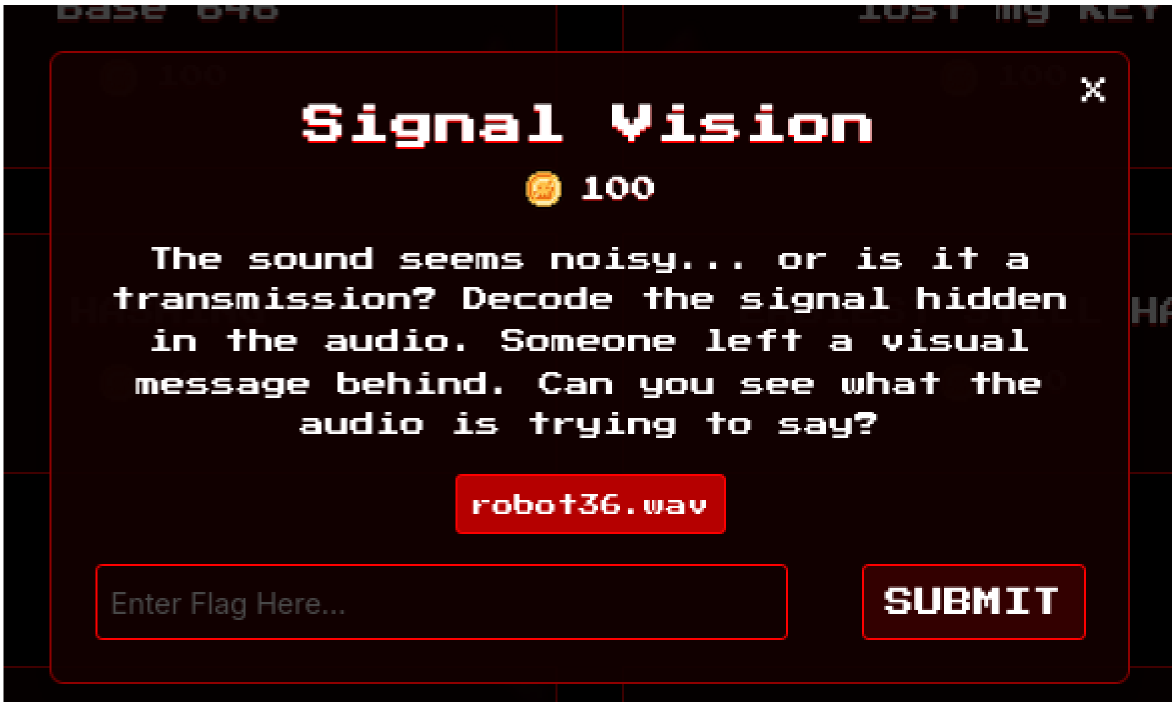

Signal Vision

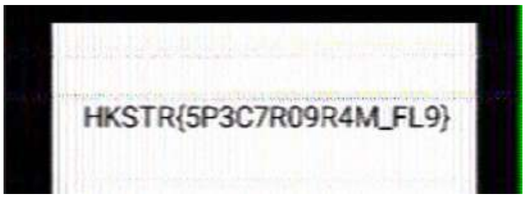
100

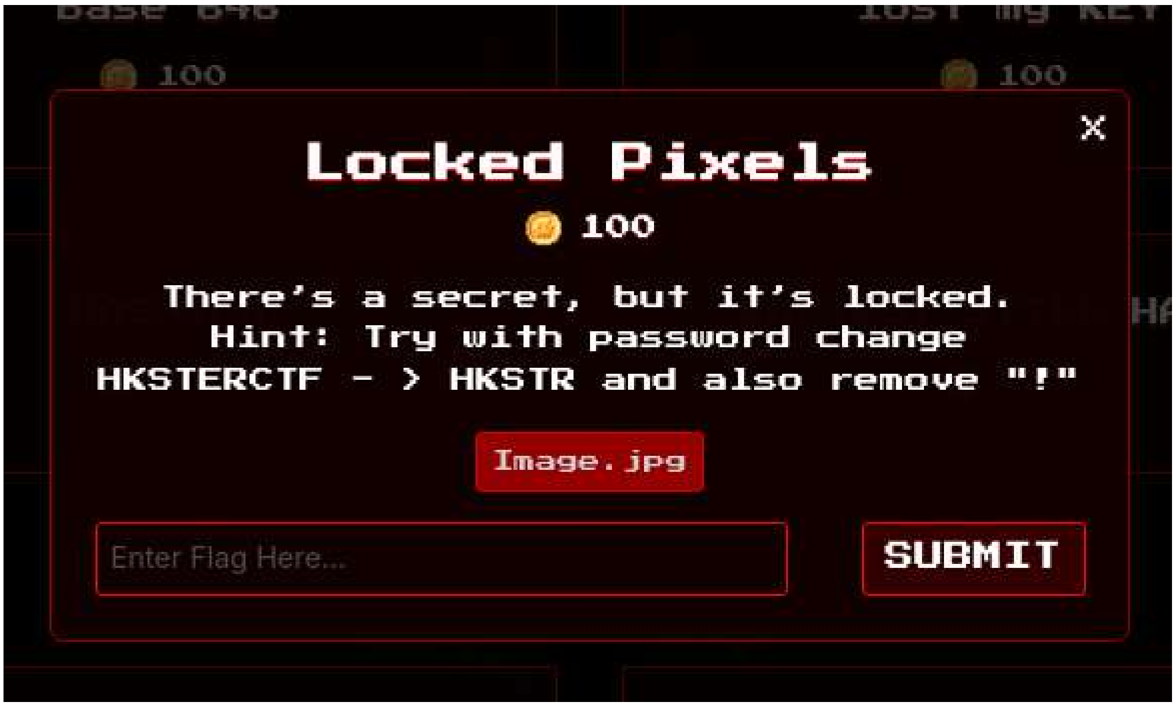
The sound seems noisy... or is it a transmission? Decode the signal hidden in the audio. Someone left a visual message behind. Can you see what the audio is trying to say?

The best way to decode SSTV audio is using the **Robot36** Android app.

Steps:

1. Download and install **Robot36: SSTV Image Decoder** from the Play Store.
2. Play the audio file on another device (or same device via file manager/audio player).
3. Open the **Robot36 app** while the audio is playing.
4. The app will **automatically start decoding** the SSTV transmission.
5. After a few seconds, an image will appear containing the **hidden flag**.


HKSTR{5P3C7R09R4M_FL9}


Locked Pixels

100

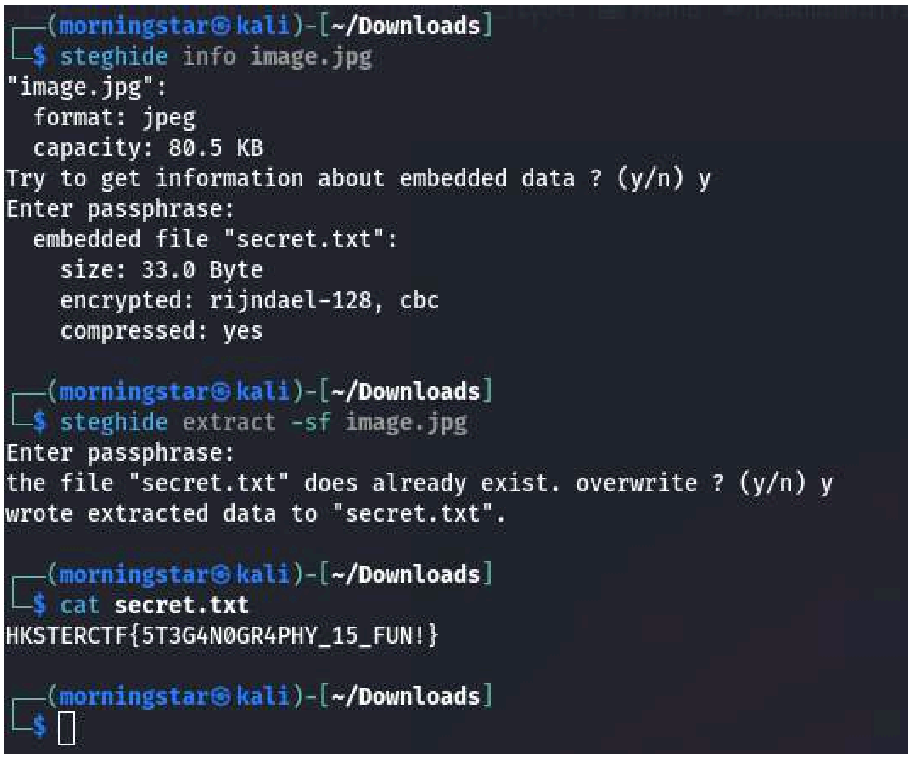
There's a secret, but it's locked.
Hint: Try with password change
HKSTERCTF -> HKSTR and also remove "!"

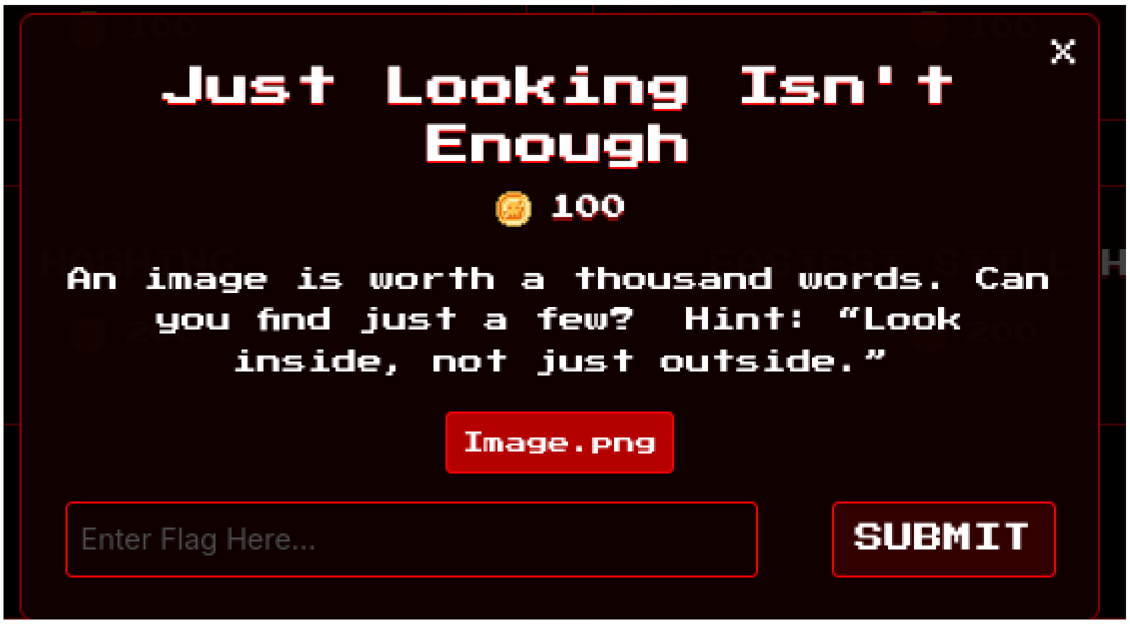
🕵️ CTF Writeup – Steganography Challenge: *"Hidden Secret"*

📄 Challenge Description:

> *There's a secret, but it's locked.*
> **Hint**: Try with password

We were given an **image**, and the challenge hinted that there was a secret hidden inside it. The word **"password"** in the hint seemed suspicious — likely it was the actual password.We checked the file format (e.g., `.jpg`, `.wav`, etc.) and suspected **Steghide** might have been used for embedding the hidden data.


```
┌──(morningstar㉿kali)-[~/Downloads]
└─$ steghide info image.jpg
"image.jpg":
  format: jpeg
  capacity: 80.5 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "secret.txt":
    size: 33.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes

┌──(morningstar㉿kali)-[~/Downloads]
└─$ steghide extract -sf image.jpg
Enter passphrase:
the file "secret.txt" does already exist. overwrite ? (y/n) y
wrote extracted data to "secret.txt".

┌──(morningstar㉿kali)-[~/Downloads]
└─$ cat secret.txt
HKSTERCTF{5T3G4N0GR4PHY_15_FUN!}

┌──(morningstar㉿kali)-[~/Downloads]
└─$ 
```



Just Looking Isn't Enough

100

An image is worth a thousand words. Can you find just a few? Hint: "Look inside, not just outside."

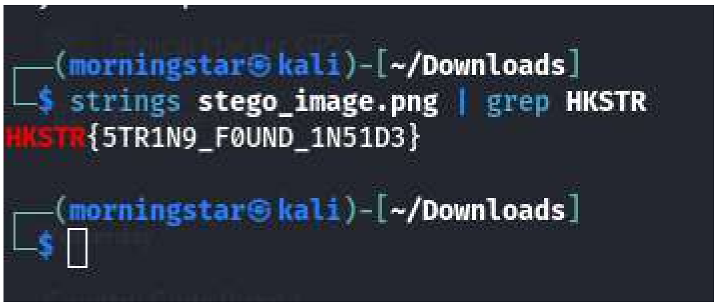
🧵 1. Use `strings` Command

We used the `strings` command to extract all readable text from the image file:

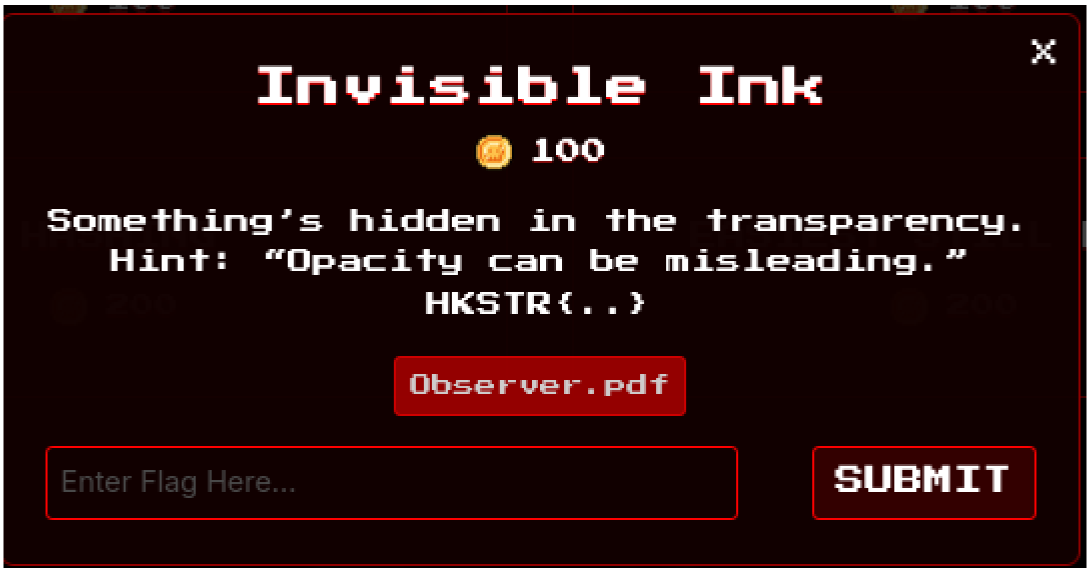
This command prints all ASCII strings from the binary file, which can sometimes contain hidden messages, flags, or clues.

If you're on Windows, you can also use online tools like:

- https://strings.utilitymill.com/
- Or use **Notepad++** and enable "Show All Characters" to scroll through hidden text.



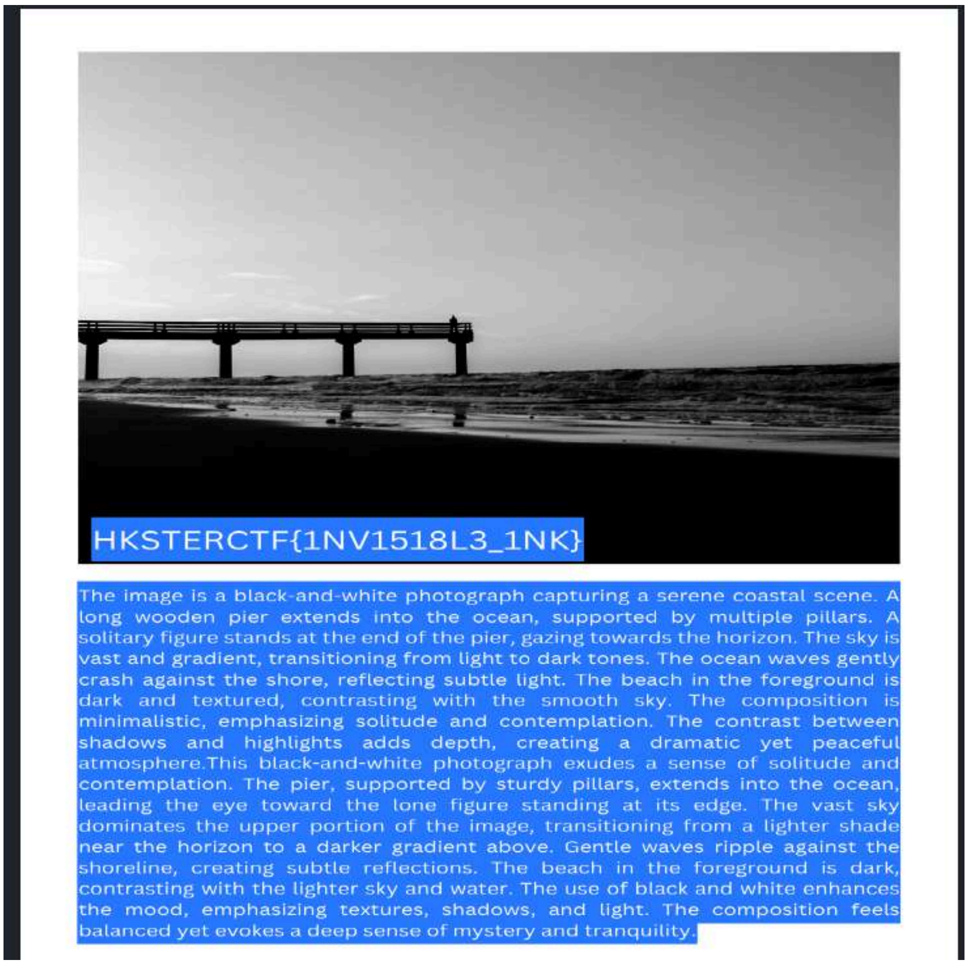
```
┌──(morningstar㉿kali)-[~/Downloads]
└─$ strings stego_image.png | grep HKSTR
HKSTR{5TR1N9_F0UND_1N51D3}

┌──(morningstar㉿kali)-[~/Downloads]
└─$ 
```


Invisible Ink

100

Something's hidden in the transparency.
Hint: "Opacity can be misleading."
HKSTR{..}

📄 **Challenge Description:**


HKSTERCTF{1NV1518L3_1NK}

The image is a black-and-white photograph capturing a serene coastal scene. A long wooden pier extends into the ocean, supported by multiple pillars. A solitary figure stands at the end of the pier, gazing towards the horizon. The sky is vast and gradient, transitioning from light to dark tones. The ocean waves gently crash against the shore, reflecting subtle light. The beach in the foreground is dark and textured, contrasting with the smooth sky. The composition is minimalistic, emphasizing solitude and contemplation. The contrast between shadows and highlights adds depth, creating a dramatic yet peaceful atmosphere.This black-and-white photograph exudes a sense of solitude and contemplation. The pier, supported by sturdy pillars, extends into the ocean, leading the eye toward the lone figure standing at its edge. The vast sky dominates the upper portion of the image, transitioning from a lighter shade near the horizon to a darker gradient above. Gentle waves ripple against the shoreline, creating subtle reflections. The beach in the foreground is dark, contrasting with the lighter sky and water. The use of black and white enhances the mood, emphasizing textures, shadows, and light. The composition feels balanced yet evokes a deep sense of mystery and tranquility.

A PDF file was given. Nothing was visible at first glance.
But remember: **What you see isn't always what you get.**

🔍 Step-by-Step Solution:

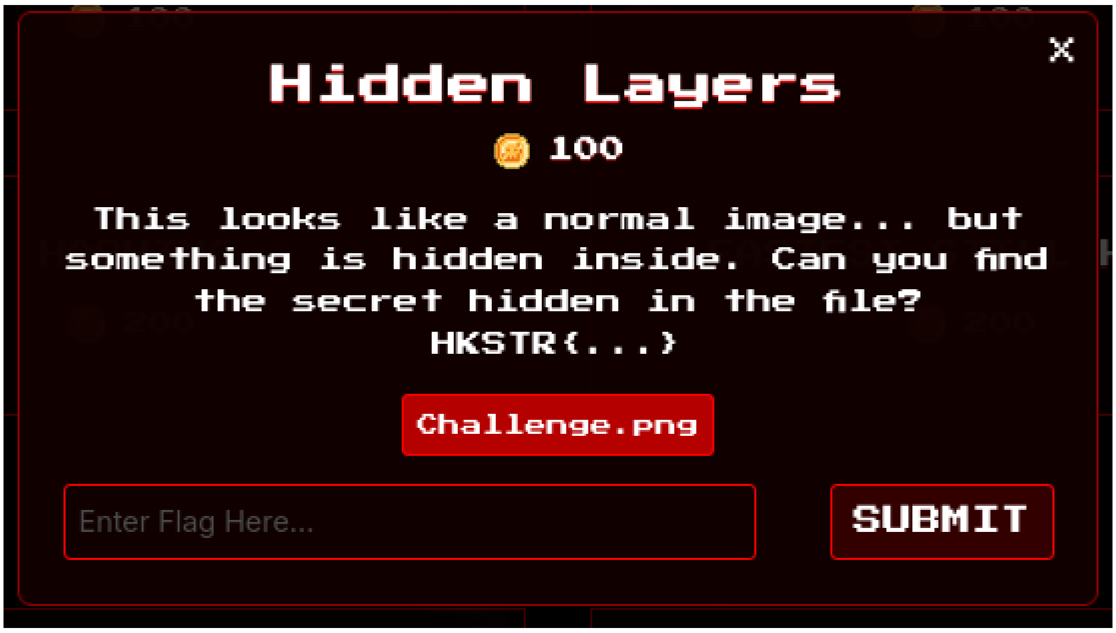
When we opened the PDF file, it appeared to be **completely black** or **empty** — no visible flag or hint.

🖱️ 1. Try Select All

We suspected the flag might be hidden in **black-colored text** on a **black background**, making it invisible to the eye.

So we did:

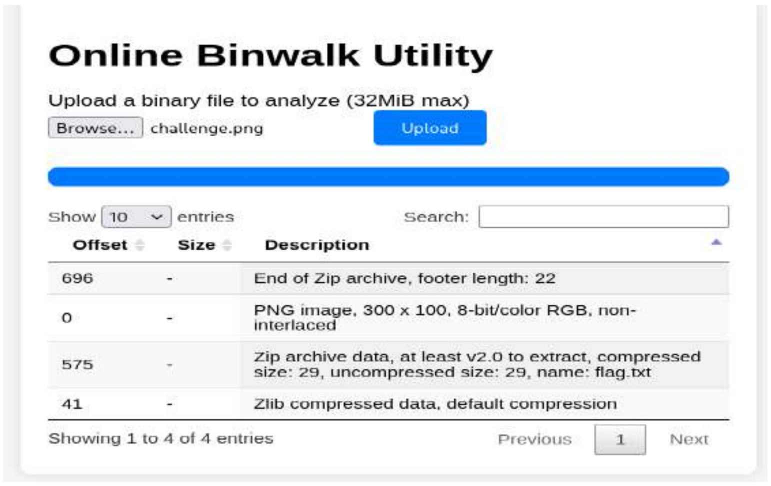
- Opened the PDF in any viewer (e.g., browser or Adobe Reader).
- Pressed `Ctrl + A` (Select All).
- Then pressed `Ctrl + C` (Copy).
- Opened a text editor (like Notepad) and pressed `Ctrl + V` (Paste).


Hidden Layers

100

This looks like a normal image... but something is hidden inside. Can you find the secret hidden in the file? HKSTR{...}

This challenge hinted that the given file might contain **embedded files or data** inside it — a perfect use case for the tool **Binwalk**.



Online Binwalk Utility

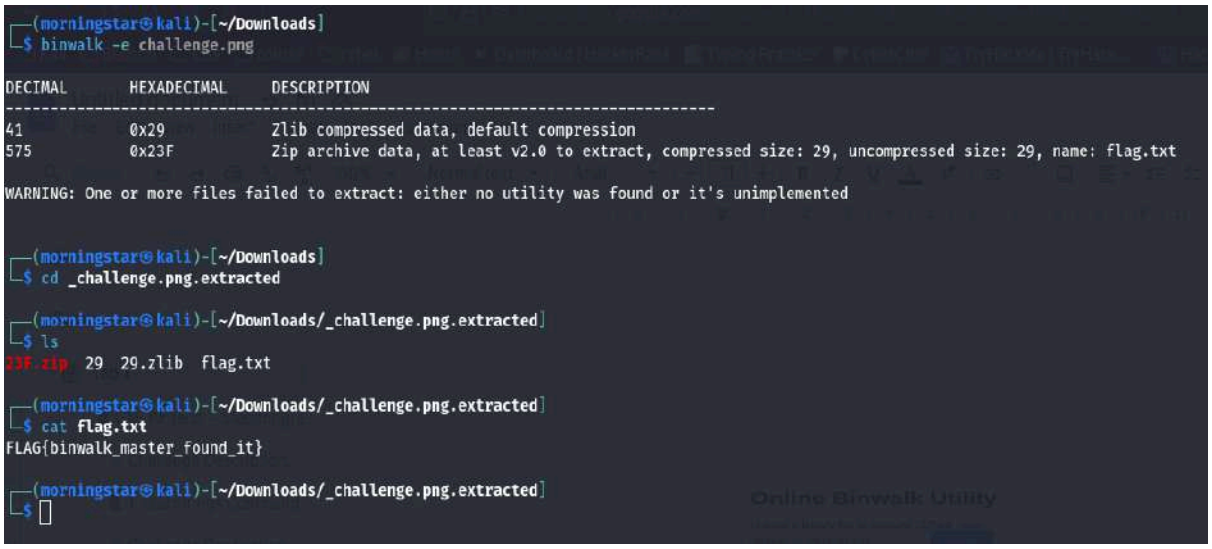
Upload a binary file to analyze (32MiB max)

Browse... challenge.png Upload

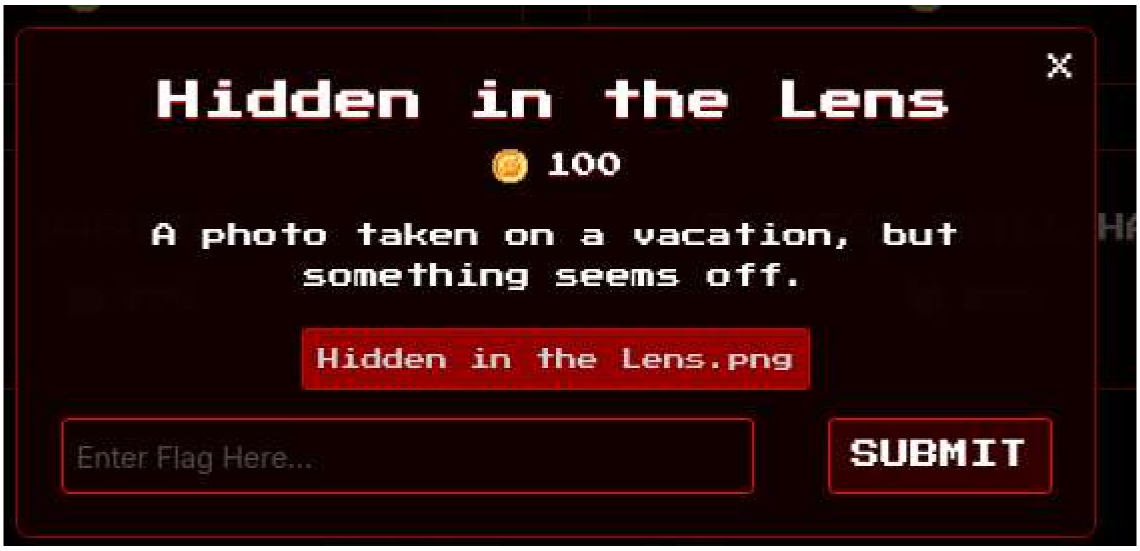
Show 10 entries Search:

Offset	Size	Description
696	-	End of Zip archive, footer length: 22
0	-	PNG image, 300 x 100, 8-bit/color RGB, non-interlaced
575	-	Zip archive data, at least v2.0 to extract, compressed size: 29, uncompressed size: 29, name: flag.txt
41	-	Zlib compressed data, default compression

Showing 1 to 4 of 4 entries Previous 1 Next



```
┌──(morningstar㉿kali)-[~/Downloads]
└─$ binwalk -e challenge.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
41            0x29            Zlib compressed data, default compression
575           0x23F           Zip archive data, at least v2.0 to extract, compressed size: 29, uncompressed size: 29, name: flag.txt

WARNING: One or more files failed to extract: either no utility was found or it's unimplemented


┌──(morningstar㉿kali)-[~/Downloads]
└─$ cd _challenge.png.extracted

┌──(morningstar㉿kali)-[~/Downloads/_challenge.png.extracted]
└─$ ls
23F.zip  29  29.zlib  flag.txt

┌──(morningstar㉿kali)-[~/Downloads/_challenge.png.extracted]
└─$ cat flag.txt
FLAG{binwalk_master_found_it}

┌──(morningstar㉿kali)-[~/Downloads/_challenge.png.extracted]
└─$ 
```

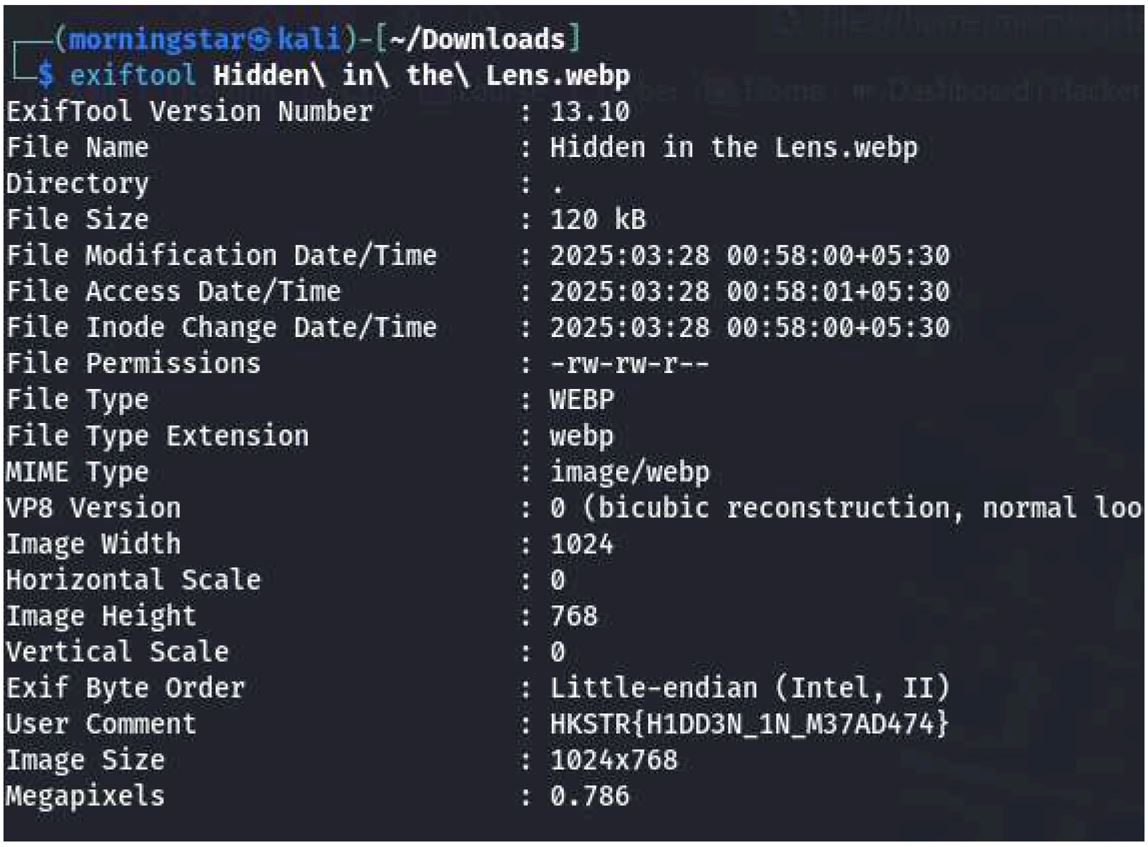


Hidden in the Lens

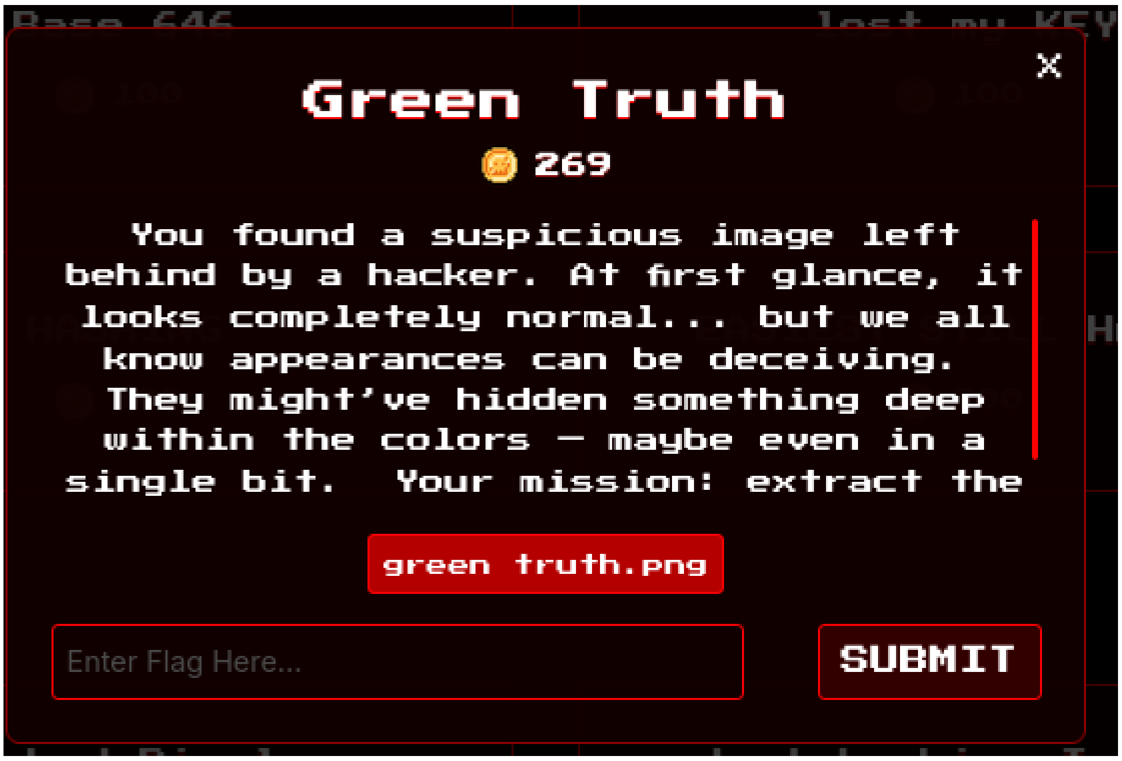
100

A photo taken on a vacation, but something seems off.

Something seems off" + title "Hidden in the Lens" points toward **EXIF metadata**, which often contains information from the camera
Flag is in the metadata.



```
┌──(morningstar㉿kali)-[~/Downloads]
└─$ exiftool Hidden\ in\ the\ Lens.webp
ExifTool Version Number         : 13.10
File Name                       : Hidden in the Lens.webp
Directory                       : .
File Size                       : 120 kB
File Modification Date/Time     : 2025:03:28 00:58:00+05:30
File Access Date/Time           : 2025:03:28 00:58:01+05:30
File Inode Change Date/Time     : 2025:03:28 00:58:00+05:30
File Permissions                : -rw-rw-r--
File Type                       : WEBP
File Type Extension             : webp
MIME Type                       : image/webp
VP8 Version                     : 0 (bicubic reconstruction, normal loo
Image Width                     : 1024
Horizontal Scale                : 0
Image Height                    : 768
Vertical Scale                  : 0
Exif Byte Order                 : Little-endian (Intel, II)
User Comment                    : HKSTR{H1DD3N_1N_M37AD474}
Image Size                      : 1024x768
Megapixels                      : 0.786
```

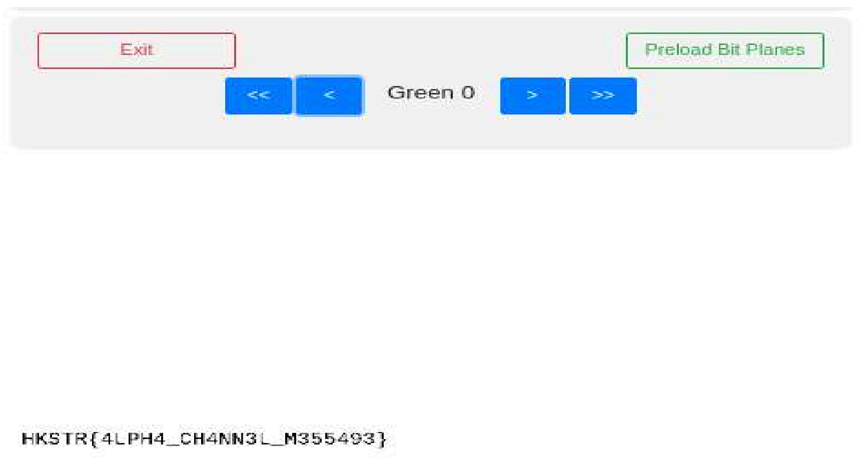


Green Truth

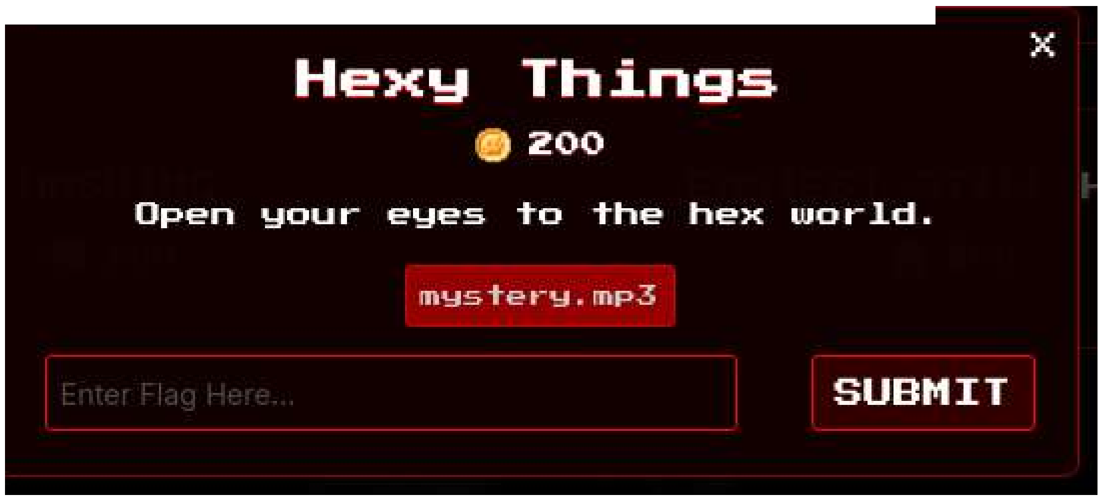
269

You found a suspicious image left behind by a hacker. At first glance, it looks completely normal... but we all know appearances can be deceiving. They might've hidden something deep within the colors — maybe even in a single bit. Your mission: extract the

This challenge is based on **Least Significant Bit (LSB) steganography**, where data is hidden in the lowest bits of pixel values green 0.



HKSTR{4LPH4_CH4NN3L_M355493}



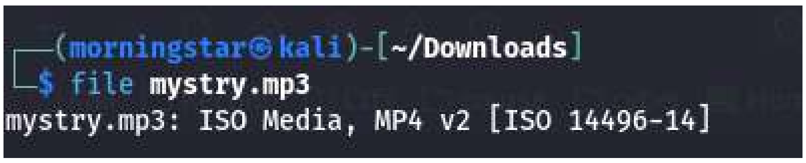
This challenge is based on **file signature spoofing** — where the **file extension and magic bytes** are intentionally tampered with to hide its true format.

🧪 1. Identify the True File Type

Although the file is named `mystery.mp3`, the description hinted it's really an **MP4** file.

We ran:

file mystery.mp3



```
┌──(morningstar㉿kali)-[~/Downloads]
└─$ file mystry.mp3
mystry.mp3: ISO Media, MP4 v2 [ISO 14496-14]
```

the signature is broken also — the **magic bytes** at the start are not matching a known file type.

🧬 2. Check Magic Bytes in a Hex Editor

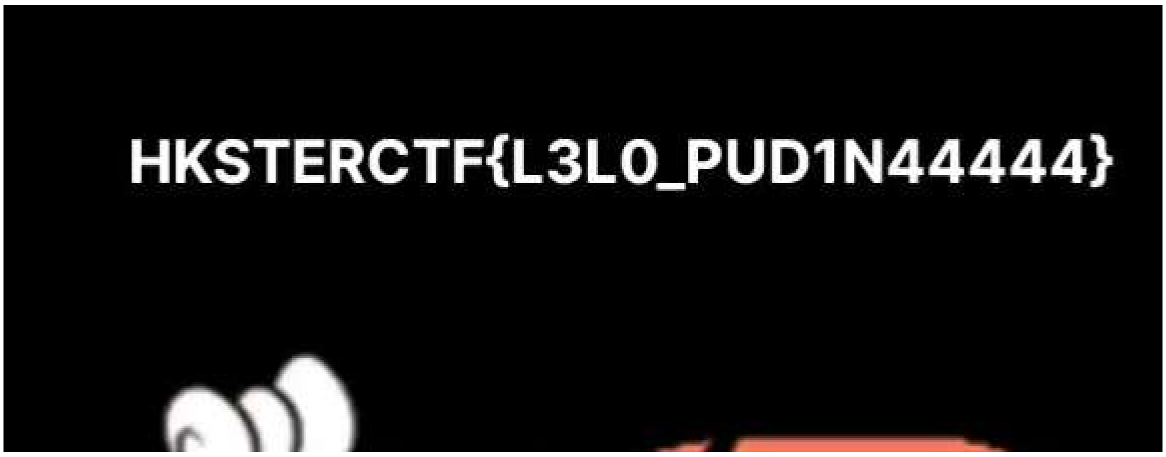
We opened the file in a hex editor like:

- **HxD (Windows)**
- **hexedit** or **xxd** (Linux)
- **bless** or **GHex (Linux GUI)**

🛠️ 3. Fix the Magic Bytes

We manually edited the first few bytes of the file to match a valid MP4 file signature, such as:

First 6 digit: 00 00 00



HKSTERCTF{L3L0_PUD1N44444}