CTF_1 Write-ups

OSTIN Write-up

What is Open-Source Intelligence? 🤔🔍

Imagine being a detective with a magnifying glass, scouring the public world for hidden clues... that's essentially what **Open-Source Intelligence (OSINT)** is all about! 📡

OSINT refers to intelligence gathered from publicly available sources. We're talking about everything from social media, blogs, websites, forums, and even Google. The goal? To collect, analyze, and make sense of this publicly available info to answer specific questions or solve problems in the world of cybersecurity. Think of it as using your sleuth skills to connect the dots!

Source: SANS Blog

Who's All About OSINT? 🤓🔐

A whole crew of cybersecurity pros rely on OSINT to stay ahead of the bad guys! Here's who's in the OSINT squad:

- **Information Security**: The defenders of the digital realm, using OSINT to spot vulnerabilities before they're exploited.
- **Cyber Threat Intelligence**: These folks keep their eyes peeled for emerging threats and malicious actors, using public data to predict the next attack.
- **Pen Testers**: Ethical hackers who use OSINT to map out the target system and look for weak spots (all in the name of security, of course!).
- **Social Engineers Awareness**: The masterminds who study online personas to manipulate or trick individuals into revealing sensitive info.

So whether you're protecting systems, investigating threats, or just trying to learn more about your digital footprint, OSINT is your go-to toolkit. And guess what? You'll use it in this CTF to find all kinds of clues! 🕵️🔎

Ready to dive in? Let's get those cyber sleuthing skills sharpened! 😎

Challenge 1: "KF - The Masterpiece near the Mosque" 🏰🕌

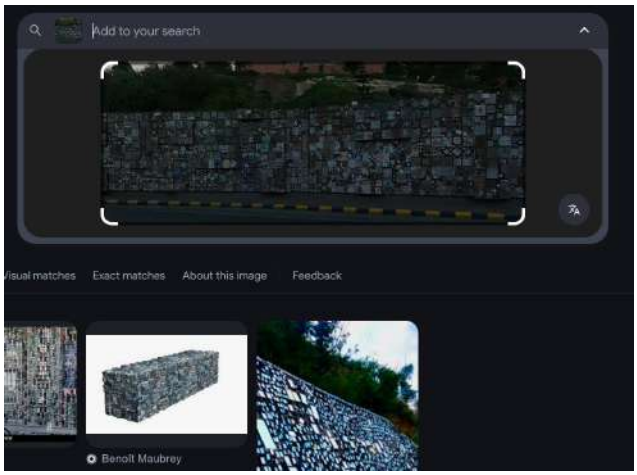
This is a masterpiece located near a well-known mosque. I wonder what it's called?

Flag{mosque_name}

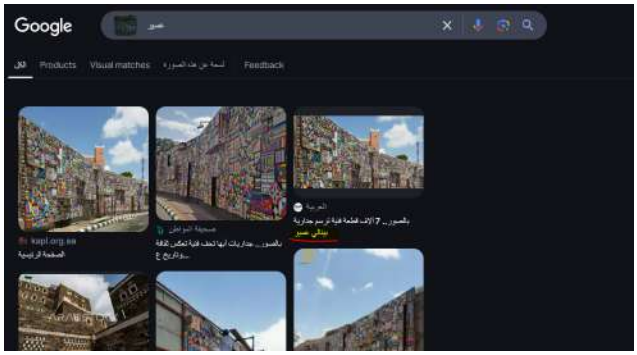
To solve this, we can look for clues or hints in the picture or even in the question itself. It mentions a masterpiece located near a well-known mosque. This part suggests that it's a popular place or street. We can see part of the street in the photo, which gives us a solid clue to move forward with!

An image search engine is one of the tools we can use to find the image, and it's available pretty much everywhere. But for now, let's stick with Google tools like Image Search and Google Earth.

Let’s begin with image search.



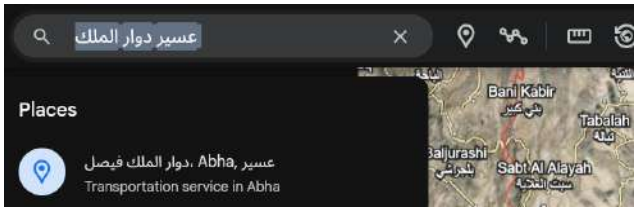
It didn’t show the result we wanted, and that’s because we need to be more specific. Let’s search using the name of the region, "Asir.”



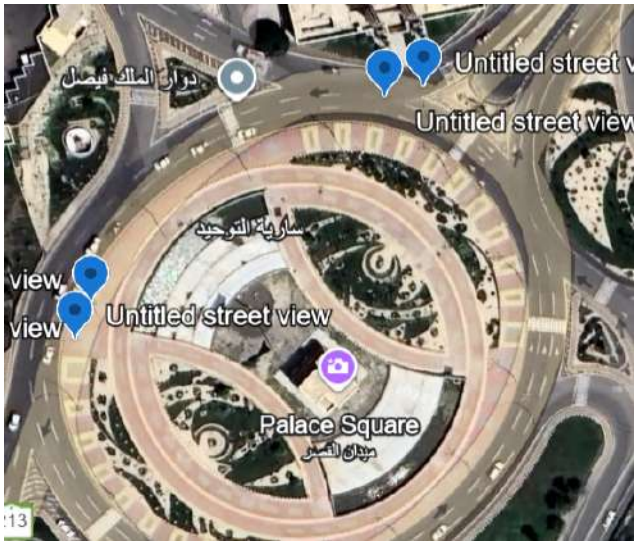
And here we are! We can see even the same building behind! 🙂

Since we’ve found the name of our masterpiece, the next step is to find the street name.

To take a clear look at this place—nothing beats Google Earth for this task!



Using Street View, we get a clear glimpse.



And done! The flag is here:

Flag: `Flag{` King Faisal Mosque `}` 🚩

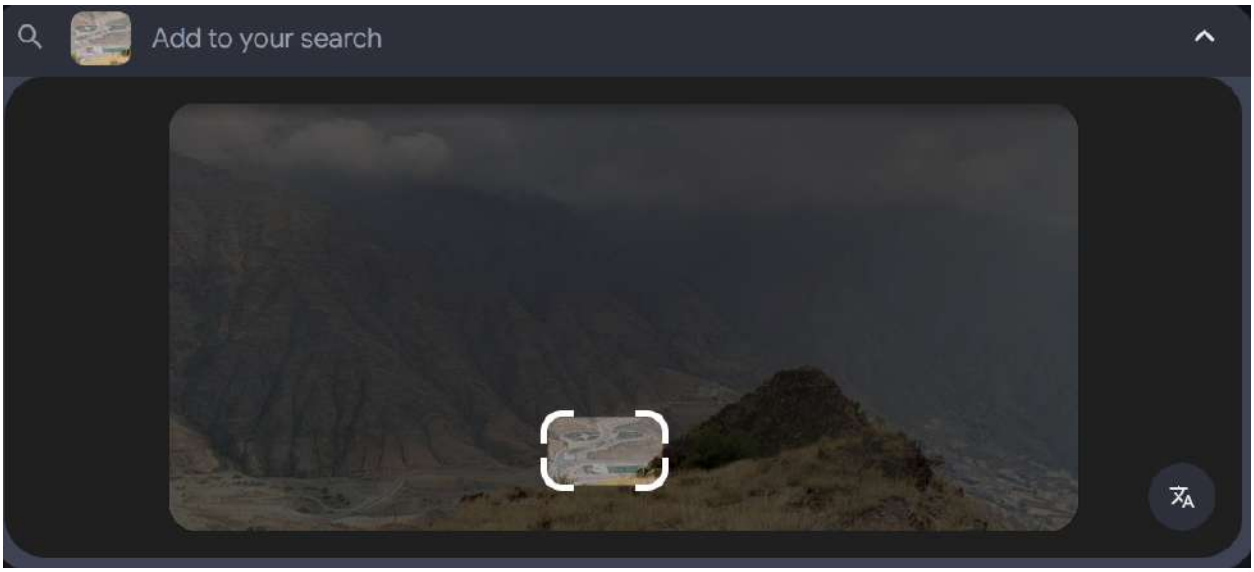
PA Challenge: "Nice View, Right? Do You Know What This Road is Called?" 🌇🚗

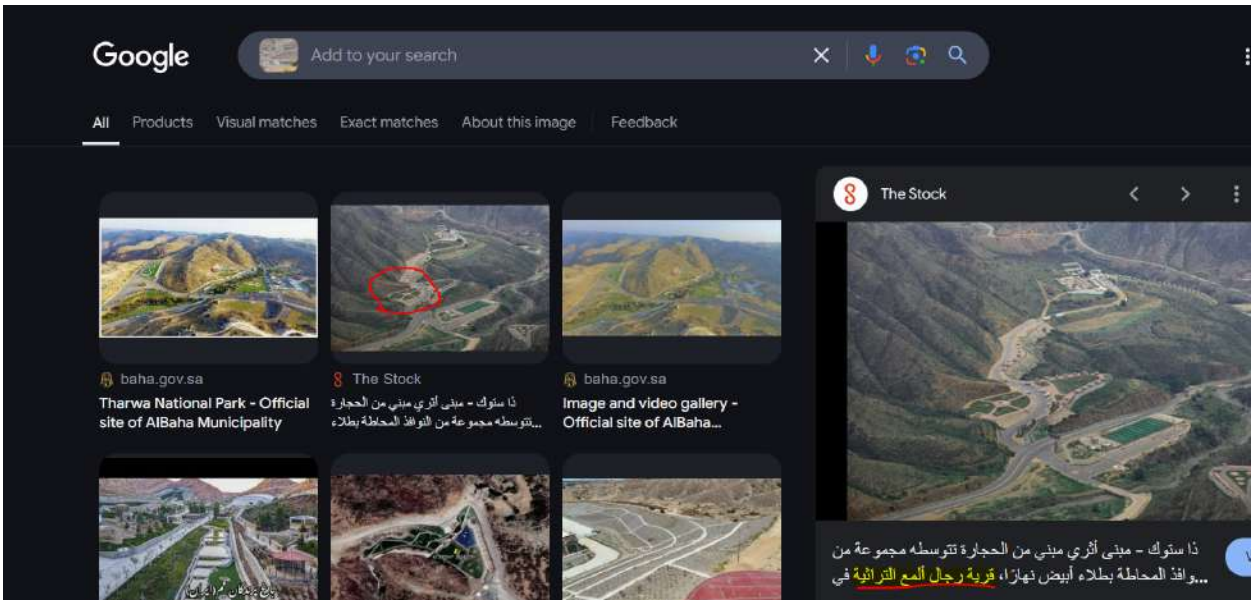
This one might seem a bit tricky because of the mountains, but trust me—it'll be easier than it seems! 🤔 Take a deep look at the picture. What do you notice about the mountains? What can you deduce from the image?

We can see two key things: a road and what looks like a rest area or park. These are our main clues.

Let's start with a trusty OSINT tool—**Image Search**. 🕵️

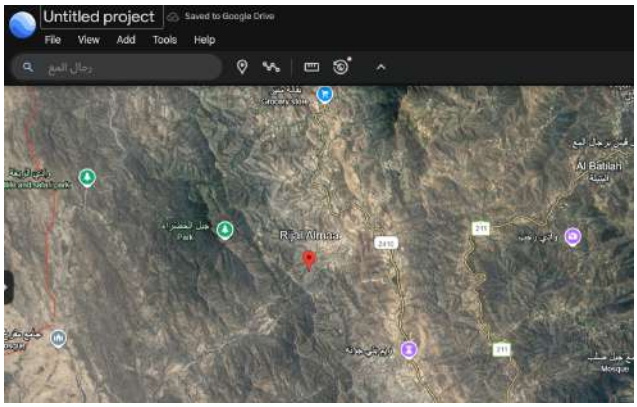
Zoom in.





Looks like the same park from the picture! This gives us a strong clue about where to search in Google Earth.

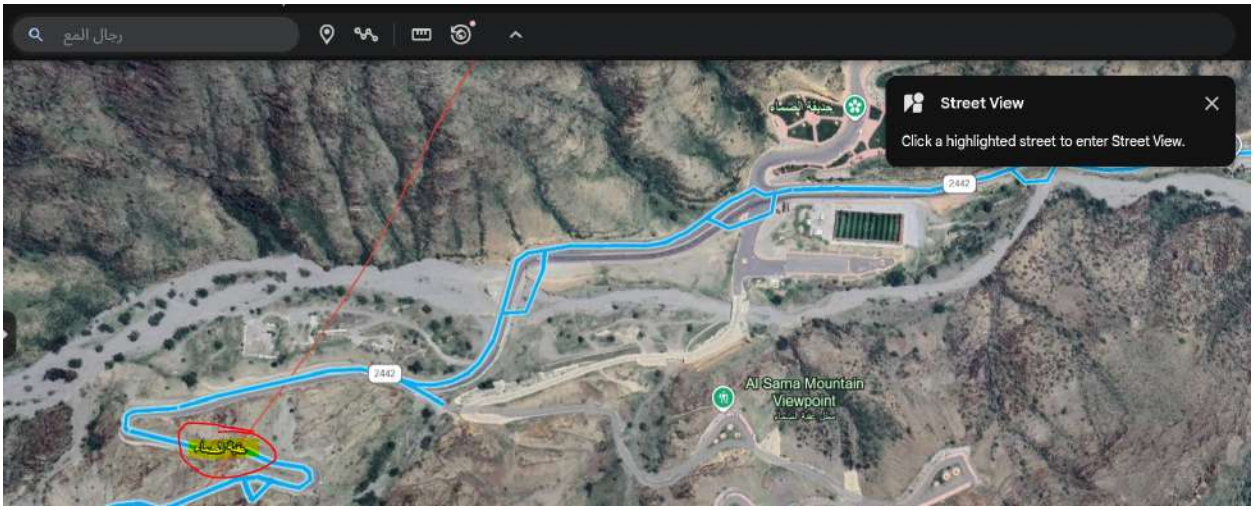
As you can see, there's no street view available in the area. But, since the image was taken from a high vantage point, we should focus on nearby mountains.



There are a few locations with street view near the village, but the best option is the closest one.

Let’s try turning the view a bit to get a clearer perspective.

And there it is! But wait, we still need the name of the road. No worries—just zoom out a bit.



Flag: `Flag{Agabah Al Sama}` 🚩

Tips for Solving OSINT Challenges Easily: 🧠💡

1. **Identify the Main Object**: 🔍

   Focus on key features in the image (e.g., buildings, roads, landmarks). These are your main clues.

2. **Be Specific with Your Search**: 🌐

   Narrow your search by including relevant details (e.g., exact locations, landmarks, or region names).

3. **Use Reverse Image Search**: 📸

If you're stuck, use tools like Google Image Search to find similar images or context.

4. **Leverage Google Earth and Street View**: 🌍

   Zoom in on locations using Street View for a clearer perspective or Google Earth for a broader overview.

5. **Look for Hidden Clues**: 🕵️

   Check for small details like street signs, objects, or labels that could lead to more information.

6. **Cross-reference Information**: 🔄

   Use multiple tools or resources to verify your findings and ensure accuracy.

7. **Think Like a Detective**: 🕵️

   Combine your knowledge and puzzle-solving skills to make connections between clues.

Crypto CTF

Cabbage

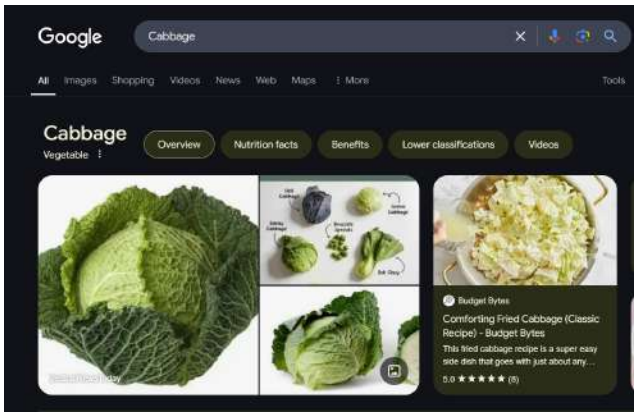
GUZCANTEEA3TQIBWHAQDKYJAGMZSANZUEAZTGIBVMEQDKNRAGJTCANRZEA3GIIBWGEQDINRAGY3CANJZEAZTGIBUM

Is the simplest way to solve this by guessing what we have here? 🤔

The name of the challenge is "Cabbage," but why? 🥬

What's the deal with cabbage?

Let's take a look:



We can see that it has layers.

So what?

How could this be related to crypto?

Later on, we were provided with a hint, and it all became clear:

HINT ⭐**:** "Cabbage is a vegetable with layers, just like crypto can have layers of different encoding methods."

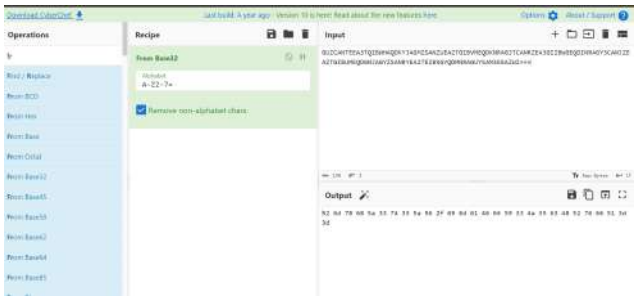
But we still need to guess the encoding method.

Just by looking, we can tell this is likely one of the base encoding methods, but which one?

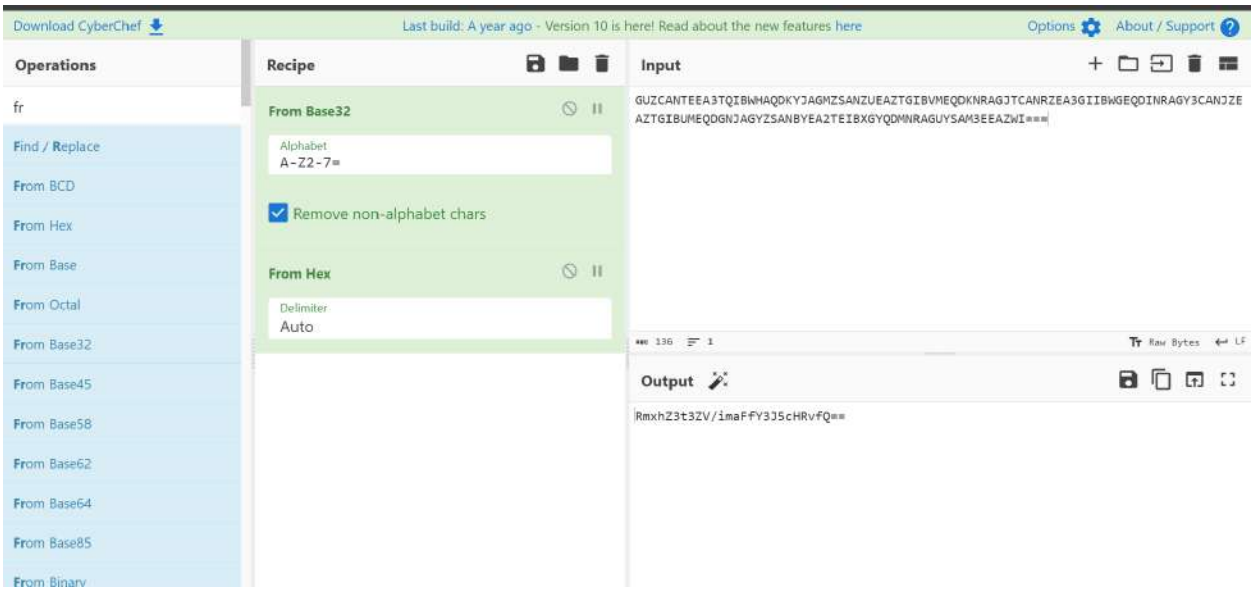
The most commonly used encoding methods in CTF challenges are base64 and base32. At least, that's what I've often encountered when dealing with base encoding.

Let’s give it a shot.

Cool, it’s working!



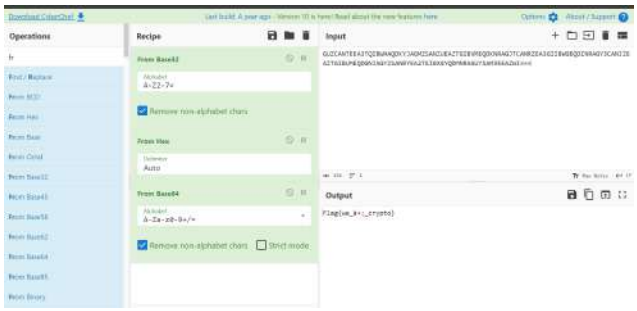
Now, we can see the characters are in a range of numbers and letters from 1 to f, which suggests it’s hexadecimal.



That also looks like it could be base64.

Let’s try base64 now.

And we find the flag!



Flag: `Flag{we_â¡_crypto}` 🚩

Tips for Solving Crypto CTF Challenges: 🧠💡

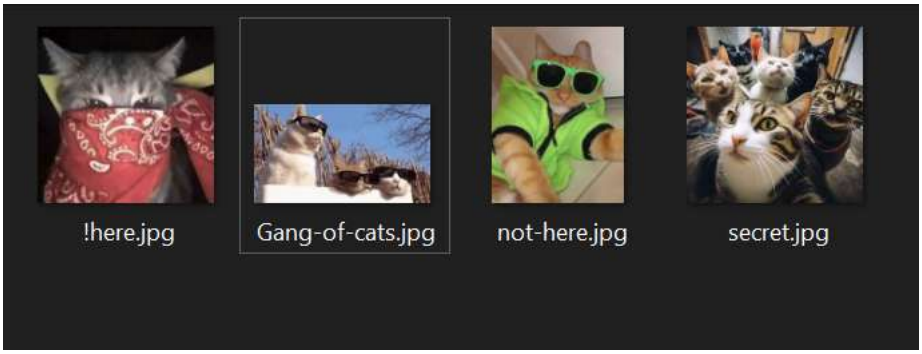
- **Look for Patterns:** Ciphertexts often follow patterns. Common encodings like base64, base32, and hexadecimal are easy to spot.
- **Use Online Tools:** Tools like CyberChef are great for quickly testing different encoding formats.
- **Know Common Encodings**

Forensics CTF

Gang-of-Cats

Our special agent has come across a suspicious file. Can you help uncover the secret hidden within? 🕵️

Inside the 7z file, we find **four images of cats**. 🐱



Each image has a weird name, but the most intriguing one is **secret.jpg**. 🤔

Let's first make sure there's nothing hidden in the folder.

```
└─$ ls -la
total 36
drwxrwxrwx 1 catroo catroo   512 Jan 10 20:20 .
drwxrwxrwx 1 catroo catroo   512 Jan  8 16:28 ..
-rwxrwxrwx 1 catroo catroo  7290 Jan  8 14:45 'Gang-of-cats.jpg'
-rwxrwxrwx 1 catroo catroo  6637 Jan  8 13:10 '!here.jpg'
-rwxrwxrwx 1 catroo catroo  7427 Jan  8 13:40  not-here.jpg
-rwxrwxrwx 1 catroo catroo 12069 Jan  8 13:13  secret.jpg
```

Hmm... nothing suspicious here, just images.

Next, let's open
secret.jpg.

Still nothing. But I have a
hunch there's something more here.

Let's try opening the other images to be sure.

None of these are showing any secrets... but wait—**the hint for this challenge** was:

"Sometimes answers are hidden in plain sight. Look beyond the surface where a mystery awaits."

Could the secret be hidden in **metadata**? 🔍

💡What's metadata? It's essentially data about the file—think of it as the file's "details." More on metadata here:

- File system metadata
- Installing and using exiftool on Linux

I'll use **ExifTool** (since we're on Linux) to check out the file's metadata.

```
└─$ exiftool secret.jpg
ExifTool Version Number         : 13.00
File Name                       : secret.jpg
Directory                       : .
File Size                       : 12 kB
File Modification Date/Time     : 2025:01:08 13:13:10-08:00
File Access Date/Time           : 2025:01:10 20:26:39-08:00
File Inode Change Date/Time     : 2025:01:08 13:42:30-08:00
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                      : not here !!
Padding                         : (Binary data 268 bytes, use -b option to extract)
Image Width                     : 225
Image Height                    : 225
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 225x225
Megapixels                      : 0.051
```

Aha!
Secret.jpg has a suspicious comment! 🤫

Let's now check out the other images. First, we take a look at

not-here.jpg.

```
└─$ ls
Gang-of-cats.jpg  '!here.jpg'   not-here.jpg   secret.jpg
```

```
└─$ exiftool not-here.jpg
ExifTool Version Number         : 13.00
File Name                       : not-here.jpg
Directory                       : .
File Size                       : 7.4 kB
File Modification Date/Time     : 2025:01:08 13:40:46-08:00
File Access Date/Time           : 2025:01:10 20:28:28-08:00
File Inode Change Date/Time     : 2025:01:08 13:42:21-08:00
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                     : 194
Image Height                    : 259
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 194x259
Megapixels                      : 0.050
```

It looks like it has no metadata, but maybe there's something else going on here? 🤔

Could the secret be hidden within the image file itself? I found a guide on how to hide text in images, but can we hide **files**? 🧐

💡Steganography

- Image steganography using StegoSuite on Linux

We know now that there might be a hidden file inside **not-here.jpg**.

Let's use
Steghide, a tool for steganography, to see if there's a hidden file inside **not-here.jpg**.

```
└─$ steghide extract -sf not-here.jpg
Enter passphrase:
steghide: could not extract any data with that passphrase!
```

It asks for a password. Let's check the other image for the password. 🐾

```
└─$ exiftool Gang-of-cats.jpg
ExifTool Version Number         : 13.00
File Name                       : Gang-of-cats.jpg
Directory                       : .
File Size                       : 7.3 kB
File Modification Date/Time     : 2025:01:08 14:45:07-08:00
File Access Date/Time           : 2025:01:10 20:29:13-08:00
File Inode Change Date/Time     : 2025:01:08 14:45:07-08:00
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)
Artist                          : xxrooziixx
XP Comment                      : Password:kitekat
XP Author                       : xxrooziixx
Padding                         : (Binary data 268 bytes, use -b option to extract)
About                           : uuid:faf5bdd5-ba3d-11da-ad31-d33d75182f1b
Creator                         : xxrooziixx
```

I found the password hidden in
Gang-of-cats.jpg!

```
└─$ steghide extract -sf not-here.jpg
Enter passphrase:
wrote extracted data to "Flag.txt".
```

After entering the password, I finally extract the hidden file. And voilà, we have the flag! 🎉

```
└─$ cat Flag.txt
Flag{Easy_right!}
```

Flag: `Flag{Easy_right!}` 🚩

Key Takeaways:

- **Check the metadata:** The metadata often hides clues that can lead you to the secret.
- **Look beyond the surface:** Sometimes, the answers are hidden in plain sight (or in plain metadata).
- **Steganography can be sneaky:** Files can be hidden within images—don't forget to check.

This is everything about this CTF competition. Need help with anything CTF-related? Hit me up anytime! 😎

Created by: @XxrooziixX