****

**CT126-3-2-CTF-PRACTICAL CTF STRATEGIES**

**GROUP ASSIGNMENT (40%)**



**Incognito CTF 6.0**

**Challenges Write-ups for Digital Forensics and Cryptography**

|  |  |  |
| --- | --- | --- |
| **GROUP NAME:** | **Sestra** | |
| **TP NUMBER** | **NAME** | **WORKLOAD** |
| TP069365 | KHEGAI OLGA | 50% |
| TP071009 | NIYAZALY ZHASSULAN | 50% |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Content**

[**1.0 Introduction** 3](#_Toc197985819)

[**1.1 Group Selfie 1** 4](#_Toc197985820)

[**1.2 Group Selfie 2** 5](#_Toc197985821)

[**1.3 Group Selfie 3** 6](#_Toc197985822)

[**2.0 Challenge: Wierd disk Image WriteUp (Difficulty – Hard, Forensics)** 7](#_Toc197985823)

[**2.1 Our Understanding of the Challenge** 9](#_Toc197985824)

[**3.0 Step-by-Step Solution** 10](#_Toc197985825)

[**4.0 Challenges and Experiences in Live CTF as Beginners** 29](#_Toc197985826)

[**5.0 Conclusion** 30](#_Toc197985827)

[**6.0 References** 31](#_Toc197985828)

[**7.0 Workload Matrix / Task Distribution** 32](#_Toc197985829)

## **1.0 Introduction**

Figure 1.0: Introduction

We have participated in The Incognito CTF 6.0 competition that was held on April 28 to April 29. Incognito CTF is a competitive cybersecurity event designed to test participants' skills across various domains, including web exploitation, reverse engineering, cryptography, and digital forensics. The platform presents a range of challenges with varying difficulty levels, encouraging both novice and experienced participants to engage in problem-solving and knowledge application.

### **1.1 Group Selfie 1**



Figure 1.1: Selfie number 1

### **1.2 Group Selfie 2**

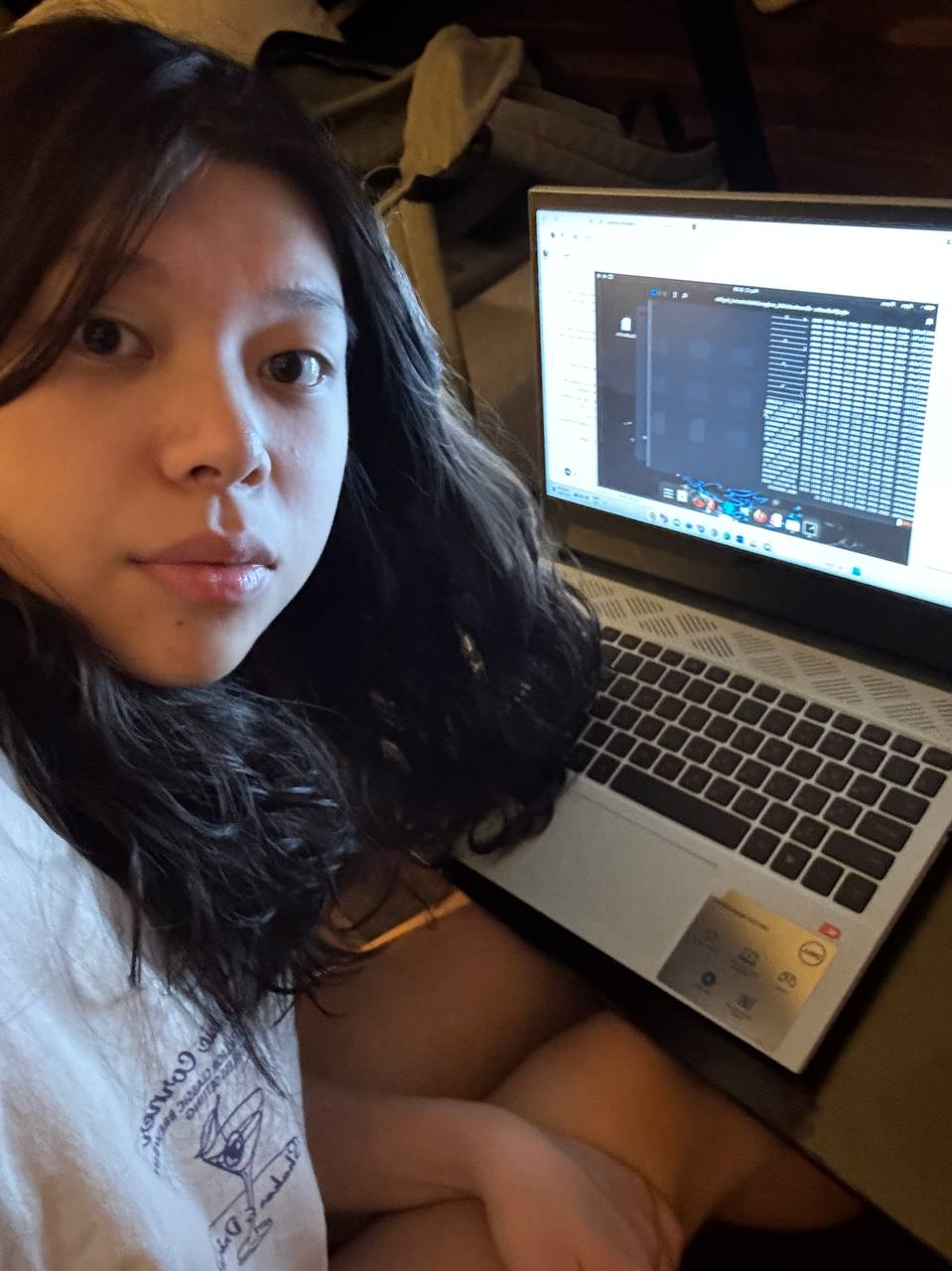


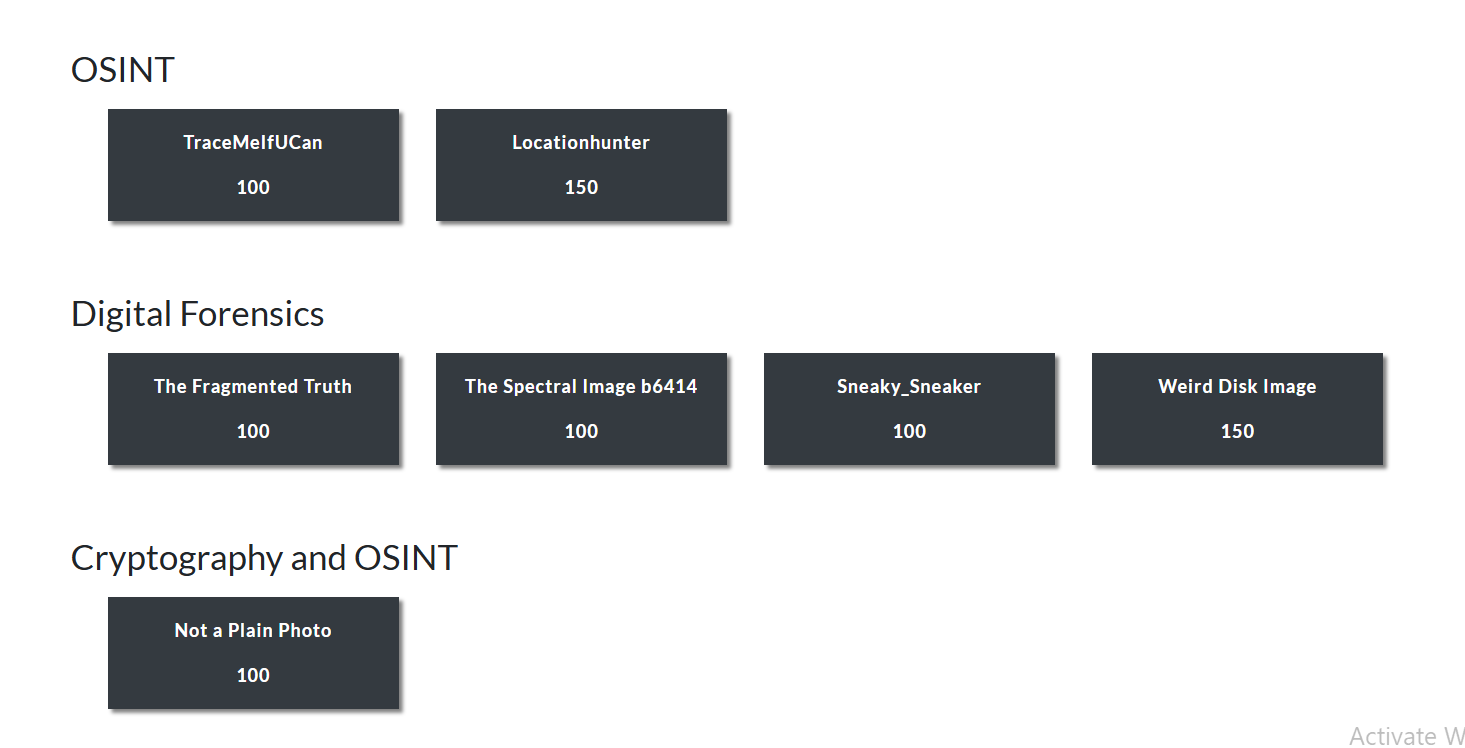
Figure 1.2: Selfie number 2

### **1.3 Group Selfie 3**

Figure 1.3: Selfie number 3

## **2.0 Challenge: Wierd disk Image WriteUp (Difficulty – Hard, Forensics)**

Figure 2.0.1: Challenge

Figure 2.0.2: Challenge cont.

In this CTF, our team has decided to choose Weird Disk Image which is one of the hard questions and it gives 150 points. And in this ctf competition we have only 2 challenges which gives 150 points. I selected this specific challenge because of its difficulty (worth **150 points**) and its requirement for multiple forensic steps, including disk analysis, data carving, and flag extraction. It seemed like a perfect opportunity to test and apply my skills in disk image investigation using Linux and forensic tools.

### **2.1 Our Understanding of the Challenge**

Upon downloading the challenge, I was given a disk image file that seemed normal on the surface. However, as the challenge name suggests, there was something “weird” about it. The task was to thoroughly analyze the disk image, uncover anything unusual or hidden, and ultimately extract the flag.

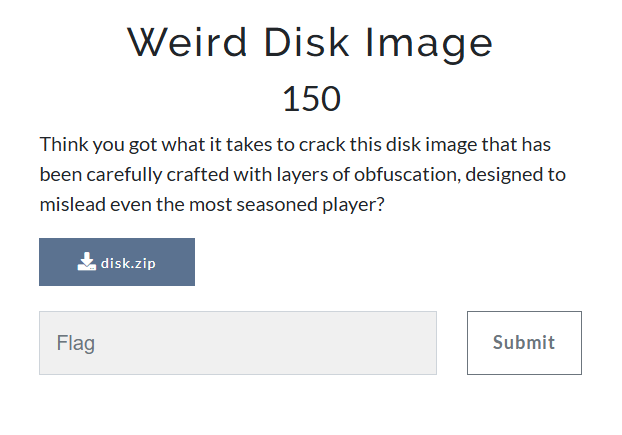


Figure 2.1: Information

## **3.0 Step-by-Step Solution**

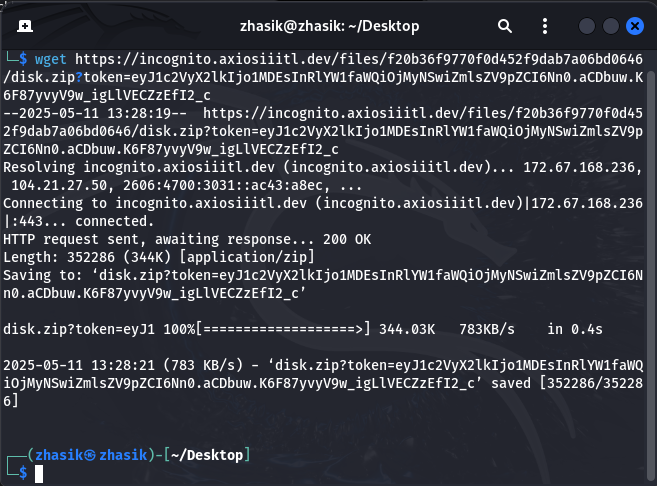


Figure 3.0: Downloading

In this screenshot, I'm on the "Weird Disk Image" challenge page from the Incognito CTF platform. This is a 150-point challenge where I’m asked to analyze a disk image that’s been carefully designed with layers of obfuscation to mislead even experienced players.

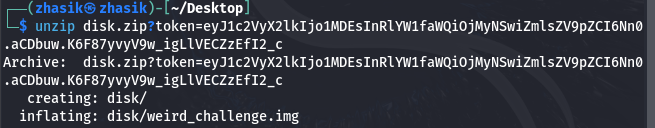
I right clicked on the disk.zip download button to bring up the browser context menu. From there, I had several options, like opening the link in a new tab or window or saving the file directly. I chose “Copy link address” so I could grab the direct URL to the disk image. This was useful for downloading it via terminal or using tools that require a direct link.

This step marked the very beginning of my approach to the challenge — getting the actual disk image so I could start analyzing its contents.

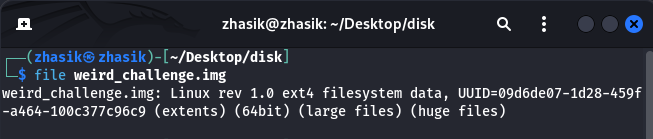
Figure 3.0.1: download

This screenshot shows the moment I downloaded the **disk.zip** file for the Weird Disk Image CTF challenge using the “wget” command in my terminal.

I used the direct download link from the challenge page and ran. This step confirmed that I had successfully obtained the disk image archive and could begin extracting and analyzing its contents for hidden clues or flags.

Figure 3.0.2: Unzipping

This screenshot shows the step where I extracted the contents of the disk.zip archive using the unzip command in my terminal. At this point, I had the raw disk image I needed to start the CTF investigation — the real analysis began from here, digging into the .img file to uncover hidden clues, partitions, and ultimately, the flag. This screenshot shows that I successfully extracted the ZIP file and now have the core file of the challenge. **weird\_challenge.img** sitting inside the disk/ folder on my Desktop. This .img file is a disk image — it’s essentially a snapshot of a Linux filesystem. From this point on, I was ready to start forensic analysis using tools like binwalk, mount, or debugfs to explore the contents, recover hidden files, and eventually uncover the flag for the challenge. This marks the beginning of the real investigation phase.

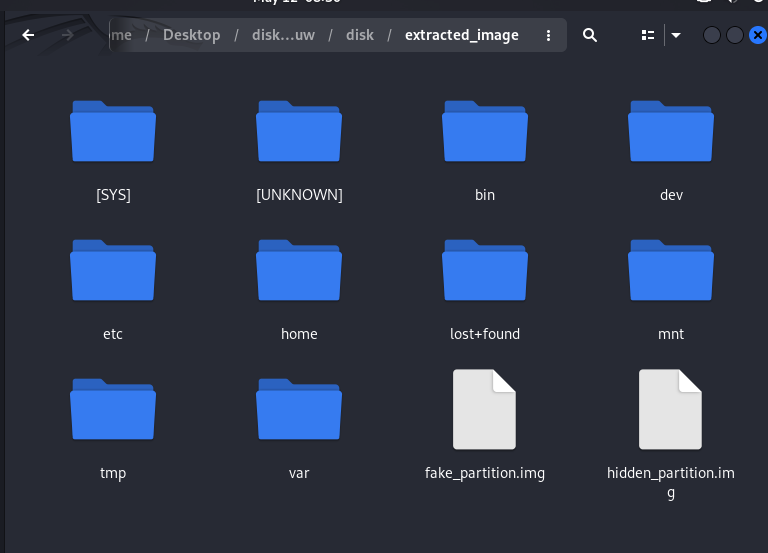
Figure 3.0.3: file

In this screenshot, I used the file command to inspect the type of the weird\_challenge.img file that I extracted earlier. This told me that weird\_challenge.img isn’t just a raw data file — it’s a full Linux disk image.

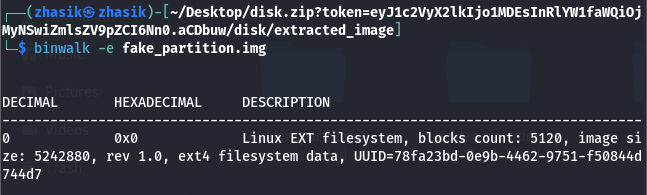
A screenshot of a computer

AI-generated content may be incorrect.Figure 3.0.4: 7zip

In this screenshot, I used 7z (7-Zip) to extract the contents of weird\_challenge.img into a folder called extracted\_image. It contains 18 folders and 20 files, meaning there's a decent amount of content to explore. The filesystem supports features like journaling, large files, metadata checksums, and extended attributes (which aligns with clues I found earlier).

Figure 3.0.5: files

This screenshot shows the fully extracted contents of the weird\_challenge.img disk image, located inside the extracted\_image folder. Let’s start from fake\_partition.img and hidden\_partition.img to check everything.

Figure 3.0.6: binwalk

We used binwalk -e to analyze the fake\_partition.img file from the extracted image directory and do same for hidden\_partition.img

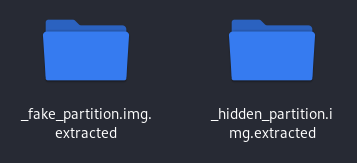
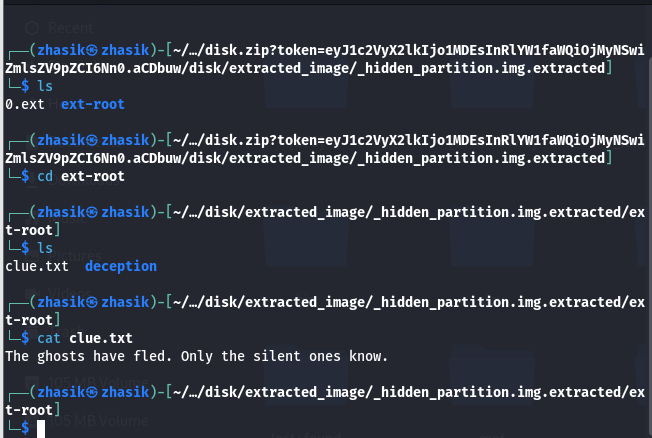
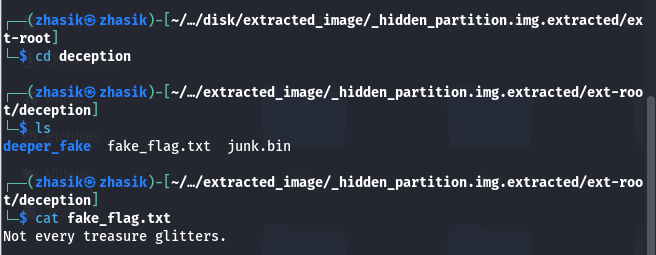


Figure 3.0.7: extracted files

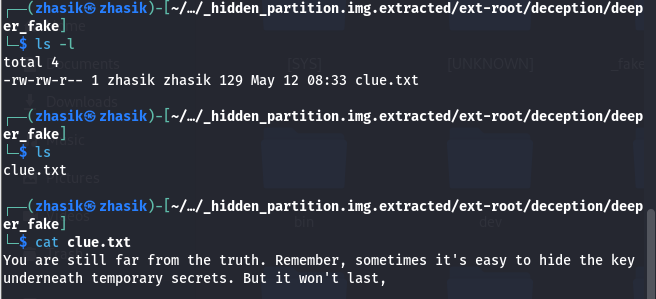
Now we have two folders with hidden extracted parts.

Figure 3.0.8: checking

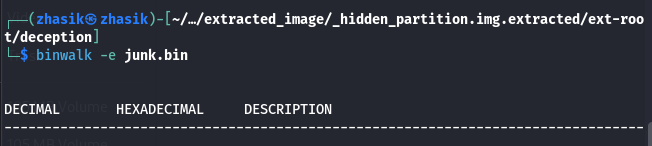
This screenshot shows exploring the contents of the extracted hidden\_partition.img. I navigated into the ext-root directory inside \_hidden\_partition.img.extracted/, where I found two files which are clue.txt and deception. When I read clue.txt, it said "The ghosts have fled. Only the silent ones know”

Figure 3.0.9: checking 2

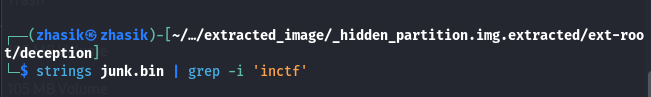
Inside the deception directory, we found three key items. First, fake\_flag.txt — which told us. "Not every treasure glitters.". This clearly meant it was a decoy, meant to mislead us.

Figure 3.0.10: checking 3

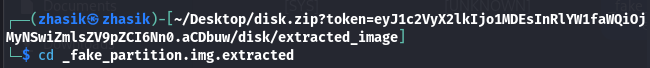
In this screenshot, we navigated into the deeper\_fake folder located inside deception, which is part of the extracted hidden\_partition.img. Inside deeper\_fake, there was a single file which is clue.txt.  
We opened it using cat, and it read. "You are still far from the truth. Remember, sometimes it's easy to hide the key underneath temporary secrets. But it won't last,"

Figure 3.0.10: bin

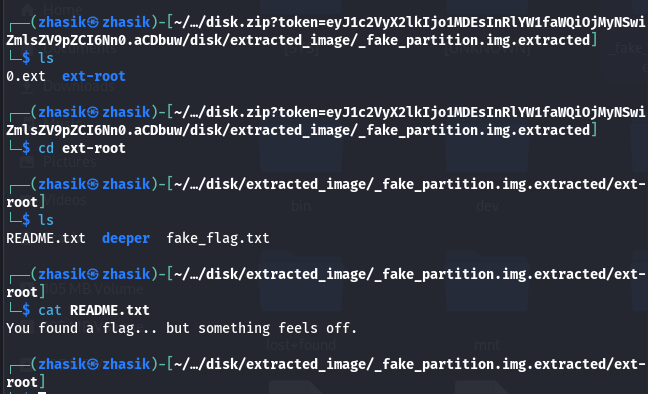
We tried to extract hidden parts of junk.bin. However, there were nothing inside.

Figure 3.0.11: grep

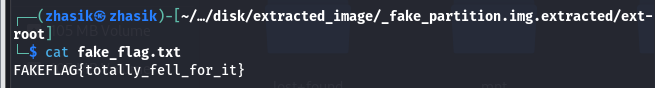
With this command, we wanted to search for any readable strings in junk.bin that contain the word **inctf**, which is the expected prefix of the flag (inctf{...}).

Figure 3.0.12: fake partition

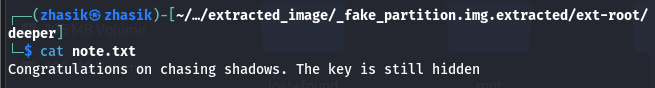
After checking the hidden\_partition.img, we went to fake\_partition.img.

Figure 3.0.13: checking 4

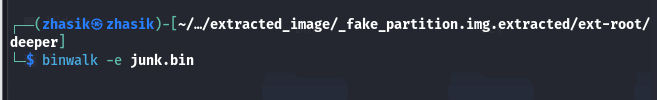
This screenshot shows our investigation into the contents of the fake\_partition.img and confirms that — just like the name suggests — it’s another intentional misdirection. We opened the extracted folder from fake\_partition.img and entered the ext-root directory. Inside, we found README.txt, fake\_flag.txt, A subdirectory called deeper. We opened README.txt, which said. "You found a flag... but something feels off."

Figure 3.0.14: fake flag

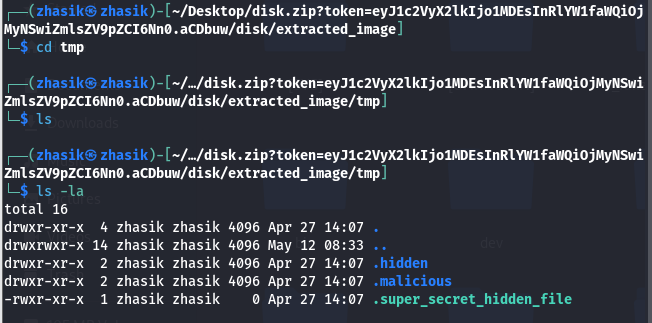
This screenshot captures the moment we confirmed that fake\_partition.img is truly a decoy in the CTF challenge. This is a deliberately fake flag, placed to mislead players who don’t dig deeper. The text "totally\_fell\_for\_it" confirms that the entire fake\_partition.img and its contents are part of the challenge’s obfuscation. It’s a humorous way of signaling: “You’re in the wrong place — keep looking.”

Figure 3.0.15: note

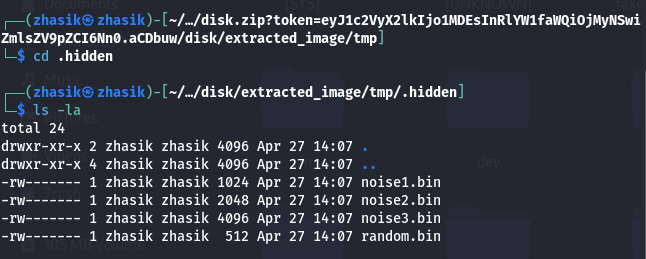
Exploring one last layer of the fake\_partition.img, specifically inside the deeper/ subfolder, which was found under ext-root. We opened note.txt using cat command. And the message is **"**Congratulations on chasing shadows. The key is still hidden**."** This is a final confirmation that the entire structure within fake\_partition.img was designed to mislead us.

Figure 3.0.16: binwalk

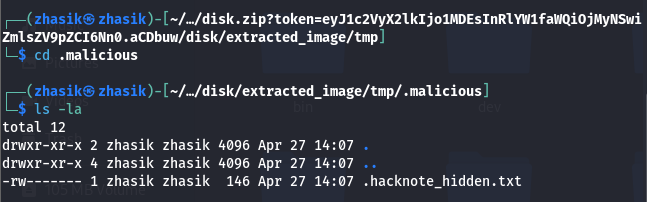
Also, checked for hidden files, however, nothing was found.

Figure 3.0.17: tmp folder

In this screenshot, we continued exploring the contents of the extracted disk image and navigated into the tmp directory — a common place in Linux for temporary or disposable files. We changed into the tmp folder. We listed its contents, including hidden files, using ls -la. There are 3 files in this tmp folder.

Figure 3.0.18: hidden

We chanced directly into hidden and there 4 bin files. Also, these files are decoys.

Figure 3.0.19: malicious file

After checking malicious folder, we found hacknote\_hidden.txt

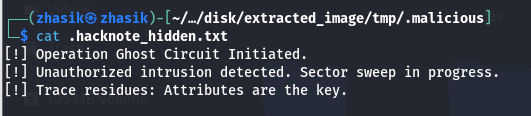


Figure 3.0.20: message

This screenshot shows us reading the contents of .hacknote\_hidden.txt inside the .malicious folder — and it delivers one of the clearest hints in the entire challenge. This message is a thematic warning, but it also gives us an incredibly important technical clue "Trace residues: Attributes are the key."

A computer screen shot of a computer program

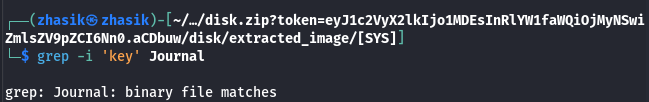
AI-generated content may be incorrect.Figure 3.0.21: grep key

This screenshot shows us performing a recursive grep search across the entire extracted image for the keyword "key" — and it powerfully validates everything we’ve suspected about this challenge. After finding that all clues key words were “key”, we used grep to find key words inside all files. As you can see from picture that we found clues and checked everything except [SYS] file.

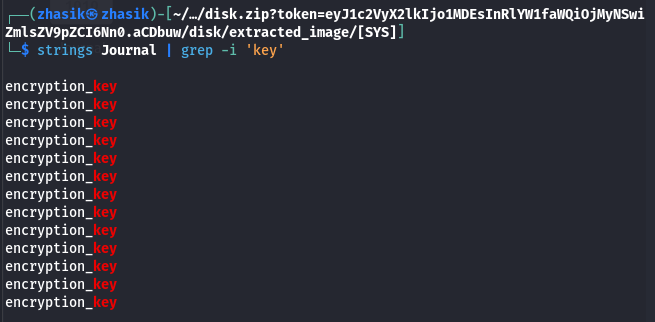
A computer screen shot of a computer program

AI-generated content may be incorrect.Figure 3.0.22: SYS file

So we went to SYS file and we’re inspecting the contents of the bin directory from the extracted disk image — and it’s clearly part of the challenge’s obfuscation layer. Using ls -la, we listed out all files.

Figure 3.0.23: grep journal

In this screenshot, we’re investigating the contents of the Journal file located in the [SYS] folder from the extracted disk image — and we’ve discovered a crucial hint hidden in a **binary system artifact**. And its says that file matches. Therefore, try to get key through other way.

Figure 3.0.24: grep key

In this screenshot, we analyzed the contents of the Journal file inside the [SYS] directory using the command strings Journal | grep -i 'key'. Output were encryption\_key multiple times. The journal file is part of the ext4 filesystem’s metadata layer — it logs recent file system activity, including deleted or moved files, and extended attributes. Seeing encryption\_key appear repeatedly in this file implies that. A key was once written, possibly to a file's extended attribute. That attribute or file might have since been deleted, moved, or overwritten — but it still left forensic traces in the journal. This validates the earlier clue from .hacknote\_hidden.txt. "Trace residues: Attributes are the key."

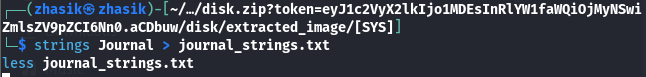


Figure 3.0.25: creating journal strings

This screenshot shows us taking a forensic step to extract readable strings from the binary Journal file and view them more comfortably. The strings command extracts all human-readable ASCII/UTF-8 text from the binary Journal file. The output is redirected into a file called journal\_strings.txt for easier viewing and searching.

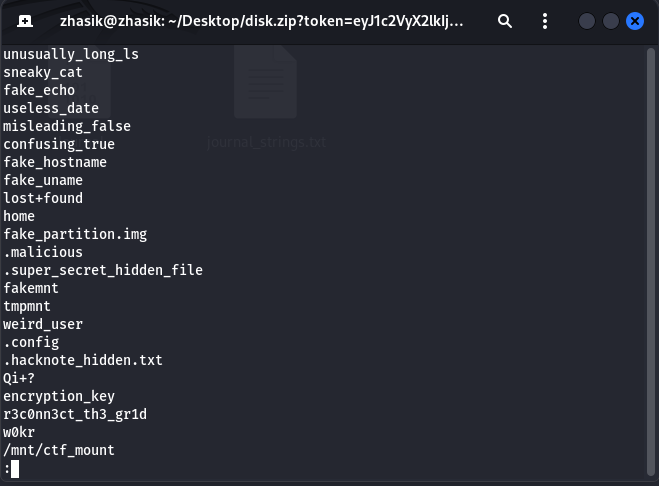


Figure 3.0.26: strings

This screenshot shows us viewing the contents of journal\_strings.txt using less, after previously extracting human-readable text from the binary Journal file. This is a critical forensic step in the CTF analysis.

What we see - The file lists several strings recovered from the ext4 journal. Suspicious files and directories are unusually\_long\_ls, sneaky\_cat, fake\_echo, etc. - fake utilities from the /bin folder. super\_secret\_hidden\_file, .hacknote\_hidden.txt — hidden files with CTF clues. fake\_partition.img, .malicious/, weird\_user/, tmpmnt — decoy or misleading paths. Clues are encryption\_key – repeatedly referenced earlier in strings Journal | grep 'key'. r3c0nn3ct\_th3\_gr1d – this looks like the actual flag, following typical CTF formatting conventions. w0kr – possibly part of the flag context or a user. mnt/ctf\_mount – possibly where the disk image was last mounted or explored

So, one of these words is key. Now let’s try to find flag to decrypt with key.

A computer screen shot of a computer program

AI-generated content may be incorrect.

Figure 3.0.27: unusually\_long\_ls

After checking all folders and files inside it. Most, of them were decoy, fake or empty. However, we found something strange inside unusually\_long\_ls.

A screen shot of a computer code

AI-generated content may be incorrect.

Figure 3.0.28: binary code

After scrolling down for 20 minutes, we finally found binary code.

00011011 01010000 00010111 01010110 00010101 00001000 00000111 01010100 01000011 00101101 00101011 00001011 01000001 01101011 00000100 00011001 00000010 00000000 00101101 00000010 00001101 01101111 00011101 01011101 01010000 01010011 00011010 00111011 01000001 00010101

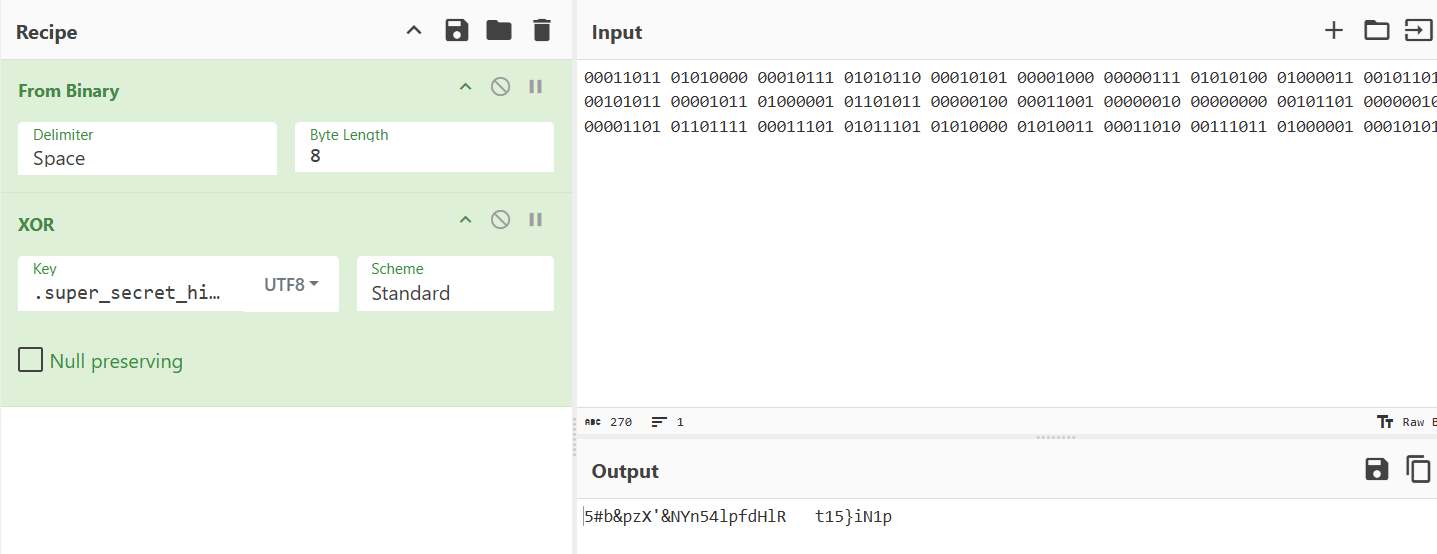


Figure 3.0.29: flag

After finding binary code, we open cyberchef enter input and add key, and first were incorrect, then we changed key to next.

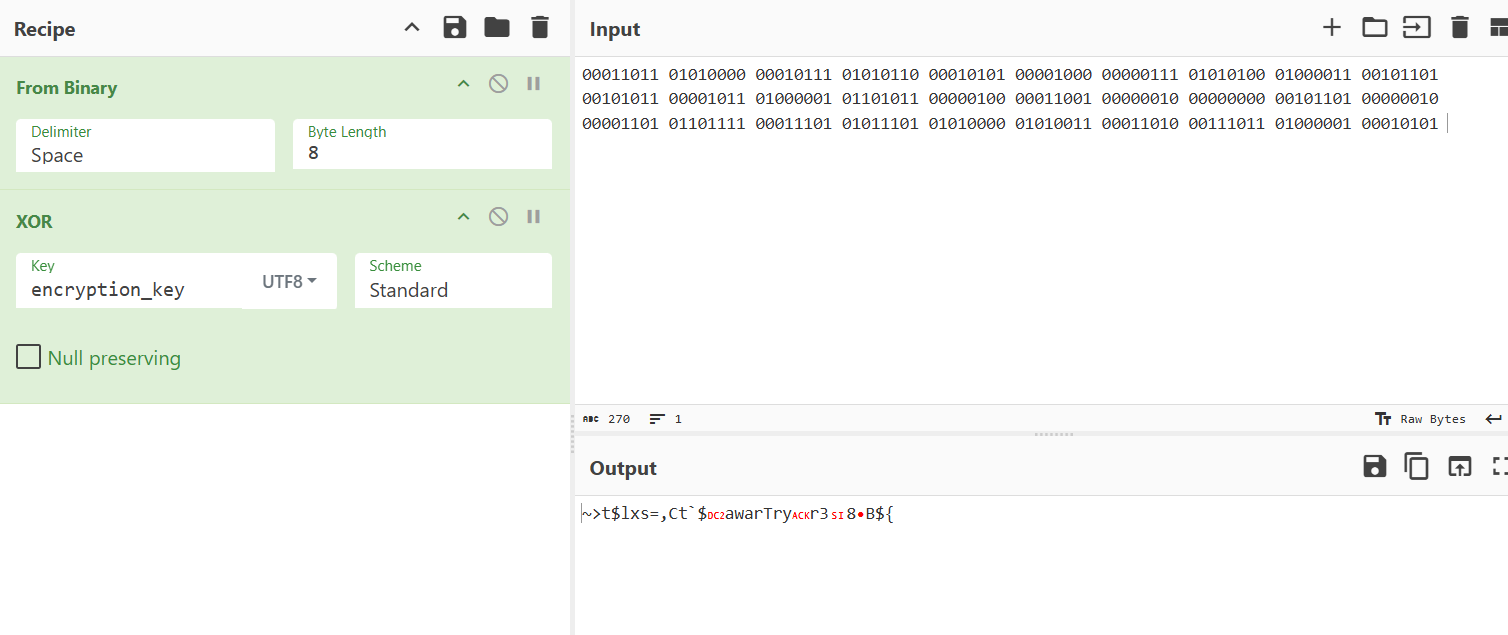


Figure 3.0.29: flag 2

Again key is incorrect, next key.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3.0.29: flag 3

After entering the next key, the flag has been successfully found. Ictf{f477r\_cr4ck3d\_1n\_s3c0nd5}

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3.0.29: success

We submitted the flag, and it was correct.

## **4.0 Challenges and Experiences in Live CTF as Beginners**

Our first experience running a live CTF gave us some useful lessons in real-time. Perhaps the most challenging task was managing time pressure — knowing what to do and organizing the division of labour in the best way possible We struggled sometimes to realize what the challenges were asking us to achieve. Some of the challenges appeared relatively straightforward, but had tricky file names or the data was very obscure. We also learned the importance of tools: sometimes their default tools let them down, whereas other tools like CyberChef saved us hours of debugging. Coordination was another important factor. Close coordination was always of utmost importance — sometimes working side by side on the same file, and sometimes separating the steps, one person to research file contents, while another would attempt methods of decryption. In the end, the CTF has expanded our appreciation of how a hacker thinks and how information can be obscured or leveraged. We also formed technical skills and confidence to work under pressure and improve in problem-solving contexts.

## **5.0 Conclusion**

Taking part in the Incognito CTF 6.0 was certainly eye-opening, especially as novice cybersecurity enthusiasts. Choosing the "Weird disk image" challenge, really opened our eyes to digital forensics, and forced us to test not just technical skills but also patience, logical reasoning, and creative thinking. Through a step-by-step investigation, we faced many decoy files, misleading folders, and clues that were deeply hidden in many possible locations - all crafted to confuse us. Despite the complexity, we were able to apply a wide range of forensic fundamentals and tools including; binwalk, strings, grep, and even 7-Zip. We learned how data can be hidden within disk structures, footnotes, metadata, journal files, and extended attributes in our disk image. This experience has demonstrated just how an attacker can hide or mislead any investigators. Forensic specialists must follow very small 'breadcrumbs' as their only reference paths, like how we learned to investigate the digital footprints. We learned how to operate and respond to time pressures, develop quick team interaction, and respond to complete failure of our tools or respond to misleading paths of data.

## **6.0 References**

*Incognito*. (n.d.). <https://incognito.axiosiiitl.dev/>

*CyberChef*. (n.d.). Crown Copyright 2016-2025. <https://gchq.github.io/CyberChef/>

*strings(1) - Linux manual page*. (n.d.). <https://man7.org/linux/man-pages/man1/strings.1.html>

*grep(1) - Linux manual page*. (n.d.). <https://man7.org/linux/man-pages/man1/grep.1.html>

*The sleuth kit*. (n.d.). <https://www.sleuthkit.org/sleuthkit/>

*Disk image -*. (n.d.). <https://forensics.wiki/disk_image/>

## **7.0 Workload Matrix / Task Distribution**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | TP Number | Contribution (%) | Tasks Covered | Signature |
| Niyazaly Zhassulan | TP071009 | 50% | Forensic analysis, disk image investigation, screenshots, partial documentation | НЖ |
| Khegai Olga | TP069365 | 50% | Forensic analysis, clue tracing, tool usage, partial documentation, formatting |  |