Lab 10: Steganography

Shivali Mate

I. Introduction

THE lab focused on exploring digital steganography, the practice of concealing hidden information within media files. Using the tools provided in Dominic Breuker's Stego-Toolkit [3], the objective was to analyze two image files 1.secret and 2.secret from [1] and uncover any concealed data. The lab emphasized understanding how forensic utilities such as exiftool, binwalk, strings, zsteg, and StegoVeritas function together to extract metadata, detect anomalies, and recover hidden information from visual and audio layers. Through this exercise, the learner gained insight into how data can be embedded in pixels, bit planes, or frequency components without visibly altering the carrier image.

II. Methodology

The methodology followed a structured forensic workflow, beginning with file format verification using the file command and progressing through metadata inspection exiftool, byte-level examination strings and binwalk, and steganographic analysis zsteg, StegoVeritas. All tools were accessed through the Dockerized stego-toolkit environment [3]. The experiments were performed separately on the two secret images.

A. 1.secret

The analysis of 1.secret started by identifying its file type using the command below. This confirmed that the file was a PNG image. Metadata was then examined using exiftool, which displayed information such as dimensions, bit depth, and color profiles. These details confirmed that the image structure was intact and ready for deeper inspection. The next step involved running strings to look for readable text or hidden fragments. Common PNG headers like IHDR and IDAT appeared along with some encoded characters. To verify whether any additional files were attached, binwalk was used to scan for embedded binaries, but none were detected. This indicated that the hidden content, if present, was likely concealed in the pixel data or LSB region.

```
file 1.secret
exiftool 1.secret
strings 1.secret
binwalk -e 1.secret
```

Further analysis was done using zsteg -a 1.secret to inspect the image's bit planes. This revealed a concealed text string gc7W[*(C hidden in the red channel's least significant bits. The result, shown in Fig. 2, confirmed the successful detection of steganographic text. To validate this, StegoVeritas was applied to generate multiple transformed versions of the image, exposing individual color planes and pixel patterns. The command below produced a folder of extracted planes, as illustrated in Fig. 2, which visually highlighted variations in pixel intensity that suggested encoded data. This combination of automated detection and visual inspection successfully recovered the embedded text while demonstrating how steganography operates within pixel-level LSB data.

```
zsteg -a 1.secret
python3 -m stegoveritas.stegoveritas
    1.secret -meta -out results/
```

Fig. 1. Zsteg output showing hidden text in the red channel of 1.secret

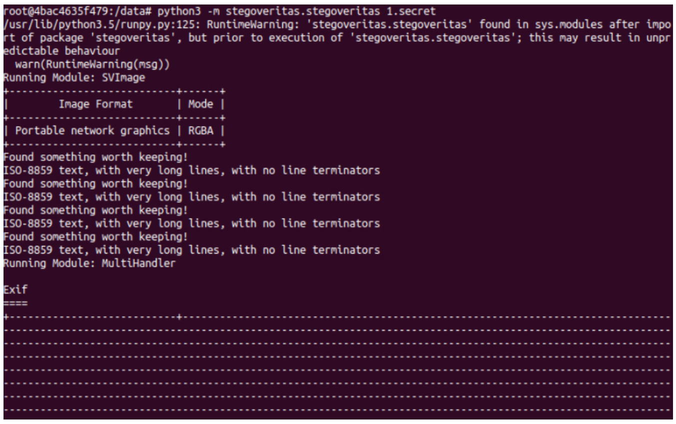

Fig. 2. StegoVeritas extracted bit-planes confirming encoded data in 1.secret

B. 2.secret

The same approach was followed for 2.secret. The file type was confirmed as PNG, and exiftool displayed standard metadata values. A quick scan using strings revealed binary symbols instead of readable text, hinting at a more complex embedding method. Running binwalk -e 2.secret displayed traces of MPEG ADTS signatures, suggesting that an audio stream might be hidden within the image data. To test this possibility, zsteg was used to perform a deeper analysis across multiple bit planes and color channels.

```
file 2.secret
exiftool 2.secret
```

```
strings 2.secret
binwalk -e 2.secret
zsteg -a 2.secret
```

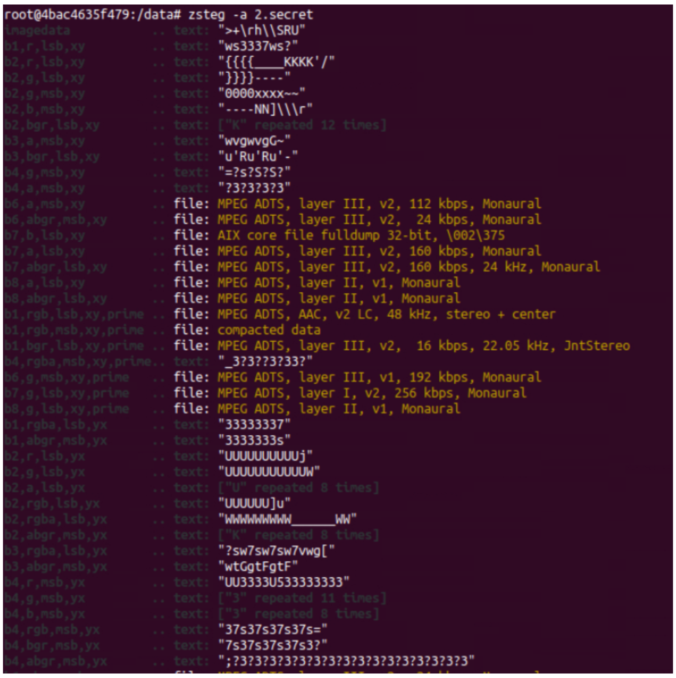

Fig. 3. Zsteg output revealing embedded MP3 fragments in 2.secret

The command below was then used to extract the discovered audio data as a new file named hidden2_alt.mp3. This file was confirmed to be an MPEG audio stream using the file command. To interpret its content, ffmpeg was used to generate a spectrogram that visualized the hidden audio frequencies. The spectrogram image, shown in Fig. 4, displayed structured horizontal frequency bands, indicating the successful recovery of encoded audio data. This process demonstrated how steganography can embed multimedia content within an image and how frequency-domain visualization can reveal its presence.

```
zsteg -E b6,abgr,msb,xy
    > hidden2_alt.mp3
file hidden2_alt.mp3
ffmpeg -i hidden2_alt.mp3 -lavfi
    showspectrumpic=s=1280x720
    spectrogram2.png
```

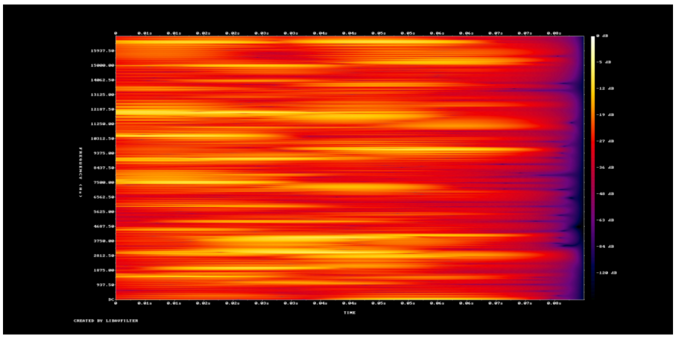

Fig. 4. Spectrogram generated from hidden audio extracted from 2.secret

III. Tools and Learning Experience

Throughout the lab, a range of forensic and steganography-specific tools was explored. The Stego-Toolkit [3] served as the primary environment, combining Linux command-line utilities with advanced steganalysis programs. Exiftool was instrumental for reading metadata, binwalk and strings for detecting appended files or hidden ASCII content, and zsteg for multi-channel LSB analysis. StegoVeritas proved especially valuable for generating visual transformations that expose subtle hidden structures, while ffmpeg enabled audio decoding and spectrogram generation. By experimenting with these tools in different configurations, the learner gained hands-on experience with both pixel-level and frequency-domain data concealment, reinforcing how steganography operates beneath standard image or audio formats.

IV. Findings and Discussion

The analysis confirmed successful extraction of hidden content from both images. In 1.secret, textual data concealed within the pixel bit-planes was retrieved and verified through statistical LSB detection. In 2.secret, an embedded MP3 file was uncovered and visualized as a frequency-based spectrogram, validating that steganography can extend beyond visual domains into multimedia. The combination of zsteg and StegoVeritas produced consistent results, while ffmpeg offered an effective bridge to analyze non-visual payloads.

While these were classical steganalysis utilities, the experiment also connected to AI-driven concepts discussed in modern digital forensics, such as automated anomaly detection in pixel distributions or spectral pattern recognition in audio. The comparison revealed that tools based on deterministic bit-level analysis, like zsteg, excel at structured LSB encodings but may fail on encrypted or non-linear embeddings, areas where machine learning models can outperform traditional techniques by identifying statistical irregularities.

V. Conclusion

The steganography lab successfully revealed how digital files can serve as carriers for concealed information. Through systematic application of open-source forensic tools, both textual and audio data were extracted and analyzed from two secret images. The first image verified pixel-level text embedding, while the second demonstrated an audio steganography case visualized through its spectrogram. The process reinforced understanding of LSB manipulation, metadata examination, and the use of hybrid image-audio recovery techniques. Overall, the lab enhanced awareness of how modern steganographic methods operate and how they can be forensically detected.

References

[1] Indiana University Canvas, Lab 10 Assets: 1.secret and 2.secret files, available under course files for Cyber Defense Laboratory.

[2] GeeksforGeeks, LSB-based Image Steganography Using MATLAB. https://www.geeksforgeeks.org/lsb-based-image-steganography-using-matlab/

[3] Dominic Breuker, Stego-Toolkit: A Docker Image for Steganography Challenges. GitHub Repository, https://github.com/DominicBreuker/stego-toolkit