Lab 11

Venkata Diwakar Reddy Kashireddy

***Abstract*—In this lab, we explored the field of steganography using various tools and techniques. Our primary goal was to uncover hidden data within images using steganographic methods, leveraging both common and advanced tools to achieve this.**

I. Introduction

Steganography, the practice of hiding secret information within non-secret media, is a critical aspect of cybersecurity. In this lab, we used various tools to detect and decode hidden messages in images. This is vital for understanding the intricacies of covert data transmission and the importance of detecting such methods in cybersecurity.

II. Tools

• KVM (Kernel-based Virtual Machine): It is a full virtualization solution for Linux. [1]
• Overleaf: It is a collaborative cloud-based LaTeX editor that helps to create documents easily by providing standard formats. [2]
• GitHub: GitHub is an Internet hosting service for software development and version control using Git. [3]
• RED: – IU Research Desktop (RED) is a virtual desktop service for users with accounts on the Carbonate research supercomputer at IU. [4]
• Steganography Toolkit: A collection of tools for detecting and decoding steganography in images. [5]

III. Stegonography Toolkit

The stego-toolkit by Dominic Breuker is a comprehensive collection of steganography and steganalysis tools, hosted in a Docker container for ease of use. It includes tools for embedding and detecting hidden data in various media, alongside forensic and image manipulation utilities. Installation was done on the VM fairly simple by running a Docker image. This toolkit is an essential resource for anyone engaged in digital steganography and forensic analysis.

IV. Discovering Hidden Data in Images

For the lab, we were given two images and were asked to find the secrets or hidden data in the images. First, I tried to understand all the tools in the container. There were various tools for various types of files. The below are the images that we needed to explore: 1.secert and 2.secret.

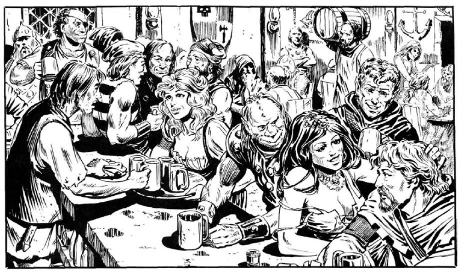


Fig. 1. 1.secret

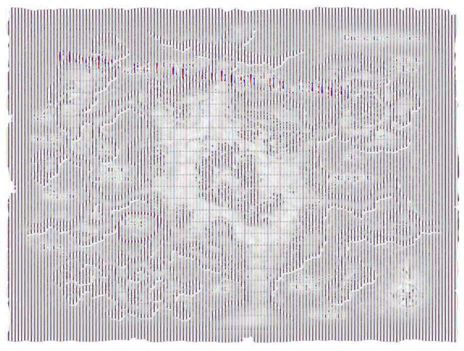
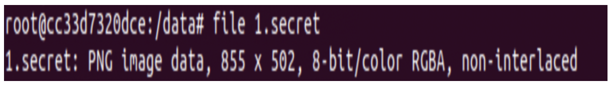


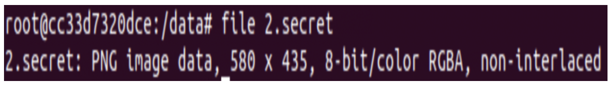
Fig. 2. 2.secret

The first step was to determine the file format using file tool command and inspect metadata using ExifTool tool command.



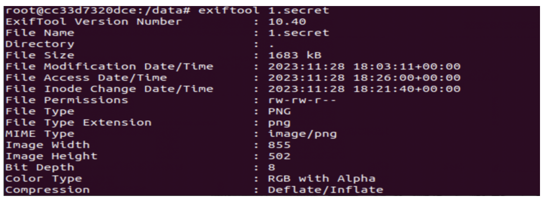
```
root@cc33d7320dce:/data# file 1.secret
1.secret: PNG image data, 855 x 502, 8-bit/color RGBA, non-interlaced
```

Fig. 3. File format 1.secret



```
root@cc33d7320dce:/data# file 2.secret
2.secret: PNG image data, 580 x 435, 8-bit/color RGBA, non-interlaced
```

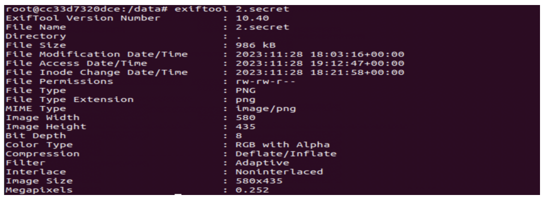
Fig. 4. File format 2.secret



```
root@cc33d7320dce:/data# exiftool 1.secret
ExifTool Version Number         : 10.40
File Name                       : 1.secret
Directory                       : .
File Size                       : 1683 kB
File Modification Date/Time     : 2023:11:28 18:03:11+00:00
File Access Date/Time           : 2023:11:28 18:26:00+00:00
File Inode Change Date/Time     : 2023:11:28 18:21:40+00:00
File Permissions                : rw-rw-r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 855
Image Height                    : 502
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
```

Fig. 5. Meta data 1.secret

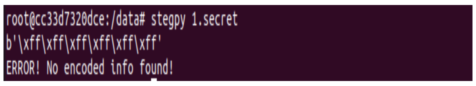
From the analysis, we found out that the both the files are in png format. This provided us with hints of other tools, we can use to perform more analysis.



```
root@cc33d7320dce:/data# exiftool 2.secret
ExifTool Version Number         : 10.40
File Name                       : 2.secret
Directory                       : .
File Size                       : 986 kB
File Modification Date/Time     : 2023:11:28 18:03:16+00:00
File Access Date/Time           : 2023:11:28 19:12:47+00:00
File Inode Change Date/Time     : 2023:11:28 18:21:58+00:00
File Permissions                : rw-rw-r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 580
Image Height                    : 435
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Image Size                      : 580x435
Megapixels                      : 0.252
```

Fig. 6. Meta data 2.secret

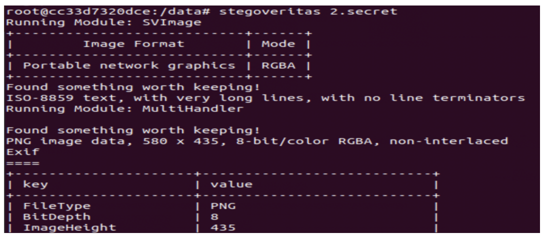
We used Binwalk to extract embedded files and Stegdetect to identify any steganographic signatures. Few tools that were used were not really helpful and few needed some sort of passwords, so those were not useful as well.



```
root@cc33d7320dce:/data# stegpy 1.secret
b'\xff\xff\xff\xff\xff\xff'
ERROR! No encoded info found!
```

Fig. 7. Impletementing stegpy tool

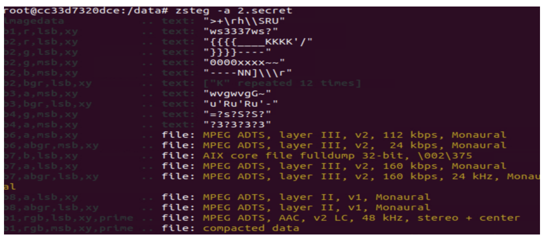
We also utilized other tools from the Stego-Toolkit to analyse the images. These tools included pngcheck, stegoVeritas, zsteg, openstego, cloackedpixel, LSBSteg, and stegpy. Each tool served a different purpose, pngcheck was used to verify the integrity and authenticity of the PNG files, stegoVeritas and zsteg provided a comprehensive analysis of potential hidden data through metadata examination and LSB steganography detection. openstego and cloackedpixel were employed for embedding and extracting covert messages using various LSB algorithms. LSBSteg and stegpy were utilized for their straightforward approach to encode and decode messages in uncompressed image formats. This toolkit enabled a thorough investigation of the images, revealing hidden details and information.



```
root@cc33d7320dce:/data# stegoveritas 2.secret
Running Module: SVImage
+-----------------------------+------+
|         Image Format        | Mode |
+-----------------------------+------+
| Portable network graphics | RGBA |
+-----------------------------+------+
Found something worth keeping!
ISO-8859 text, with very long lines, with no line terminators
Running Module: MultiHandler

Found something worth keeping!
PNG image data, 580 x 435, 8-bit/color RGBA, non-interlaced
Exif
====
+---------------------+-----------------------------+
| key                 | value                       |
+---------------------+-----------------------------+
| FileType            | PNG                         |
| BitDepth            | 8                           |
| ImageHeight         | 435                         |
```

Fig. 8. stegoveritas tool analysis



```
root@cc33d7320dce:/data# zsteg -a 2.secret
imagedata               .. text: ">+\rh\\SRU"
b1,r,lsb,xy             .. text: "ws3337ws?"
b2,r,lsb,xy             .. text: "{{{{____KKKK'/"
b2,g,lsb,xy             .. text: "}}}}----"
b2,g,msb,xy             .. text: "0000xxxx~~"
b2,b,msb,xy             .. text: "----NN]\\\r"
b2,bgr,lsb,xy           .. text: ["K" repeated 12 times]
b3,a,msb,xy             .. text: "wvgwvgG~"
b3,bgr,lsb,xy           .. text: "u'Ru'Ru'-"
b4,g,msb,xy             .. text: "=?s?S?S?"
b4,a,msb,xy             .. text: "?3?3?3?3"
b6,a,msb,xy             .. file: MPEG ADTS, layer III, v2, 112 kbps, Monaural
b6,abgr,msb,xy          .. file: MPEG ADTS, layer III, v2,  24 kbps, Monaural
b7,b,lsb,xy             .. file: AIX core file fulldump 32-bit, \002\375
b7,a,lsb,xy             .. file: MPEG ADTS, layer III, v2, 160 kbps, Monaural
b7,abgr,lsb,xy          .. file: MPEG ADTS, layer III, v2, 160 kbps, 24 kHz, Monau
al
b8,a,lsb,xy             .. file: MPEG ADTS, layer II, v1, Monaural
b8,abgr,lsb,xy          .. file: MPEG ADTS, layer II, v1, Monaural
b1,rgb,lsb,xy,prime     .. file: MPEG ADTS, AAC, v2 LC, 48 kHz, stereo + center
b1,rgb,msb,xy,prime     .. file: compacted data
```

Fig. 9. Zteg tool analysis on 2.secret

V. Decoding the Hidden Messages

After identifying the presence of steganographic data, we used the Steganography Toolkit to decode the messages. This process required careful analysis and sometimes a trial-and-error approach with different tools to successfully extract the hidden information. Since we have already established that the file format was png, we have narrowed down the tools we can use to only the ones that can decode png files.

After using all possible tools, we were able to find out the hidden data in the 1st images. It was retreived as an result of extracting embedded files from the original image. The hidden message was: flag woah such as secret message.

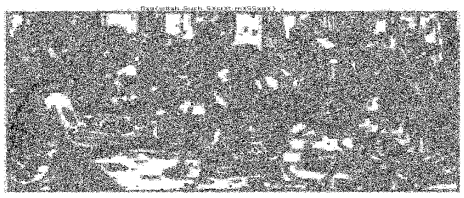


Fig. 10. Retreived images of 1.secret with the hidden message.

As for the second image, it looked like an image of a map with names of the locations. A location ”Ledge” was highlighted in the map. The challenges faced here shows the sophistication of modern steganographic techniques.

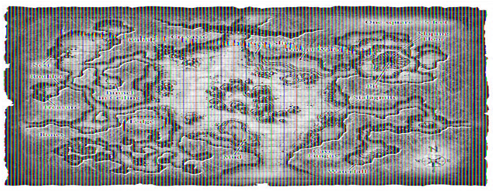


Fig. 11. Retreived images of 2.secret with the hidden message.

VI. Conclusion

This lab provided valuable insights into the realm of steganography. It emphasized the importance of understanding and detecting hidden information in digital media, a skill increasingly relevant in cybersecurity. The ability to uncover and interpret hidden data not only demonstrates the power of steganography but also highlights the potential security risks associated with its use.

References

[1] KVM
https://www.linux-kvm.org/page/MainPage..

[2] OverLeaf
https://www.overleaf.com/.

[3] Github
https://en.wikipedia.org/wiki/GitHub.

[4] RED
https://kb.iu.edu/d/apum.

[5] stego-toolkit
https://github.com/DominicBreuker/stego-toolkit

[6] LSB based Image steganography
https://www.geeksforgeeks.org/lsb-based-image-steganography-using-matlab/