totes accurate depiction of what analyzing hard drive looks like

FORENSICS

Beginner Series Talk

By Aya Abdelgawad and Aly Abdulatif

- FUZZILY DEFINED
- Finding flag that's been embedded/hidden in a file, image, disk image, etc.
  - any file that's not code (because that's reversing)
  - usually interested in how file has been tampered

What is Forensics?

What is a file?

- collection of bits (1's and 0's)
- computer will present them to you in friendly human-readable format, but it's really
  - thousands or
  - millions or
  - billions of bits
- easy for computers to read
  - blargh for humans

Binary Crash Course

- probably familiar with decimal system (base 10)
- Starting at the right and moving left, each position represents a power of [base]
- digit tells us what to multiply the power by

Decimal (base 10)

$$\begin{array}{cccc} \multicolumn{4}{c}{\boxed{1 \quad 5 \quad 7 \quad 2}} \\ 10^{3} & 10^{2} & 10^{1} & 10^{0} \end{array}$$

$$1000+500+70+2=1{,}572$$

Binary (base 2)

$$\begin{array}{ccccccccccc} \multicolumn{11}{c}{\boxed{1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0}} \\ 2^{10} & 2^{9} & 2^{8} & 2^{7} & 2^{6} & 2^{5} & 2^{4} & 2^{3} & 2^{2} & 2^{1} & 2^{0} \end{array}$$

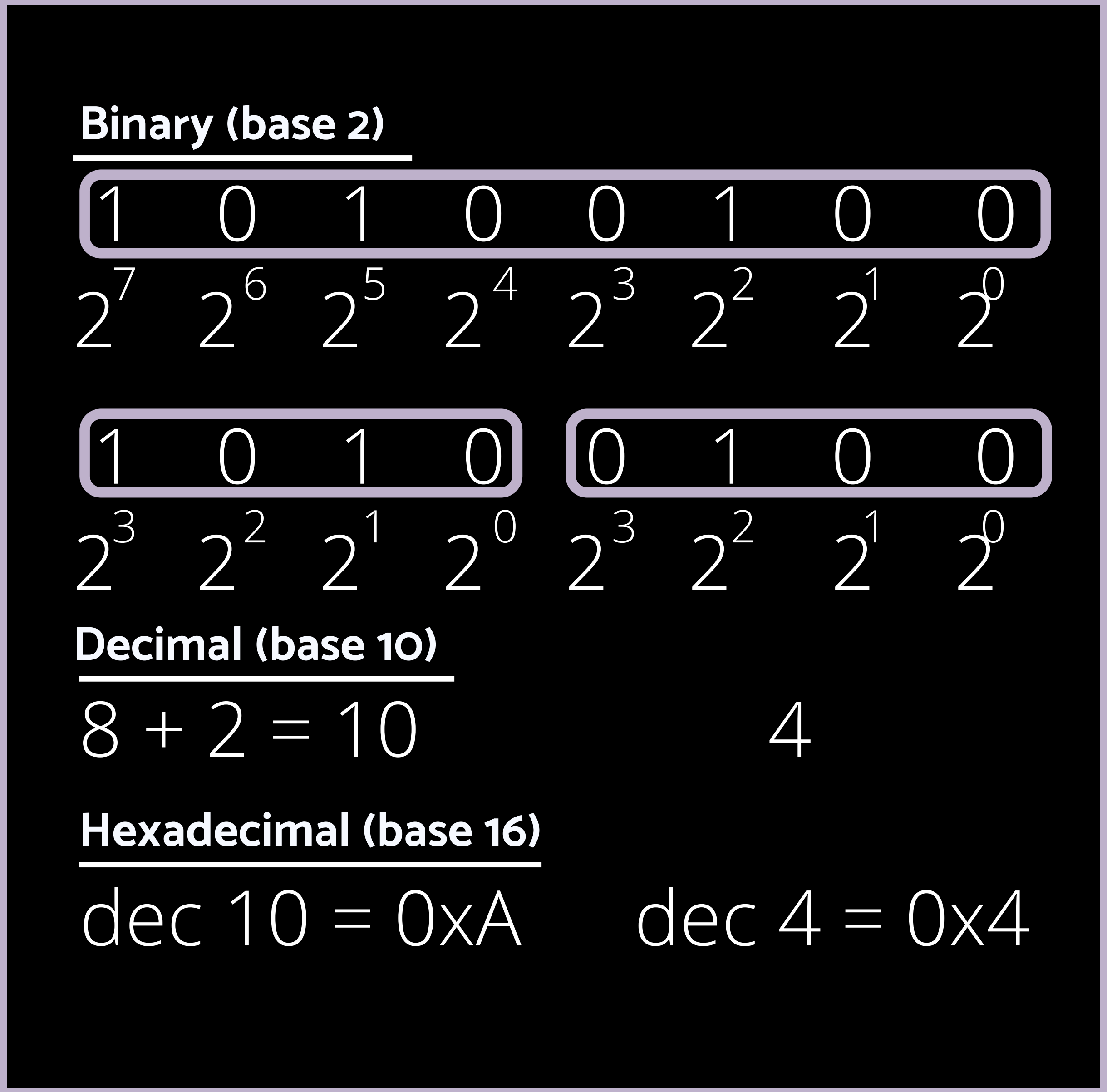
$$1024+512+32+4=1{,}572$$

so long! D:

Hex Crash Course

- Hexadecimal
  - Base 16: Numbers 0-9 and A-F represent 0-15, usually prefixed by 0x
  - Since 16 = 2^4, we can only represent 4 bits of data with one hex character

- **Most of the time you will see files represented in hexadecimal**, since it's more compact than binary but also easier to think about than base 32 or 64



Pop quiz!

Determines your grade in ISSS so no pressure or anything

You want to convert a word doc into a PDF. Which of the following will NOT do that?

A. Use a random online PDF converter

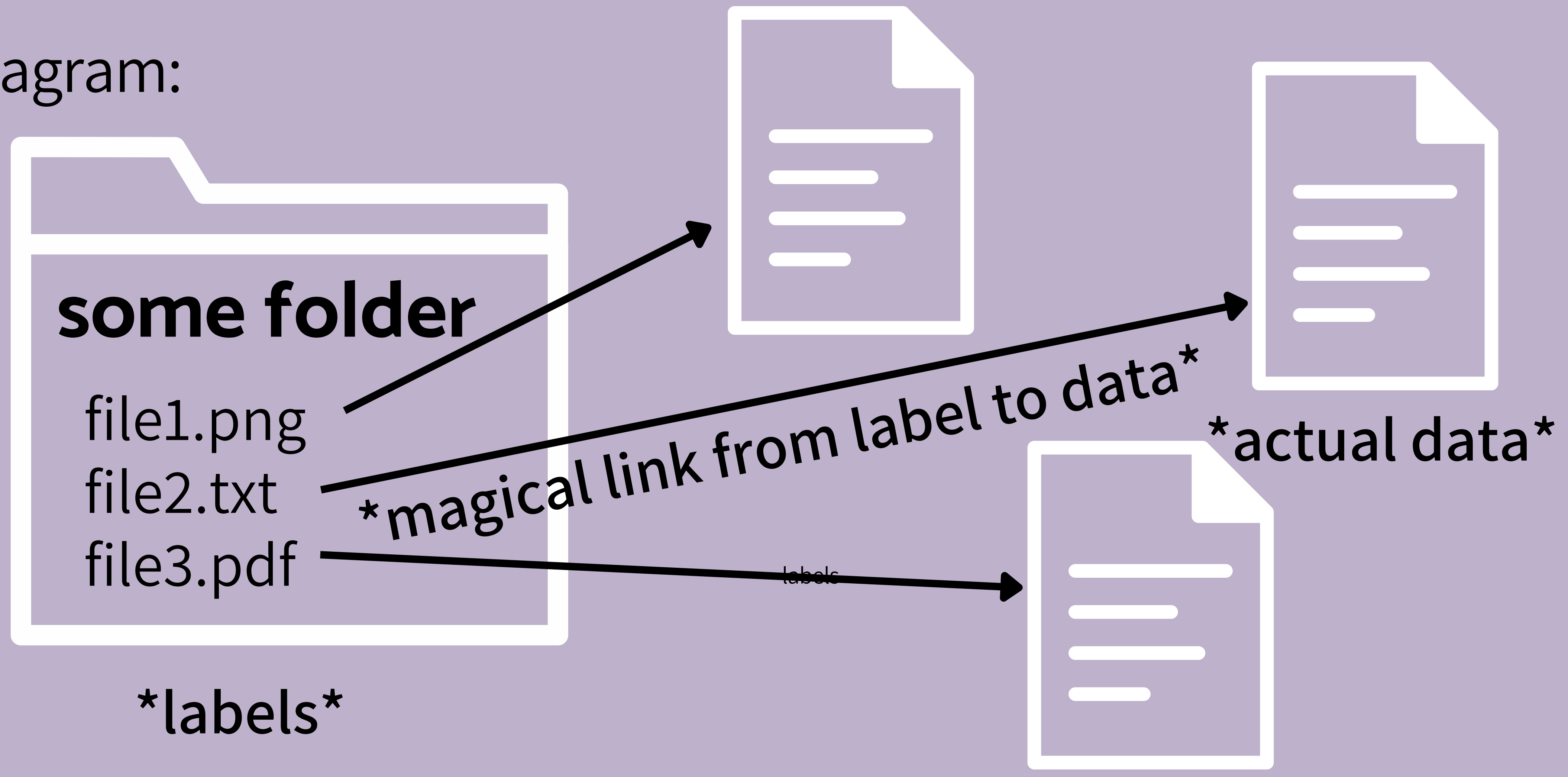
B. Use your text editor (File > Download as PDF, or something similar)

C. Change the extension to `.pdf`

Why does this not work?

Change the extension to `.pdf`

- Extension part of file's name
- Name and associated data of files are separate
- Oversimplified diagram:



How do I learn the TRUTH?

- run **`file`** to learn what the file type really is
    - plus some extra metadata (esp. for images)

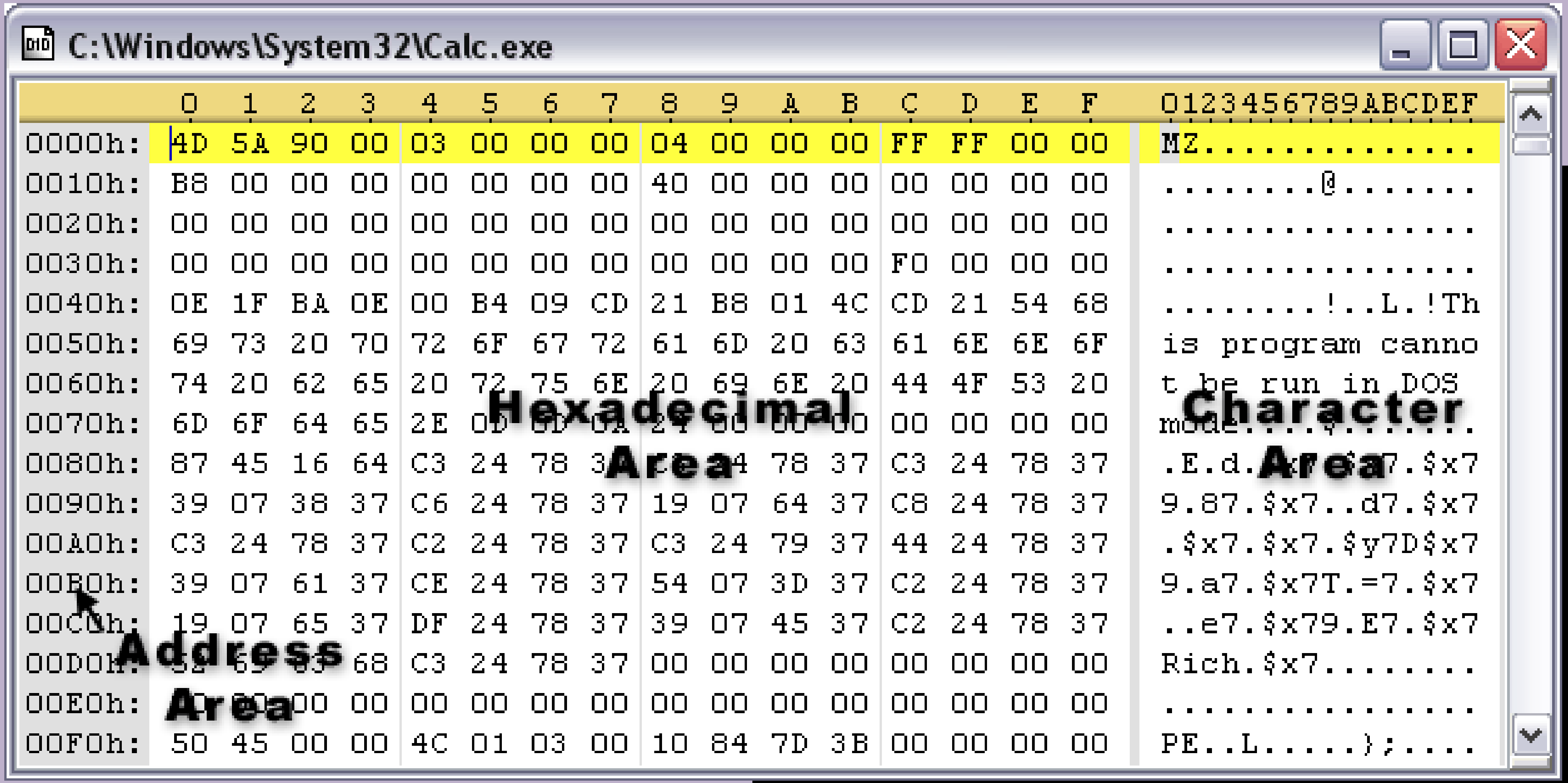
```
$ file foo.txt
foo.txt: JPEG image data, JFIF standard 1.01, resolution
(DPI), density 96x96, segment length 16, Exif Standard:
[TIFF image data, big-endian, direntries=5], baseline,
precision 8, 600x320, components 3
```

Wait...How does the computer know the TRUTH?

- Magic headers/magic bytes/file headers/file signatures
  - Special sequences of bytes at the beginning or end to identify the file type
    - ex: PDF's start with **25 50 44 46 2D** (ASCII rep of "**%PDF-**")
- Wikipedia has a handy-dandy ginormous list of file headers
- Fixing headers a common CTF problem
  - May be missing or altered
  - (note if the file header is broken, `file` will not be accurate)

Hex Editors

- Can use a hex editor to alter the file and fix headers
  - GHex (Linux only)
  - Bless (Linux and Windows, but crashes a lot from my experience)



```
         0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h:  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00   MZ..............
0010h:  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0020h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0030h:  00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00   ................
0040h:  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68   ........!..L.!Th
0050h:  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F   is program canno
0060h:  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20   t be run in DOS 
0070h:  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00   mode....$.......
0080h:  87 45 16 64 C3 24 78 37 C3 24 78 37 C3 24 78 37   .E.d.$x7.$x7.$x7
0090h:  39 07 38 37 C6 24 78 37 19 07 64 37 C8 24 78 37   9.87.$x7..d7.$x7
00A0h:  C3 24 78 37 C2 24 78 37 C3 24 79 37 44 24 78 37   .$x7.$x7.$y7D$x7
00B0h:  39 07 61 37 CE 24 78 37 54 07 3D 37 C2 24 78 37   9.a7.$x7T.=7.$x7
00C0h:  19 07 65 37 DF 24 78 37 39 07 45 37 C2 24 78 37   ..e7.$x79.E7.$x7
00D0h:  52 69 63 68 C3 24 78 37 00 00 00 00 00 00 00 00   Rich.$x7........
00E0h:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00F0h:  50 45 00 00 4C 01 03 00 10 84 7D 3B 00 00 00 00   PE..L.....};....
```

Strings

- Common easy problem: hide flag in metadata of a file, or somewhere in the data of the file itself
- You can use **`strings`** to look for any strings in the file, and then pass it to **`grep`** to search for the flag format

```
$ strings foo | grep utflag
utflag{you_found_me}
```

- This can work on non-forensics problems too!
  - E.g. reversing problems where the flag is harcoded somewhere in the program

EXIF

- Exchangeable Image File
  - data included in image/video files captured by cameras
- Flag (or relevant info) may be hidden in metadata

```
$ strings foo | grep utflag
utflag{you_found_me}
```

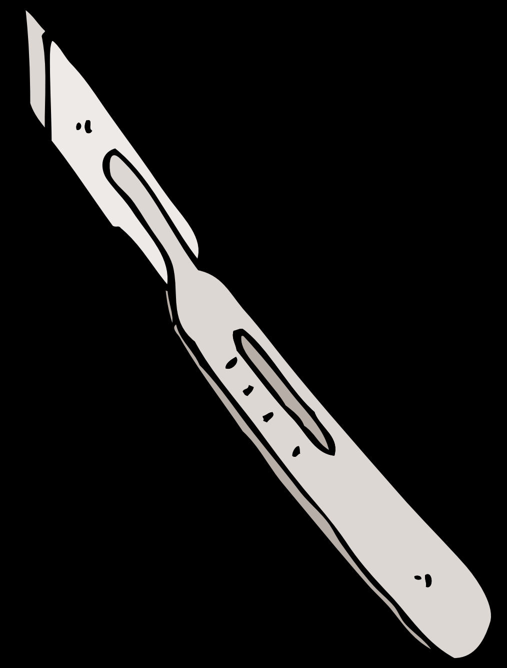
File Carving

- A common type of forensics CTF problem you'll see is called file carving
- Hide files within other files, or concatenate multiple files together into one big file
    - In order to see what's inside of a file you're given, use **`binwalk`**

```
$ binwalk foo
DECIMAL   HEXADECIMAL   DESCRIPTION
--------------------------------------------------------------
0         0x0           JPEG image data, JFIF standard 1.01
30        0x1E          TIFF image data, big-endian
66995     0x105B3       Zip archive data, at least v1.0 to extract
69256     0x10E88       End of Zip archive, footer length: 22
```

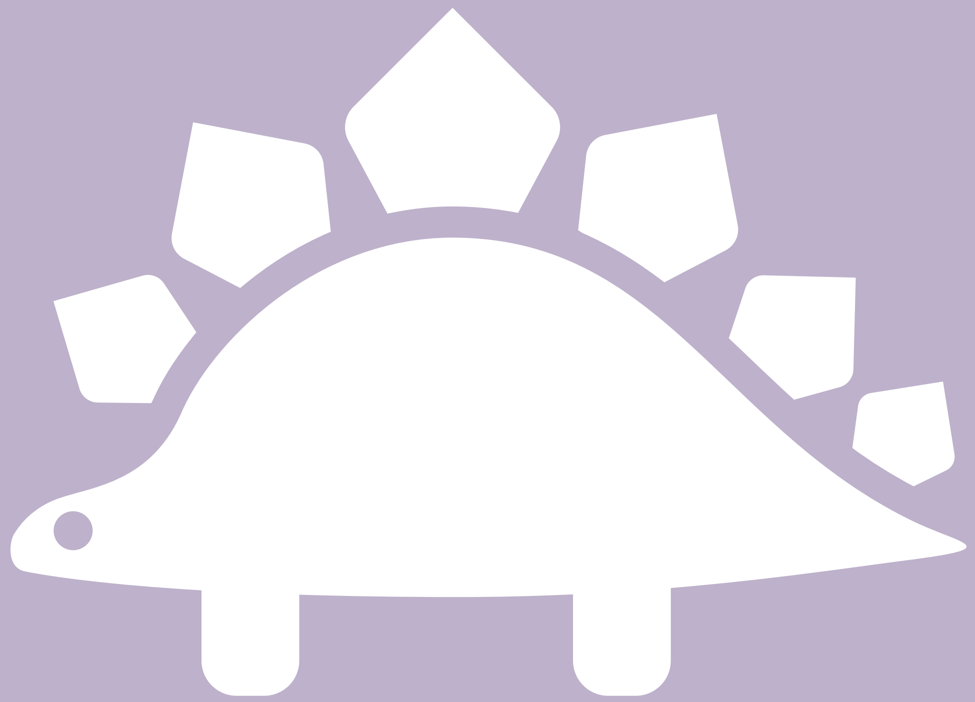
File Carving

- To extract these files, you can use
  - **`binwalk -e`**
  - Foremost
  - Scalpel



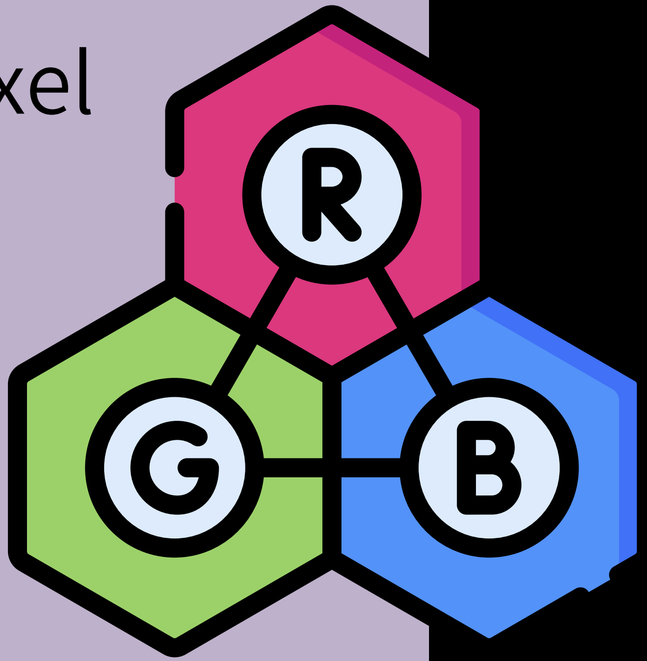
Steganography

- Hiding information AND hiding the fact that you hid information
- For CTFs
    - usually info hidden in image files
    - may also see info hidden in audio files too
    - (technically, you can hide any file in any type of file, but these are the most common)



LSB Steganography

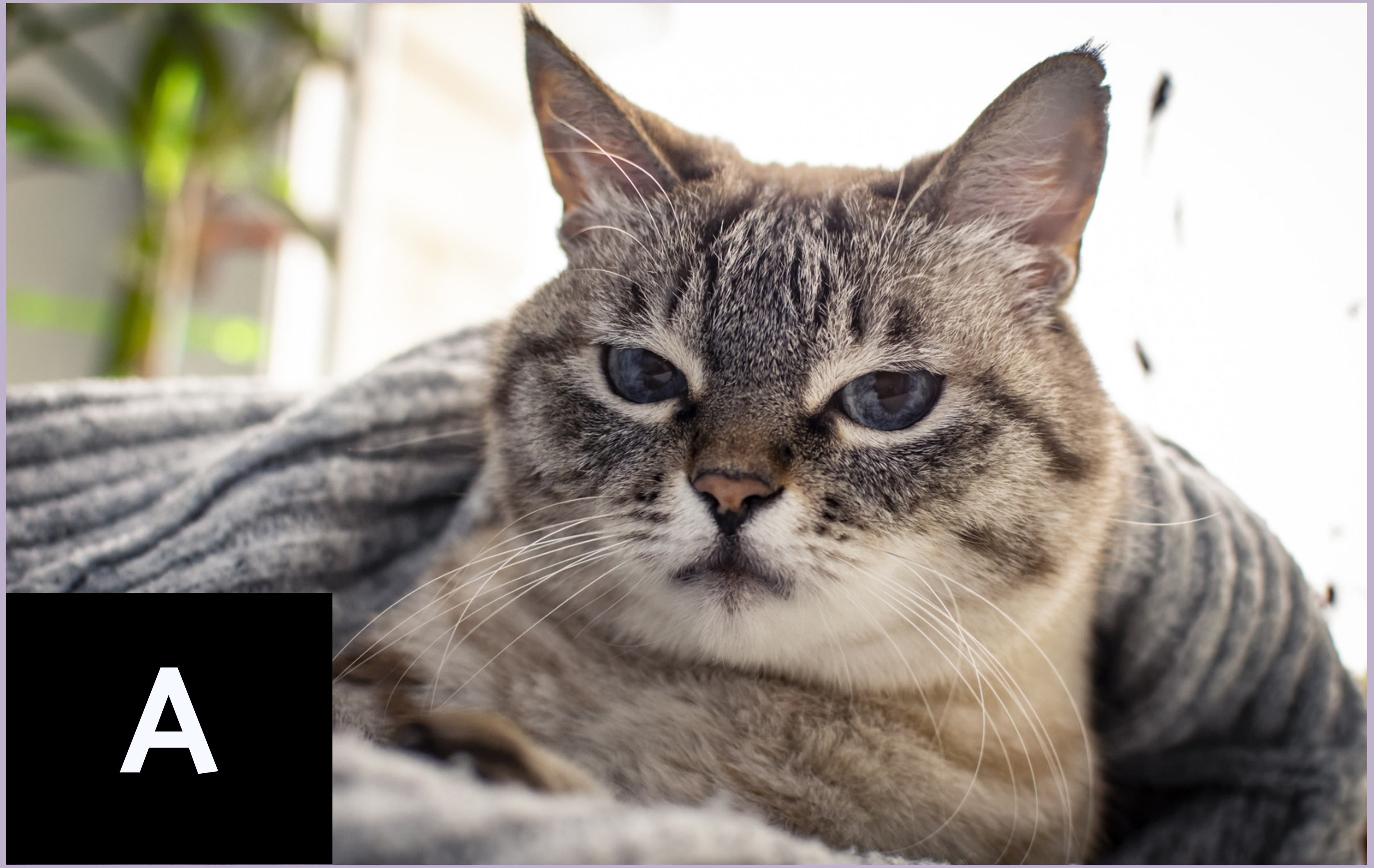
- Least Significant Bits
  - Embed message in the least x significant bits of the image
  - The more bits you use, the more obvious it is that you are hiding a message
- Images are typically encoded with 1 byte per color channel per pixel
  - To represent 1 pixel, you'll need 3 bytes:
    - 1 byte for red
    - 1 byte for green
    - 1 byte for blue
- Interpret your message/file to binary, then change the last bit of each of these bytes to the values of your secret

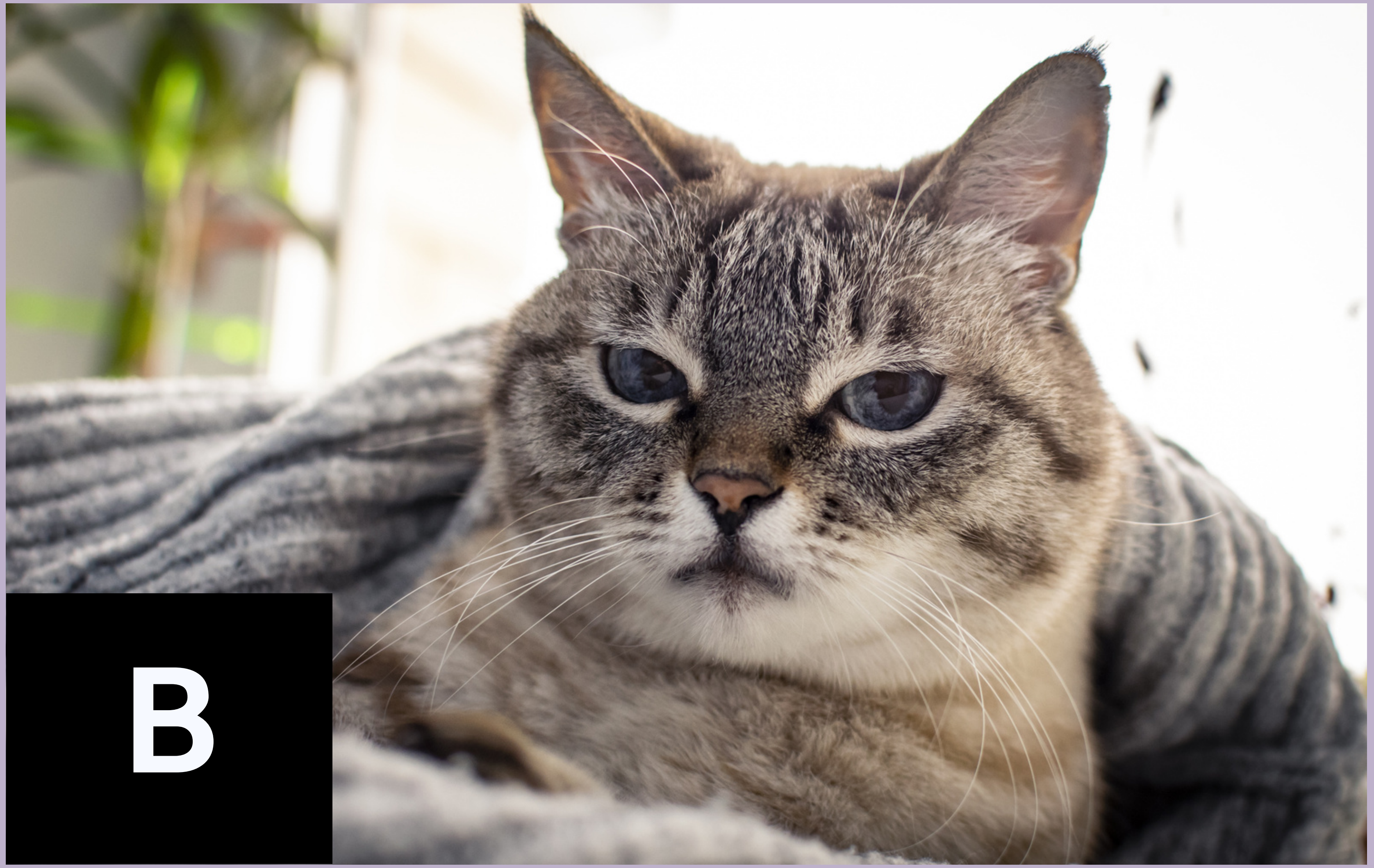


Observe:

	Red	Green	Blue
🟪	1011111**0**	1011000**1**	1100101**1**
🟪	1011111**1**	1011000**1**	1100101**1**
🟪	1011111**1**	1011000**0**	1100101**0**

Which is the original?



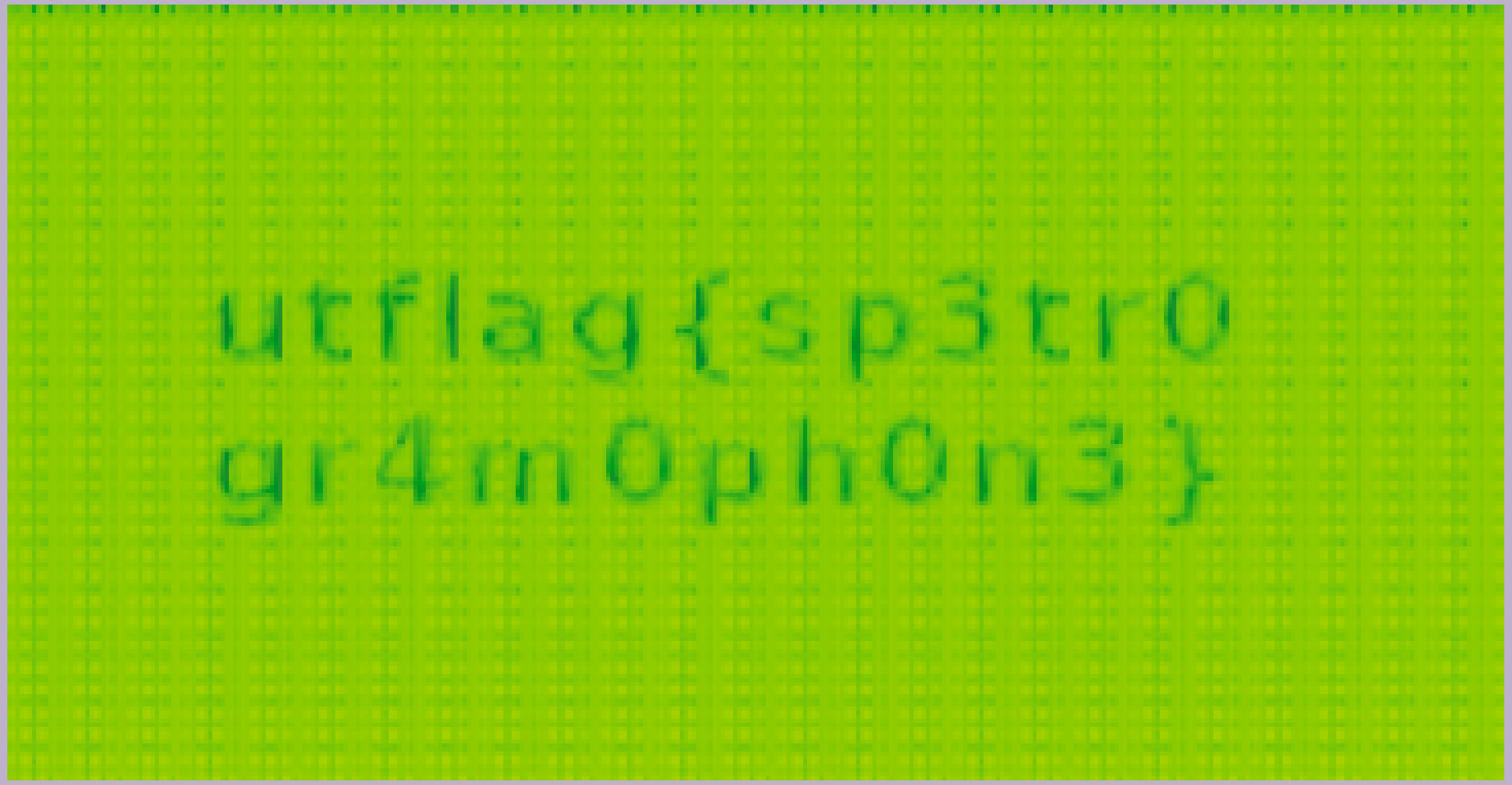


Steganography Tools

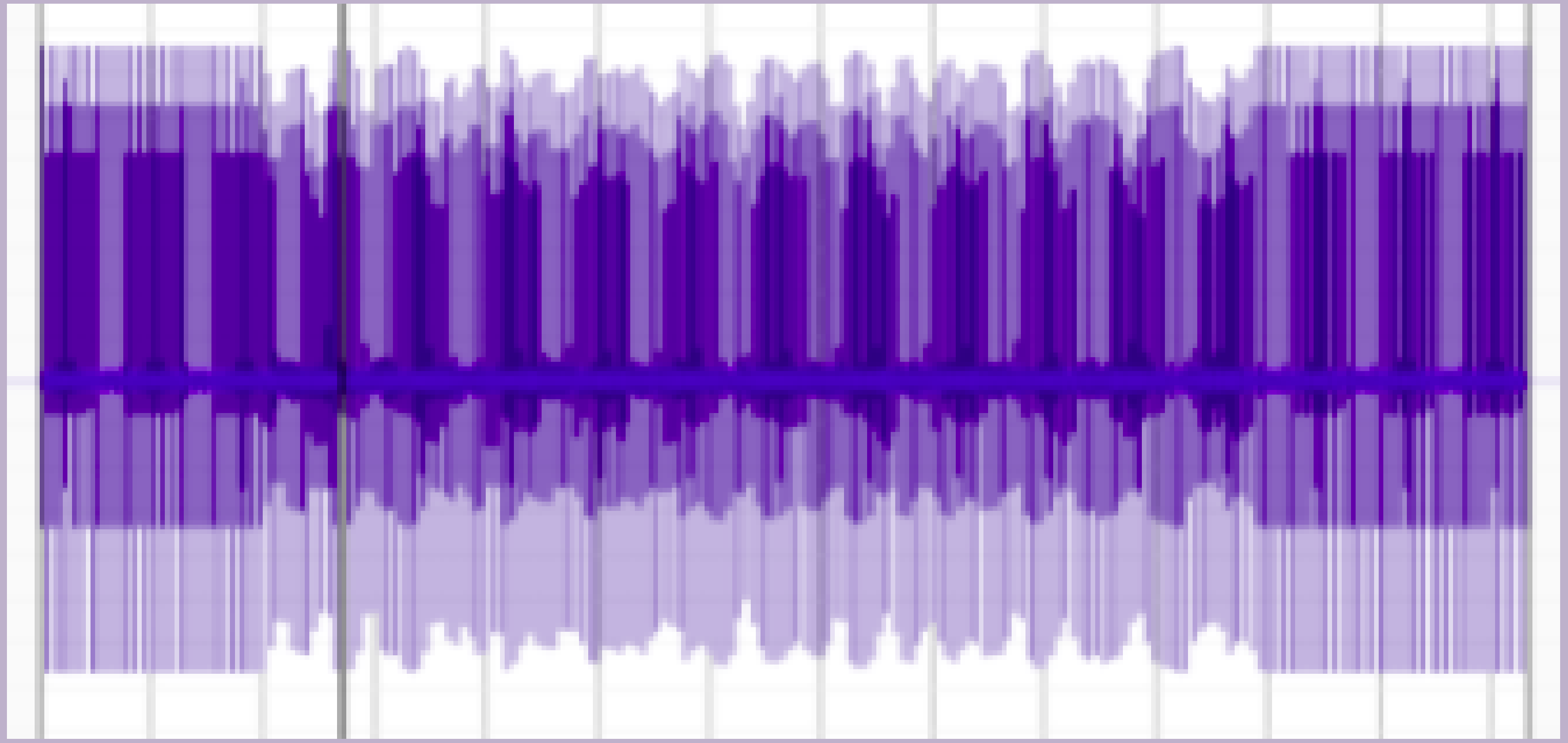
- To recover the message, you can write your own program to recover the bits of the secret, or you can use some of these tools
    - Stegsolve
    - Steghide
    - zSteg
    - Online decoder
    - Cyberchef

Audio Steganography

- Hide text or images in audio files that can only be seen by looking at the spectrogram
  - Spectrogram: visual representation of the spectrum of frequencies
- Can use tool like Sonic Visualizer or Audacity (both work on win, *nix, mac) to view spectrogram



spectrogram



waveform (default representation)

pane > add spectrogram

Disk Images & Partitions

- Sometimes you may even be given a disk image to work with
  - file that is really a copy of chunk (probably a partition) of hard drive
- Partition
  - explicit allocation of space on a drive
  - partitions isolated from one another
  - each can have their own file system type/OS/etc.
  - only boot to 1 at a time
  - access multiple at once

Mounting

- Way to access another partition
  - "attach" the other partition onto a folder (mount point)
  - Loop mounting: treat a file as an entire disk device

```
$ mkdir mntpt
$ ls mntpt
$ sudo mount [src] mntpt
$ ls mntpt
bin     dev    lib64    media        proc    sbin   swapfile  usr
boot    etc    lib      libx32       var     root   snap      sys
cdrom   home   lib32    lost+found   opt     run    srv       tmp
```

1. Create the mount point
 (Note there is nothing there currently)
2. Command to mount (scr will be file name)
(Note we can now see contents of disk

Practice!

- Visit forever.isss.io for some practice forensics problems
- Feel free to ask your neighbor or an officer for help if you're stuck :)
    - Can also leave a message later on in #foreverctf on the ISSS discord (isss.io/discord)
- Happy hacking! :)