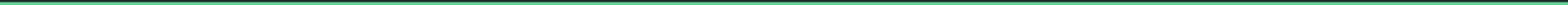


# FORENSICS

A magnifying glass icon with a blue circular lens and an orange handle, positioned to the right of the word "FORENSICS".

# What is it?

- Looking for digital “clues”
  - Data hiding in audio, video, images
  - Encoded data
  - Deleted or corrupted data
- It is a profession
- There are lots of videos of people talking about catching people hiding info, which are funny.

Here's one: DEFCON 21 - Forensic Fails

[https://www.youtube.com/watch?v=NG9Cq\\_vBKOg](https://www.youtube.com/watch?v=NG9Cq_vBKOg)

# Plan

- Briefly discuss tools and methods
- Give a demo (from DEFCON)
- Try some challenges

# Image Steganography

- Hiding data in images
- Techniques:
  - Hiding data in metadata
  - LSB
  - Hiding data in the image source
  - Encrypting data and “mixing” it into the image
  - Hiding data in the image itself

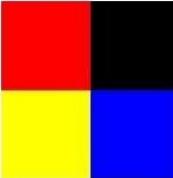
# Hiding data in the metadata:

- Metadata - gives basic information about the data (ie. its size, its location, creation time, etc)
- Sometimes data (by data I mean flags) is in the Comments of the metadata
- To read metadata: exiftool [FILENAME]
- You can also use exiftool to write to the metadata
- Ex:

```
crazyights@kali: ~/Downloads/ctf_primer_01/forensics$ exiftool -a f.02.wav
ExifTool Version Number      : 11.80
File Name                   : f.02.wav
Directory                   :
File Size                    : 394 kB
File Modification Date/Time : 2019:10:26 13:57:46-04:00
File Access Date/Time       : 2020:02:02 19:41:43-05:00
File Inode Change Date/Time : 2020:02:02 19:41:41-05:00
File Permissions            : rw-rw-r--
File Type                   : WAV
File Type Extension         : wav
MTIME/TIME                 : 2020-02-02 19:41:41 -0500
```

# LSB (Least Significant Bit):

Original Image

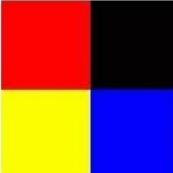


11111111	00000000
00000000	00000000
00000000	00000000

11111111	00000000
11111111	00000000
00000000	11111111

Least Significant Bit  
Steganography

Stego Image



11111101	00000011
00000010	00000001
00000000	00000010

11111100	00000011
11111101	00000001
00000001	11111100

c      a      t

01 10 00 11    01 10 00 01    01 11 01 00

A red curly brace groups the bottom row of the stego image's binary data, specifically the columns for the red pixels, which correspond to the letters 'c', 'a', and 't'.

More:

<https://www.cybrary.it/Op3n/hide-secret-message-inside-image-using-lsb-steganography/>

# Hiding data in the image source

Example:

```
sphil@athens:~/Documents/Screenshots/steg images$ hexdump anon.jpg | tail -10
*
000c150 28 00 a2 8a 28 00 a2 8a 28 00 a2 8a 28 00 a7 0f
000c160 ba 69 b4 e1 d0 d0 03 68 a2 8a 00 28 a2 8a 00 28
000c170 a2 8a 00 28 a2 8a 00 28 a2 8a 00 28 a2 8a 00 28
*
000c1a0 a2 8a 00 28 a2 8a 00 2b fa e5 ff 00 82 61 ff 00
000c1b0 c9 8d fc 35 ff 00 b8 cf fe 9e 2f 6b f9 1a af eb
000c1c0 97 fe 09 87 ff 00 26 37 f0 d7 fe e3 3f fa 78 bd
000c1d0 a0 0f ff d9
000c1d4
sphil@athens:~/Documents/Screenshots/steg images$ echo 'hello world' >> anon.jpg
sphil@athens:~/Documents/Screenshots/steg images$ hexdump anon.jpg | tail -10
*
000c150 28 00 a2 8a 28 00 a2 8a 28 00 a2 8a 28 00 a7 0f
000c160 ba 69 b4 e1 d0 d0 03 68 a2 8a 00 28 a2 8a 00 28
000c170 a2 8a 00 28 a2 8a 00 28 a2 8a 00 28 a2 8a 00 28
*
000c1a0 a2 8a 00 28 a2 8a 00 2b fa e5 ff 00 82 61 ff 00
000c1b0 c9 8d fc 35 ff 00 b8 cf fe 9e 2f 6b f9 1a af eb
000c1c0 97 fe 09 87 ff 00 26 37 f0 d7 fe e3 3f fa 78 bd
000c1d0 a0 0f ff d9 68 65 6c 6c 6f 20 77 6f 72 6c 64 0a
000c1e0
sphil@athens:~/Documents/Screenshots/steg images$
```



# Using steghide, stegosuite, ....

Steghide and stegosuite let you encrypt files and hide them inside another files content

- Limitations:

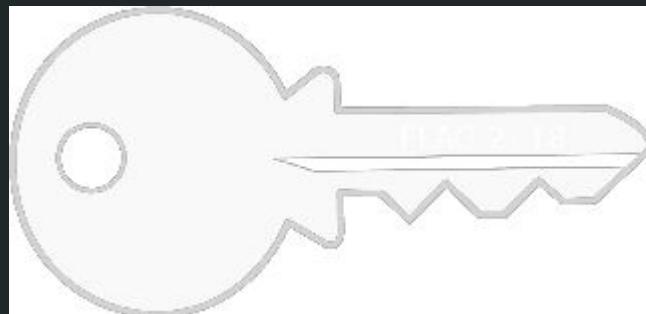
- The file used for attaching the hidden files (cover files) are limited to certain formats (JPEG, BMP, ...)
- The cover file must be bigger than the file you want to hide

According to its manual, this is what it does.

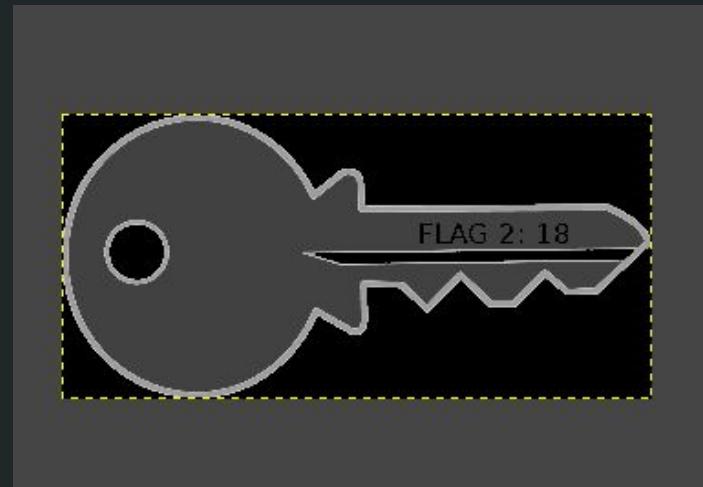
If you hide a text file using an image, then after ‘Steghide’ encrypts and compresses the text file, it’ll also ‘convert’ its data to look like it holds data of an image (pixels!). If you use an audio file instead of the image, then it’ll ‘convert’ the text data into ‘audio samples’. That’s how the data becomes ‘invisible’ once gets mixed.

# Hiding data in the image itself

- Used less frequently
- Text is hidden through a series of filters to the image
- Reversing the filters reveals to text



Linear Invert



# A quick example:

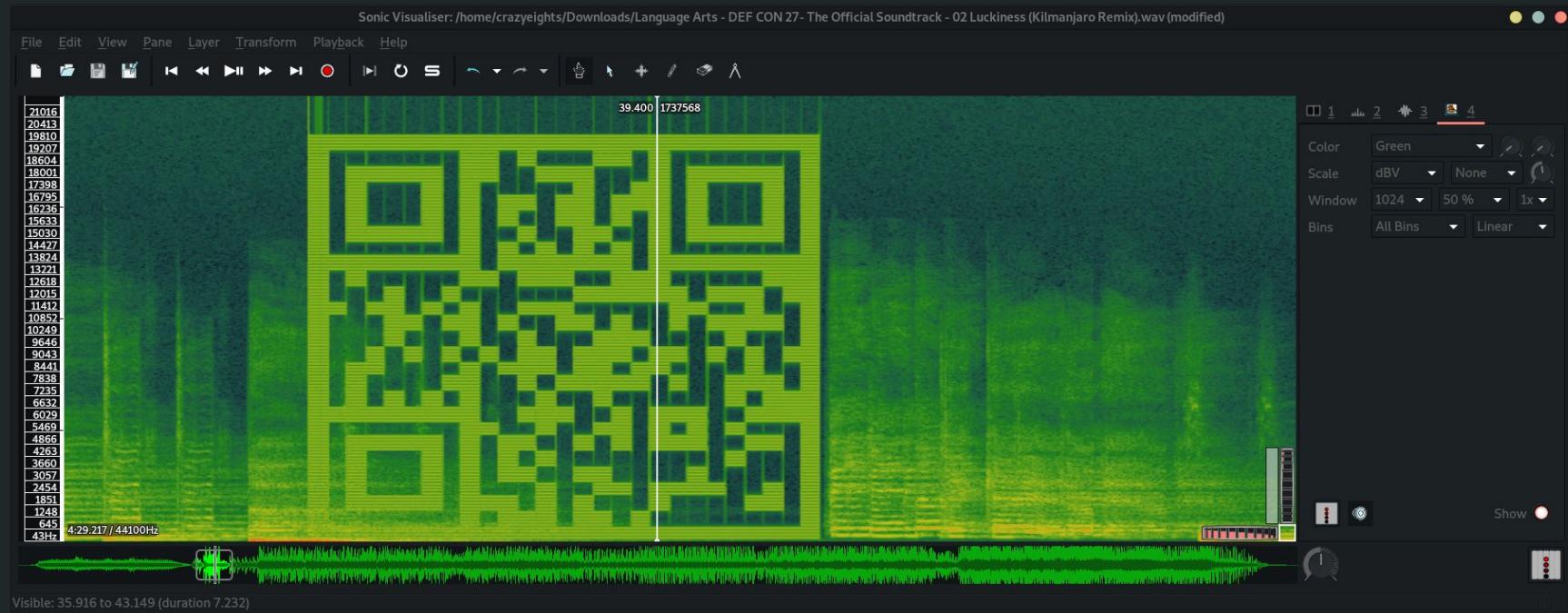
Find the flag given this audio file. The music actually quite pleasant :)

The screenshot shows a Windows context menu for a file named "Language Arts - DEF CON 27- The Official Soundtrack - 02 Luckiness (Kilmanjaro Remix).wav". The "Properties" tab is selected. The "General" section displays the following metadata:

Attribute	Value
Title	Luckiness (Kilmanjaro Remix)
Artist	Language Arts
Album	DEF CON 27: The Official Soundtrack
Year	2019
Duration	4 minutes 29 seconds
Comment	Visit <a href="http://defconcommunications.bandcamp.com">http://defconcommunications.bandcamp.com</a>
Container	WAV

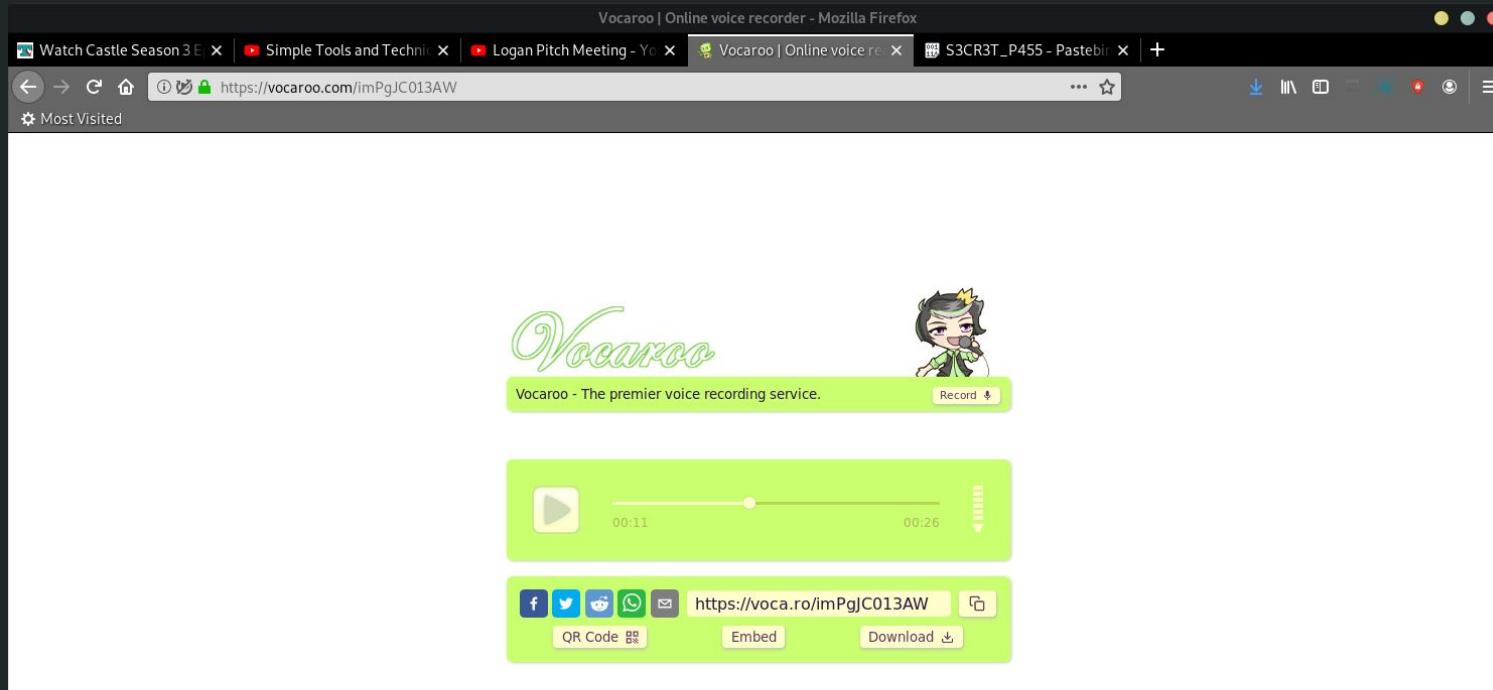
# A quick example:

Using Sonic Visualiser open the song, and add stegrogram layer:



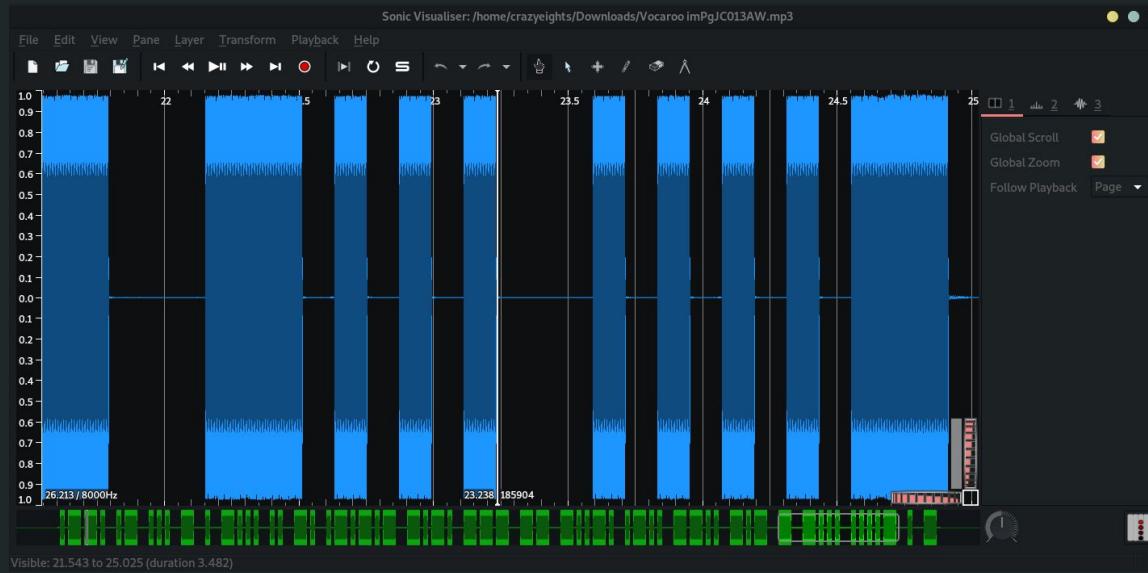
# A quick example:

Used a QR reader to get the corresponding url



# A quick example:

Listened to it and it just some beeps. Its morse code. Since I am not fast enough to translate just by listening I downloaded it and analysed the wavelengths: (You can see that dots are shorter and dashes are longer)



# A quick example:

I got:

```
Open ▾ +  
p a s t e b i n . c o m / l z k t b 4 e t  
pastebin.com/lzktb4et
```

The screenshot shows the Pastebin website interface. At the top, there's a navigation bar with links for 'GO PRO', 'API', 'TOOLS', 'FAQ', 'DEALS', and a search bar. Below the navigation is a post titled 'Untitled' by 'A GUEST' posted on 'JAN 22ND, 2020' with '40' views and 'NEVER' expiration. The post content area is empty. Below it, a message encourages users to sign up for more features. The 'RAW Paste Data' section contains the text 'S3CR3T\_P455'. To the right, a sidebar lists recent public pastes from other users, such as 'tes', 'Untitled', 'Anal Mathematics', and 'emptyChest'. A 'steadfast' logo is visible. At the bottom, there's a cookie consent message and a 'Sign Up' button.

Untitled  
A GUEST JAN 22ND, 2020 40 NEVER

Not a member of Pastebin yet? [Sign Up](#). It unlocks many cool features!

text 0.01 KB  
1. S3CR3T\_P455

raw download clone embed report print

**RAW Paste Data**

S3CR3T\_P455

Recommended by

We use cookies for various purposes including analytics. By continuing to use Pastebin, you agree to our use of cookies as described in the [Cookies Policy](#).

OK, I Understand

Waiting for pastebin.com...

HOSTED BY **steadfast**

Not a member of Pastebin yet?  
[Sign Up](#), it unlocks many cool features!

# A quick example:

Use the password to retrieve the flag from the image:

```
crazyeights@kali:~/Downloads$ steghide extract -sf "Language Arts - DEF CON 27- The Official Soundtrack - 02 Luckiness (Kilmanjaro Remix).wav" -p S3CR3T_P455
wrote extracted data to "secret".
crazyeights@kali:~/Downloads$
```

```
crazyeights@kali:~/Downloads$ cat secret
THM{f0und_m3!}
crazyeights@kali:~/Downloads$
```

# Tools Needed:

- exiftool
- steghide
- binwalk
- wireshark
- sonic visualizer
- zsteg
- stegoveritas
- Hex editor (I use ghex)
- CYBERCHEF (or other online tools)
- <https://www.dcode.fr> => Has many different decoders
- <https://copy.sh/brainfuck/> ⇒ Brainf\*ck compiler

# Types of codes:

- Different Data Formats: base85, base64, base62, base32, base16
  - Ascii is base10
- Different Encoding types: ROT13 (CAESAR CIPHER), ROT47, VIGENERE CIPHER
  - Characters are rotated, ie. ROT3 A ⇒ D
- XOR cipher, morse code
- Different Languages: Aurebesh, Pigpen Cipher, Brainf\*ck

Challenges, ya let's go! 🏆👽

---

# CTF: functf: this was retired now. :(

<https://www.tryhackme.com/room/functf>

No SSH Connection Required

The screenshot shows the TryHackMe website interface. At the top, there is a navigation bar with a cloud icon containing binary code (10 10 1110 0101 0101 01), the text "Try Hack Me", and links for "Forum", "Feedback", "Profile", and "Logout". Below the navigation bar, the main content area features a title "CaptureTheFlag" with a green flag icon, described as a "beginner level CTF". There are up and downvote buttons on the left. On the right, there are "Share" and "Options" buttons. Below the title, there are tabs for "Chart", "Scoreboard", "Chat", "Writeups", and "More", with "Chart" being the active tab. A "Difficulty" slider is set to yellow. At the bottom, it says "Highest Scoring Users" and shows the number 400.

In my github:

<https://github.com/crazeights225/CCSC/tree/master/functf-retired>

Challenges in own folder with challenge description

Flag format: tryhackme{...}

[Part 1] #1 Do Images have strings?

---

# #1 Do Images have strings?

The hint here is strings

strings - print the sequences of printable characters in files

```
root@kali:~/Downloads# strings Basic.jpg
```

JFIF

ICC\_PROFILE

...

tryhackme{7h1s\_i5\_wh4t\_strings\_d0es} ← ANSWER TO #1

[Part 1] #2 Metadata or EXIF data?....ah!! I'm so confused

---

## #2 Metadata or EXIF data?....ah!! I'm so confused

- Metadata or Exif data can be viewed with exiftool

```
root@kali:~/Downloads# exiftool Basic.jpg
```

...

Comment	: dHJ5aGFja2llezRsd2F5NV9jaDNja19tM3Q0ZGE3NH0K
Image Width	: 404
Image Height	: 404
Encoding Process	: Progressive DCT, Huffman coding

# #2 Metadata or EXIF data?....ah!! I'm so confused

The screenshot shows the CyberChef web application interface. On the left, there's a sidebar with various operations listed under 'Operations' and 'Favourites'. The 'Favourites' section includes 'To Base64', 'From Base64', 'To Hex', 'From Hex', 'To Hexdump', 'From Hexdump', and 'URL Decode'. The main workspace shows a 'Recipe' card titled 'From Base64'. It has settings for 'Alphabet' (set to 'A-Za-z0-9+/='), a checked checkbox for 'Remove non-alphabet chars', and a status bar indicating 'start: 44 end: 44 length: 44 lines: 1'. Below the card, the input 'dHJ5aGFja21leRsd2F5NV9jaDNja19tM3Q0ZGE3NH0K' is processed, resulting in an output of 'tryhackme{4lway5\_ch3ck\_m3t4da74}'.

Download CyberChef [Download](#)

Last build: 7 days ago - v9 supports multiple inputs and a Node...

Options [⚙️](#) About / Support

**Operations**

Search...

**Favourites** [★](#)

- To Base64
- From Base64
- To Hex
- From Hex
- To Hexdump
- From Hexdump
- URL Decode

**Recipe**

From Base64

Alphabet  
A-Za-z0-9+/=

Remove non-alphabet  
chars

**Input**

start: 44 end: 44 length: 44 lines: 1

dHJ5aGFja21leRsd2F5NV9jaDNja19tM3Q0ZGE3NH0K

**Output**

start: 33 end: 33 time: 23ms length: 33 lines: 2

length: 0

tryhackme{4lway5\_ch3ck\_m3t4da74}

[Part 2] #1 Find the flag.

---

# #1 Find the flag.

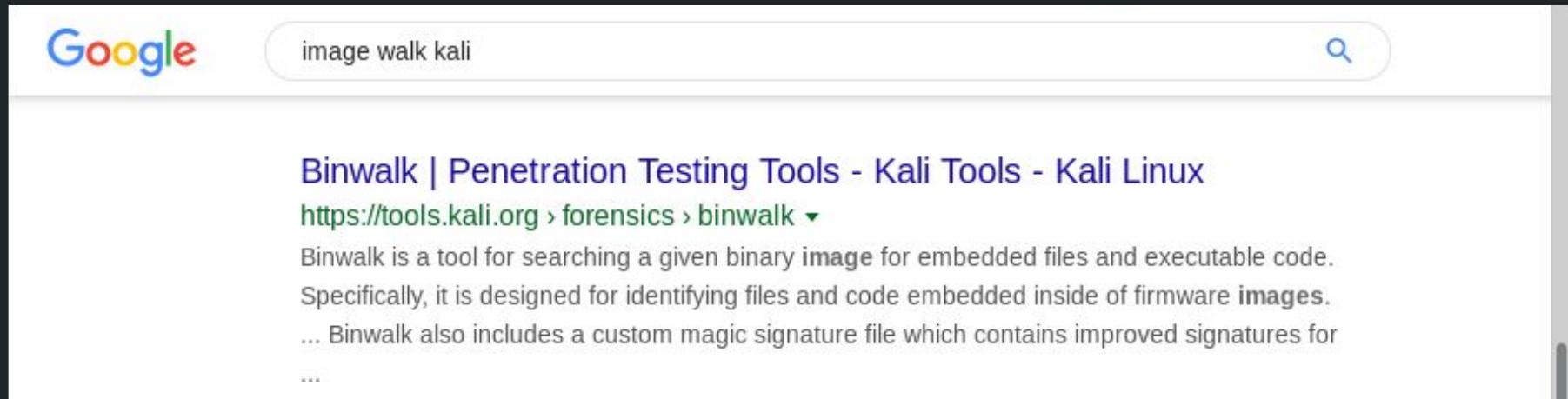
Download the next image: walk.jpg

I can make this easy just by telling you the tool or maybe you can read the title again and figure out your self.

P.S - It's a very famous, open source tool :)

# #1 Find the flag.

The image name is walk, so:



A screenshot of a Google search results page. The search bar at the top contains the query "image walk kali". Below the search bar, the first result is a link to the Binwalk tool on the Kali Linux website. The link text is "Binwalk | Penetration Testing Tools - Kali Tools - Kali Linux" and the URL is "https://tools.kali.org › forensics › binwalk ▾". A snippet of the page content below the link describes Binwalk as a tool for searching binary images for embedded files and executable code, specifically for identifying files and code embedded inside firmware images. Ellipses indicate more content follows.

Binwalk | Penetration Testing Tools - Kali Tools - Kali Linux  
<https://tools.kali.org/forensics/binwalk> ▾

Binwalk is a tool for searching a given binary image for embedded files and executable code. Specifically, it is designed for identifying files and code embedded inside of firmware images.

... Binwalk also includes a custom magic signature file which contains improved signatures for

...

The tool is binwalk

# #1 Find the flag.

binwalk:

binwalk - tool for searching binary images for embedded files and executable code

Param:

-e, --extract      Automatically extract known file types

# #1 Find the flag.

```
root@kali:~/Downloads# binwalk -e walk.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
-----		
0	0x0	JPEG image data, JFIF standard 1.01
30	0x1E	TIFF image data, big-endian, offset of first image directory: 8
170610	0x29A72	gzip compressed data, from Unix, last modified: 2019-04-21 08:25:56

```
root@kali:~/Downloads# ls
```

\_walk.jpg.extracted

# #1 Find the flag.

```
root@kali:~/Downloads# cd _walk.jpg.extracted/
root@kali:~/Downloads/_walk.jpg.extracted# ls
29A72 29A72.gz
root@kali:~/Downloads/_walk.jpg.extracted# cat 29A72
PaxHeader/flag.txt000644 001750 001751 00000000066 13457024252 014506
xustar00mzfrmzfr000000 000000 30 mtime=1555835050.729934811
24 SCHILY.fflags=extent
flag.txt000644 001750 001751 00000000352 13457024252 012533
Oustar00mzfrmzfr000000 000000 hmm..So you've got the flag.txt file good!!
Now let's play a bit with bases
This is the flag but it's encoded twice with 2 different bases. Figure it out

T1JaSFMyREJNTIZXMIpMM01JWVc0NVpVTIJWVjZNRFNMNVREQTRSVE5WWFRL
NUQ1Qkk9PT09PT0K
```

# #1 Find the flag.

I love cyberchef

Download CyberChef [↓](#)

From Binary

To Octal

From Octal

To Base64

From Base64

Show Base64 offsets

To Base32

From Base32

To Base58

From Base58

Recipe [🔗](#) [📁](#) [📄](#) [🗑️](#)

From Base64 [🔗](#) [⏸️](#)

Alphabet A-Za-z0-9+ ...

Remove non-alphabet chars

From Base32 [🔗](#) [⏸️](#)

Alphabet A-Z2-7=

Remove non-alphabet chars

Input length: 76  
lines: 1 [+ ↻](#) [📁](#) [📄](#) [🗑️](#) [🔗](#)

T1JaSFM...jZNR  
FNMNVREQTRSVE5wWFRNUQ1Qkk9PT09PT0K

Output time: 7ms  
length: 31  
lines: 2

tryhackme{b1nw4lk\_0r\_f0r3mo5t}

# [Part 3] #1 Find the Flag

---

# #1 Find the flag.

Download the next image: hide.jpg

Hint: You know the drill, focus on the Title.

# #1 Find the flag.

This tool is really popular:

steghide - a steganography program

To extract:

Example:

```
$ steghide extract -sf picture.jpg
```

Enter passphrase:

wrote extracted data to "secret.txt".

# #1 Find the flag.

```
root@kali:~/Downloads# steghide extract -sf hide.jpg
```

Enter passphrase:

```
steghide: could not extract any data with that passphrase!
```

```
root@kali:~/Downloads#
```

Oh No. The passphase must be hidden in the image.

# #1 Find the flag.

You can find the password 2 ways:

```
root@kali:~/Downloads# strings hide.jpg
JFIF
ORZHS2BUMNVW2MYK
$3br
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
      #3R
&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
H@9l|
3_x
NM+U
V[$2\
```

```
EAuy-
root@kali:~/Downloads# exiftool hide.jpg
ExifTool Version Number      : 11.77
File Name                    : hide.jpg
Directory                   : .
File Size                   : 56 kB
File Modification Date/Time : 2019:12:27 19:23:3
File Access Date/Time       : 2019:12:27 19:28:4
File Inode Change Date/Time : 2019:12:27 19:23:4
File Permissions            : rw-r--r--
File Type                   : JPEG
File Type Extension         : jpg
MIME Type                   : image/jpeg
JFIF Version                : 1.01
Resolution Unit              : inches
X Resolution                : 300
Y Resolution                : 300
Comment                      : ORZHS2BUMNVW2MYK
Image Width                 : 426
```

# #1 Find the flag.

- Tried ORZHS2BUMNVW2MYK
- Realized it was encoded
- Used cyberchef

The screenshot shows the CyberChef interface with two main sections: "Input" and "Output".

**Input:** The input string is "ORZHS2BUMNVW2MYK". Above the input, there are statistics: start: 17, end: 17, length: 17, lines: 0, and length: 2.

**From Base64:** This section is active, indicated by a grey background. It includes:

- A dropdown menu for "Alphabet" containing "A-Za-zA-Z0-9+=".
- A checkbox labeled "Remove non-alphabet chars" which is checked.

**From Base32:** This section is shown below the first one, with a green background. It includes:

- A dropdown menu for "Alphabet" containing "A-Za-zA-Z0-78-7".

**Output:** The output string is "tryh4ckm3". Above the output, there are statistics: time: 3ms, length: 10, and lines: 2.

# #1 Find the flag.

```
root@kali:~/Downloads# steghide extract -sf hide.jpg
```

```
Enter passphrase:
```

wrote extracted data to "flag-1.txt".

```
root@kali:~/Downloads# cat flag-1.txt
```

Steghide is a great tool to find some hidden data that couldn't be extracted using binwalk.

Note: steghide doesn't need password always

tryhackme{st3gh1d3\_i5\_l0v3}

[Part 4] #1 Find the flag.

---

# #1 Find the flag.

Download: stegano.png

Hint:

Hiding data in LSB are a very common process. Especially in CTFs.

The most famous tool used for this is KDE68

P.S: Name of the tool is encrypted in a version of ROT cipher.

P.P.S: I repeat decode KDE68 to find the name of the tool.

(Hint look up ROT13 variants)

# #1 Find the flag.

Decode KDE68

- Tried a bunch of different things until something worked



Search for a tool

★ SEARCH A TOOL ON DCODE BY KEYWORDS:  
e.g. type caesar  GO

Results

**zsteg**  

ROT-47 Cipher - [uCode](#)

Tag(s) : Substitution Cipher, Internet

## ROT-47 CIPHER

Cryptography > Substitution Cipher > ROT-47 Cipher

### ROT47 Decoder

★ ROT47 CIPHERTEXT

KDE68

**DECRYPT ROT47**

See also: [ROT Cipher](#) — [ROT-13 Cipher](#) — [Caesar Cipher](#)

# #1 Find the flag.

- You have to download zsteg

<https://github.com/zed-0xff/zsteg>

- Extract and run:

```
root@kali:~/zsteg-master# gem install zsteg
```

# #1 Find the flag.

```
root@kali:~/Downloads# zsteg stegano.png
imagedata          .. text: "yxw46+%)"
b1,bgr,lsb,xy     .. text:
"=flag=4wbWVHV1VA43QJtvWdw8pLCwkADDQ7ZdYkz39KsKaXUeLtPy9DShWSp\n
....
```

# #1 Find the flag.

I love cyberchef

Last build: 7 days ago - v9 supports multiple inputs and a Node API ... Options About / Support

**Recipe** **Input** length: 55  
lines: 1

Remove non-alphabet chars

**From Base58**   
Alphabet 123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ ...

Remove non-alphabet chars

**From Base32**   
Alphabet A-Z2-7=

Remove non-alphabet chars

**Output** start: 24 time: 10ms  
end: 24 length: 24  
length: 0 lines: 1

tryhackme{lsb\_4r3\_l1t!!}

The screenshot shows the CyberChef interface with a pipeline for decoding a Base58 string. The input is a long Base58 encoded string: 4wbW...Sp. The first step in the recipe is to "Remove non-alphabet chars". Below this, there are two conversion steps: "From Base58" and "From Base32". Both steps have their "Remove non-alphabet chars" checkboxes checked. The output of the pipeline is the ASCII string: tryhackme{lsb\_4r3\_l1t!!}.

[Part 5] #1 Since you've been working hard..I wanted to hand out the flag to you but my dumb friend messed the whole image. Fix the image to get the flag.

---

# #1 Fix the image.

Download: flag.png

There are a lot of ways to mess a file. The most common one is to play with its headers.

NOTE: The flag is not in the tryhackme{}. For submission add tryhackme{} around the found flag.

# #1 Fix the image.

Open the image with ghex and check the file signature: 17 23 44 28 0D 0A 1A ..

The screenshot shows the GHex hex editor interface with the title bar "flag.png - GHex". The menu bar includes File, Edit, View, Windows, and Help. The main window displays the hex dump of the file. The first few bytes are highlighted with red boxes: byte 0 (17), byte 1 (23), byte 2 (44), byte 3 (28), byte 4 (0D), byte 5 (0A), and byte 6 (1A). The text area shows the following bytes:

Address	Value	Text
00000000	17 23 44 28 0D 0A 1A 0A	IHDR
00000013	E8 00 00 03 E8 08 02 00	C...p
00000026	48 59 73 00 00 2E 23 00	HYS.#.#.x.?v...
00000039	07 74 49 4D 45 07 E3 04	.tIME....#/0...
0000004C	19 74 45 58 74 43 6F 6D	.tEXtComment.Create
0000005F	6D 65 6E 74 00 43 72 65	d with GIMPW.... .
00000072	61 74 65 F2 16 IDATx...gW#K.....	
00000085	33 E6 FE FF FF EB 59 7B	Y{.#..@..p..M
00000098	06 23 EF 91 40 C0 F9 70	....#S...fk@.....
000000AB	1D E2 4D 3....	
0000008C	8A 8C 8C 08 B1 .<zyy.B..."D...!B...	

At the bottom, there are three status bars: "Signed 8 bit": 02, "Signed 32 bit": 675554071, and "Unsigned decimal": 17.

# #1 Fix the image.

Lookup the file signature for png and compare it with:

17 23 44 28 0D 0A 1A ..

89 50 4E 47 0D 0A 1A 0A	.PNG....	0	png	Image encoded in the <a href="#">Portable Network Graphics format</a> <sup>[13]</sup>
----------------------------	----------	---	-----	--

This doesn't match.

Edit the hex on flag.png to match, and then save it.

# #1 Fix the image.

flag.png - GHex

File Edit View Windows Help

```
00000000 89 50 4E 47 0D 0A 1A 0A 00 00 00 00 0D 49 48 44 52 00 00 00 03 .PNG. .... IHDR ...
00000013 E8 00 00 03 E8 08 02 00 00 00 C2 C1 43 B3 00 00 00 09 70 ..... C.... p
00000026 48 59 73 00 00 2E 23 00 00 2E 23 01 78 A5 3F 76 00 00 00 HYS....#....#.x.?v...
00000039 07 74 49 4D 45 07 E3 04 15 0B 07 23 2F 30 CA B5 00 00 00 .tIME.....#/0....
0000004C 19 74 45 58 74 43 6F 6D 6D 65 6E 74 00 43 72 65 61 74 65 .tEXtComment.Create
0000005F 64 20 77 69 74 68 20 47 49 4D 50 57 81 0E 17 00 00 20 00 d with GIMPW.... .
00000072 49 44 41 54 78 DA EC BD 67 57 23 4B D3 B4 DB 80 10 F2 16 IDATx...gW#K.... .
00000085 33 E6 FE FF FF EB 59 7B 06 23 EF 91 40 C0 F9 70 1D E2 4D 3.....Y{.#.@..p..M
00000098 AA BA 1B 01 12 23 53 F9 81 C5 66 6B 40 EA EE AA 8A 8C 8C .....#S...fk@.... .
000000AB 8C 3C 7A 79 79 89 42 84 08 11 22 44 88 10 21 42 84 08 B1 .<zyy.B...."D...!B...
```

Signed 8 bit: 13      Signed 32 bit: 169478669      Hexadecimal: 0D

#1 Fix the image.

The fixed image is:

And the flag is:

tryhackme{LoL\_m355ed\_H34D3  
R5\_FoR\_th15?}

LoL\_m355ed  
-H34D3R5 -  
FoR\_th15?

# [Part 6] #1 Audio?!

---

# #1 Audio?!

Download flag.wav

Hint:

---

HACKER1: FBI is onto me that is why I am sending you a hidden message in an audio file.

HACKER2: What? Audio file...how the hell is that safe.

H1: It is because audio has nothing to do with it.

H2: So how can I see it.

H1: Just check the spectro.....

-----DISCONNECTED-----

This was the conversation intercepted by FBI between two hackers. FBI has provided you with the audio file can you help then find the message?

# #1 Audio?!

In the hint it says check the spectro  
After much searching I found a tool:

sonic-visualiser/kali-rolling 4.0-1 amd64  
viewing and analysing the contents of music audio files

Downloaded it and opened the file

# Sonic Visualiser: /root/Downloads/flag.wav



File Edit View Pane Layer Transform Playback Help



Click and drag to navigate; use mouse-wheel or trackpad-scroll to zoom; hold Shift and drag to zoom to an area

# #1 Audio?!

To reveal the flag:

Layer > Add Spectrogram

File Edit View Layer Transform Playback Help



Visible: 0.394 to 16.700 (duration 16.305)

# #1 Audio?!

That is so cool...

Flag is:

tryhackme{NOWUS33M3}

[Part 7] #1 Let's start with the basic

---

# #1 Let's start with the basic

Let's start with the basic:

Aopz pz h Jhlzhy jpwoly zopmalk zlclu wvzpapvuz zv h pz lxbpchslua av o huk  
zv vu.

Doha fvb ullk pz h mshn ypnoa ayfohjrtl{Uv\_jhlzhy\_Uv\_Jyfwav}

## #1 Let's start with the basic

Text has been shifted. We have to figure out how much.

The last bit in the phrase is obviously the flag:

ayfohjrtl{Uv\_jhlzhy\_Uv\_Jyfwav}

ayfohjrtl == tryhackme

# #1 Let's start with the basic

## Using ROT13

ROT13 - CyberChef - Mozilla Firefox

Home My Drive - Google Drive CCSC: Stego - Google S image walk kali - Google TryHackMe | functf ROT13 - CyberChef

https://gchq.github.io/CyberChef/#recipe=ROT13(true,true,19)&input=YXlmb2hqcNRs

Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB GHDB MSFU

Last build: 7 days ago - v9 supports multiple inputs and a Node API allowing you to program with Cyb... Options About / Support

Download CyberChef

Triple DES Encrypt

Triple DES Decrypt

RC2 Encrypt

RC2 Decrypt

RC4

RC4 Drop

ROT13

ROT47

**Recipe**

ROT13

Rotate lower case chars  Rotate upper case chars

Amount  
19

**Input**

ayfohjrtl

length: 9  
lines: 1

**Output**

time: 3ms  
length: 9  
lines: 1

The screenshot shows the CyberChef interface in a Mozilla Firefox browser. The title bar reads "ROT13 - CyberChef - Mozilla Firefox". The address bar shows the URL "https://gchq.github.io/CyberChef/#recipe=ROT13(true,true,19)&input=YXlmb2hqcNRs". Below the address bar is a toolbar with various Kali Linux links. The main interface has a sidebar on the left with encryption/decryption options like Triple DES, RC2, RC4, etc. The central area is divided into Recipe, Input, and Output sections. The Recipe section shows "ROT13" selected with checkboxes for "Rotate lower case chars" and "Rotate upper case chars", and an "Amount" input set to 19. The Input section contains the string "ayfohjrtl". The Output section shows the result "YXlmb2hqcNRs". Status bars at the bottom indicate "length: 9", "lines: 1", "time: 3ms", and "length: 9", "lines: 1".

# #1 Let's start with the basic

ROT13 - CyberChef - Mozilla Firefox

Home My Drive - Google Drive CCSC: Stego - Google S image walk kali TryHackMe | functf ROT13 - CyberChef

https://gchq.github.io/CyberChef/#recipe=ROT13(true,true,19)&input=YXlmb2hqcNRsCgpBb3E

Last build: 7 days ago - v9 supports multiple inputs and a Node API allowing you to program with Cyb... Options About / Support

Download CyberChef

Triple DES Encrypt

Triple DES Decrypt

RC2 Encrypt

RC2 Decrypt

RC4

RC4 Drop

ROT13

ROT47

XOR

XOR Brute Force

**Recipe**

ROT13

Rotate lower case chars

Rotate upper case chars

Amount 19

**Input**

ayfohjrtl

Aopz pz h Jhlzhy jpwoly zopmalk zlclu wvzpapvuz zv h pz lxbpchsua av o huk zv vu.  
Doha fvb ullk pz h mshn ypnoa ayfohjrtl{Uv\_jhlzhy\_Uv\_Jyfwav}

**Output**

tryhackme

This is a Caesar cipher shifted seven positions so a is equivalent to h and so on.  
What you need is a flag right tryhackme{No\_caesar\_No\_Crypto}

The screenshot shows the CyberChef web application interface. On the left, a sidebar lists various encryption and decryption recipes. The main area is titled 'ROT13 - CyberChef - Mozilla Firefox'. It has tabs for 'Recipe' (selected), 'Input', and 'Output'. Under 'Recipe', 'ROT13' is chosen with options for rotating lowercase and uppercase characters (both are checked). The 'Amount' is set to 19. In the 'Input' section, the text 'ayfohjrtl' is entered. Below it, the decrypted output 'tryhackme' is shown. A note at the bottom explains that this is a Caesar cipher shift and provides the correct flag format.

[Part 7] #2 Let's start with the basic

---

## #2 Let's start with the basic

Guvf gvzr gurl ner fuvsgrq guvegrra cbfvgvbaf gung vf jul vg'f pnyrq EBG  
guvegrra.

SYNT: gelunpxzr{ebg\_guvegrra\_vf\_nyfb\_pnrfne\_pvcure}

## #2 Let's start with the basic

Focusing on this

SYNT: gelunpxzr{ebg\_guvegrra\_vf\_nyfb\_pnrfne\_pvcure}

SYNT is hint maybe?

- (Its actually flag)
- Using the same technique

# #1 Let's start with the basic

ROT13 - CyberChef - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Home My Drive - Google Drive CCSC: Stego - Google Slides image walk kali - Google Search TryHackMe | functf ROT13 - CyberChef +

Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB GHDB MSFU

Download CyberChef [Download](#)

Last build: 7 days ago - v9 supports multiple inputs and a Node API allowing you to program with CyberChef... Options [About / Support](#) ?

Recipe: ROT13  
 Rotate lower case chars  
 Rotate upper case chars  
Amount: 39

Input:  
SYNT: gelunpxzr  
Guvf gvzr gurl ner fuvsgrq guvegrra cbfgvbaf gung vf jul vg'f pnyrq EBG guvegrra.  
SYNT: gelunpxzr{ebg\_guvegrra\_vf\_nyfb\_pnrfne\_pvcure}

Output:  
start: 154 end: 154 length: 154 lines: 6  
start: 154 time: 17ms end: 154 length: 0 lines: 6  
FLAG: tryhackme  
This time they are shifted thirteen positions that is why it's called ROT thirteen.  
FLAG: tryhackme{rot\_thirteen\_is\_also\_caesar\_cipher}

[Part 7] #3 What the hell is this?

---

## #3 What the hell is this?

(@29]]]H:== E9:D 6G6C DE@An x >62? H6 42? ;FDE D9:7E E@ 2?J 2>@F?E @7  
A@D:E:@?D H:E9 H926G6C 492C24E6C D6E]  
ECJ924<>6Lu=2v0xD0p==0x0Hp?E0?@0q\$N

## #3 What the hell is this?

(@29]]]H:== E9:D 6G6C DE@An x >62? H6 42? ;FDE D9:7E E@ 2?J 2>@F?E @7  
A@D:E:@?D H:E9 H926G6C 492C24E6C D6E]  
ECJ924<>6Lu=2v0xD0p==0x0Hp?E0?@0q\$N

- They have probably shifted more than just letters
- ⇒ The encoding that does that is ROT47

# #3 What the hell is this?

ROT47 - CyberChef - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Home My Drive - Google Drive CCSC: Stego - Google S image walk kali - Google TryHackMe | functf ROT47 - CyberChef +

Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB GHDB MSFU

https://gchq.github.io/CyberChef/#recipe=ROT47(47)&input=KEAyOV1dXUg6PT0gRTk6RCA2 ... ⚡ ☆

Download CyberChef [Download](#) Last build: 7 days ago - v9 supports multiple inputs and a Node API allowing you to program with CyberChef Options [⚙️](#) About / Support [?](#)

Triple DES Encrypt

Triple DES Decrypt

RC2 Encrypt

RC2 Decrypt

RC4

RC4 Drop

ROT13

ROT47

XOR

XOR Brute Force

**Recipe** [🔗](#) [📁](#) [trash](#)

**ROT47** [🔗](#) [⏸](#)

Amount 47 [⬆️](#) [⬇️](#)

**Input** start: 143 end: 143 length: 143 lines: 1

(@29]]]H:== E9:D 6G6C DE@An x >62? H6 42? ;FDE D9:7E E@ 2?J 2>@F?E @7 A@D:E:@?D H:E9 H926G6C 492C24E6C D6E] ECJ924<>6Lu=2v0xD0p==0x0Hp?E0?@0q\$N

**Output** start: 143 time: 5ms end: 143 length: 143 lines: 1

Woah...will this ever stop? I mean we can just shift to any amount of positions with whatever character set. tryhackme{FlaG\_Is\_All\_I\_wAnt\_no\_BS}

# [Part 10] #1 Exclusive Or Random

---

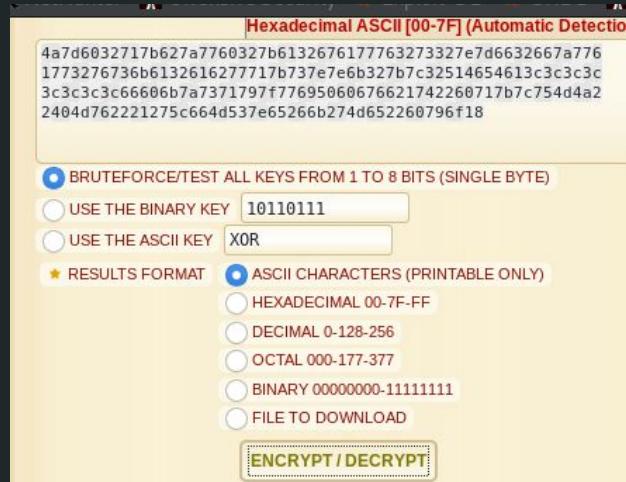
# #1 Exclusive Or Random

Hint: You know you can take any two beautiful messages(strings) and mesh them together and they'll come out complete random.

4a7d6032717b627a7760327b6132676177763273327e7d6632667a7761773276  
736b6132616277717b737e7e6b327b7c32514654613c3c3c3c3c3c3c66606b7  
a7371797f77695060676621742260717b7c754d4a22404d762221275c664d537  
e65266b274d652260796f18

# #1 Exclusive Or Random

```
1...  
  
PV]LEGOIA_fVQP0B0VGMC{|@v{@000jP  
{eH50}0{S0VOY.  
  
[00010010]: Xor cipher is used a lot these  
days specially in  
010110000110111 CTFs.....tryhackme{Brut3f0rcing_  
1... g_XOR_d035Nt_Alw4y5_w0rk}  
  
mZG0V\EPG0  
[00100111]: \F0@FPO@TYZA@A]PFP@QTLF@FEPV  
011011010101101 \TYL@  
0... \[@vasF00000000AGL]TV^XPNwG@A0$0G  
V\[Rjm0gjQ000[AjtYB0L0]B0G^H?  
000^000000^0  
^0  
000^0000  
  
Using dCode, yo
```



You have to brute force it:

[00010010]: 010110000110111...Xor cipher is used a lot these days specially in  
CTFs.....tryhackme{Brut3f0rcing\_XOR\_d035Nt\_Alw4y5\_w0rk}

# [Part 8] #1 Ancient Times

---

# #1 Ancient Times

לְרֹאשׁוֹת־מִזְרָחֶם וְבַעֲדֵי־מִזְרָחֶם  
בְּנֵי־יִשְׂרָאֵל וְבְנֵי־יִשְׂרָאֵל

# #1 Ancient Times

- Looked around until I found out what it was
- Pigpen cipher
- Found a site to decode it

## Search for a tool

★ SEARCH A TOOL ON DCODE BY KEYWORDS:

e.g. type random

GO

## Results



↑↓

↑↓

(Original)

LOOKEDLIKESOMEALIENLANGUAGETOMETR  
YHACKMEPIGANDPEN

#,\*,\*,\*

PSSOEDPIOEJSQEAPIERPARGLAGEKSQEKV  
YHACOQETIGARDTER

#,\*,\*,\*

EKKCJHERCJTKGJBERJIEBINVBNJTKGJTQ  
UPBFCGJMRNBIHMJI

La Buse

Heinrich von  
Nettelsheim

JMMKEDJIKEPMKEAJIELJALGUAGEQMKEQO  
YHACKKENIGALDNEL

#,\*,\*,\*

CFFBRQCVBRWFDRNCVRECNETYNTRXFDRXI  
LUNPBDRGVVTNEQGRE

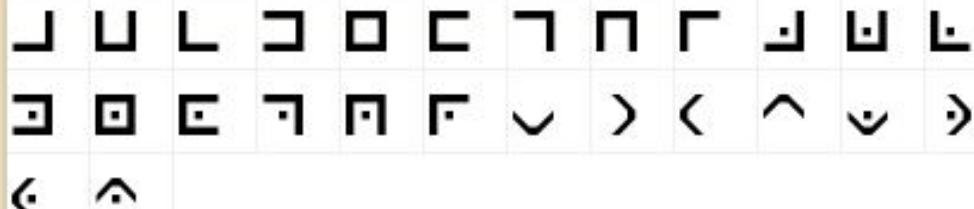
TWNSMLTQSMAWUMITQMVTVOCIOBMWBUMBZ

## PigPen Cipher

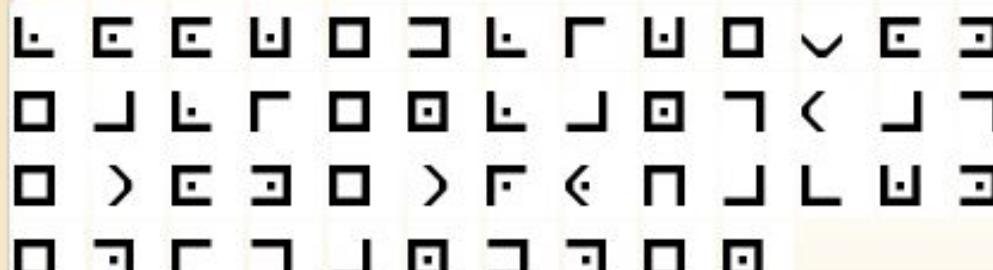
Sponsored ads

## PigPen Decoder

★ SYMBOLS OF THE PIGPEN ALPHABET (CLICK TO ADD)



★ PIG-PEN CIPHERTEXT



DECRYPT

# FORENSICS 02 : FIND THE FLAG



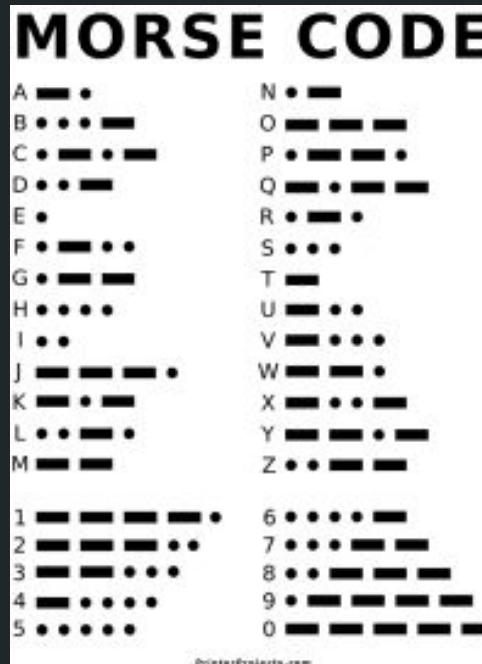
# FORENSICS 02: THE FLAG

Check out f.02.wav

```
crazyeights@kali:~/Downloads/ctf_primer_01/forensics$ exiftool -a f.02.wav
ExifTool Version Number      : 11.80
File Name                   : f.02.wav
Directory                   : .
File Size                   : 394 kB
File Modification Date/Time : 2019:10:26 13:57:46-04:00
File Access Date/Time       : 2020:02:02 19:41:43-05:00
File Inode Change Date/Time: 2020:02:02 19:41:41-05:00
File Permissions            : rw-rw-r--
File Type                   : WAV
File Type Extension         : wav
MIME Type                   : audio/x-wav
Encoding                     : Microsoft PCM
Num Channels                : 1
Sample Rate                 : 11050
Avg Bytes Per Sec          : 11050
Bits Per Sample              : 8
Duration                    : 0:00:37
crazyeights@kali:~/Downloads/ctf_primer_01/forensics$ 
```

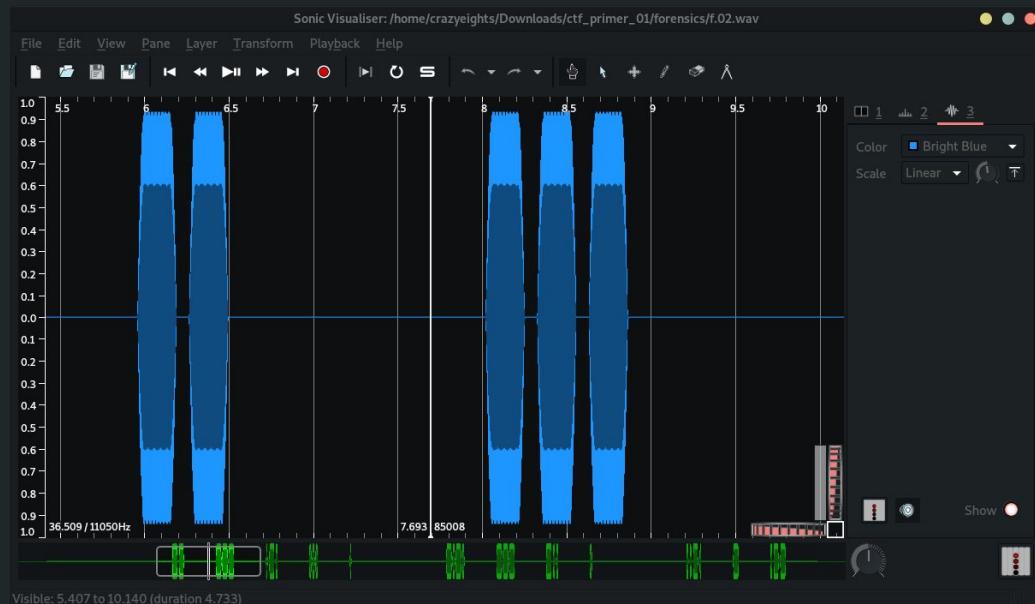
# FORENSICS 02: THE FLAG

Listening to it you hear very loud beeps. Hmmm could it be morse?



# FORENSICS 02: THE FLAG

Opening it with Sonic Visualiser, and translating from morse based on size of sound wave:



\*Untitled Document 1

MORSE CODE FTW

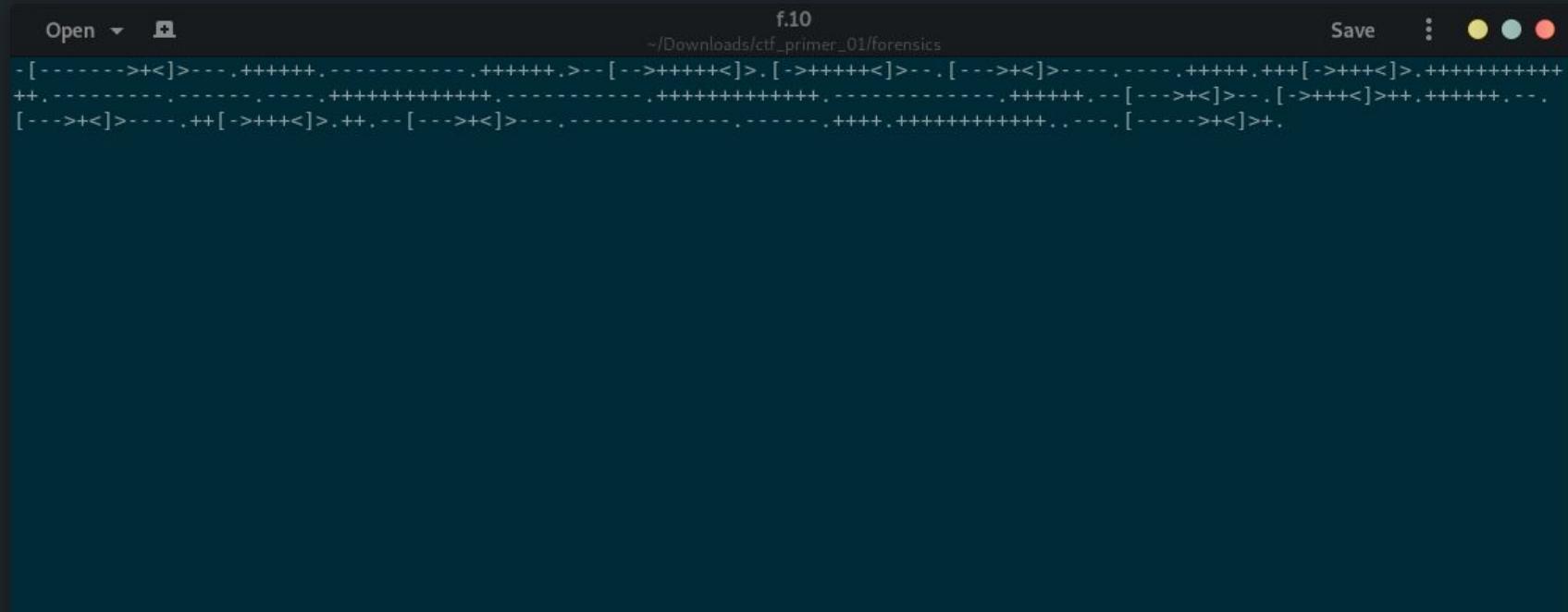
Plain Text Tab Width: 8 Ln 1, Col 15 INS

The screenshot shows a text editor window titled "\*Untitled Document 1". The text "MORSE CODE FTW" is displayed in the main area. Below the text, the editor's status bar shows "Plain Text", "Tab Width: 8", "Ln 1, Col 15", and "INS". The interface has a dark theme.

# Forensics 10: Could it be the brain??

---

# Forensics 10: Could it be the brain?



A screenshot of a terminal window titled "f.10" located at the path "/Downloads/ctf\_primer\_01/forensics". The window has a dark background and light-colored text. At the top, there are "Open" and "Save" buttons, along with a file icon. To the right of the title, there are four colored dots (yellow, cyan, magenta, black) used for font color selection. The main content of the window is a large block of binary data represented by various symbols like '-' and '+'.

```
-[ ->+<]>---.+++++.-----+----.+++++.>--[ ->+++++<]>.[->+++++<]>---.[->+<]>-----.-.----.+++++.++[ ->+++<]>.+++++-----  
++.-.----.----.----.+++++-----+----.+++++-----+----.----.+++++.--[ ->+<]>-.[->+++<]>++.+++++.--.  
[ ->+<]>---.++[ ->+++<]>.++.--[ ->+<]>---.-----.----.++++.+++++-----.,---,[ ->+<]>+.
```

# Forensics 10: Could it be the brain?

```
1. +++++.--[--->+<]>--.[->+++<]>++.++++++.-.[--->+<]>----.+[->+++<]>.++.--[--->+<]>----.----.----.++++.+++++++.----.[--->+<]>+.
```

Load example program ▾

Have fun! :-)

You can add special chars to the input field:  
Decimal: \65 (same as "A")  
Hexadecimal: \x7E (same as "~")  
Control characters: \r \n \t

For any kind of feedback, toss me a mail to  
-.[--->+<]>----.++++++.-[--->+<]>+.++  
[->+++<],+++++++.----.----.-[--->+<]>+.----.  
-<>-.----.

Links:

- [Wikipedia on Brainfuck](#)
- [The Brainfuck archive](#)
- [Brainfuck snippets](#)
- [Text generator](#)
- [BF code compressor](#)

Options

Cell size (Bits):  8  16  32

Dynamic (infinite) Memory:

Memory size: 30000

Memory overflow behaviour:

- undefined (fast)
- warn
- abort

run stop load from server link to this code view memory view generated code minify

Finished in 9 ms.

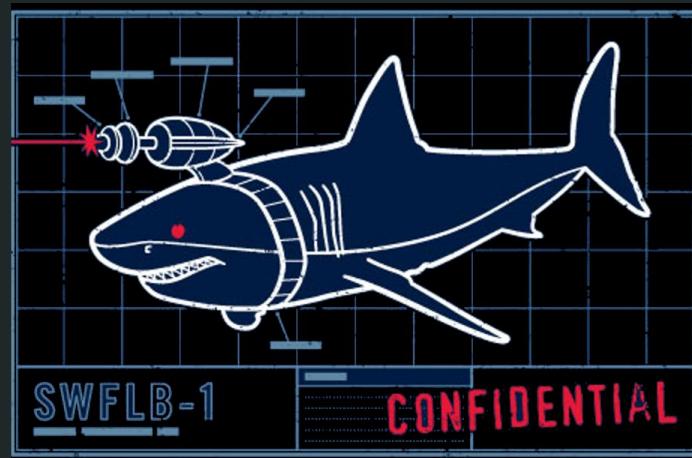
FLAG{esoteric\_lanaguages\_are\_cool}

# [Part 12]: What is in the bin?

---

## [Part 12] What is in the bin?

- Download flag binary file
- Find the hidden flag



# [Part 12] What is in the bin?

```
crazyeights@kali:~/Downloads$ cd Downloads/
crazyeights@kali:~/Downloads$ strings flag
/lib64/ld-linux-x86-64.so.2
libc.so.6
puts
printf
_cxa_finalize
__libc_start_main
GLIBC 2.2.5
_ITM_deregisterTMCCloneTable
_gmon_start_
_ITM_registerTMCCloneTable
u3UH
[]A\A]A^A_
Oh so you are here to get the flag huh?!
bG9sLi55b3UgdGhvdWdodCB0aGlzIHdvdWxkIGJlIHRoYXQgZWFeSBodWg/Cg==
Well Good luck!
dHJ5aGFja21le3M3cjFuZ3M/XzByX3I0YjFuMj99Cg==
;*3$"
GCC: (GNU) 8.2.1 20180831
GCC: (GNU) 8.2.1 20181127
init.c
crtstuff.c
deregister_tm_clones
```

The screenshot shows the CyberChef interface. In the 'Input' section, the string `dHJ5aGFja21le3M3cjFuZ3M/XzByX3I0YjFuMj99Cg==` is pasted. The 'Recipe' dropdown is set to 'From Base64'. Below it, the 'Alphabet' dropdown shows 'A-Za-zA-Z0-9+='. A checked checkbox labeled 'Remove non-alphabet chars' is visible. In the 'Output' section, the decoded string `tryhackme{s7r1ngs?_0r_r4b1n2?}` is displayed. The CyberChef header indicates it was last built 2 months ago and supports multiple inputs and a Node API.

# Analysing PCAPs Part 1

---

# PCAP1

Download n.01.pcap

- Open with wireshark
- Hunt for flag in conversation

# PCAP1

n.01.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...

No.	Time	Source	Destination	Protocol	Length	Info
19	21.974066659	10.0.2.15	192.168.0.231	TCP	76	42842 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3...
20	21.974316749	192.168.0.231	10.0.2.15	TCP	62	80 → 42842 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
21	21.974358439	10.0.2.15	192.168.0.231	TCP	56	42842 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
22	21.974465569	10.0.2.15	192.168.0.231	HTTP	478	GET /login.php?username=admin&password=FLAG{n0w_y0ur_g3tting_it} HT...
23	21.974541629	192.168.0.231	10.0.2.15	TCP	62	80 → 42842 [ACK] Seq=1 Ack=423 Win=65535 Len=0
24	21.974847969	192.168.0.231	10.0.2.15	TCP	73	80 → 42842 [PSH, ACK] Seq=1 Ack=423 Win=65535 Len=17 [TCP segment o...

► Frame 22: 478 bytes on wire (3824 bits), 478 bytes captured (3824 bits) on interface 0  
► Linux cooked capture  
► Internet Protocol Version 4, Src: 10.0.2.15, Dst: 192.168.0.231  
► Transmission Control Protocol, Src Port: 42842, Dst Port: 80, Seq: 1, Ack: 1, Len: 422  
► Hypertext Transfer Protocol

0000 00 04 00 01 00 06 08 00 27 74 17 d4 00 00 08 00 . . . . 't .  
0010 45 00 01 ce 96 0b 40 00 40 06 d5 80 0a 00 02 0f E . . @ . @ .  
0020 c0 a8 00 e7 a7 5a 00 50 2f cb 97 7d 01 0e 82 02 . . Z P / } .  
0030 50 18 fa f0 cf 5e 00 00 47 45 54 20 2f 6c 6f 67 P . ^ . GET /log  
0040 69 6e 2e 70 68 70 3f 75 73 65 72 6e 61 6d 65 3d in.php?u ssername=  
0050 61 64 6d 69 6e 26 70 61 73 73 77 6f 72 64 3d 46 admin&pa ssword=F  
0060 4c 41 47 7b 6e 30 77 5f 79 30 75 72 5f 67 33 74 LAG{n0w\_ y0ur\_g3t

Packets: 46 · Displayed: 46 (100.0%) Profile: Default

# Analysing PCAPs Part 2

---

# PCAP2

Download n.02.pcap

- Open with wireshark
- Hunt for flag in conversation

# PCAP2

Wireshark · Follow HTTP Stream (tcp.stream eq 3) · n.02.pcap

```
POST /login.php HTTP/1.1
Host: 192.168.0.231
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 46

username=admin&password=FLAG{1_am_th3_p0stm4n}<head>
<title>Error response</title>
</head>
<body>
<h1>Error response</h1>
<p>Error code 501.
<p>Message: Unsupported method ('POST').
<p>Error code explanation: 501 = Server does not support this operation.
</body>
```

# Analysing PCAPs Part 3

---

# PCAP3

Download n.03.pcap

- Open with wireshark
- Hunt for flag in conversation

# PCAP3

```
Wireshark · Follow TCP Stream (tcp.stream eq 3) · n.03.pcap
```

```
GET /help.php HTTP/1.1
Host: 192.168.0.231
User-Agent: Mozilla/5.0 (FLAG{s3cr3t_ag3nt}) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache

HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/2.7.17rc1
Date: Sat, 26 Oct 2019 11:19:28 GMT
Content-type: application/octet-stream
Content-Length: 14
Last-Modified: Sat, 26 Oct 2019 11:19:22 GMT

Help yourself
```

# Analysing PCAPs Part 4

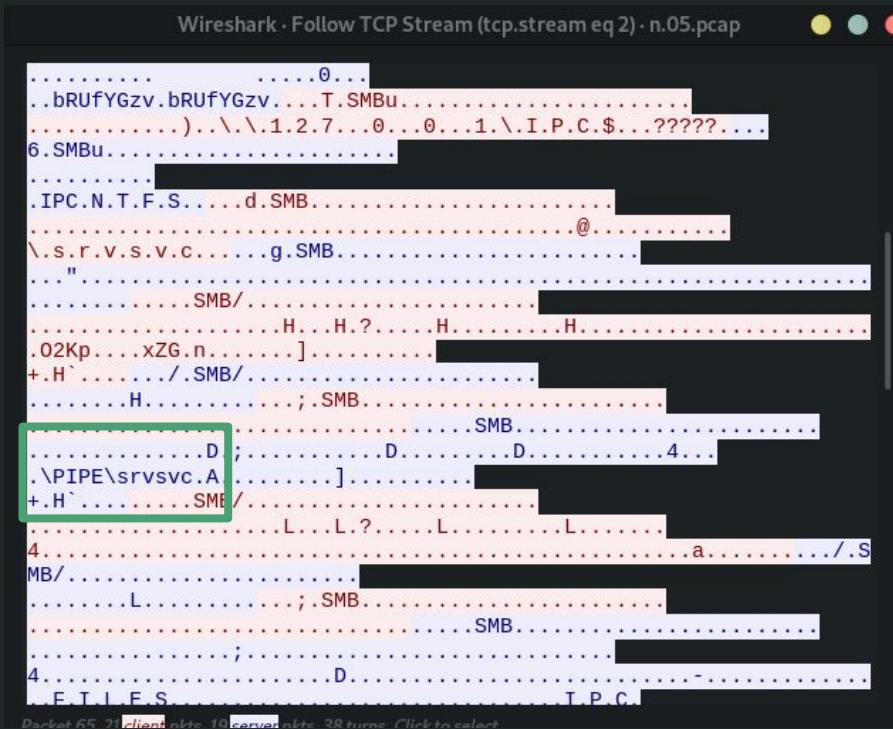
---

# PCAP4

Download n.05.pcap

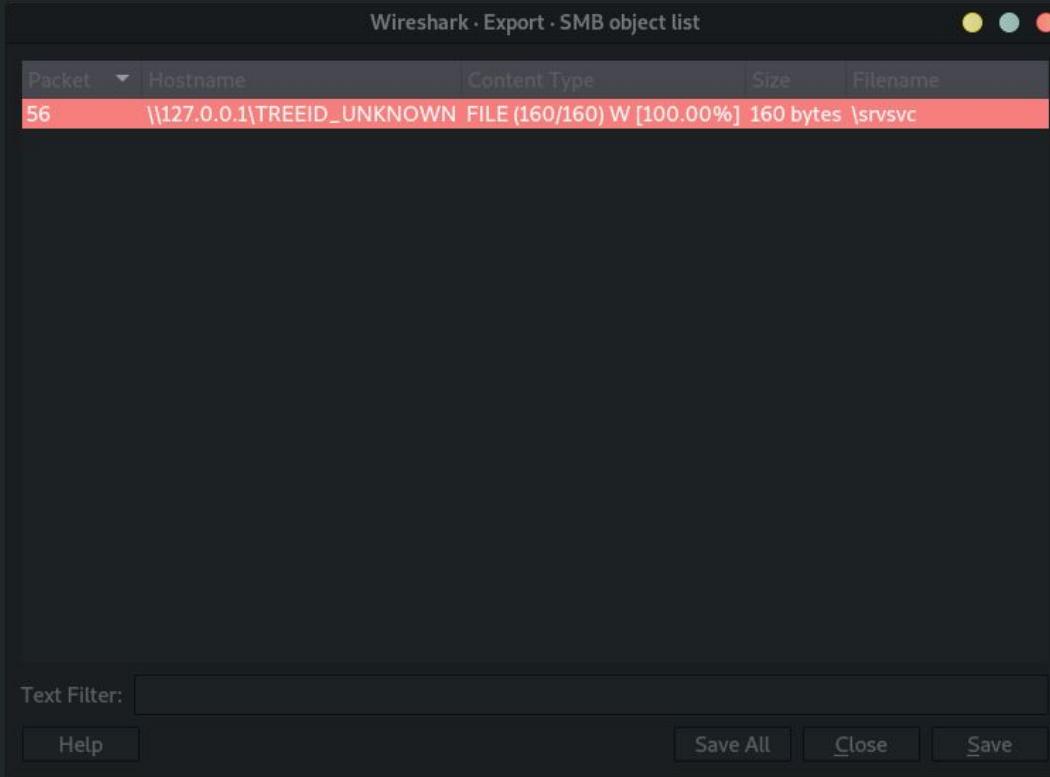
- Open with wireshark
- Find and export the file from the capture

# PCAP4



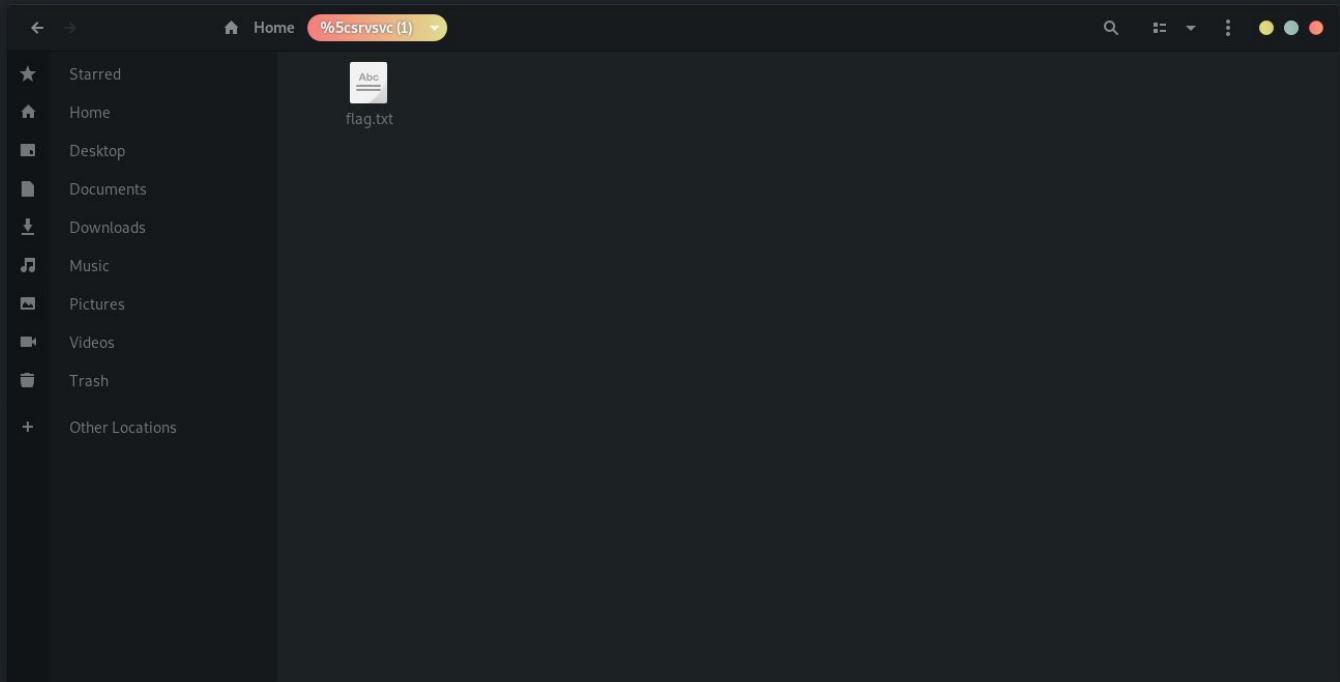
# PCAP4

Go to File > Export  
Objects > SMB



# PCPAP4

Decompress the archive to get the flag



BONUS 1: [Part 7] #4 What the hell is  
this?

---

## #4 What the hell is this?

Fmeorcbi gc rmd gyowyb sp sw gd. Afy gybiq gi hewr geld xfo jjkk  
rbcfkgiwi{TsKcxipo\_gGzLcB\_mQ\_MeCcep\_mmNrIp}

P.S: Don't forget to use your brain ;)

## #4 What the hell is this?

Keeping only the flag part:

Three different amounts that were shifted by:

tdehmikyk{VuMezkrq\_ilbNeD\_oS\_OgEegr\_ooPtKr}

hrsvawymy{JiAsnyfe\_wWpBsR\_cG\_CuSsuf\_ccDhYf}

nxybgcese{PoGytelk\_cCvHyX\_iM\_laYyal\_iiJnEl}

--> You can see that together they make tryhackme

## #4 What the hell is this?

--> Replace the characters that are in the wrong position with # to make clearer

t##h##k##{V##e##r#\_#l##e#r\_#s\_##E#r\_##P##r}

#r##a##m##i##n##e##p##R##\_##C##s##\_c##h##}

##y##c##e##G##e##\_##c##H##\_i##\_a##a##\_##i##E##}

the flag is:

tryhackme{ViGenere\_cIpHeR\_iS\_CaEsar\_ciPhEr}

--> I figured this out with pencil and paper I am sure there is faster way

# BONUS 3: [Part 9]: #1 Genetics

---

# #1 Genetics

I heard scientist found ways to hide data in DNA and stuff. Is it really true?

```
CTCAAAATAATCTTGATTACAATGATTAGTACATTGAAACACACATTGCCACAG  
AGAACATCACGTTGAAAATCCGACATACTAGAACATCACGTTGCATATGTTGATAAAAA  
AGGACATTGCATAACTACAAGACACTTGATAAACACAGCAGAAAACGACATTGCA  
GACAAAGCCACACACATTGGTAAACAAGTAGTTGCCGACTATGTTGAAGAA  
ACACACACAGTTGGAGTTGAGCCCACAGCATTGATCACAAACAAATTGATAC  
GATTGAATAAAATAATCTTGACCAGTAAAACGTTGGAAACAGCTTCGCATTCAA  
AGTGAGACCCAGAC
```

# #1 Genetics

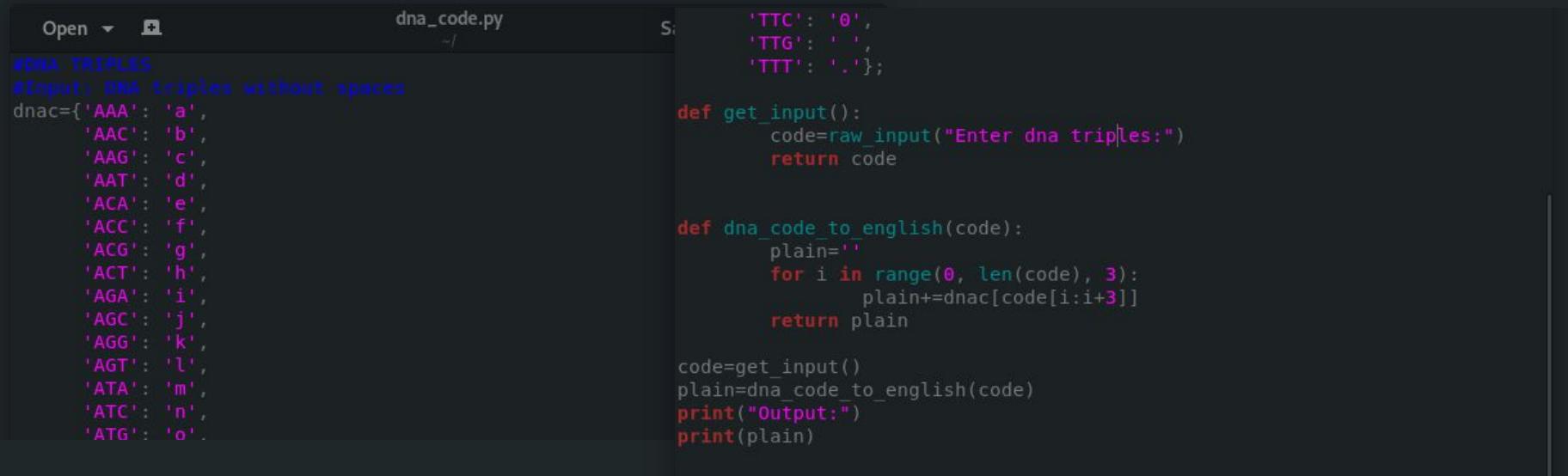
(This actually took me a long time to figure out)

First I looked up dna code to english and scrolled until I found this promising table:

DNA CODE								
Codon	English	Codon	English	Codon	English	Codon	English	Codon
AAA	a	CAA	q	GAA	G	TAA	w	
AAC	b	CAC	r	GAC	H	TAC	X	
AAG	c	CAG	s	GAG	I	TAG	Y	
AAT	d	CAT	t	GAT	J	TAT	Z	
ACA	e	CCA	u	GCA	K	TCA	l	
ACC	f	CCC	v	GCC	L	TCC	2	
ACG	g	CCG	w	GCG	M	TCG	3	
ACT	h	CCT	x	GCT	N	TCT	4	
AGA	i	CGA	y	GGA	O	TGA	5	
AGC	j	CGC	z	GGC	P	TGC	6	
AGG	k	CGG	A	GGG	Q	TGG	7	
AGT	l	CGT	B	GGT	R	TGT	8	
ATA	m	CTA	C	GTA	S	TTA	9	
ATC	n	CTC	D	GTC	T	TTC	0	
ATG	o	CTG	E	GTG	U	TTG	space	
ATT	p	CTT	F	GTT	V	TTT	. (period)	

# #1 Genetics

I started manually translating it, but then gave up and wrote a program to do it for me.



The image shows a code editor window with a dark theme. On the left, there is a sidebar with file navigation icons. The main area displays a Python script named `dna_code.py`. The code defines a dictionary `dnac` mapping 20 DNA triplets to letters (a-s). It includes a function `get_input` to read DNA triples from the user, and a function `dna_code_to_english` to translate them into plain text. Finally, it calls `get_input`, `dna_code_to_english`, and prints the result.

```
Open ▾ dna_code.py
#DNA TRIPLES
#Input: DNA triples without spaces
dnac={'AAA': 'a',
      'AAC': 'b',
      'AAG': 'c',
      'AAT': 'd',
      'ACA': 'e',
      'ACC': 'f',
      'ACG': 'g',
      'ACT': 'h',
      'AGA': 'i',
      'AGC': 'j',
      'AGG': 'k',
      'AGT': 'l',
      'ATA': 'm',
      'ATC': 'n',
      'ATG': 'o'}
      'TTC': 'p',
      'TTG': 'q',
      'TTT': 'r'};
def get_input():
    code=raw_input("Enter dna triples:")
    return code

def dna_code_to_english(code):
    plain=""
    for i in range(0, len(code), 3):
        plain+=dnac[code[i:i+3]]
    return plain

code=get_input()
plain=dna_code_to_english(code)
print("Output:")
print(plain)
```

# #1 Genetics

Program Output:

```
root@kali:~# python dna_code.py
Enter dna triples:CTCAAAATAATCTTGATTACAATGATTAGTACATTGAAACACACATTGCCACAGAGAATCAC
GTTGAAAATCCGACATACTAGAACATCGTTGCATATGTTGATAAAAAGGACATTGCATACTACAAGACACTTGATAACAC
AGCAGAAAACGACATTGCAGACAAAGCCACACACATTGGATAACAGTAGTTGCCACTATGTTGAAGAACACACACA
CAGTTGGAGTTGAGCCCACAGCATTGATCACAAACAAATTGATACGATTGAATAAAATACTTGACCAGTAAACGTT
GGAAACAGCTTCGCATTCAAAGTGAGACCCAGAC
Output:
Damn people are using anything to make their message secure. Well who cares I ju
st need my damn flag GeN3t1c5HuH
root@kali:~#
```

Flag: tryhackme{GeN3t1c5HuH}

# BONUS 4: [Part 11] #1 Morse Code

---

# #1 Morse Code

Download the file: `morse.txt`

Hint: Morse code is being used for a very long time. And since then there has been a lot of versions like using your eyebrows, flashing torches, tapping etc.

# #1 Morse Code

Found a table that had the conversion:

The screenshot shows a web application for Morse code conversion. At the top, there's a navigation bar with tabs for "International" (selected), "American", and "More". The URL "SCPhillips.com" is also visible. Below the table, there's a note: "the highlighted letters or symbols the morse sound will be played".

Letter	Morse
A	di-dah
B	dah-di-di-dit
C	dah-di-dah-dit
D	dah-di-dit
E	dit
F	di-di-dah-dit
G	dah-dah-dit
H	di-di-di-dit
I	di-dit

Letter	Morse
N	dah-dit
O	dah-dah-dah
P	di-dah-dah-dit
Q	dah-dah-di-dah
R	di-dah-dit
S	di-di-dit
T	dah
U	di-di-dah
V	di-di-di-dah

Digit	Morse
0	dah-dah-dah-dah-dah
1	di-dah-dah-dah-dah
2	di-di-dah-dah-dah
3	di-di-di-dah-dah
4	di-di-di-di-dah
5	di-di-di-di-dit
6	dah-di-di-di-dit
7	dah-dah-di-di-dit
8	dah-dah-dah-di-dit

**Sound Controls**

Pitch ⓘ: 550

Speed ⓘ: 20

Farnsworth speed ⓘ: 20

# #1 Morse Code

Wrote a program to  
perform the decoding:

```
root@kali:~# python morse_to_en.py
Enter Morse code:dah dah-dah-dah di-dah-di-dit dah-di-dit dah-di-dah-dah dah-dah
-dah di-di-dah dah di-di-di-dit dit dah-di-dah-dah di-dah di-dah-dit dit di-di-d
ah di-di-dit di-dit dah-dit dah-dah-dit di-dah dah-dit dah-di-dah-dah dah di-di-
di-dit di-dit dah-dit dah-dah-dit dah dah-dah-dah dit dah-dit dah-di-dah-dit di-
dah-dit dah-di-dah-dah di-dah-dah-dit dah dah di-di-di-dit dit di-di-dit dit dah
-di-dit di-dah dah-di-dah-dah di-di-dit di-dah-di-dah-di-dah di-di-dah-dit di-da
h-di-dit di-dah dah-dah-dit di-dit di-di-dit di-dit dah-dit dah di-di-di-dah-dah
di-dah-dit dah-dit di-di-di-dah dah di-dit dah-dah-dah-dah dah-dit di-di-
-di-dah di-dah-di-dit dah-dah dah-dah-dah-dah di-dah-dit di-di-dit di-di-
di-dah-dah dah-di-dah-dit dah-dah-dah-dah dah di-dit di-di-di-dah-dah
Output:
TOLDYOU THEY ARE USING ANYTHING TO ENCRYPT THESE DAYS. FLAG IS INT3RN4TION4L M0RS3C0D3
root@kali:~#
```

```
'dah-di-dah-di-dah-dah': ' ',  
'di-dah-di-dah-di-dah': ',',  
'dah-di-di-di-di-dah': '-+',  
'di-dah-di-dah-dit': '+',  
'di-dah-di-di-dah-dit': ")",  
'di-di-dah-dah-di-dit': "?",  
'dah-di-di-dah-dit': '\\\\'  
};  
  
def get_input():  
    code=raw_input("Enter Morse code:")  
    return code  
  
  
def morse_code_to_english(code):  
    plain=""  
    code_arr=code.split(' ')  
    for i in range(0, len(code_arr)):  
        plain+=morse_dict[code_arr[i]]  
    return plain  
  
code=get_input()  
plain=morse_code_to_english(code)  
print("Output:")  
print(plain)
```

End.

There are about 5 more. But they are about reverse engineering, and are unrelated to the topic.

There are a ton of challenges similar to these on this site, and hack the box.

I used to do these challenges in first year:

<https://cryptopals.com/sets/1/challenges/>

<https://www.mysterytwisterc3.org/en/challenges/>