TryHackMe Room Write-up

Title: c4ptur3-th3-fl4g

Description: A beginner level CTF challenge

Author: Igor Buszta

Translation & Shifting

Description: *Translate, shift and decode the following;*

Answers are all case sensitive.

Tools: Cyberchef, online decoders, macOS

Objectives:

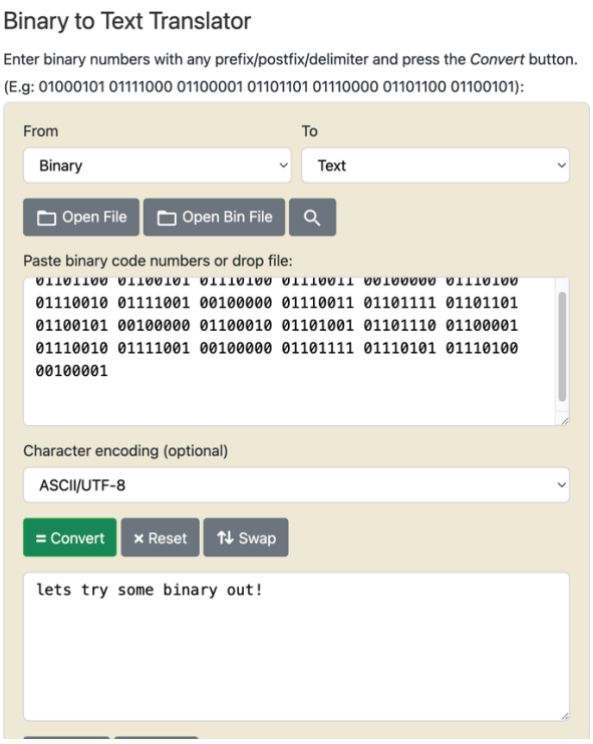
c4n y0u c4p7u23 7h3 f149?

This one is straight forward, we need to substitute the digits to get this text in all-characters:

can you caputre the flag?

01101100 01100101 01110100 01110011 00100000 01110100 01110010 01111001 00100000 01110011 01101111 01101101 01100101 00100000 01100010 01101001 01101110 01100001 01110010 01111001 00100000 01101111 01110101 01110100 00100001

This is straight binary, let's quickly decode it online.



Picture 1: Decoding binary online.

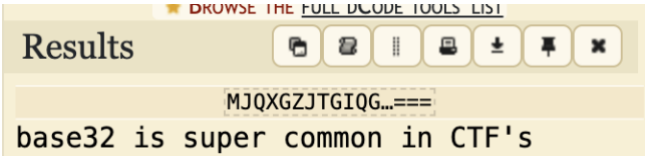
MJQXGZJTGIQGS4ZAON2XAZLSEBRW63LNN5XCA2LOEBBVIRRHOM======

Here we have to have basic knowledge of character representations. Most popular one, ending with one or two '=' is base64.

Base64 divides the message into groups 3 bytes each. Later these groups are divided to 4 groups 6 bits each (it adds up to 64 bits in one group). Each character is represented by a number from 0 to 64. If there are not enough characters to fill a group, base64 uses padding (=) to fil lit up – there can be no more than 2 equal signs.

Because there are so many paddings, be should suspect another base variant. Base32 uses set of 32 character to represent data and can have up to 6 '=' as padding.

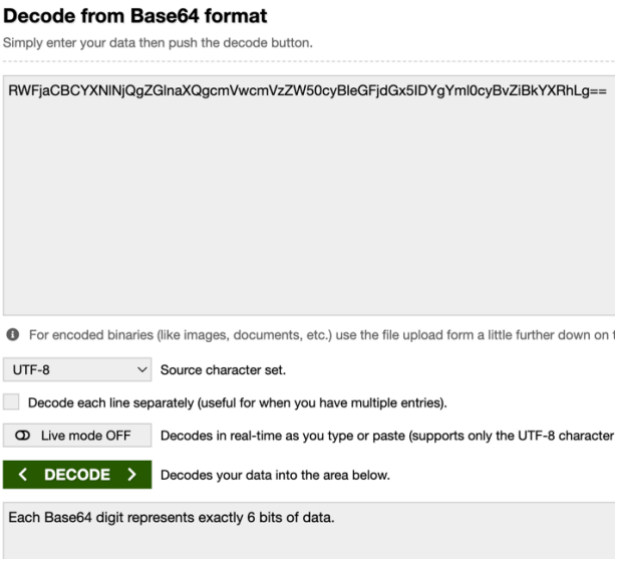
Let's find proper online decoder and get our flag.



Picture 2: Online base32 decoding.

RWFjaCBCYXNlNjQgZGlnaXQgcmVwcmVzZW50cyBleGFjdGx5IDYgYml0cyBvZiBkYXRhLg==

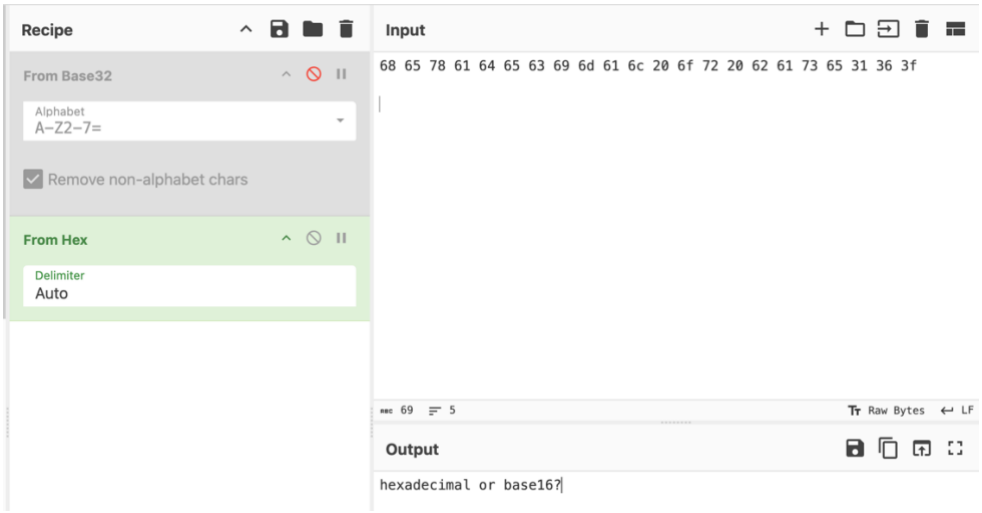
Knowing what we know about base, we see (by the number of characters and padding) that this is base64.



Picture 3: Online base64 decoding.

68 65 78 61 64 65 63 69 6d 61 6c 20 6f 72 20 62 61 73 65 31 36 3f

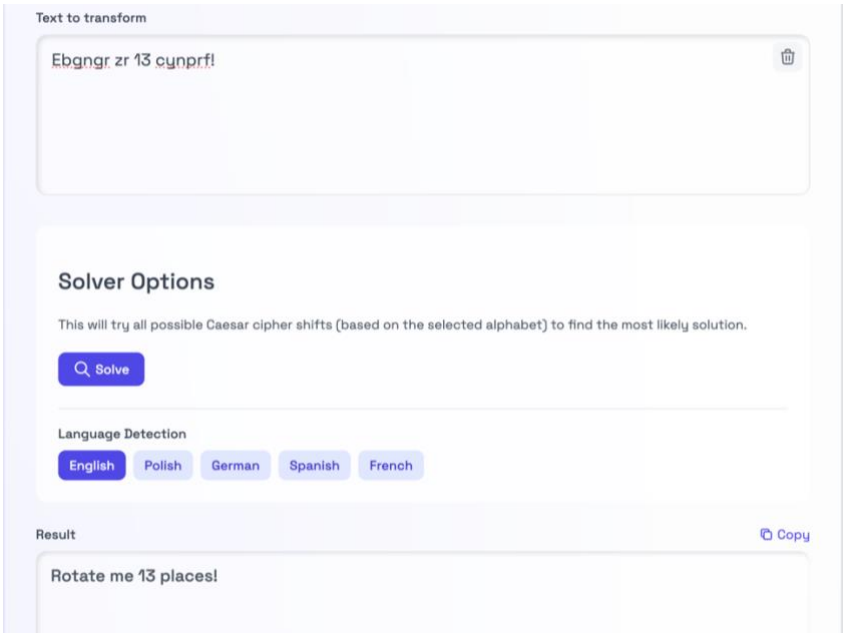
If we know a little bit about number systems, we can recognize that this one is hexadecimal (represented by 16 characters – also called base16) it's using characters from A-F and digits 0-9. It's usually represented in pairs, hence the recognition.



Picture 4: Decoding via Cyberchef.

Ebgngr zr 13 cynprf!

This looks like they are meant to be words, but characters are switched or substituted. We can tell the number 13 which might suggest the **transposition cypher** something like Cesear's Cypher based on alphabet. Let's try.

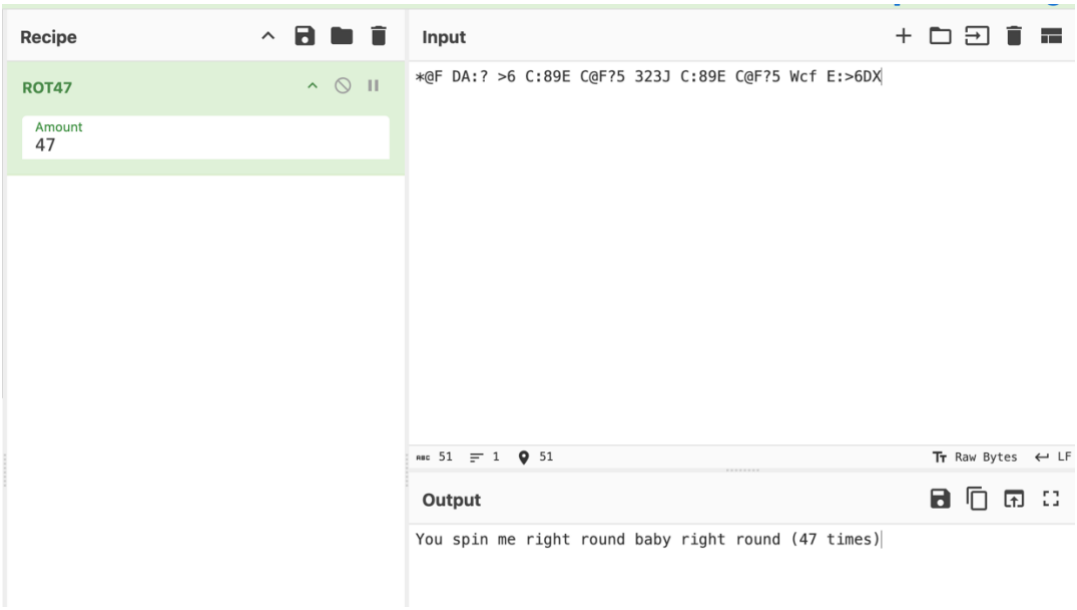


Picture 5: Transposition decyphering online.

***@F DA:? >6 C:89E C@F?5 323J C:89E C@F?5 Wcf E:>6DX**

What is that supposed to be? We can see that the spaces seem to seperate different words in a sentence but characters are once again mixed up. We still recognize the ASCII characters – capital and small with cyphers and signs. This should be another transposition cypher but not on the alphabet but on ASCII table.

After research be find ROT cyphers fitting our description. Let's try the most popular one – ROT 47.

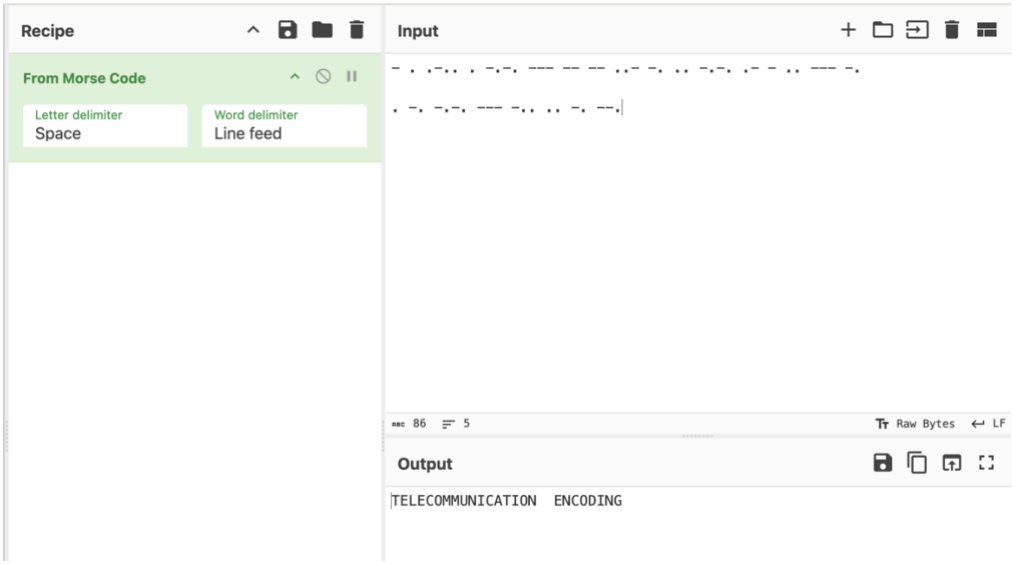


Picture 6: Decoding ROT47 via Cyberchef.

- . .-.. . -.-. --- -- -- ..- -. .. -.-. .- - .. --- -.

. -. -.-. --- -.. .. -. --.

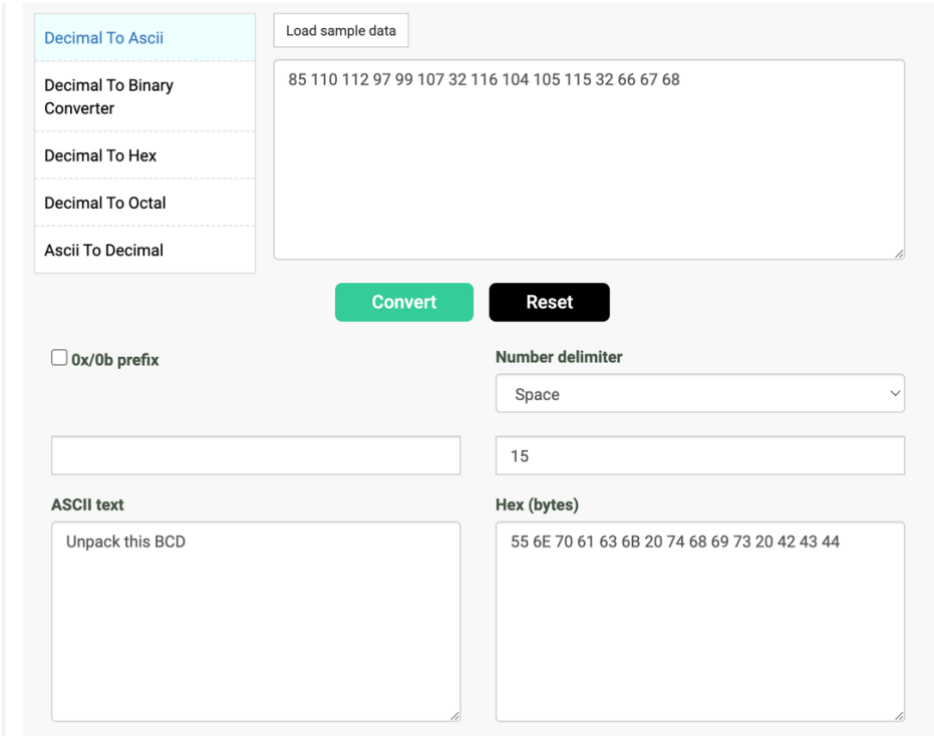
This one is clearly a morse code (alphabet represented by dots and minuses or sound/light)



Picture 7: Decoding morse via Cyberchef.

85 110 112 97 99 107 32 116 104 105 115 32 66 67 68

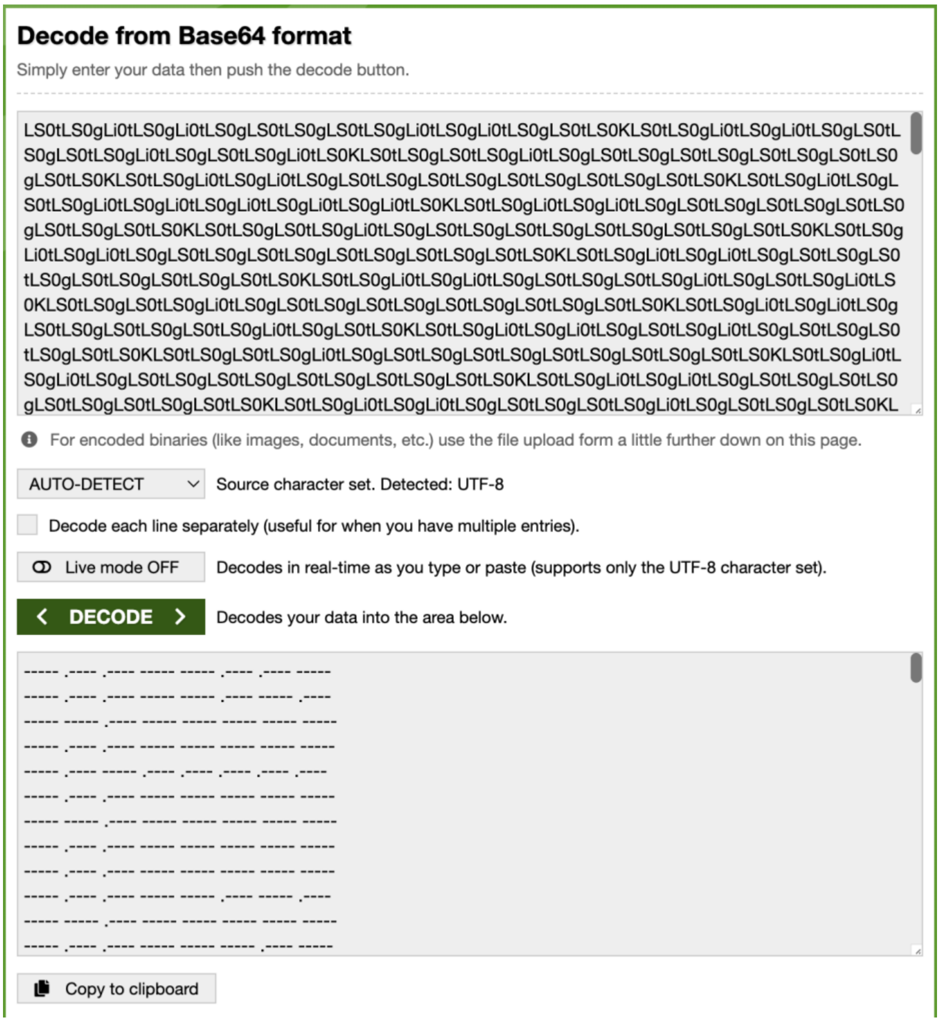
And what's that? First thought – another HEX? Nah, there are 3 digits in one segment. Second thought, ASCII codes represented in decimal? Should be right.



Picture 8: Converting DEC to ASCII online.

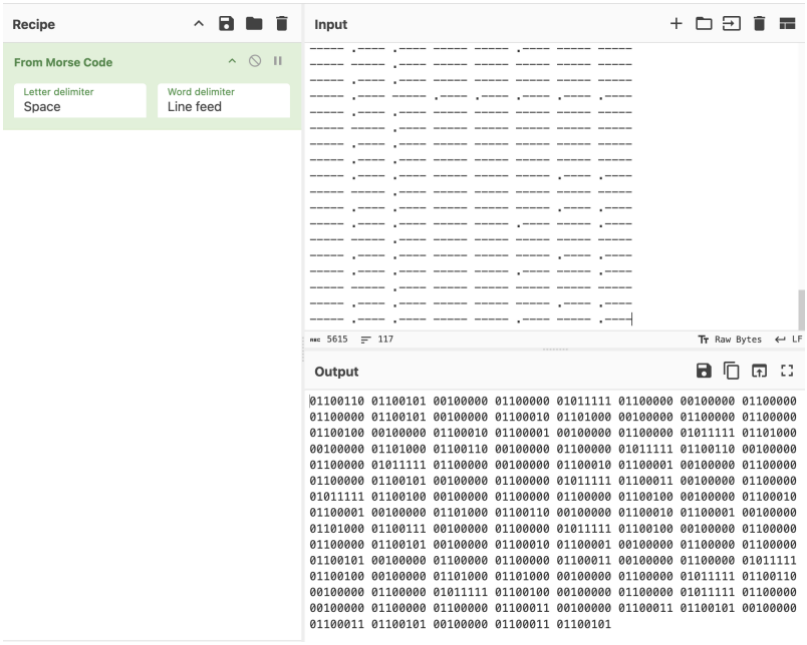
[omitted]S0tLS0KLS0tLS0gLi0tLS0gLi0tLS0gLS0tLS0gLS0tLS0gLS0tLS0gLi0tLS0gLi0tLS0KLS0tLS0gLi0tLS0gLi0tLS0gLS0tLS0gLS0tLS0gLi0tLS0gLS0tLS0gLi0tLS0KLS0tLS0gLS0tLS0gLi0tLS0gLS0tLS0gLS0tLS0gLS0tLS0gLS0tLS0gLS0tLS0KLS0tLS0gLi0tLS0gLi0tLS0gLS0tLS0gLS0tLS0gLS0tLS0gLi0tLS0gLi0tLS0KLS0tLS0gLi0tLS0gLi0tLS0gLS0tLS0gLS0tLS0gLi0tLS0gLS0tLS0gLi0tLS0=

This one doesn't even fit on the screen, but once I saw the qual sign (padding for base64) I knew what to do.



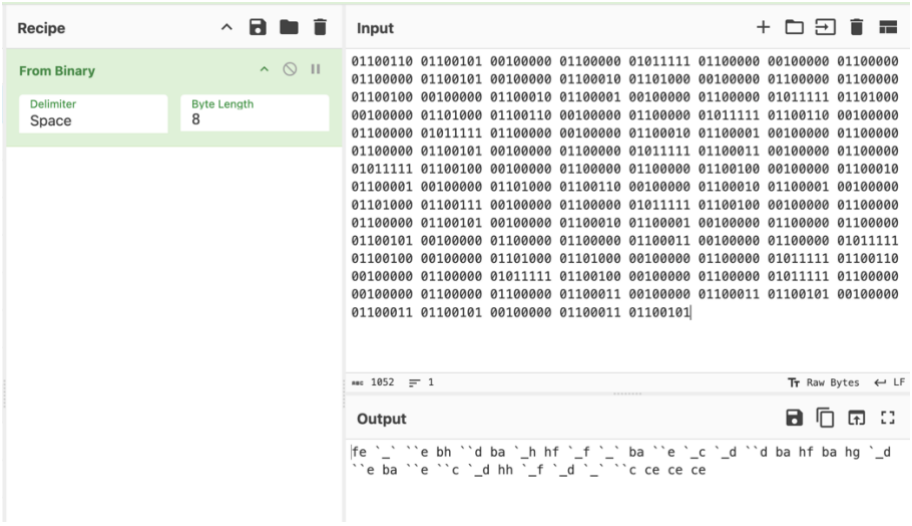
Picture 9: Decoding base64 online.

Would you look at that. It's a morse code.



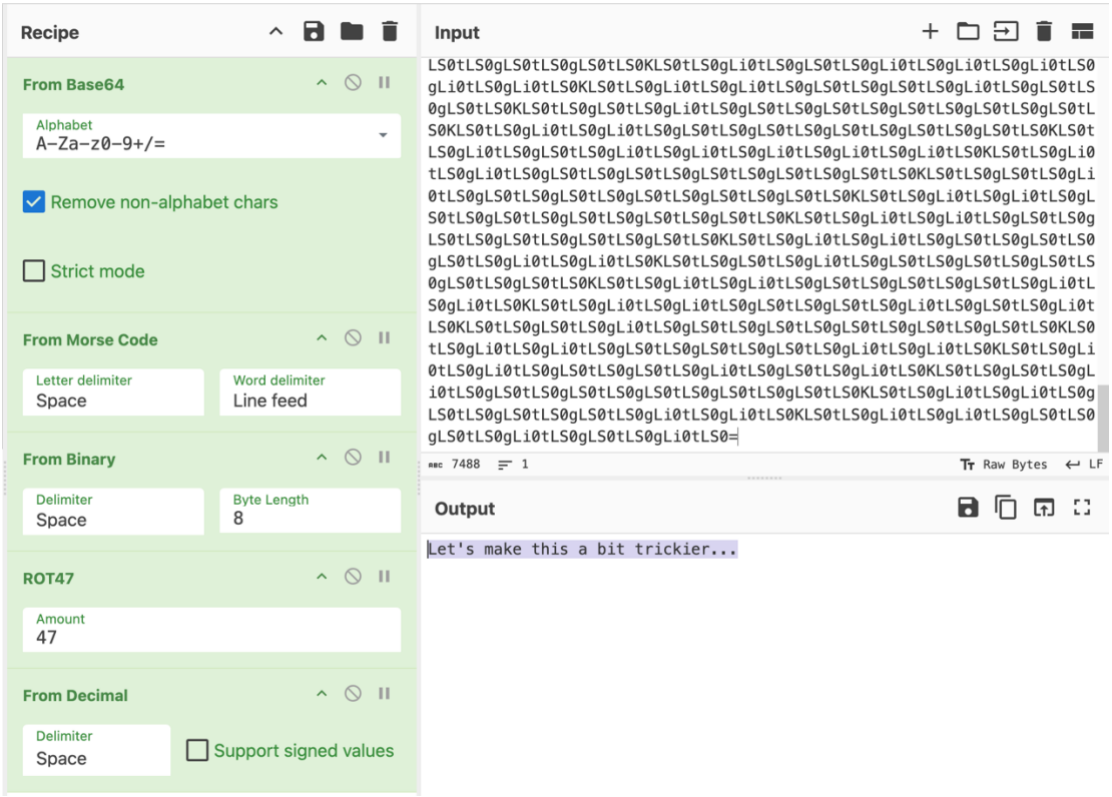
Picture 10: Decoding morse code via Cyberchef.

It’s binary. Looks like it’s the combination of the previous flags. Let’s keep decoding.



Picture 11: Decoding binary via Cyberchef.

This is complete gibberish. It’s hard to even recognize any of it, but seeing the scheme, this might be ROT47. Let’s hope I’m right.



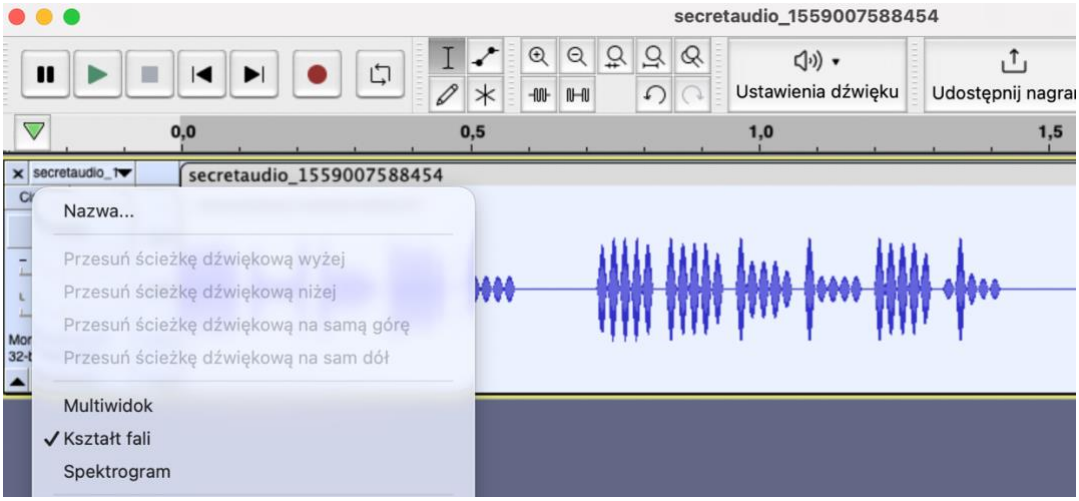
Picture 12: Complete recipe in Cyberchef.

Spectograms

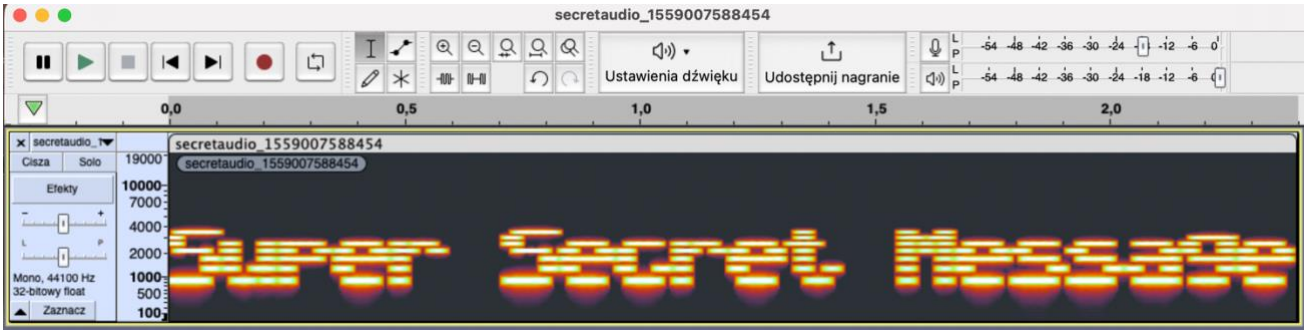
Objective: Download the file.

Tools: Audacity, macOS

A spectrogram is a visual representation of the spectrum of frequencies of a signal. In CTF analogy, the frequencies are used to form a message visible in spectogram view. To access that in Audacity (popular sound editing software), import the track, click black arrow next to the name of the track and select Spectogram.



Picture 13: Audacity – selecting spectogram view



Picture 14: Spectogram view in Audacity.

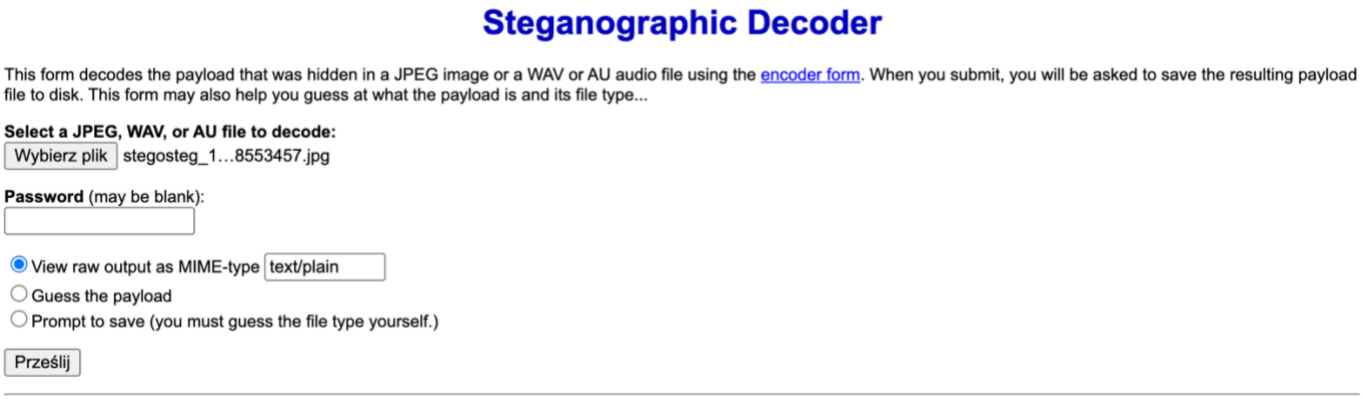
That's our flag.

Steganography

Objective: Decode the image to reveal the answer.

Tools: Online steganography decoder, macOS

Steganography is actually a field of study of hiding information in plain and common things like images, sounds or text files. The main purpose of it is to hide the fact that there i sany communication occuring in the first place. Luckly there are online tools to help find those messages.



Picture 15: Online steganographic decoder.

Picture 16: Hidden message inside the image.

Security through obscurity

Description: *Security through obscurity is the reliance in security engineering on the secrecy of the design or implementation as the main method of providing security for a system or component of a system.*

Objective: Download and get 'inside' the file. What is the first filename & extension?

Get inside the archive and inspect the file carefully. Find the hidden text.

Tools: binwalk, strings, macOS

What does the author mean, by getting ‘inside’ the file?

It usually means that there are other files embedded to the one we downloaded. What tool helps us identify and extract those files? Say hello to binwalk, our CTF friend.

```
> binwalk meme_1559010886025.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
74407         0x122A7         RAR archive data, version 5.x
74478         0x122EE         PNG image, 147 x 37, 8-bit/color RGBA, non-interla
ced
74629         0x12385         Zlib compressed data, default compression
```

Picture 17: binwalk output on meme file.

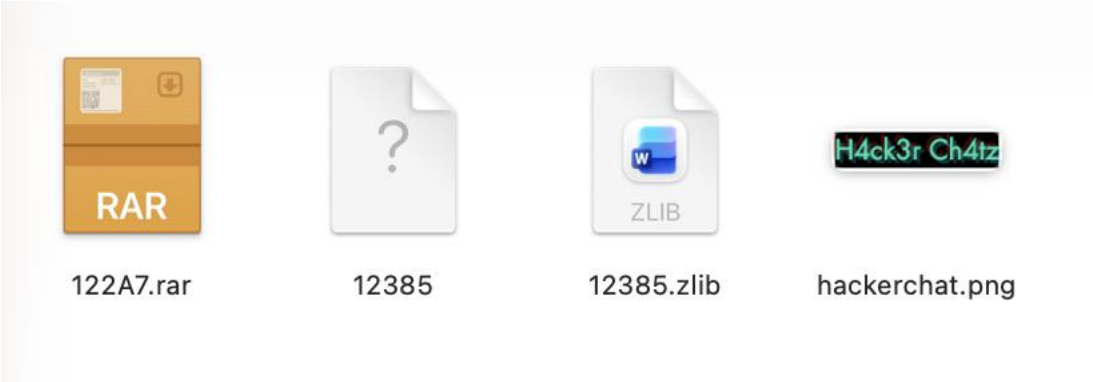
We can see at least 3 different files embedded in our picture. First two are the information of encoding and extension, so we omit that.

Let's extract them using '-e' flag and move to the created directory (directory is named [filename].extracted).

```
> ls
122A7.rar       12385           12385.zlib
```

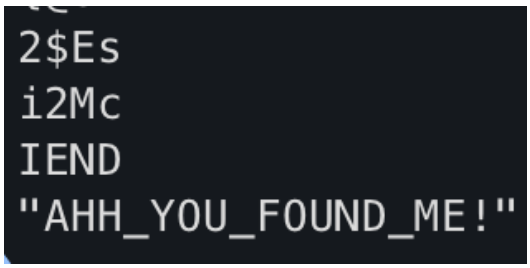
Picture 18: Extracted files from meme file.

This .rar archive seems promising. Let's open and see what's inside



Picture 19: Wanted image extracted from .rar.

Our last task is to inspect the file carefully to find the flag. Let's use our know tool, strings, to see if there are any flags hidden in plain text in this image.



Picture 20: Selected strings output with wanted flag.