

# CTF HACKON 2021

## STEGO – THEY ARE COMING FOR US



🌐 <https://www.hackon.es/>  
🐦 @HackOnURJC



### ENUNCIADO

My friend J.P. García has sent to me this image. He says that it has a hidden message, but the file seems to be corrupt and I can not recover it. Could you help me to get the message?



### FLAG

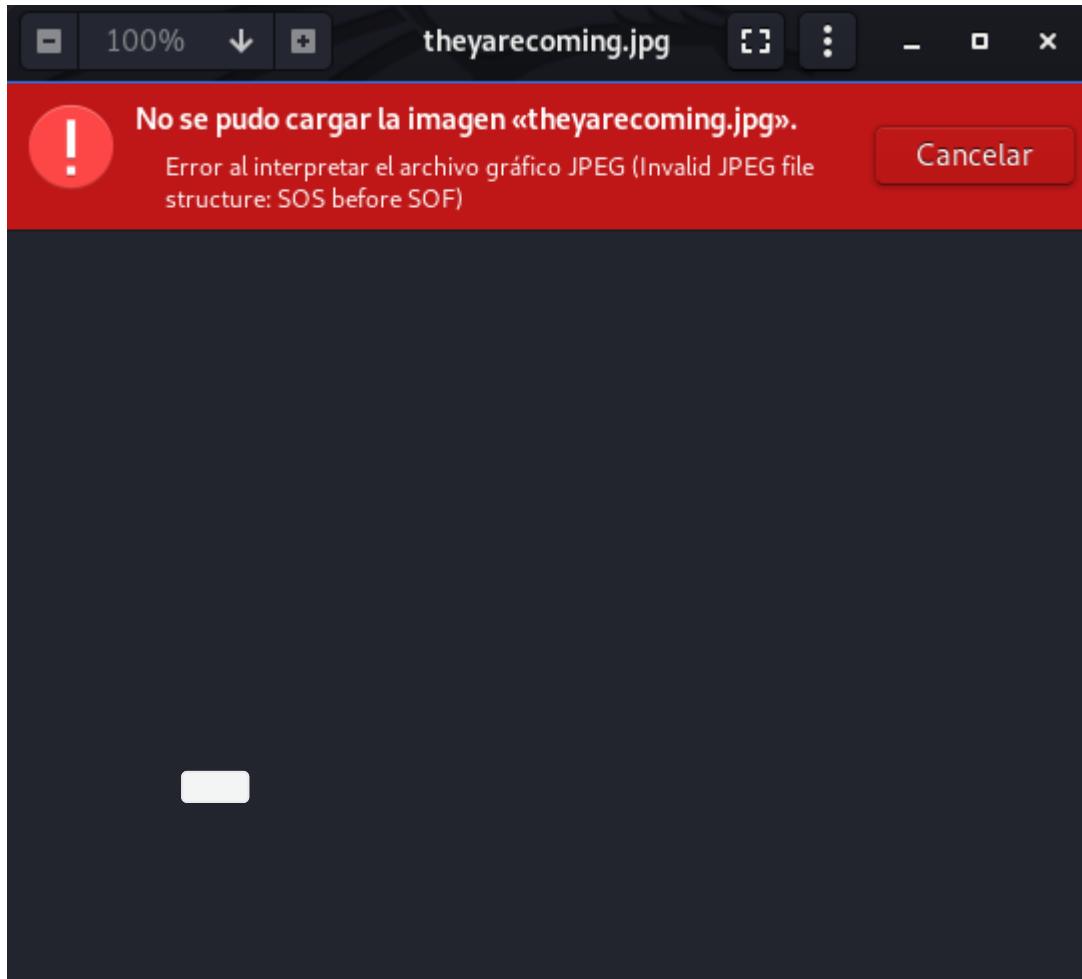
HackOn{5p3ct0gr4m\_4\_Th3\_W1n}



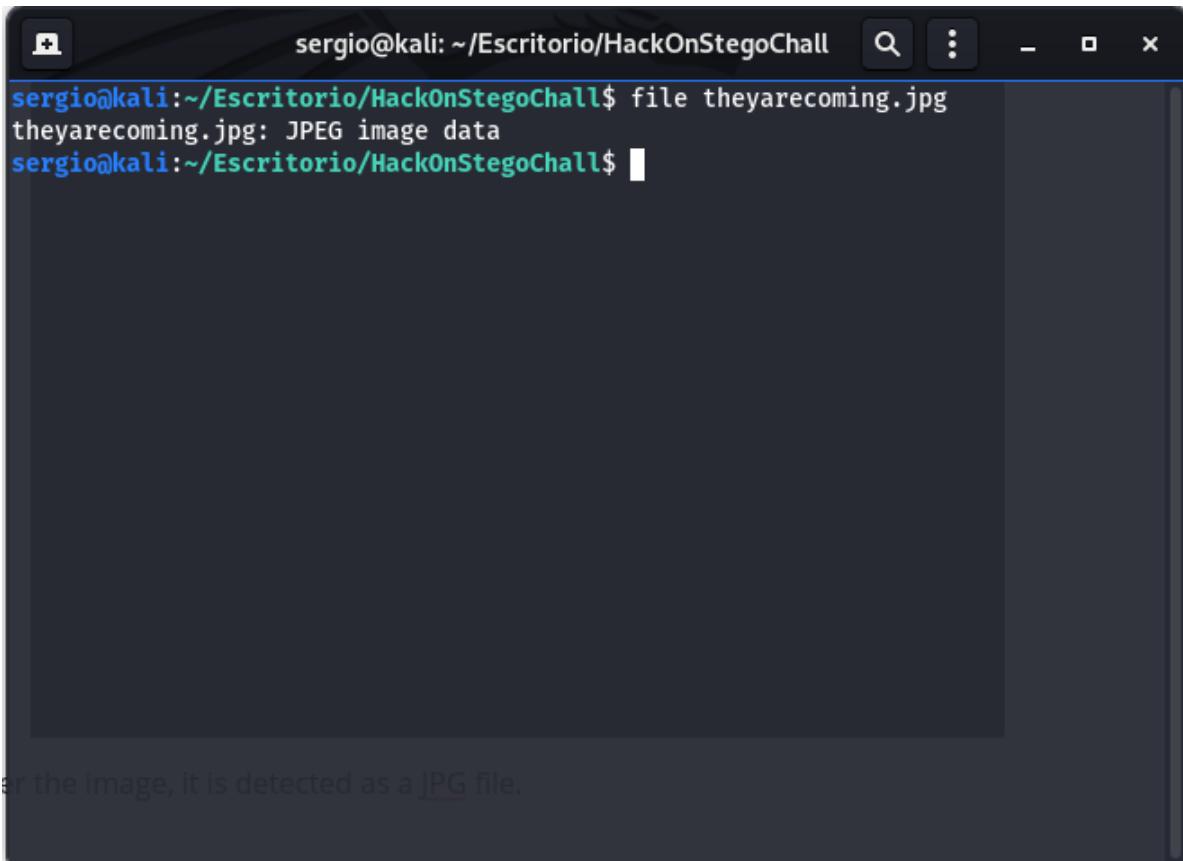
### SOLUCIÓN

#### First steps

As we can see, if we download the image and try to open it, we have an error message that indicate us that the file is corrupted:



And executing the `file` command over the image, it is detected as a JPG file.



A terminal window titled "sergio@kali: ~/Escritorio/HackOnStegoChall". The command "file theyarecoming.jpg" is run, resulting in the output "theyarecoming.jpg: JPEG image data". Below the terminal is a blurred screenshot of a web browser showing a corrupted image.

```
sergio@kali:~/Escritorio/HackOnStegoChall$ file theyarecoming.jpg
theyarecoming.jpg: JPEG image data
sergio@kali:~/Escritorio/HackOnStegoChall$
```

So, it seems clear that we have a corrupted file that we should fix to recover the image.

## Let's go to the hexeditor

If we take a look into the file header with a hex editor:

```
sergio@kali:~/Escritorio/HackOnStegoChall$ hexeditor theyarecoming.jpg
```

And we remember the file signature of a JPG file ([https://en.wikipedia.org/wiki/List\\_of\\_file\\_signatures](https://en.wikipedia.org/wiki/List_of_file_signatures)) we clearly see that there is a problem with the file header:

		ASCII Offset: 0x00000000 / 0x012BB128 (%00)																
00000000	FF D8 FF E0 FF FF FF FF	49	46	00	01	01	01	00	48	.	.	.	.	IF	.	H		
00000010	00 48 00 00 FF E1 1E 46	45	78	69	66	00	00	49	49	.	H	.	F	Exif	.	II		
00000020	2A 00 08 00 00 00 06 00	12	01	03	00	01	00	00	00	*	.	.	.	.	.	.		
00000030	01 00 00 00 1A 01 05 00	01	00	00	00	56	00	00	00	.	.	.	.	V	.			
00000040	1B 01 05 00 01 00 00 00	5E	00	00	00	28	01	03	00	.	.	.	.	^	.	(	)	
00000050	01 00 00 00 02 00 00 00	31	01	02	00	0D	00	00	00	.	.	.	.	1	.			
00000060	66 00 00 00 32 01 02 00	14	00	00	00	74	00	00	00	f	.	2	.	t	.			
00000070	88 00 00 00 48 00 00 00	01	00	00	00	48	00	00	00	.	H	.	.	H	.			
00000080	01 00 00 00 47 49 4D 50	20	32	2E	31	30	2E	32	32	.	.	.	.	GIMP	2.10.22			
00000090	00 00 32 30 32 31 3A 30	31	3A	31	31	20	31	38	3A	..	2021:01:11	18:	.	.	.			
000000A0	34 36 3A 35 36 00 08 00	00	01	04	00	01	00	00	00	46:56.	.	.	.	.	.	.		
000000B0	00 01 00 00 01 01 04 00	01	00	00	00	AA	00	00	00	.	.	.	.	.	.	.		
000000C0	02 01 03 00 03 00 00 00	EE	00	00	00	03	01	03	00	.	.	.	.	.	.	.		
000000D0	01 00 00 00 06 00 00 00	06	01	03	00	01	00	00	00	.	.	.	.	.	.	.		
000000E0	06 00 00 00 15 01 03 00	01	00	00	00	03	00	00	00	.	.	.	.	.	.	.		
000000F0	01 02 04 00 01 00 00 00	F4	00	00	00	02	02	04	00	.	.	.	.	.	.	.		
00000100	01 00 00 00 49 1D 00 00	00	00	00	00	08	00	08	00	....I	.	.	.	.	.	.		
00000110	08 00 FF D8 FF E0 00 10	4A	46	49	46	00	01	01	00	JFIF	.	.	.	.	.	.		
00000120	00 01 00 01 00 00 FF DB	00	43	00	08	06	06	07	06	C	.	.	.	.	.	.		
00000130	05 08 07 07 07 09 09 08	0A	0C	14	0D	0C	0B	0B	0C	.	.	.	.	.	.	.		
00000140	19 12 13 0F 14 1D 1A 1F	1E	1D	1A	1C	1C	20	24	2E	\$	.	.	.	.	.	.		
00000150	27 20 22 2C 23 1C 1C 28	37	29	2C	30	31	34	34	34	' ",#..(7),01444	.	.	.	.	.	.		

So we correct the file header and...



Voilá! We can see the image!

## Decoding the hidden message

The text in the image is clearly a Base64 encoded string, so we decode it using any tool like [Cyberchef](#) or the linux command line:

```
sergio@kali:~/Escritorio$ echo -n "cm9ja3lvdWlzeW91cmZyaWVuZA==" | base64 -d
```

Obtaining the string:

```
rockyouisyourfriend
```

It seems to suggest that we should to bruteforce the image to extract the hidden message...

## Binwalk

At first, we can think that we must bruteforce the image, but if we check it using binwalk, we see something interesting:

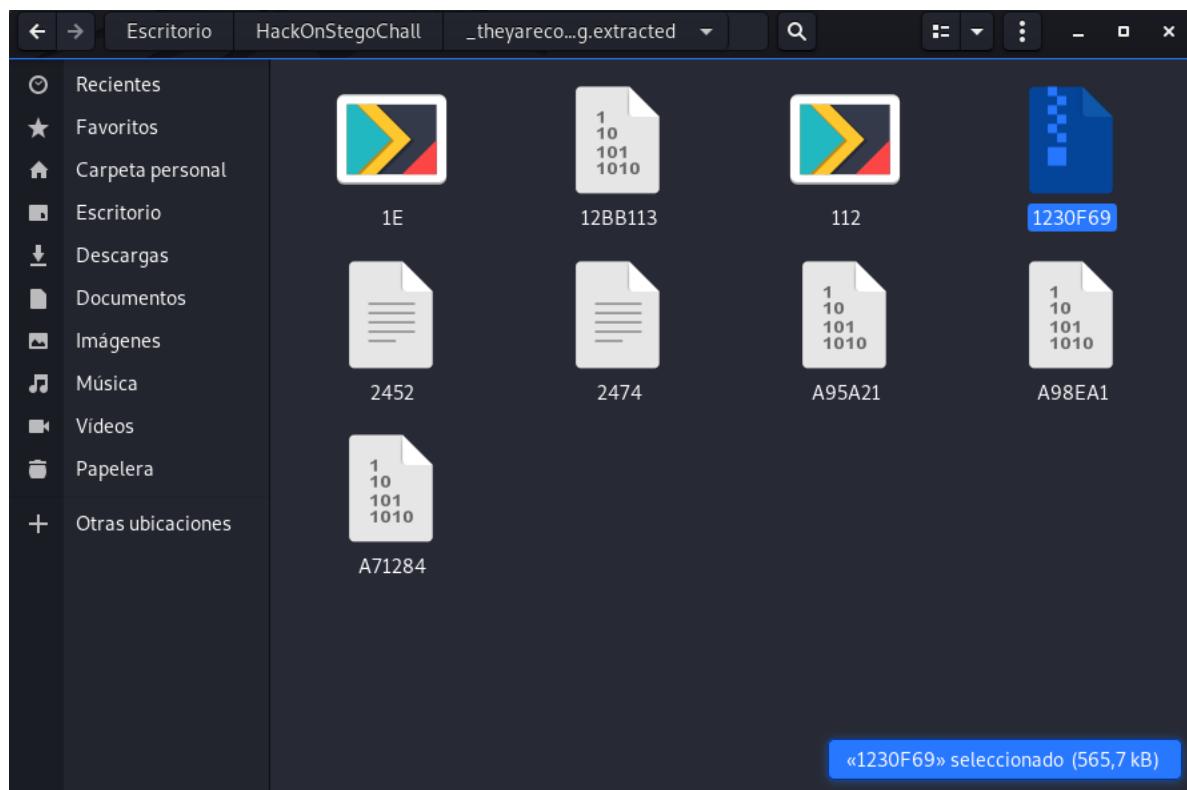
```
sergio@kali:~/Escritorio/HackOnStegoChall$ binwalk theyarecoming.jpg

DECIMAL      HEXADECIMAL      DESCRIPTION
-----
30          0x1E          TIFF image data, little-endian offset of first
image directory: 8
274          0x112         JPEG image data, JFIF standard 1.01
9298         0x2452        Copyright string: "CopyrightOwner> <rdf:Seq/>
</plus:CopyrightOwner> <plus:Licensor> <rdf:Seq/> </plus:Licensor>
</rdf:Description> </rdf:RDF> </x:>""
9332         0x2474        Copyright string: "CopyrightOwner> <plus:Licensor>
<rdf:Seq/> </plus:Licensor> </rdf:Description> </rdf:RDF> </x:xmpmeta>   "
10949252     0xA71284       Intel x86 or x64 microcode, sig 0x00000004, pf_mask
0x20000000, 2000-08-10, rev 0x-80000000, size 2048
11098657     0xA95A21       Intel x86 or x64 microcode, sig 0x09908000, pf_mask
0x2000100d, 2000-01-02, rev 0x0090, size 131072
11112097     0xA98EA1       Intel x86 or x64 microcode, sig 0x08000181, pf_mask
0x61000, 1A34-04-01, size 262160
19074921     0x1230F69       Zip archive data, encrypted compressed size:
565524, uncompressed size: 705644, name: flag.wav
19640595     0x12BB113      End of Zip archive, footer length: 22
```

We can see an encrypted Zip file in the image, which contains a flag.wav file inside. So we extract the zip file using binwalk:

```
sergio@kali:~/Escritorio/HackOnStegoChall$ binwalk --dd='.*' theyarecoming.jpg
```

And now we can try to crack its password



## Time for cracking

Now, having obtained the clue of using rockyou in the first step of the challenge and the Zip file, we will try to crack it and extract the WAV file. To do this, we use [John The Ripper](#) and the zip2john utility to obtain the hash of the file.

```
sergio@kali:~/Escritorio/HackOnStegoChall/_theyarecoming.jpg.extracted$ zip2john  
1230F69 > ../ziphash
```

And execute john with rockyou dictionary:

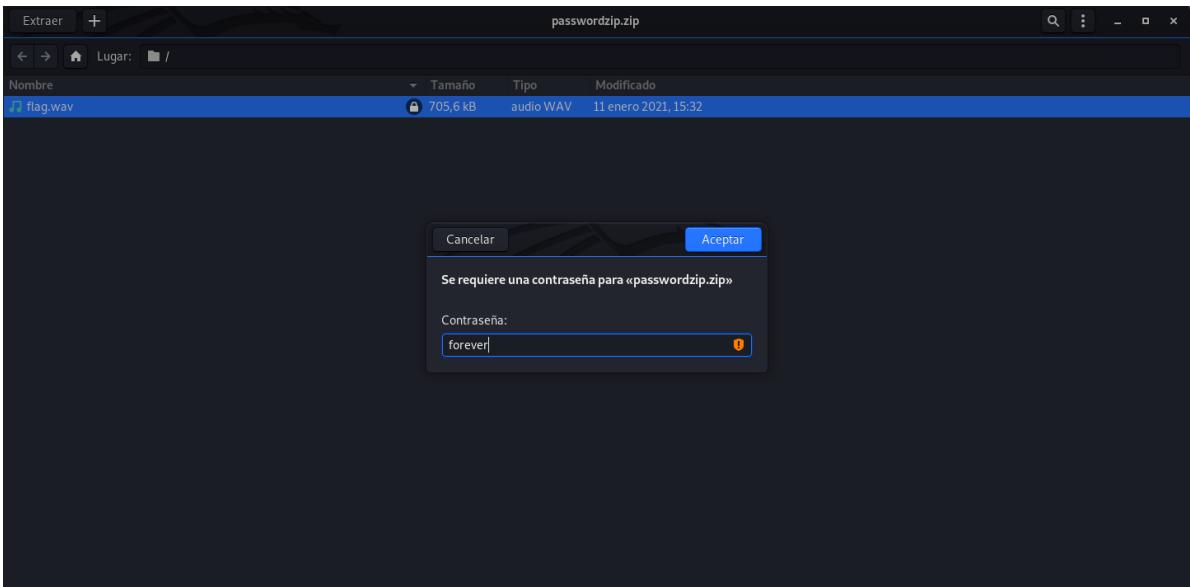
```
sergio@kali:~/Escritorio/HackOnStegoChall$ john ziphash --  
wordlist=/usr/share/wordlists/rockyou.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 256/256 AVX2 8x])  
Press 'q' or Ctrl-C to abort, almost any other key for status  
forever          (passwordzip.zip.flag.wav)  
1g 0:00:00:00 DONE (2021-01-11 19:44) 7.142g/s 914.2p/s 914.2c/s 914.2C/s  
samantha..diamond  
Use the "--show" option to display all of the cracked passwords reliably  
Session completed
```

In few seconds/minutes, depending on your machine, we obtain the cracked password:

```
sergio@kali:~/Escritorio/HackOnStegoChall$ john ziphash --show  
passwordzip.zip.flag.wav:forever:flag.wav:passwordzip.zip:passwordzip.zip  
  
1 password hash cracked, 0 left
```

## Extracting and... obtaining the flag?

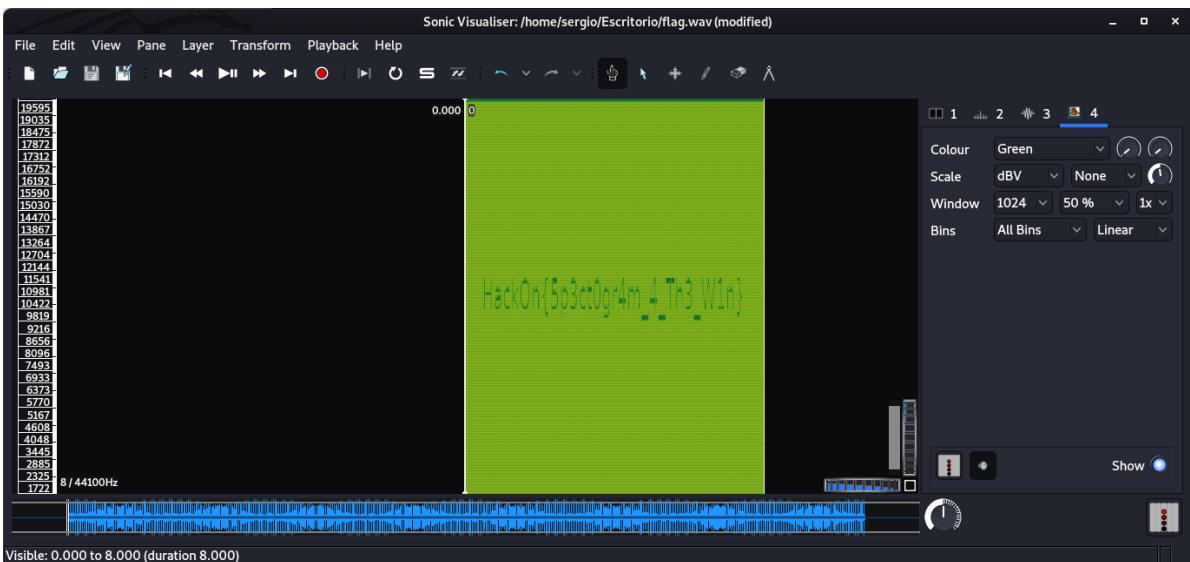
Using `forever` as password, we can extract the WAV file and if we listen it, we obtain the flag... or not.



If we open the file, we can not understand anything! Frustrating...

## The spectogram is your friend

When we face with audio stego challenges, one of the first steps that we should take is to look into the wave of the file, looking for morse encoded messages, binary encoded messages, etc. For this task, we have tools like [Audacity](#) or [Sonic visualiser](#). Analyzing the wave of the WAV file with Audacity, we cannot see any pattern, so we can assume that there is no hidden messages in the wave. Its tame to go over the second typicall step in audio stego challenges: analyze the spectogram. We do this with sonic visualiser and... BINGO!



**FLAG: HackOn{5p3ct0gr4m\_4\_Th3\_W1n}**

Hope you have enjoyed this challenge and learned something :)