

Forencics

Writeup Report

1. Babushka

Challenge Description

We were given a zipped file named forencics, which contained an image called Babushka.jpg. The challenge hinted that, much like a real Russian nesting doll, this image would contain multiple layers of hidden information. Our goal: to find the hidden flag of the format CTFLib{...}.

Inspection (Hex Analysis)

- The zipped file ‘forencics’ has been downloaded.
- The image Babushka.jpg has been opened via [Notepad++](#) Hex editor plugin.
- By clicking View in Hex in the text editor plugins, we see the raw hexadecimal representation of the image data. Since the challenge describes the CTFLib{..} would be hidden inside, we search for this ANSI string within the hex dump. Next up, the results show

Address 0 1 2 3 4 5 6 7 8 9 a b c d e f Dump
00382240 00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 00
00382250 4c 61 79 65 72 5f 31 2f 4c 61 79 65 72 5f 32 2f Layer_1/Layer_2/
00382260 4c 61 79 65 72 5f 33 2f 50 4b 03 04 14 00 00 00 Layer_3/PK.....
00382270 00 00 23 14 46 54 67 67 de 41 d0 00 00 00 d0 00 ..#.FTggDAD...D.
00382280 00 00 20 00 00 00 4c 61 79 65 72 5f 31 2f 4c 61 ...Layer_1/La
00382290 79 65 72 5f 32 2f 4c 61 79 65 72 5f 33 2f 46 6c yer_2/Layer_3/F1
003822a0 61 67 2e 7a 69 70 50 4b 03 04 0a 00 00 00 00 ag.zipPK.....
003822b0 ae 0d 45 54 58 0b 12 bb 3a 00 00 3a 00 00 00 @.ETX..>:....
003822c0 08 00 00 00 46 6c 61 67 2e 74 78 74 43 54 46 4c ...Flag.txtCTFL
003822d0 69 62 7b 59 6f 75 5f 68 61 76 65 5f 75 6e 66 6f ib(You_have_unfo
003822e0 6c 64 65 64 5f 74 68 65 5f 73 65 63 72 65 74 73 lded_the_secrets
003822f0 5f 6f 66 5f 74 68 65 5f 52 75 73 73 69 61 6e 5f _of_the_Russian_
00382300 64 6f 6c 6c 21 7d 50 4b 01 02 3f 00 0a 00 00 00 doll!)PK..?....
00382310 00 00 ae 0d 45 54 58 0b 12 bb 3a 00 00 00 3a 00 ..@.ETX..>:....
00382320 00 00 08 00 24 00 00 00 00 00 00 20 00 00 00\$.
00382330 00 00 00 00 46 6c 61 67 2e 74 78 74 0a 00 20 00Flag.txt..
00382340 00 00 00 00 01 00 18 00 da f8 96 48 21 1a d8 01Úø-H!.Ø.-ßØe Ø.
00382350 da f8 96 48 21 1a d8 01 ac df d8 65 20 1a d8 01 Úø-H!.Ø.-ßØe Ø.
00382360 50 4b 05 06 00 00 00 01 00 01 00 5a 00 00 00 PK.....Z...
.....

...that the string is indeed found, and also there is the message inside the Flag.txt that shows the encrypted message, CTFLib{You_have_unfolded_the_secrets_of_the_Russian_doll!}

The hex dump revealed references to Flag.txt and the text. This confirmed that there was a hidden flag embedded in the image.

So there is one more step to be made.

- d. An internal structure is noticed, indicating an embedded ZIP archive:
Layer1/Layer2/Layer3/Flag.zipPK.
- e. Locating the ZIP's starting signature (50 4B 03 04 in hex) and its ending signature (50 4B 05 06), which marks the end of the central directory record,
 1. copying the whole raw byte block and
 2. pasting it in a new file as a binary paste,
 3. saving the file as bab.zip, we finally get the result.

Using the archive tool 7-zip, we open the bab.zip and inside we're now able to see the Flag.txt which inside contains the flag clearly now seen :

CTFLib{You_have_unfolded_the_secrets_of_the_Russian_doll!}

Conclusion

In this challenge, the Babushka.jpg file was modified to contain a nested ZIP archive. By analyzing the image in a hex editor and extracting the embedded archive, we revealed the hidden **Flag.txt** file. Ultimately, we recovered the flag:

CTFLib{You_have_unfolded_the_secrets_of_the_Russian_doll!}

2. Hacked

Challenge Description

We were given an image file named Hacked.jpg that looked normal at first glance. Our objective was to investigate whether it contained any hidden data or had been otherwise tampered with, and to find a flag in the format CTFLib{...} if present.

Hex Analysis with HxD

- a. For this challenge we use the tool [HxD](#) (a Windows-based hex editor) and open the Hacked.jpg. We're able to see the bytes of the file image.
- b. We search again for the String CTFLib{ in ANSI to see if there is somewhere in the file.

- c. A standard JPG file typically ends with the bytes FF D9 and the file actually we see that indeed starts the standard way with FF D8

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01	ÿà..JFIF.....
00000010	00 01 00 00 FF DB 00 43 00 05 03 04 04 04 03 05ÿÛ.C.....
00000020	04 04 04 05 05 05 06 07 0C 08 07 07 07 07 0F 0B
00000030	0B 09 0C 11 0F 12 12 11 0F 11 11 13 16 1C 17 13

.. but it doesn't really end with the expected bytes. Instead :

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
0003F010	F1 5C E5 09 6D 1F ED 9B 12 74 79 8F 71 E1 A5 E7	ñ\å.m.i>.ty.qåÙç
0003F020	ED 9A 66 03 D1 36 77 D7 9C 34 04 50 2B CE 87 8D	i..ñéwæø4.PÍ+í.
0003F030	71 E7 20 69 54 97 CF B7 0F 80 09 BF 4D C0 A2 20	gq_1T-1..é..MÄc
0003F040	76 1D 30 DE 03 04 2A EF 48 BC 6A DC 64 89 C9	v.ØP..iHMjÜd..kÉ
0003F050	F2 9F OC 0C 8B 0A 40 83 B1 8F F3 24 2B 42 43	öY.åç..ßt..öÙ+BC
0003F060	C5 D2 BE FE B1 39 FA D3 AA 64 9F F2 C0 C3 49	Åøqø+iøqó*dýùAAI
0003F070	A5 01 BC 5F 97 15 71 A3 45 32 E4 7C 1A F7 92 93	¥.4..-qÆZä .-"
0003F080	22 AC 2F 19 60 4B 44 F2 59 7F 9C 02 71 2E BA	"-/..KDöY.?æ.q.."
0003F090	B7 93 10 67 B6 2E BD 1E 72 0E A1 67 D9 EF DE 21	".,.q..r.;gÙb!
0003F0A0	4E C3 4B 4E 9C 85 FD B8 35 FD 50 99 ED 32 1A 22 45	NÄKNœ...ÿ..5Pm12..E
0003F0B0	45 DA 39 E9 BO FA 52 AB 3A 21 FC 98 00 82 FC A7	EÜ9þ*üRæ!:ÿ..,us
0003FOCO	73 C0 3F 9B 86 B2 F2 E6 14 DB 55 93 0E 71 0D D2	æ?..+*ðæ..ðU..ç.ó
0003F0DO	61 69 AF 55 FD E2 4F A9 C7 DF FA 7A CB BC FF 00	ai~ÜyåOçQûzE*ý.
0003F0EO	1F 57 59 5C 33 BF 9C 01 C3 E4 90 F8 C4 CA 35 C1	.WY!3æ..Å..æÅSÁ
0003F0FO	3E CD 8F BC 8B 0A AF SE DC A0 C6 C0 B4 66 91 C4	>í..åc..~Ü..ZÄ..fí..
0003F100	F3 2A 45 55 39 98 E9 77 83 B9 9F 63 18 35 86 40	ó*EU9*ewf!ÿ.C.5të
0003F110	8F 49 EF 05 9A 7F 61 3F 9D FF B0 48 51 27 67 24	.II..å..æ..b..Hø'gø
0003F120	CB CB 00 20 B5 87 88 78 0E 31 7D 30 D1 24 E0 24	EE..p#*x..l)ñNSAS
0003F130	7A E7 03 A9 0E DD 53 B7 75 E5 E7 0E D3 76 81 4B	zç..é..ÝS..uåç..Öv.K
0003F140	1B 49 5F FE 65 3C 7B 9A 9D 0F 8E F9 C4 EB 49 0D	.I..þe<í..žúñéI.
0003F150	22 76 62 4F 68 58 1B 41 BE 6F E7 2A 66 9C 8D ED	"vbOñX.AWøç!fø.i
0003F160	8C DD BD 9A 3C B3 BE 4C AF 74 C9 10 D6 1D 27 EF	óYùå<..ñL..tÉ..ó..í
0003F170	2D DF F3 F1 9A 13 2D BC E3 AE 87 9E B0 95 84 B7	-òññ..-iøøþz°..-
0003F180	87 3E FF 00 43 54 46 4C 69 62 7B 54 68 69 73 5F	#?y..CTFLib{This_is_the_flag_hidden_in_hex_form!}
0003F190	69 73 5F 74 68 65 5F 66 6C 61 67 5F 68 69 64 64	is_the_flag_hidden_in_hex_form!
0003F1A0	65 6E 5F 69 6E 5F 68 65 78 5F 66 6F 72 6D 21 7D	

- d. we observed that the file's ending bytes include FF 00 followed immediately by an ASCII string, the **CTFLib{This_is_the_flag_hidden_in_hex_form!}** so it is definitely modified.
e. Extra data (the hidden flag) has been intentionally appended after what would be the normal image content. The image appears normal while hiding a secret message, so it's indeed Hacked.

Conclusion

By analyzing Hacked.jpg in a hex editor, we discovered that the image's data does not conclude with the usual FF D9 marker. Instead, extra data beginning with FF 00 is appended, which contains the hidden flag:

CTFLib{This_is_the_flag_hidden_in_hex_form!}

..confirming the "Hacked" nature of the challenge.

3. Information

Challenge Description

We have the image `Information.jpg` that appears to show an adorable missing dog. The hint suggests we should look for details about the dog's owner, implying the crucial data may be hidden in the image's metadata.

Metadata Analysis

- a. For this challenge we use the [ExifTool](#) and we run the `exiftool Information.jpg` in command line. (there is a small report about how the exiftool was directly used from cmd).
- b. In the **Artist**, **Creator**, or **Image Creator** fields, we discovered a string that looked encoded or encrypted:

`CTFLib{Gur_vasbezngvba_vf_va_gur_Zrgnqngn!}`

```
Directory : f_Forensics
File Size : 904 kB
Zone Identifier : Exists
File Modification Date/Time : 2025:03:10 03:43:24+02:00
File Access Date/Time : 2025:03:11 00:25:55+02:00
File Creation Date/Time : 2023:08:23 15:46:04+03:00
File Permissions : -rw-rw-rw-
File Type : JPEG
File Type Extension : jpg
MIME Type : image/jpeg
JFIF Version : 1.01
Resolution Unit : inches
X Resolution : 72
Y Resolution : 72
Exif Byte Order : Big-endian (Motorola MM)
Artist : CTFLib{Gur_vasbezngvba_vf_va_gur_Zrgnqngn!}
Current IPTC Digest : 5eff5fc0d5295914c5bb29c450e7a6ad
Coded Character Set : UTF8
Envelope Record Version : 4
By-line : CTFLib{Gur_vasbezngvba_vf_va_gur
Application Record Version : 4
XMP Toolkit : Image::ExifTool 12.36
Creator : CTFLib{Gur_vasbezngvba_vf_va_gur_Zrgnqngn!}
Image Creator Name : CTFLib{Gur_vasbezngvba_vf_va_gur_Zrgnqngn!}
Profile CMM Type : Linotronic
Profile Version : 2.1.0
Profile Class : Display Device Profile
Color Space Data : RGB
Profile Connection Space : XYZ
Profile Date Time : 1998:02:09 06:49:00
Profile File Signature : acsp
Primary Platform : Microsoft Corporation
CMM Flags : Not Embedded, Independent
Device Manufacturer : Hewlett-Packard
Device Model : sRGB
Device Attributes : Reflective, Glossy, Positive, Color
Rendering Intent : Perceptual
Connection Space Illuminant : 0.9642 1 0.82491
Profile Creator : Hewlett-Packard
Profile ID : 0
Profile Copyright : Copyright (c) 1998 Hewlett-Packard Company
```

There is not an encryption suspicion because headers and structure of the file follows the expected. So the message is just encoded.

- c. Using the Cyber Chef from <https://gchq.github.io/CyberChef/> we try to decode the message by experimenting with multiple common encryption/encoding schemes, ROT13 was the one that gave us the result:

The screenshot shows the CyberChef interface. On the left, there's a sidebar with various encryption/decryption options like GOST Sign, GOST Verify, GOST Key Wrap, GOST Key Unwrap, ROT13, ROT13 Brute Force, ROT47, ROT47 Brute Force, ROT8000, XOR, XOR Brute Force, Vigenère Encode, Vigenère Decode, and XXTEA Encrypt. The main area has tabs for 'Recipe' (set to 'ROT13') and 'Input' (containing the encoded string). Below the input is a section for 'Amount' with a dropdown set to '13'. The 'Output' tab shows the decrypted message: 'The_information_is_in_the_Metadata!'.

- d. There isn't a name in the end. We only found the message:

The_information_is_in_the_Metadata!

The challenge is not done; This is not the name of the owner. We should try find its owner through physical means... 🔎

Conclusion

Using [ExifTool](#), we examined the metadata of Information.jpg and discovered a ROT13-encoded string in the Artist/Creator/Image Creator field. After trying common cipher methods, ROT13 seem to correctly decode the message. This revealed the final flag:

CTFLib{The_information_is_in_the_Metadata!}