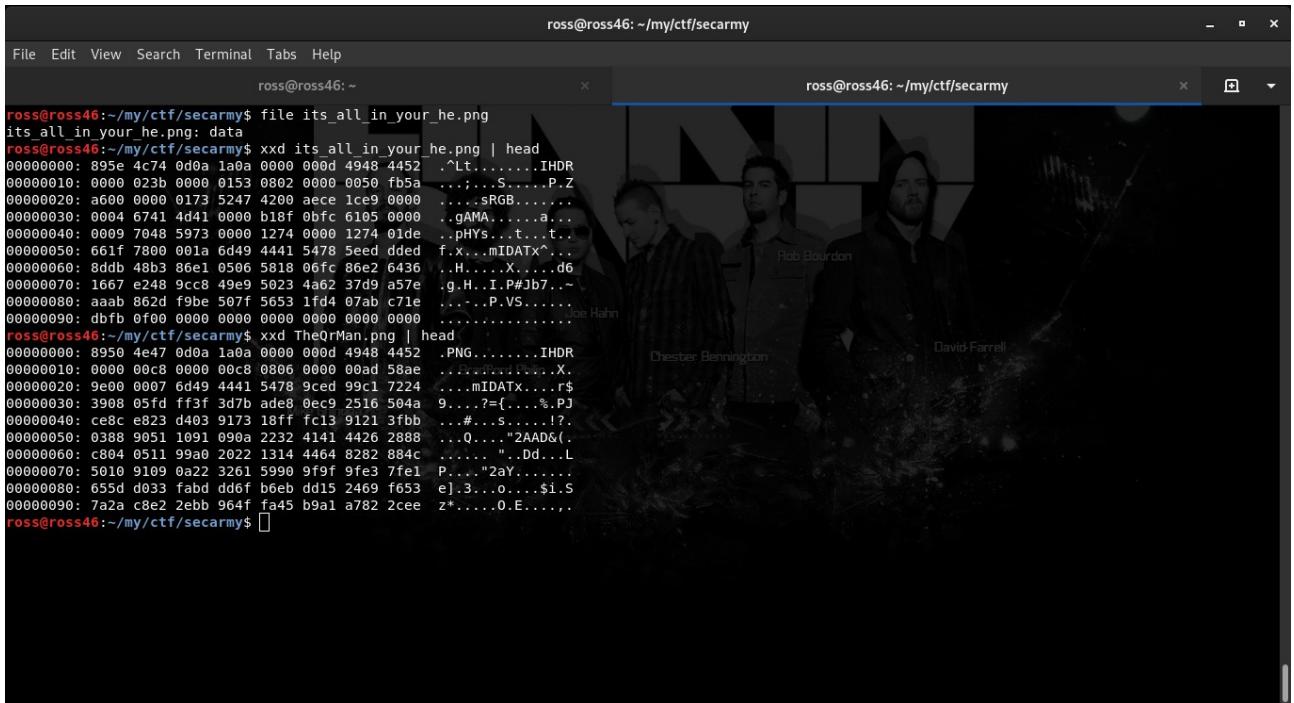


PART II of my writeups

its_all_in_your_head.png:

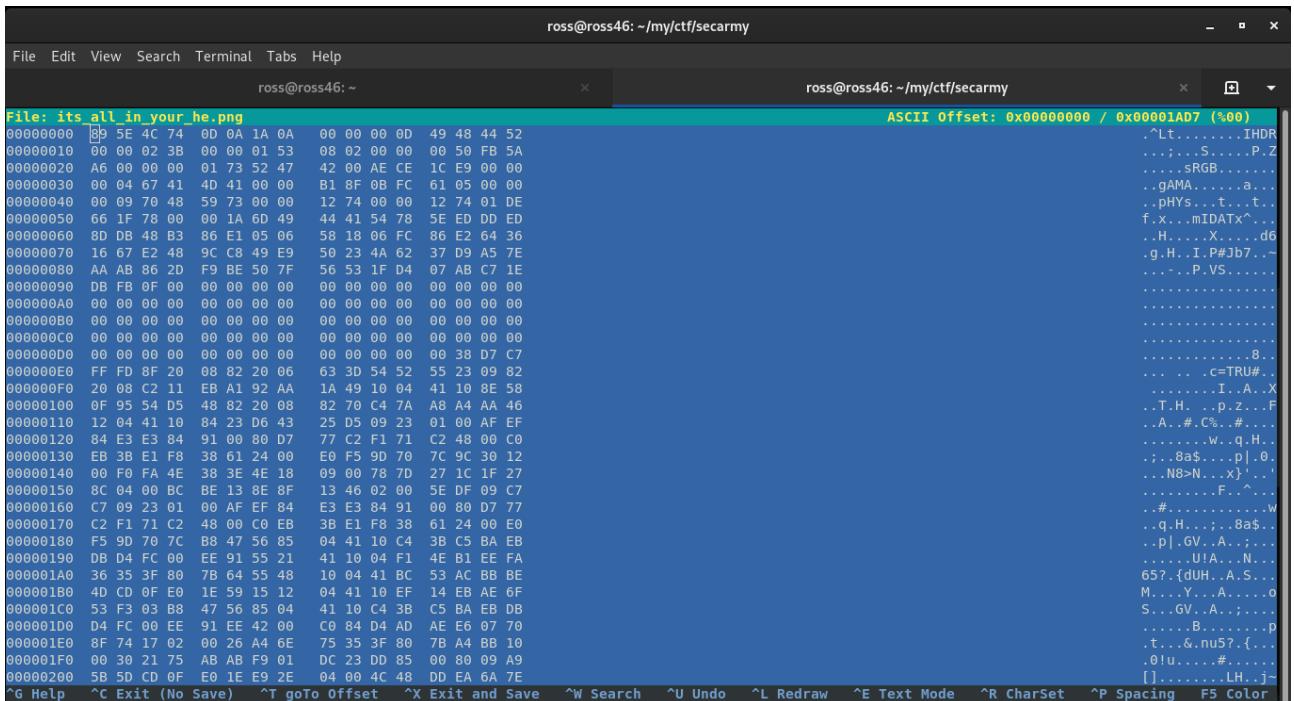
analyzing the file type we get: its_all_in_your_head.png: data

we check the magic bytes of it as its extension is .png



```
ross@ross46: ~/my/ctf/seccarmy
File Edit View Search Terminal Tabs Help
ross@ross46: ~
ross@ross46: ~/my/ctf/seccarmy$ file its_all_in_your_head.png
its all in your he.png: data
ross@ross46: ~/my/ctf/seccarmy$ xxd its_all_in_your_head.png | head
00000000: 895e 4c74 000a 1a0a 0000 000d 4948 4452 .^Lt.....IHDR
00000010: 0000 023b 0000 0153 0802 0000 0050 fb5a ....S....P.Z
00000020: a600 0000 0173 5247 4200 aece 1ce9 0000 .....SRGB.....
00000030: 0004 6741 4041 0000 b1bf 0bfc 6105 0000 ..gAMA.....a...
00000040: 0009 7048 5973 0000 1274 0000 1274 01de ..pHYS....t...
00000050: 661f 7800 001a 6d49 4441 5478 5eed dded f.x..mIDATx^...
00000060: 8dd0 48b3 86e1 0506 5818 06fc 86e2 6436 ..H....X....d6
00000070: 1667 e248 9cc8 49e9 5023 4a62 37d9 a57e .g.H..I.P#jb7..~
00000080: aaab 862d f9b8 507f 5653 1fd4 07ab c71e .....P.VS.....
00000090: dbfb 0f00 0000 0000 0000 0000 0000 0000 .....P.I...
ross@ross46: ~/my/ctf/seccarmy$ xxd TheQrMan.png | head
00000000: 8950 4e47 0000 1a0a 0000 000d 4948 4452 .PNG.....IHDR
00000010: 0000 00c0 0000 00c8 0800 0000 00ad 58ae .....S....X.
00000020: 9e00 0007 6049 4441 5478 9ced 99c1 7224 ..mIDATx^...r$ 
00000030: 3908 05fd ff3f 3d7b ade8 0ec9 2516 504a 9....?={....%PJ
00000040: ce8c e923 d493 9173 18ff fc13 9121 3fb8 ..#.S....!?
00000050: 0388 9051 1091 090a 2232 4141 4426 2888 ..Q...."ZADS<.
00000060: c804 0511 99a0 2022 1314 4464 8282 884c ....."Dd..L
00000070: 5010 9109 0a22 3261 5990 9f9f 9fe3 7fe1 P...."2aY.....
00000080: 655d d033 fabd dd6f b6eb dd15 2469 f653 ej.3....o....$i.S
00000090: 7a2a c8e2 2ebb 964f fa45 b9a1 a782 2cee z*....0.E.....
ross@ross46: ~/my/ctf/seccarmy$
```

Comparing magic bytes of a proper PNG and its_all_in_your_head.png you can see there is a slight variation in the magic bytes.



```
ross@ross46: ~/my/ctf/seccarmy
File Edit View Search Terminal Tabs Help
ross@ross46: ~
ross@ross46: ~/my/ctf/seccarmy
ASCII Offset: 0x00000000 / 0x00001AD7 (%00)
00000000: 89 5E 4C 74 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 .^Lt.....IHDR
00000010: 00 00 02 3B 00 00 01 53 08 02 00 00 00 50 FB 5A ....S....P.Z
00000020: A6 00 00 00 01 73 52 47 42 00 AE CE 1C E9 00 00 .....SRGB.....
00000030: 00 04 67 41 4D 41 00 00 B1 8F 0B FC 61 05 00 00 ..gAMA.....a...
00000040: 00 09 70 48 59 73 00 00 12 74 00 00 12 74 01 DE ..pHYS....t...
00000050: 66 1F 78 00 00 1A 6D 49 44 41 54 78 5E ED DD ED f.x..mIDATx^...
00000060: 8D DB 48 B3 86 E1 05 06 58 18 06 FC 86 E2 64 36 ..H....X....d6
00000070: 16 67 E2 48 9C CB 49 E9 50 23 4A 62 37 D9 A5 7E .g.H..I.P#jb7..~
00000080: AA AB 86 2D F9 BE 50 7F 56 53 1F 04 07 AB C7 1E .....P.VS.....
00000090: DB FB 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 .....P.I...
000000A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....P.I...
000000B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....P.I...
000000C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....P.I...
000000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....P.I...
000000E0: FF FD 8F 20 08 82 20 06 63 3D 54 52 55 23 09 82 .....8...
000000F0: 20 08 C2 11 EB A1 92 AA 1A 49 10 04 41 10 8E 58 ..c=TRU#...
00000100: 0F 95 54 D5 48 82 20 08 82 70 C4 7A A8 A4 AA 46 .....I.A..X
00000110: 12 04 41 10 84 23 D6 43 25 D5 09 23 01 00 AF EF ..T.H..P.z...F
00000120: 84 E3 E3 84 91 00 80 D7 77 C2 F1 71 C2 48 00 C0 ..A.#.C%.#...
00000130: EB 3B E1 F8 38 61 24 00 E0 F5 9D 70 7C 9C 30 12 ..A.w.q.H...
00000140: 00 F0 FA 4E 38 3E 4E 18 09 00 78 7D 27 1C 1F 27 ..B.B$...p|0...
00000150: 8C 04 00 BC BE 13 8E 8F 13 46 02 00 5E DF 09 C7 ..N8>N...x}'...
00000160: C7 09 23 01 00 AF EF 84 E3 E3 84 91 00 80 07 77 ..F.^...
00000170: C2 F1 71 C2 48 00 C0 EB 3B E1 F8 38 61 24 00 E0 ..#....w...
00000180: F5 9D 70 7C B8 47 56 85 04 41 10 C4 3B C5 BA EB ..q.H...;...8$...
00000190: DB D4 FC 00 EE 91 55 21 41 10 04 F1 4E B1 EE FA ..p|GV..A.;...
000001A0: 36 35 3F 80 7B 64 55 48 10 04 41 BC 53 AC BB BE ..U.IA..N...
000001B0: 40 CD 0F E0 1E 59 15 12 04 41 10 EF 14 EB AE 6F 657.{0UH..A.S...
000001C0: 53 F3 03 B8 47 56 85 04 41 10 C4 3B C5 BA EB DB M...Y..A...o...
000001D0: D4 FC 00 EE 91 EE 42 00 C0 84 D4 AD AE E6 07 70 S...GV..A;...
000001E0: 8F 74 17 02 00 26 A4 6E 75 35 3F 80 7B A4 BB 10 ..t...&nu5?{...
000001F0: 00 30 21 75 AB AB F9 01 DC 23 DD 85 00 80 09 A9 ..0iu....#...
00000200: 5B 5D CD 0F E0 1E E9 2E 04 00 04 4C 48 DD EA 6A 7E [...].....LH..j~
```

We open the its_all_in_your_head.png with hexeditor to edit its magic byte.

Hexeditor -b its_all_in_your_head.png and change the bytes

89 5E 4C 74 > 89 50 4E 47

How we got the magic bytes:

check the link and image

https://en.wikipedia.org/wiki/List_of_file_signatures

https://en.wikipedia.org/wiki/List_of_file_signatures			
52 61 72 21 1A 07 00	Rar!...	0	rar
52 61 72 21 1A 07 01 00	Rar!....	0	rar
7F 45 4C 46	.ELF	0	Executable and Linkable Format
89 50 4E 47 0D 0A 1A 0A	.PNG....	0	Image encoded in the Portable Network Graphics format ^[13]
CA FE BA BE	����	0	Java class file, Mach-O Fat Binary
EF BB BF	���	0	UTF-8 encoded Unicode byte order mark, commonly seen in text files.
FE ED FA CE	0 0x1000	Mach-O binary (32-bit)
FF ED FA CE		0	Mach-O binary (64-bit)

so the proper format it

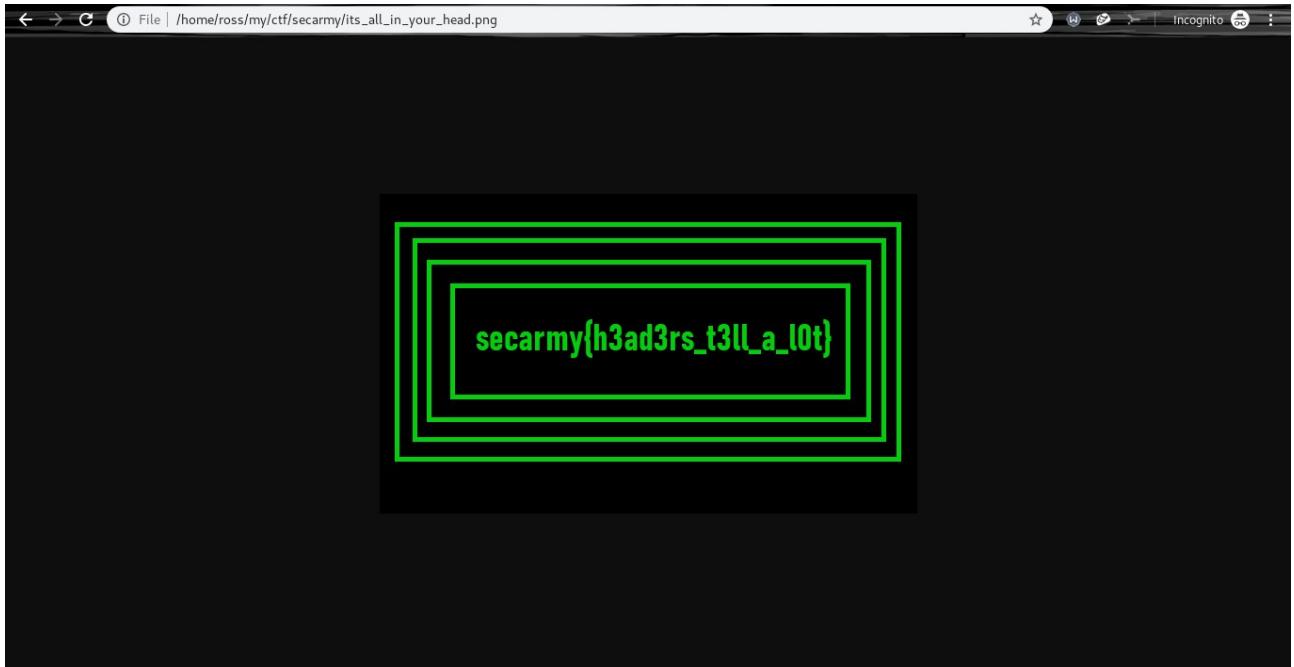
```
ross@ross46: ~/my/ctf/seccarmy
File Edit View Search Terminal Tabs Help
ross@ross46: ~
File: its_all_in_your_head.png          ASCII Offset: 0x00000000 / 0x000001AD7 (%00)
00 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52
00000010 00 00 02 3B 00 00 01 53 08 02 00 00 00 50 FB 5A
00000020 A6 00 00 00 01 73 52 47 42 00 AE CE 1C E9 00 00
00000030 00 04 67 41 4D 41 00 00 B1 8F 0B FC 61 05 00 00
00000040 00 09 70 48 59 73 00 00 12 74 00 00 12 74 01 DE
00000050 66 1F 78 00 00 1A 6D 49 44 41 54 78 5E ED DD ED
00000060 8D DB 48 B3 8E E1 05 06 58 18 00 FC 86 E2 64 36
00000070 16 67 E2 48 9C C8 49 E9 50 23 4A 62 37 D9 A5 7E
00000080 AA AB 86 2D F9 BE 50 7F 56 53 1F D4 07 AB C7 1E
00000090 DB FB 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
000000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000D0 00 00 00 00 00 00 00 00 00 00 00 00 38 D7 C7
000000E0 FF FD 8F 20 08 82 20 06 63 3D 54 52 55 23 09 82
000000F0 20 08 C2 11 EB A1 92 AA 1A 49 10 04 41 10 8E 58
00000100 0F 95 54 D5 48 82 20 08 82 70 C4 7A A8 A4 AA 46
00000110 12 04 41 10 84 23 D6 43 25 D5 09 23 01 00 AF EF
00000120 84 E3 E3 84 91 00 80 D7 77 C2 F1 71 C2 48 00 C0
00000130 EB 3B E1 F8 38 61 24 00 E0 F5 9D 70 7C 9C 30 12
00000140 00 F6 FA 4E 38 3E 4E 18 09 00 78 7D 27 1C 1F 27
00000150 BC 04 00 BC BE 13 8E 8F 13 46 02 00 5E DF 09 C7
00000160 C7 09 23 01 00 AF EF 84 E3 E3 84 91 00 80 D7 77
00000170 C2 F1 71 C2 48 00 C0 EB 3B E1 F8 38 61 24 00 E0
00000180 F5 9D 70 7C B8 47 56 85 04 41 10 C4 3B C5 BA EB
00000190 DB D4 FC 00 EE 91 55 21 41 10 04 F1 4E B1 EE FA
000001A0 36 35 3F 80 7B 64 55 48 10 04 41 BC 53 A1 BB BE
000001B0 4D CD 0F E0 1E 59 15 12 04 41 10 EF 14 EB AE 6F
000001C0 53 F3 03 B8 47 56 85 04 41 10 C4 3B C5 BA EB DB
000001D0 D4 FC 00 EE 91 EE 42 00 C0 84 D4 AD AE E6 07 76
000001E0 8F 74 17 02 00 26 A4 6E 75 35 3F 80 7B A4 BB 16
000001F0 00 30 21 75 AB AB F9 01 DC 23 DD 85 00 80 09 A9
00000200 5B 5D CD 0F E0 1E E9 2E 04 00 4C 48 DD EA 6A 7E
^G Help ^C Exit (No Save) ^T goTo Offset ^X Exit and Save ^W Search ^U Undo ^L Redraw ^E Text Mode ^R CharSet ^P Spacing F5 Color
```

now when we type :

file its_all_in_your_head.png

We get: its_all_in_your_head.png: PNG image data, 571 x 339, 8-bit/color RGB, non-interlaced

Open it and you get the flag



Flag: secarmy{h3ad3rs_t3ll_a_l0t}

The_Confusion:

Unzip the flags.zip we get 4 images:

FLAG3.png
flag1_Rebuilt.png
flag1.png
FILE2.png

once again we use the help of our faithful steganography tool:

<http://stylesuxx.github.io/steganography/>

We open all the 4 images:

The screenshot shows the Steganography Online interface. At the top, there are 'Encode' and 'Decode' buttons, with 'Decode' being the active one. Below this is a section titled 'Decode image' containing instructions: 'To decode a hidden message from an image, just choose an image and hit the **Decode** button.' and 'Neither the image nor the message that has been hidden will be at any moment transmitted over the web, all the magic happens within your browser.' A file input field shows 'Choose file flag1.png' and a 'Decode' button. Below this is a 'Hidden message' section with a scrollable text area containing the decoded message: 'secarmy{h3r3_is_y0u4_fl@g}'. An 'Input' section shows a thumbnail of the image 'flag1.png', which is a yellow circle with the word 'FLAG' in green.

The screenshot shows the Steganography Online interface. At the top, there are 'Encode' and 'Decode' buttons, with 'Decode' being the active one. Below this is a section titled 'Decode image' containing instructions: 'To decode a hidden message from an image, just choose an image and hit the **Decode** button.' and 'Neither the image nor the message that has been hidden will be at any moment transmitted over the web, all the magic happens within your browser.' A file input field shows 'Choose file FILE2.png' and a 'Decode' button. Below this is a 'Hidden message' section with a scrollable text area containing the decoded message: 'secarmy{01101100 01101111 01101100 01110100 01101000 01101001 01101001 01110011 01101001 01110011 01110100 01101000 01100101 01100110 01101100 01000000 01100111} /THIS IS YOUR FLAG*'. An 'Input' section shows a thumbnail of the image 'FILE2.png', which is a yellow circle with the word 'FILE' in green.

Not secure | stylesuxx.github.io/steganography/ Incognito

Steganography Online

Encode Decode

Decode image

To decode a hidden message from an image, just choose an image and hit the **Decode** button.

Neither the image nor the message that has been hidden will be at any moment transmitted over the web, all the magic happens within your browser.

flag1_Rebuilt.png

Decode

Hidden message

```
frpnezl{JN3_V7_f
```

Input



Not secure | stylesuxx.github.io/steganography/ Incognito

Steganography Online

Encode Decode

Decode image

To decode a hidden message from an image, just choose an image and hit the **Decode** button.

Neither the image nor the message that has been hidden will be at any moment transmitted over the web, all the magic happens within your browser.

FLAG3.png

FLAG3.png

Hidden message

```
_04_?7Fb:?
```

Input



We get 4 strings:

secarmy{h3r3_is_y0u4_fl@g}
secarmy{01101100 01101111 01101100 01110100 01101000 01101001 01110011 01101001
01110011 01110100 01101000 01100101 01100110 01101100 01000000 01100111}
frpnezl{JN3_V7_f
04?7Fb:?8N

The first one is obviously ruled out cause it is the flag format, mentioned in welcome
second one also is wrong

now the suspicious strings :
frpnezl{JN3_V7_f
04?7Fb:?8N

Lets try ROT

Site: dcode.fr

Why I prefer this is?, you have brute-force option which saves a lot of time
it performs ROT from 0-47 including special characters at once and has a lot of options

when we try to ROT together we get weird strings

so lets try individually :

first part brute force we get: "secarmy{WA3_I7_s" from ROT-13

second part we get: “0_c0nfu3ing}” from ROT-47

combining it together

Flag: secarmy{WA3_I7_s0_c0nfu3ing}

Save them:

We get a zip file called `cute_cats.zip`. Unzipping it we get a `.jpg` image named `cute_cats.jpg`. Using `stegextract` we find a zip file called `cute_cats_dumps`.

It consists of directory flag1

moving into that directory we find two files flags and a hidden file .sauce

opening flags we get:

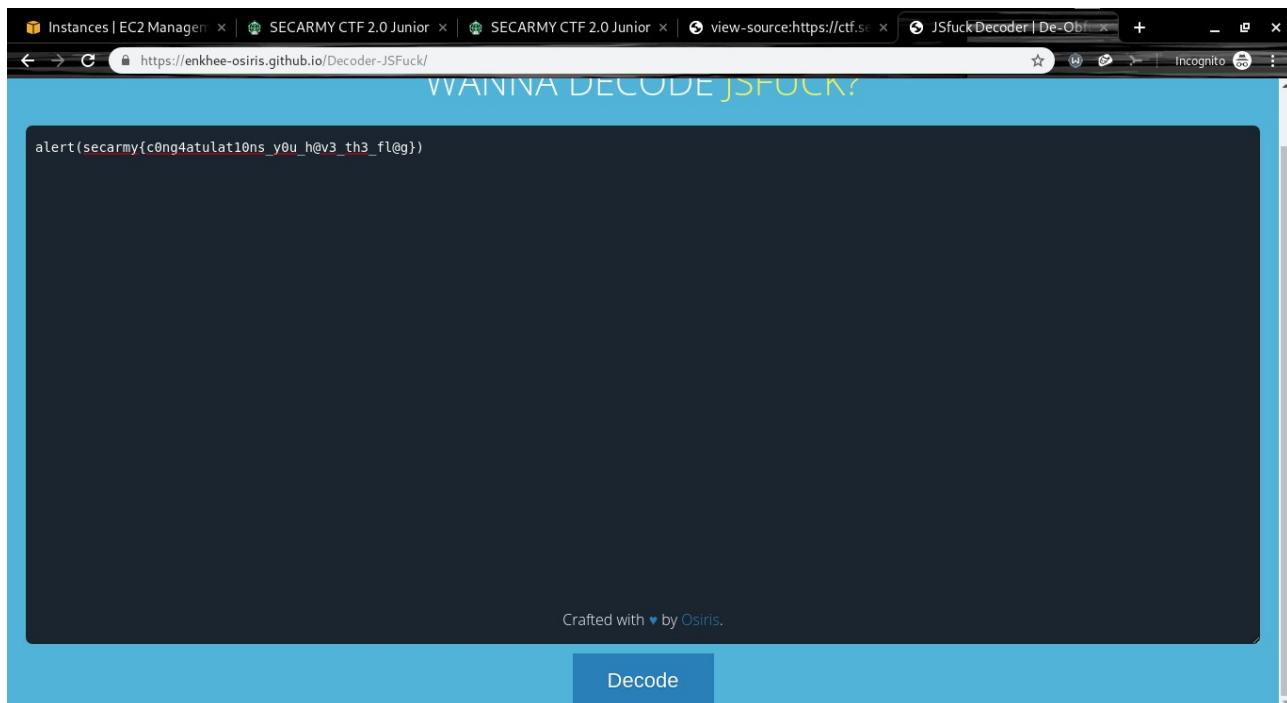
Here is your flag :- <https://pastebin.com/4ffrJKECh><https://pastebin.com/mN89JJE2> which is a hoax so we open .sauce file

we get a jsfuck code, now we gotta decode
link: <https://enkhee-osiris.github.io/Decoder-JSFuck/>



The screenshot shows a browser window with multiple tabs open. The active tab is titled "JSFuckDecoder | De-Obf" and contains the URL "https://enkhee-osiris.github.io/Decoder-JSFuck/". The main content area displays the text "WANNA DECODE JSFUCK?" in yellow. Below it is a massive block of encoded JSFuck code, which is a series of nested functions and operators like "+", "[", "]", and "()", all encoded using various URL-safe characters.

pasting the jsfuck code and decoding we get:
alert(secarmy{c0ng4atulat10ns_y0u_h@v3_th3_fl@g})



The screenshot shows the "JSFuckDecoder | De-Obf" tool interface. At the top, there's a browser-like header with tabs and a URL bar showing "https://enkhee-osiris.github.io/Decoder-JSFuck/". Below the header, the text "WANNA DECODE JSFUCK?" is displayed in yellow. A large black box contains the decoded JavaScript code: "alert(secarmy{c0ng4atulat10ns_y0u_h@v3_th3_fl@g})". At the bottom of the black box, the text "Crafted with ❤ by Osiris." is visible. In the bottom right corner of the main area, there is a blue button labeled "Decode".

Flag: secarmy{c0ng4atulat10ns_y0u_h@v3_th3_fl@g}

Thats the end of forensics

MISC:

Travel:

Question:

Well , I need to discover the path the packet takes . ,

Answer

secarmy{traceroute}

Get_Me:

We have a hint: BTW I LOVE ASCII

Unzip helpmetosolvezip.zip, we get flag2.txt

opening it we get a base32 string, how is it base32? cuz of “====” at the end

Link:https://emn178.github.io/online-tools/base32_decode.html

Use the link to decode, we get:

secarmyIL0V3ASC11

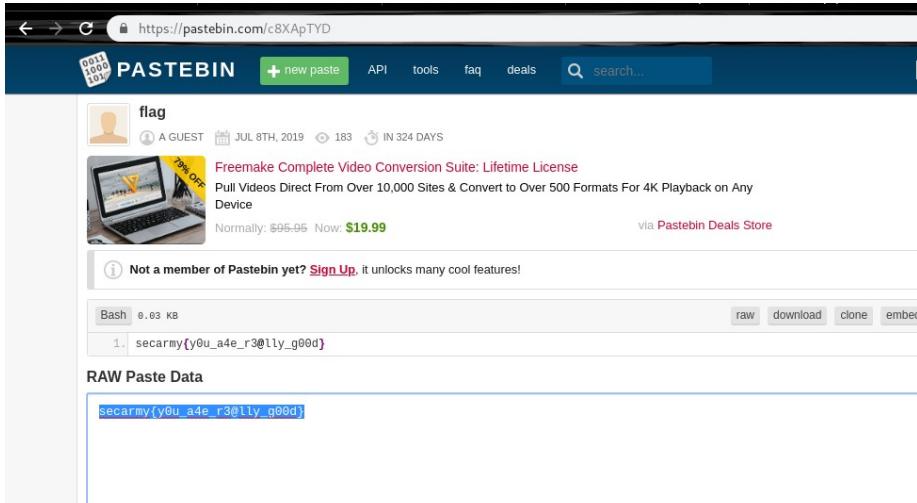
note that it is not letter “O” but Zero “0”

						SHA512/256	SHA512/256
						SHA3-224	SHA3-224
						SHA3-256	SHA3-256
,adPPYba, ,adPPYba, ,adPPYba, ,adPPYYba, 8b,dPPYba, 88,dPbBa,,adPbBa, I8[" " a8P_____88 a8" " " " `Y8 88P' "Y8 88P' "88" "8a `Y8ba, 8PP"""""" 8b ,adPPP88 88 88 88 88 aa]8I "8b, ,aa "8a, ,aa 88, ,88 88 88 88 88 `"YbbdP'" `Ybbd8'" `Ybbd8'" `8bbdP"Y8 88 88 88 88						SHA3-384	SHA3-384
						SHA3-512	SHA3-512
						Keccak-224	Keccak-224
						Keccak-256	Keccak-256
						Keccak-384	Keccak-384
						Keccak-512	Keccak-512
						Shake-128	Shake-128
						Shake-256	Shake-256
					Encode	Decode	
					Base32	Base32	
					Base32 File	Base32 File	
					Base64	Base64	
					Base64 File	Base64 File	
					HTML	HTML	
					URL	URL	
					Misc	Syntax Highlight	

so we get Flag: secarmy{I_L0V3_ASC11}

THE PASTE:

we get “c8XApTYD”, it says i love bash and pasted and internet will help you.
So a hint it is a paste bin link:
<https://pastebin.com/c8XApTYD>



The screenshot shows a browser window with the URL <https://pastebin.com/c8XApTYD>. The page title is "PASTEBIN". The main content area displays a shared link titled "flag" by "A GUEST" posted on "JUL 8TH, 2019" with "183" views and "IN 324 DAYS". Below the title, there is a thumbnail image of a laptop displaying a video player interface with the text "Freemake Complete Video Conversion Suite: Lifetime License". It also mentions "Normally: \$95.95 Now: \$19.99" and "via Pastebin Deals Store". A note below the thumbnail says "Not a member of Pastebin yet? [Sign Up](#), it unlocks many cool features!". The code in the paste is listed under "RAW Paste Data":

```
secarmy{y0u_a4e_r3@lly_g00d}
```

Flag:[secarmy{y0u_a4e_r3@lly_g00d}](https://pastebin.com/c8XApTYD)

Constructor Ninja:

opening the cpp code we see no explicitly defined constructor but for:

```
Student s1 = Student(1, "John", 89);
```

answer:

```
secarmy{parameterizedconstructor}
```

Poor prisoner:

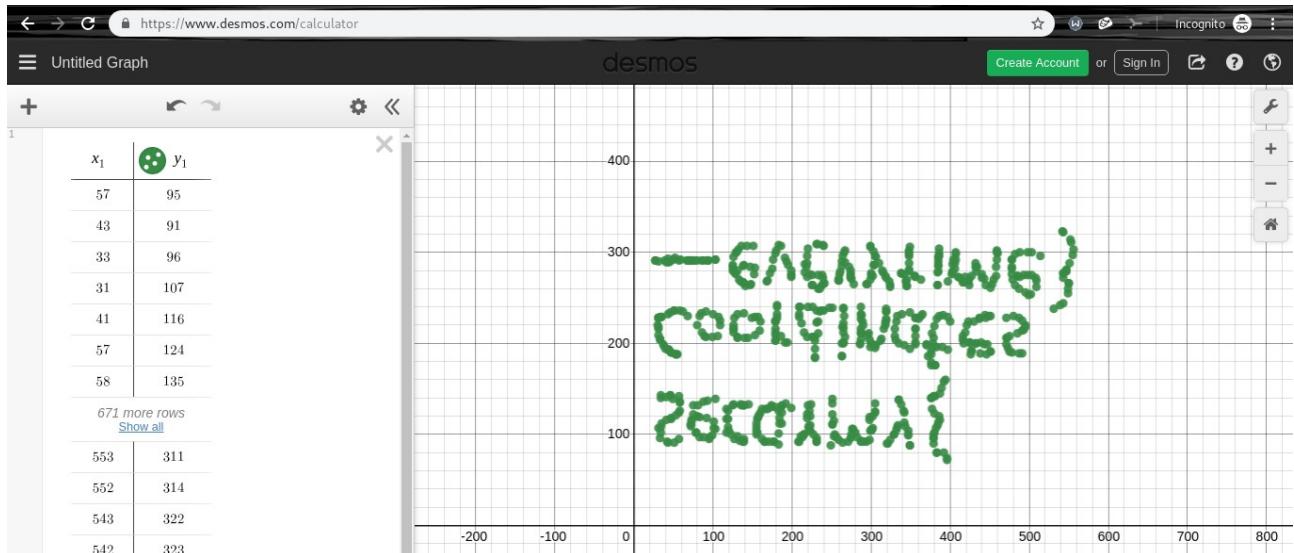
so we get a file radio_capture.txt which consists of co ordinates:

we need to plot those, we can use:

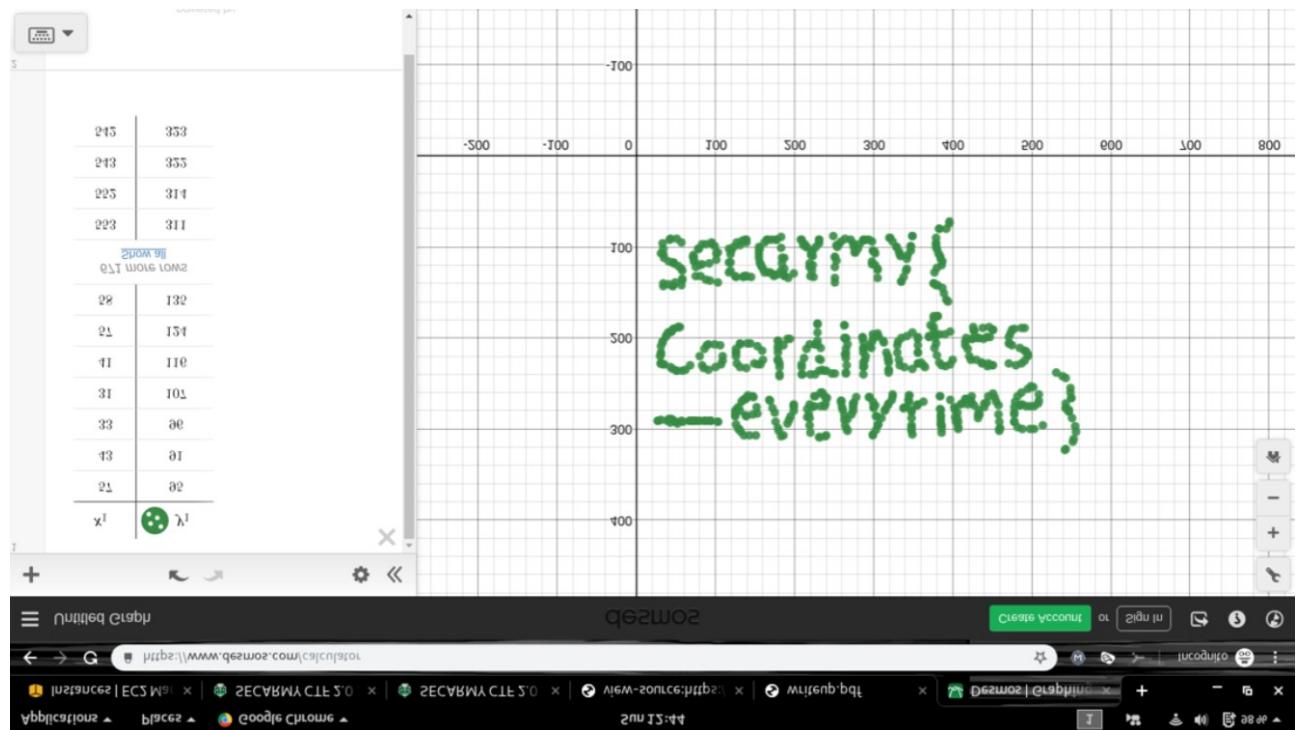
<https://www.desmos.com/calculator>

copy the entire co ordinates and paste it.

We get the key in reverse order



Mirroring the image and inverting it we get the flag:



Flag: secarmy{coordinates-everytime}.

NOTE: it is a “-” and not a “_”

WEB:

Prizes:

link: <https://ctf.sec.army/prizes>

view the source code:



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
  <meta charset="utf-8">
  <title>SECARMY Forum</title>
  <link rel="stylesheet" href="style.css">
  <link href="https://fonts.googleapis.com/css?family=Ubuntu&display=swap" rel="stylesheet">
  <script type="text/javascript" src="index.js"></script>
  <script type="text/javascript" src="process.js"></script>
</head>
<body>
  <p>SECARMY Forum</p>
  <ul>
    <li><a href="https://weare.sec.army/hall-of-fame" target="_blank">Hall of Fame</a></li>
    <li>Digital Certificate</li>
  </ul>
  <ul>
    <li><a href="http://forum.sec.army/" target="_blank">SECARMY Forum</a> 2000 Bytes</li>
  </ul>
  <h3>3rd Prize</h3>
  <p></p>
  <ul>
    <li>Arduino Nano (IOT Hardware)</li>
    <li><a href="https://bugdiscover.com" target="_blank">Bug Discover</a> Swag and Stickers</li>
    <li><a href="https://sec.army" target="_blank">SECARMY</a> Swag and Stickers</li>
    <li><a href="https://bugcrowd.com" target="_blank">Bugcrowd</a> Stickers</li>
    <li><a href="https://weare.sec.army/hall-of-fame" target="_blank">Hall of Fame</a></li>
    <li>Digital Certificate</li>
    <li><a href="http://forum.sec.army/" target="_blank">SECARMY Forum</a> 1000 Bytes</li>
  </ul>
  <h3>4th - 10th Prize</h3>
  <p></p>
  <ul>
    <li>Digital Certificate</li>
  </ul>
  <!-- One step closer to prizes: c2VjYXJteXtzMHVyYzNfaTVfbjNjZXM1YXJ5fQo= -->
</p>
</div>
</main>
<footer class="footer">
  <ul>
    <li>SECARMY</li>
    <li>SECARMY</li>
  </ul>
  <small>SECARMY</small>
</footer>
</body>
</html>
```

We have a base64 string: c2VjYXJteXtzMHVyYzNfaTVfbjNjZXM1YXJ5fQo=

decode it: secarmy{s0urc3_i5_n3ces5ary}

Flag: secarmy{s0urc3_i5_n3ces5ary}

Silly Mongolian 2.0:

The most easy and loved challenge:

view source code:

first part of flag



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
  <meta charset="utf-8">
  <title>Silly Mongolian 2.0</title>
  <link rel="stylesheet" href="style.css">
  <link href="https://fonts.googleapis.com/css?family=Ubuntu&display=swap" rel="stylesheet">
  <script type="text/javascript" src="index.js"></script>
  <script type="text/javascript" src="process.js"></script>
</head>
<body>
  <p>Silly Mongolian 2.0</p>
  <form class="login" action="index.html" method="post" >
    <input type="text" name="Username" placeholder="Enter Username" required><br>
    <input type="password" name="Password" placeholder="Enter Password" class="pass" required><br>
    <button type="button" name="Button" class="btn" onclick="Login()"><b>Login</b></button>
  </form>
  <nav>
    <p><b>Challenge by Umair747</b>|<u>Challenge by Umair747</u>|</p>
    <!--Here's the first part of flag : secarmy{why -->
    </nav>
  </body>
</html>
```

Second part is in index.js

```
function returnb() {
    window.open("index.html");
}
const utf8bytes = code => {
    code >>>= 0;
    let array;
    if(code < 0x0080) {
        array = [
            code
        ];
    } else if(code < 0x0800) {
        array = [
            0xC0 | ((code & 0x780) >> 5),
            0x80 | (code & 0x07f)
        ];
    } else if(code < 0x10000) {
        array = [
            0x80 | ((code & 0x0f00) >> 12),
            0x80 | ((code & 0x0f0) >> 6),
            0x80 | (code & 0x003f)
        ];
    } else if(code < 0x200000) {
        array = [
            0x80 | ((code & 0x1c0000) >> 18),
            0x80 | ((code & 0x03f000) >> 12),
            0x80 | ((code & 0x000fc0) >> 6),
            0x80 | (code & 0x00003f)
        ];
    } else {
        throw "undefined code: U+" + code.toString(16);
    }
    return new Uint8Array(array);
}
console.log(flag);
/*Here's the second part of flag :
_is_this_
*/
};
```

Third part is in process.js:

```
function login() {
    window.open("error404.html");
    /*Here's the third part of flag :
    m0ngolian_ such
    */
}
```

```
body{
    background-color: #212376;
    font-family: Ubuntu;
    color: white;
    font-weight: bold;
    text-align: center;
}
.heading{
    font-size: 72px;
}

.pass{
    margin-top: 20px;
}
/*Here's the fourth part of flag :
_a_foot*/
input{
    height: 40px;
    width: 320px;
    border-radius: 18px;
    border: none;
    text-align: center;
    background-color: #181818;
    color: white;
    font-family: Ubuntu;
}
.btn{
    margin-top: 20px;
    height: 30px;
    width: 120px;
    border: none;
    border-radius: 18px;
    font-family: Ubuntu;
}
nav{
    padding-top: 30px;
}
```

Fourth part is in css:

the four parts are:

secarmy{why

_1s_th1s_

m0ng0li@n_Such

_@_f001}

Flag: secarmy{why_1s_th1s_m0ng0li@n_\$uch_@_f00l}

Cookie bank:

View the cookies in console: application:

Name	Value	Domain	Path	Size	HTTP	Secure	SameSite
Cookie1	VGhpcoyBhaW4ndCpdcBjzGlZg==	sec-army.ml	/cookie_bank	35			
Cookie10	U3RhcnRpbnmcZnJvbSB0aGUgbGFzdCBlHNIZS4uLg==	sec-army.ml	/cookie_bank	52			
Cookie2	WW91Hnlob3VszCBwcm9tYYJseSBzdWJzdJhY3QgOC0z	sec-army.ml	/cookie_bank	51			
Cookie3	SSBqdXN0IGldhdmUgeW91IGEgaGlueCBhbHJIYWR5ISE=	sec-army.ml	/cookie_bank	51			
Cookie4	YXJlIHlvdSBzZXJpb3VzPz8/Pw==	sec-army.ml	/cookie_bank	35			
Cookie5	c2VjYXlteXt0aGVtJGh5X2MwMGtpZV93MXRoMW59	sec-army.ml	/cookie_bank	47			
Cookie6	ZZVOIGEgdGtZSBtYWN0aW5lIlvSBuMDBi	sec-army.ml	/cookie_bank	43			
Cookie7	SGV5IHlvdSdyZSBzdGlsbCBhbGI2ZT8=	sec-army.ml	/cookie_bank	39			
Cookie8	U2lyIGZpeCB5b3VylEdQuw==	sec-army.ml	/cookie_bank	31			

Cookie 5:c2VjYXlteXt0aGVfjGh5X2MwMGtpZV93MXRoMW59

base64 decoded: secarmy{the\$_hy_c00kie_w1th1n}

Name	Value	Domain	Path	Size	HTTP	Secure	SameSite
Cookie3	SSBqdXN0IGldhdmUgeW91IGEgaGlueCBhbHJIYWR5ISE=	sec-army.ml	/cookie_bank	51			
Cookie4	YXJlIHlvdSBzZXJpb3VzPz8/Pw==	sec-army.ml	/cookie_bank	35			
Cookie5	c2VjYXlteXt0aGVtJGh5X2MwMGtpZV93MXRoMW59	sec-army.ml	/cookie_bank	47			
Cookie6	ZZVOIGEgdGtZSBtYWN0aW5lIlvSBuMDBi	sec-army.ml	/cookie_bank	43			
Cookie7	SGV5IHlvdSdyZSBzdGlsbCBhbGI2ZT8=	sec-army.ml	/cookie_bank	39			
Cookie8	U2lyIGZpeCB5b3VylEdQuw==	sec-army.ml	/cookie_bank	31			
PHPSESSID	U2lyIHlvdSBuZWVkhRvIHJldGhpbnmsgYWJvdXQgeW91cIBkZ...4e2170e7666d40f08004ea279c78e377	sec-army.ml	/	67			
cfduid	d679bc95f8428d3c3fd9leb8f88761f81566107860	sec-army.ml	/	51	✓	✓	

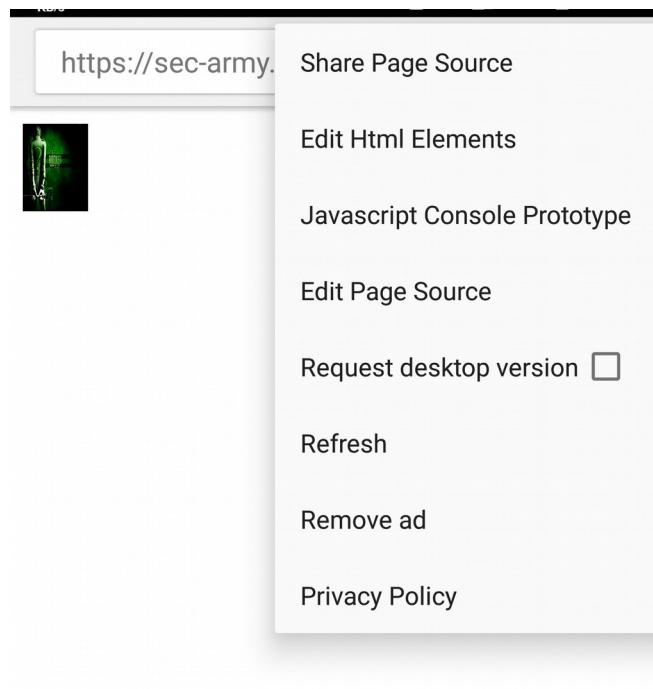
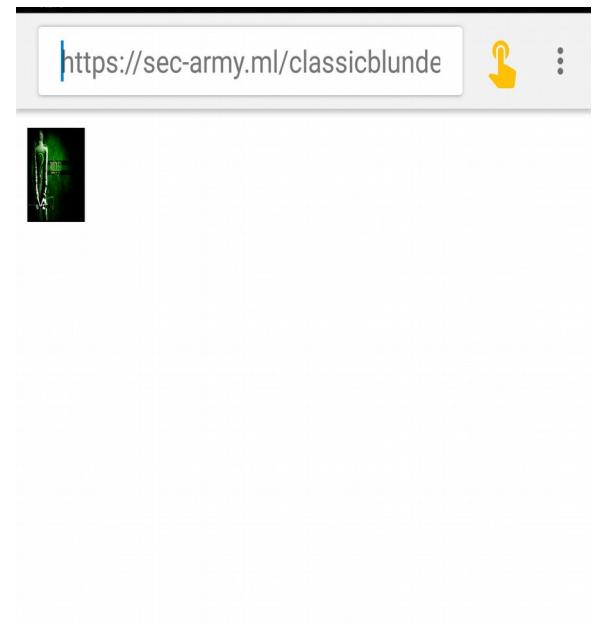
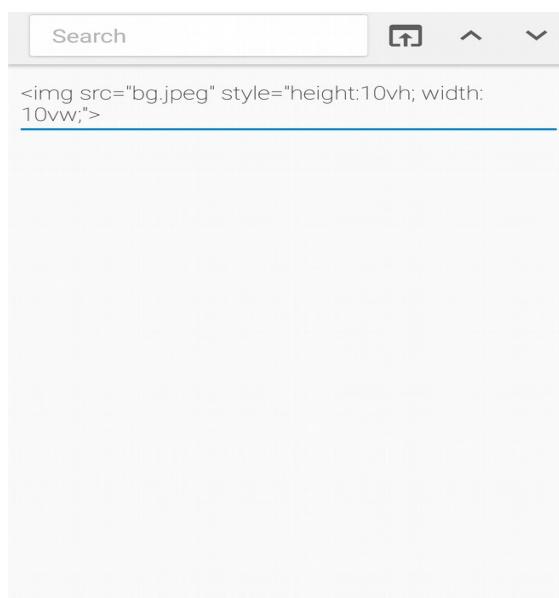
Classic blunder:

Honestly i have no idea what happened and why this happened, i was travelling and dint have access to my laptop, so tried it in my phone using an app, it was total fluke the way i got this.

I dont have any explanation at the time of writing this.

link:<https://apkpure.com/inspect-and-edit-html-live/web.dassem.livewebsiteditorfree>

I present the screenshots from my phone:



What i did :

1. Opened the link: <http://sec-army.ml/classicblunder>
2. opened the inspect element and resized the image from 100vh 100wh to 10vh 10wh
3. opened the source code, and i have the Flag: secarmy{D0M_C110B3R1NG_n_liveOverflow}

CRYPTOGRAPHY

Exchange:

we get a `text.zip` file, unzipping the file we get a `imp_file`.

Well I only know this :-prcsoix{v3hh_ptepq1qtq34}

prcsoix is secarmy

so cipher is quipquip, i had a wild guess for this

link:<https://quipquip.com/>

← → C https://quipqiup.com ☆ 🔍 🌐 Incognito

quipqiup BETA

quipqiup is a fast and automated cryptogram solver by [Edwin Olson](#). It can solve simple substitution ciphers often found in newspapers, including puzzles like cryptoquips (in which word boundaries are preserved) and patristocrats (inwhi chwör dbourn dārie sāren).

Puzzle:
prcssoix{v3hh_ptepqlqtq34}

Clues: For example G=R QWW=THE
prcssoix=secarmy

auto

Solve

0	-2.942	se car my{ i3ll_s of stl to t34}
1	-2.951	se carmy{ o3ff_ sins x1xix34}
2	-2.997	secarmy{i3ll_subst1tut34}
3	-3.065	secarmy{i3ll_suvsk1kuk34}
4	-3.085	secarmy{i3pp_sudst1tut34}
5	-3.107	se carmy{ o3ff_ six splp i p34}
6	-3.130	secarmy{i3ll_software34}

decoding it: secarmy{i3ll_subst1tut34}

Flag:secarmy{w3ll_subst1tut34}

SQUARE:

we get a zip file ThrQrMan.zip having ThrQrMan.png which is a qr code

uploading and scanning it we get:

```
43 15 13 11 42 32 54 { 41 42 _ 25 33 0 52 3_ 44 23 3_ 52 @ 54 }
```

Using polybius cipher we can decode it:

link: cryptii.com:

The screenshot shows the Cryptii interface for decoding. The input text is '43 15 13 11 42 32 54 { 41 42 _ 25 33 0 52 3_ 44 23 3_ 52 @ 54 }'. The output text is 'sec a r m y { q r _ k n 0 w _ o r n_w@y }'. The central panel shows the Polybius square settings: ALPHABET (abcdefghijklmnopqrstuvwxyz), ROWS (12345), COLUMNS (12345), SEPARATOR (empty), CASE SENSITIVITY (Yes No), FOREIGN CHARS (Include Ignore). A message at the bottom says '→ Decoded 47 chars in 0.21ms'.

Flag: secarmy{qr_kn0w_orn_w@y}

Flag Basket:

we have an image flag_basket.png with text in. We use OCR to extract the characters

Ch wjgjonyl mywolcns, u buweyl cm migylyh qbi ziwomym ih mywolcns gywbuhcmgmz iwjgjonyl uhx jfusozf{hyn_qile_msm_nygm}. Qbcfy chwfoxche nbimy qbi yhxyupil ni mnlyhanbyh mowbaln(gywbl_u_hmcg), cn cni gillyznyh omyx vs nbv gummm gyxcu uhx jjoful alyunliwof noly) ni lyzyl ni nbimy qbi mye uwyyymm xymjcn ybymy mywolcns gymolym. Nbun cm, nby gyxcu jihlusm nby 'buweyl' urn u pcftuch. xisioeg(Hhypyl_nby_fymm), julnm iz nby momwofnoly myy nbycl ucg ch willywncha mywolcns hinbyly(jii_vfy_gm) uhx omry nby qlx ch u jmcncpy myhmy. Qbcny bun cm nby hugy acpyh ni ynbewlf wjgjonyl buweylm, qbi oncfcty buwecha ch u eyimnvs(byfj_zof) qus. Qbcny buncu uly ywyigcha u hywymnuls juln iz nby chzilguncih mywolcns zcyfx. Gimn cijinluhnfs, byly cm siol zfuia mywulgsl{WL_c5_fcn} Nbys ijyluny ohxyl u wixy, qbcwb uwehiqfyaxm nbun vlyuecha chni inbyl jyifjy'm wjgjonylm cm vux,

Registration will give you access to additional features not available to guest users: convert multipage PDF (more 15 pages), large images and ZIP archives, choose recognition languages, convert into editable formats and other settings. [SIGN UP](#)

It is cipher text, which is bruteforced in dcode.fr for substitution cipher,

it is a shift of 20 alphabets and we get our flag

The screenshot shows the dcode.fr Caesar Cipher tool interface. On the left, there's a sidebar with a 'Results' section containing a message from the hacker about security measures. The main area has two tabs: 'Caesar Cipher Decoder' and 'Caesar Encoder'. Under 'Decoder', the text 'secarmy{OCR_i5_lit}' is entered, and the shift is set to 20. The 'DECRYPT CAESAR CODE' button is highlighted. To the right, a sidebar lists various cipher-related topics like 'How to encrypt using Caesar cipher?' and 'How to decrypt Caesar cipher?'. Below the sidebar, there's a 'Similar tools' section with links to ROT Cipher, Shift Cipher, Vigenere Cipher, etc.

Flag: secarmy{OCR_i5_lit}

Exclusive OR Non-Exclusive:

We get a weird string:

190f090b18071311125a1835035f35080b5f0309350c5a183559040918131a1e035a045f17

Name has it XOR,

trying few different keys I did not get anything , but the dcode.fr site gave me an hint.

“USE THE BINARY KEY” has 0 and 1 so I just thought of using iiii (i*8) in place of 1, did not work out but i tried jjjjjjjj (j*8) i got the key, Flag: secarmy{x0r_i5_ba5ic_f0r_3ncrypti0n5}

The screenshot shows the dcode.fr XOR Cipher tool interface. A search bar at the top finds 'XOR Cipher'. The main area has tabs for 'XOR Decoder' and 'XOR Encoder'. In 'Decoder', the string '190f090b18071311125a1835035f35080b5f0309350c5a183559040918131a1e035a045f17' is entered, and the mode is set to 'TEXT TO XOR'. The 'TEST ALL KEYS FROM 1 TO 8 BITS' button is checked. To the right, a sidebar lists topics like 'XOR Decoder', 'XOR Operation', and 'How to encrypt using XOR cipher?'. Below the sidebar, there's a 'Similar tools' section with a link to 'ASCII Code'.

The 3ASY:

This challenge was right on your face but u missed it,

use <https://cryptii.com> and <https://v2.cryptii.com> and <https://dcode.fr> for this

we get a string which is combination of number letters and morse code:

25111111251111251111111111251111111111112511111111112511ndaekjaadadaabaaaaajmadebab.-

Decoding the number: polybius square

251111125111251111111125111111111125111111111251111111112511 =>

kaaakaakaaaaakaaaaakaaaaaka

Decoding the alphabets: trifid cipher

ndaekjaadadaabaaaaajmadebab => kkkakkaaaaakaaaaaaaaaakaakkak

Decoding morse code:

..... = >

akaakakaakakaakaaaaaaaakk

combining all together

kaaakaakaaaaakaaaaakakkkakkaaaaakaaaaaaaaaakaakkaakakaakakaakaaaaaaaakk

It is bacon cipher, decoding it: SECARMYBACNWLLLAD

Flag: secarmy{bacnwlllad} does not make any sense, but that is the flag (-_-),

I know it hurt you guys spent a lot of time, but this was it.

The screenshot shows a web browser window with the URL <https://www.dcode.fr/bacon-cipher>. The page title is "BACON CIPHER". Below it, the navigation path is "Cryptography > Substitution Cipher > Bacon Cipher". A sidebar on the right is titled "Summary" and lists several related links. The main content area contains a search bar for tools, a search form for keywords, and a results section showing two examples of Baconian cipher decoding. One example shows the plaintext "SECARMYBACNWLLAD" corresponding to the ciphertext "A=A, B=K (αβ1)". Another example shows the plaintext "RECAQLWBACMUKKKAD" corresponding to the ciphertext "A=A, B=K (αβ2) 0??PUJ??7TLVV??".

Search for a tool

* SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type random

Results

A=A, B=K (αβ1) SECARMYBACNWLLAD
A=A, B=K (αβ2) RECAQLWBACMUKKKAD
A=K, B=A (αβ2) 0??PUJ??7TLVV??
A=K, B=A (αβ1) P??QWK??UMXXX??

DECRYPT BACON

Rivest Shamir Adelman:

We have 3 files, the encryption file, encrypted image and the private key,
the code used for it:

<https://pastebin.com/vwYqhbP>

use this code and run: python2 <filename.py>

you have the decoded image:



Flag:secarmy[RSA_ba51c5tobe_13arn7]

Old Days:

only 90's kids will remember this XD

The numbers correspond to the number of times it has to be pressed to get the letters in the phone
which was before smartphone era,

The pain of typing long text.... phew.

Well lets solve:

```
File Edit Search Options Help
1 777 33 6 33 6 22 33 777 0 8 44 666 7777 33 0 666 555 3 0 3 2 999 7777 0 9 44 33 66 0 9 33 0 55 33 33 7 0 666 66 0 7777 6 2 7777
44 444 66 4 0 666 88 777 0 55 33 999 7 2 3 0 5 88 7777 8 0 8 666 0 7777 33 66 3 0 2 0 7777 444 66 4 555 33 0 6 33 7777 7777 2 4
33 0 9 33 555 555 0 555 33 8 7777 0 777 33 555 444 888 33 0 666 88 777 0 666 555 3 0 4 666 555 3 33 66 0 3 2 999 7777 0 9 444 8
44 0 8 33 2 0 2 66 3 0 7777 666 6 33 0 333 555 2 4 7777 0 7777 33 222 2 777 6 999 777 33 555 444 888 33 8 44 33 666 555 3 3 2 999
7777 7777 33 222 2 777 6 999 0 7777 666 0 8 44 33 0 333 555 2 4 0 444 7777 0 8 44 33 0 7777 8 777 444 66 4 0 22 33 8 9 33 33 66 0
8 44 33 0 7777 33 222 2 777 6 999 0 9 44 444 222 44 0 444 7777 0 777 33 555 444 888 33 8 44 33 666 555 3 3 2 999 7777 0 44 666 7
33 0 999 666 88 0 2 777 33 0 66 666 8 0 222 666 66 333 88 7777 33 3 0 4 666 666 3 0 555 88 222 55
```

each number represents the alphabet a converted format looks like this:



```
1777 e m e m b e r 0 t h o s e 0 o l d 0 d a y s 0 w h e n 0 w e 0 k e 33 p 0 o n 0 s m a s h i n g 0 o u r 0 k e y p a d 0 j u s
t 0 t o 0 s e n d 0 a 0 s i n g l e 0 m e s 7777 a g e 0 w e l 555 0 l e t s 0 r e l i v e 0 o u r 0 o l d 0 g o l d e n 0 d a y
s 0 w i t h 0 t e a 0 a n d 0 s o m e 0 f l a g s 0 s e c a r m y r e l i v e t h e o l d d a y s . s e c a r m y 0 s o o 0 t h e 0 f
l a g 0 i s 0 t h e 0 s t r i n g 0 b e t w e 33 n 0 t h e 0 s e c a r m y 0 w h i c h 0 i s 0 r e l i v e t h e o l d 3 a y s 0
h o p e 0 y o u u 0 a r e 0 n o t 0 c o n f u s e d 0 g o 666 d 0 l u c 55
2
3 secarmy{relivetheolddays}
4
5
6
```

Flag: secarmy{relivetheolddays}

BINARY/REVERSING

Smash it:

classic buffer flow:

enter a very long string, and you are done XD

Flag: secarmy{sm@sh1ng_st@ck_1s_t00_much_fun}

if you run ltrace you can see there is strncmp()

```
strncmp("START", "START", 5)
```

It is checking for a limit of 5 chars,

you enter a lengthy string you bypass it and you get the flag.

Backyard COWs:

we have a elf file called moo

we use radare2 here, a crazy tool to do so

Flow:

1. r2 moo => load the file
 2. aa => to start analyzing the binary
 3. s => main to start analyzing from main
 4. p => to change the view mode

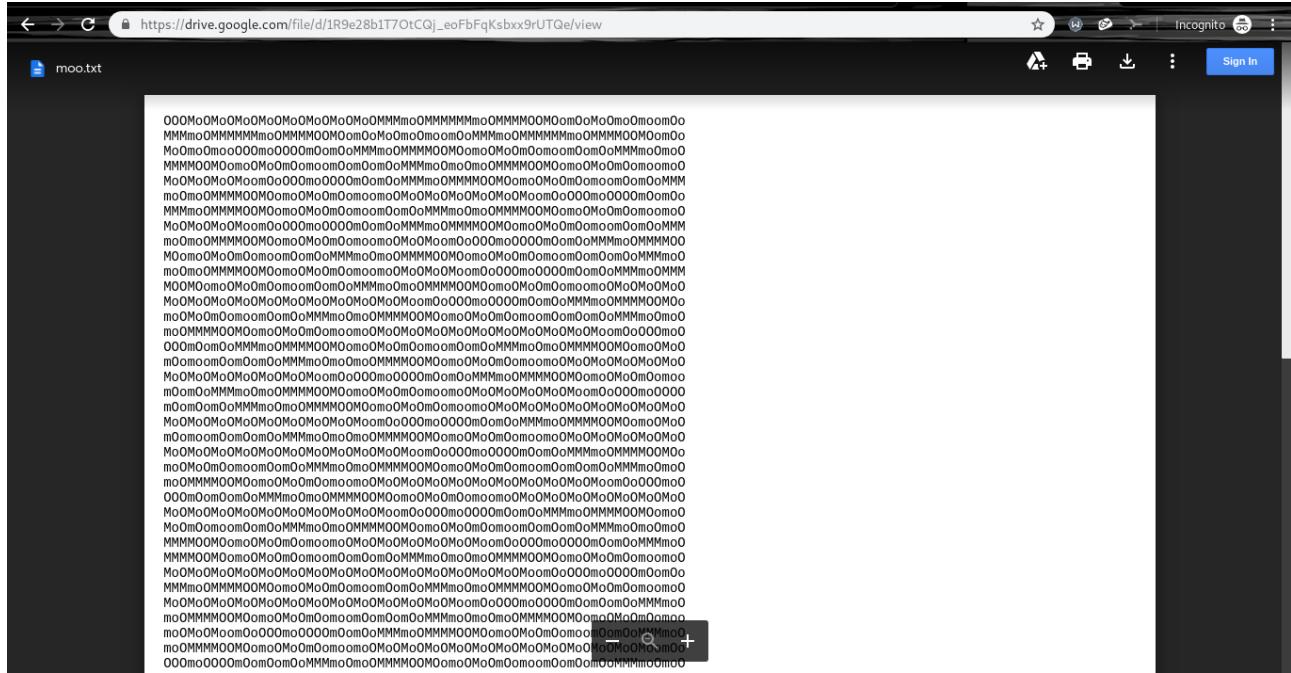
5. scroll down a bit and you will find a strange chars:

```
File Edit View Search Terminal Tabs Help
ross@ross46: ~/my/ctf/secarmy
[0x00001175 [xAdvc] 26% 215 moo]> pd @ main+16 # 0x1175
0x00001175    48c745db0000. mov qword [local_20], 0
0x0000117d    48c745e00000. mov qword [local_20], 0
0x00001185    c645e800    mov byte [local_10], 0
0x00001189    c645d008    mov byte [local_08], 0x68 ; 'h'
0x0000118d    c645d174    mov byte [local_1f], 0x74 ; 't'
0x00001191    c645d274    mov byte [local_20], 0x74 ; 't'
0x00001195    c645d370    mov byte [local_20], 0x70 ; 'p'
0x00001199    c645d43a    mov byte [local_30], 0x3a ; ':'
0x0000119d    c645d52f    mov byte [local_20], 0x2f ; '/'
0x000011a1    c645d62f    mov byte [local_20], 0x2f ; '/'
0x000011a5    c645d762    mov byte [local_20], 0x62 ; 'b'
0x000011a9    c645d869    mov byte [local_20], 0x69 ; 'i'
0x000011ad    c645d974    mov byte [local_27], 0x74 ; 't'
0x000011b1    c645da2e    mov byte [local_26], 0xe ; '.'
0x000011b5    c645dbe6    mov byte [local_25], 0x6c ; 'l'
0x000011b9    c645dc79    mov byte [local_24], 0x79 ; 'y'
0x000011bd    c645dd2f    mov byte [local_23], 0x2f ; '/'
0x000011c1    c645de6d    mov byte [local_22], 0x6d ; 'm'
0x000011c5    c645df30    mov byte [local_21], 0x30 ; '0'
0x000011c9    c645e030    mov byte [local_20], 0x30 ; '0'
0x000011cd    c645e15f    mov byte [local_1f], 0x5f ; '_'
0x000011d1    c645e26d    mov byte [local_1e], 0x6d ; 'm'
0x000011d5    c645e330    mov byte [local_1d], 0x30 ; '0'
0x000011d9    c645e430    mov byte [local_1c], 0x30 ; '0'
0x000011dd    488d3d240e00. lea rdi, qword str.    ; 0x2008 ; * ; rdi=0x2008 -> 0x5f5f5f20
0x000011e4    e857feffff    call sym.imp.puts      ;[] ; int puts(const char *) ; rsp=0xfffffffffffffff8 ; rip=0x1040 -> 0x2fdad2
0x000011e9    488d3d380e00. lea rdi, qword str.moo_select_your_language ; 0x2028 ; < moo! select your language! > ; rdi=0x2028 -> 0x2047 ; -----
0x000011f0    e84bfeffff    call sym.imp.puts      ;[] ; int puts(const char *) ; rsp=0xfffffffffffffff8 ; rip=0x1040 -> 0x2fdad2
0x000011f5    488d3d4b0e00. lea rdi, qword str.    ; 0x2047 ; ----- ; rdi=0x2047 -> 0x2d2d2d20
0x000011fc    e83ffeffff    call sym.imp.puts      ;[] ; int puts(const char *) ; rsp=0xfffffffffffffff8 ; rip=0x1040 -> 0x2fdad2
0x00001201    488d3d5d0e00. lea rdi, qword str.    ; 0x2065 ; _ \ ^ _ ; rdi=0x2065 -> 0x20202020
0x00001208    e833feffff    call sym.imp.puts      ;[] ; int puts(const char *) ; rsp=0xfffffffffffffff8 ; rip=0x1040 -> 0x2fdad2
0x0000120d    488d3d620e00. lea rdi, qword str.oo  ; 0x2076 ; _ \ ^ _ (oo)\____ ; rdi=0x2076 -> 0x20202020

```

http://bit.ly/m00_m00

leads you to a google drive containing some text in cow lang



now we need to decode it,

Link: <https://tio.run>

choose cow and decode it

The screenshot shows a terminal window on the TIO.run website. The URL in the address bar is <https://tio.run/#cow>. The terminal output displays a large amount of encoded text consisting mostly of 'M' and 'O' characters. Below the terminal, there are several sections: 'Footer', 'Input', 'Arguments', 'Output' (which contains the flag), and 'Debug'. The 'Debug' section shows performance metrics: Real time: 0.016 s, User time: 0.007 s, Sys. time: 0.007 s, CPU share: 37.87 %, and Exit code: 0.

```
Mo0m0oomOomOoMMmo0mo0MMMO0Oomo0Mo0m0oomo0Mo0Mo0Mo0Mo0Mo0  
Mo0Moom0o000mo000m0om0o0MmMo0mo0MMMO0Mo0mo0Mo0m0oom0o0m0o  
MMmMo0mo0mo0MMMO0Mo0mo0Mo0m0omo0Mo0Mo0Mo0m000mo000m0m0o  
MMmMo0MMMO0Mo0mo0Mo0m0omo0m0o0m0o0MmMo0mo0mo0MMMO0Mo0mo0  
moomoMo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0000  
m0m0oMMMo0MMMO0Mo0mo0Mo0m0omo0m0o0m0o0MmMo0mo0mo0MMMO0mo0  
Mo0m0omo0mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0000  
mo000m0o0MMMo0MMMO0Mo0mo0Mo0m0omo0m0o0m0o0MmMo0mo0mo0MMMO0  
MoomoMo0mo0mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0  
m0o00m0000m0o0MmMo0MMMO0Mo0mo0Mo0m0omo0m0o0m0o0MmMo0mo0mo0  
MMMM0Mo0mo0Mo0m0omo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0  
Mo0Mo0mo0000m0o0m0o0MmMo0mo0MMMO0Mo0mo0Mo0m0omo0m0o0m0o0  
MMMo0mo0mo0MMMO0Mo0mo0Mo0m0omo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0  
Mo0Mo0Mo0Mo0Mo0m0000m0o0m0o0MmMo0MMMO0Mo0mo0Mo0m0omo0m0o0  
m0oMMMo0mo0MMMO0Mo0mo0Mo0m0omo0m0o0m0o0MmMo0mo0mo0MMMO0mo0  
Mo0m0omo0mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0Mo0m0o0m0o0
```

- ▶ Footer
- ▶ Input
- ▶ Arguments
- ▼ Output
- secarmy{d0_y0u_l1k3_c0w_languag3____?}
- ▼ Debug

```
Real time: 0.016 s
User time: 0.007 s
Sys. time: 0.007 s
CPU share: 37.87 %
Exit code: 0
```

Flag: secarmy{d0_y0u_l1k3_c0w_languag3____?}