

ASGAMA KOMATIK

CTF 25

BAYU PUTRA IBANA

24/536830/PA/22776



ASGAMA Questions

1. Forensics

Nama ↑

☰ LOGO_ASGAMA.zip 👤

In the Forensics folder, there is a zip file, first i will download it.
Once done, I tried to unzip it:

```
(kali㉿kali)-[~/Downloads]
$ unzip LOGO_ASGAMA.zip
```

```
(kali㉿kali)-[~/Downloads]
$ unzip LOGO_ASGAMA.zip
Archive:  LOGO_ASGAMA.zip
[LOGO_ASGAMA.zip] LOGO_ASGAMA.png password: 
```

Turns out you need a password for it.
Let's analyze the file using exiftool:

```
(kali㉿kali)-[~/Downloads]
$ exiftool LOGO_ASGAMA.zip
ExifTool Version Number      : 12.76
File Name                    : LOGO_ASGAMA.zip
Directory                    : .
File Size                    : 327 kB
File Modification Date/Time   : 2024:11:19 03:19:48-05:00
File Access Date/Time        : 2024:11:19 03:20:50-05:00
File Inode Change Date/Time   : 2024:11:19 03:19:48-05:00
File Permissions              : -rw-rw-r--
File Type                    : ZIP
File Type Extension          : zip
MIME Type                    : application/zip
Zip Required Version          : 20
Zip Bit Flag                  : 0x0001
Zip Compression               : Deflated
Zip Modify Date               : 2024:11:15 22:01:56
Zip CRC                       : 0x2f40304c
Zip Compressed Size           : 327092
Zip Uncompressed Size         : 330738
Zip File Name                 : LOGO_ASGAMA.png
```

There is a png there, but still no other info.

Maybe we have to guess, or crack the password.

I decided to use **JohnTheRipper**.

```
(kali@kali)-[~/Downloads]
$ zip2john LOGO_ASGAMA.zip > hash.txt
Created directory: /home/kali/.john
ver 2.0 LOGO_ASGAMA.zip/LOGO_ASGAMA.png PKZIP Encr: cmplen=327092, decmplen=330738, crc=2F40304C ts=B03C cs=2f40 type=8
```

First extract the hash using zip2john to a text file.

Then use the rockyou.txt, which is a text file containing passwords, and john will crack it.

```
(kali@kali)-[~/Downloads]
$ john --wordlist=rockyou.txt hash.txt

Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
No password hashes left to crack (see FAQ)
```

Then use john --show to show the cracked password

```
(kali@kali)-[~/Downloads]
$ john --show hash.txt
LOGO_ASGAMA.zip/LOGO_ASGAMA.png:popsicles:LOGO_ASGAMA.png:LOGO_ASGAMA.zip::LOGO_ASGAMA.zip
1 password hash cracked, 0 left
```

It says : popsicles, let's try that

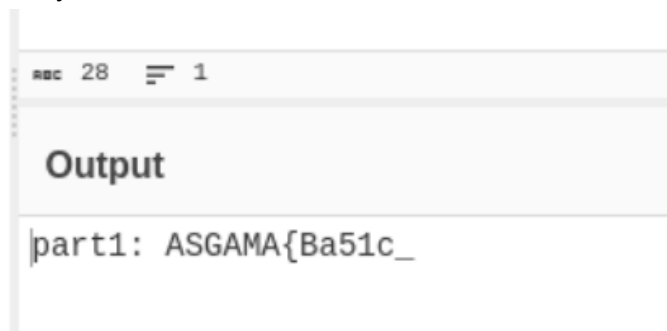
```
(kali@kali)-[~/Downloads]
$ unzip LOGO_ASGAMA.zip
Archive: LOGO_ASGAMA.zip
[LOGO_ASGAMA.zip] LOGO_ASGAMA.png password:
  inflating: LOGO_ASGAMA.png
```

It works, now we analyze the png file to see if the flag is there.

```
(kali@kali)-[~/Downloads]
$ exiftool LOGO_ASGAMA.png
ExifTool Version Number      : 12.76
File Name                    : LOGO_ASGAMA.png
Directory                   : .
File Size                    : 331 kB
File Modification Date/Time   : 2024:11:15 22:01:56-05:00
File Access Date/Time        : 2024:11:19 03:58:32-05:00
File Inode Change Date/Time   : 2024:11:19 03:58:31-05:00
File Permissions              : -rw-rw-r--
File Type                    : PNG
File Type Extension          : png
MIME Type                    : image/png
Image Width                  : 962
Image Height                 : 948
Bit Depth                    : 8
Color Type                   : RGB with Alpha
Compression                  : Deflate/Inflate
Filter                       : Adaptive
Interlace                    : Noninterlaced
sRGB Rendering                : Perceptual
Warning                      : [minor] Text/EXIF chunk(s) found after PNG IDAT (may be ignored by some readers)
Exif Byte Order               : Big-endian (Motorola, MM)
X Resolution                  : 72
Y Resolution                  : 72
Resolution Unit               : inches
Y Cb Cr Positioning           : Centered
Exif Version                  : 0232
Components Configuration     : Y, Cb, Cr, -
User Comment                  : cGFydDE6IEFTR0FNQXtCYTUxY18K
Flashpix Version              : 0100
Color Space                   : Uncalibrated
Image Size                   : 962x948
Megapixels                   : 0.912
```

There is a comment for the user, I think that can be decoded.

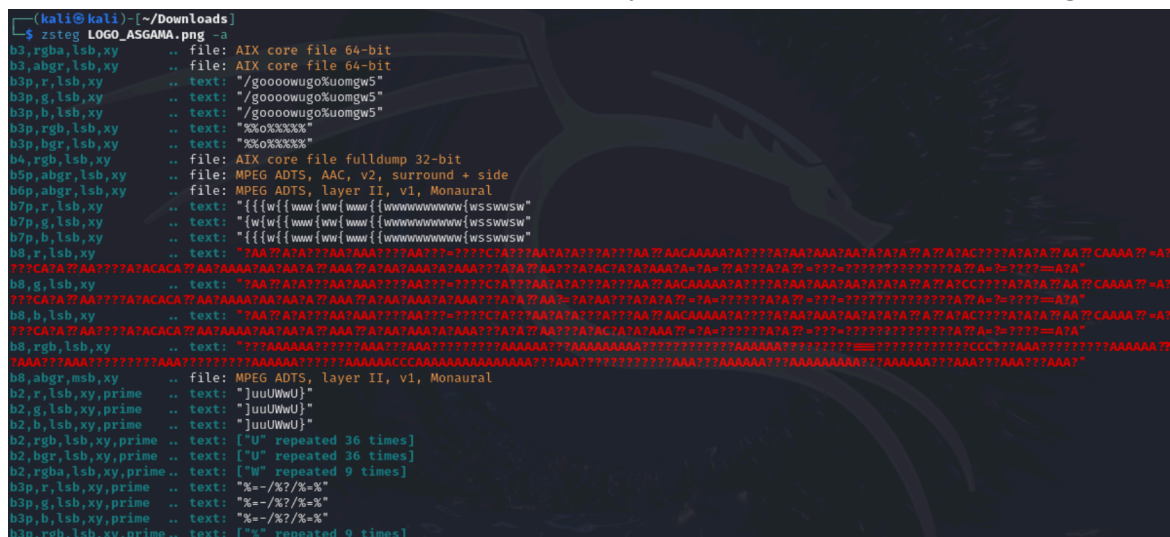
I used cyberchef to decode it, I tried from Base64 first and got it correct:



Turns out it is only 1 part of the flag: **ASGAMA{Ba51c_**

Now we have to find the other parts of the flag.

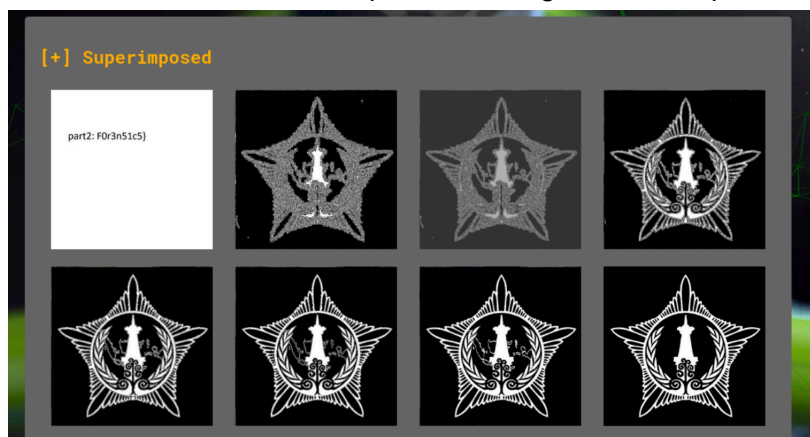
I tried to use other tools to see if i can extract any other hidden data, such as **zsteg**



But I didn't find anything.

Then next I uploaded the image to Aperi'Solve to look for other clues.

Then I stumbled on the 2nd part of the flag, which completes it.



So the full flag is : **ASGAMA{Ba51c_ F0r3n51c5}**

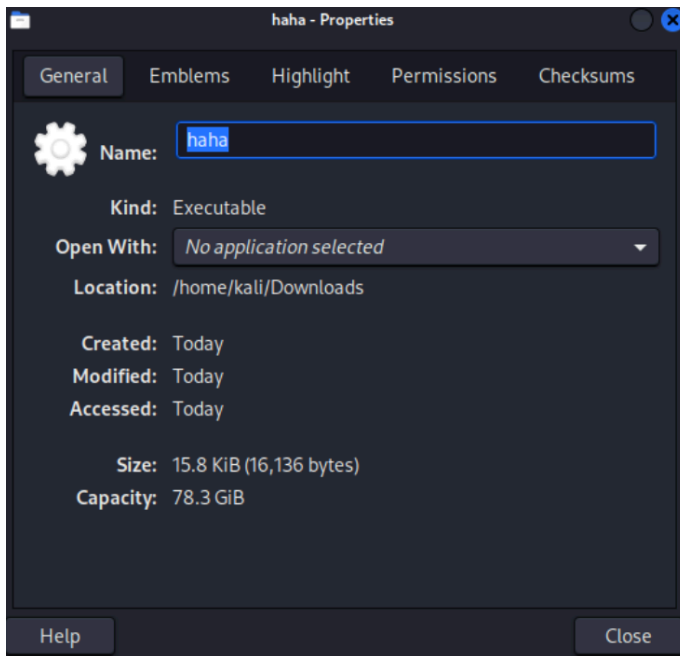
2. haha

Magang CTF Oprec AS... > Reverse Engineering

Nama ↑

haha

In the Reverse Engineering file, we can download a file , haha.



It is an executable file, so I tried running it.

```
(kali@kali)-[~/Downloads]
$ ./haha
zsh: permission denied: ./haha
```

Since the permission is denied, I figured out that you need to give it executable permissions.

```
(kali@kali)-[~/Downloads]
$ chmod +x haha
```

Then, I ran the program file.

```
(kali@kali)-[~/Downloads]
$ ./haha
Here is a huge number: 65l83l71l65l77l65l123l84l48l111l95l51l122l95l88l48l82l95l121l48l117l95l72l117l104l49l63l125l
```

We get a huge number, maybe this can be decoded.

After trying a few decodes, like Base64 or XOR, and other ciphers, but I did not find the flag. However i noticed that there is a '1' separating the numbers, by removing that maybe i can decode it into ASCII using from decimal, So in cyberchef there is a 'split' that you can use to remove the delimiter of '1' and then combine the recipe with From Decimal to decode it.

The screenshot shows the CyberChef web application interface. On the left, the 'Recipe' panel contains two steps: 'Split' and 'From Decimal'. The 'Split' step has a 'Split delimiter' set to '1'. The 'From Decimal' step has a 'Delimiter' set to 'Space' and the 'Support signed values' checkbox is unchecked. At the bottom of the recipe panel is a green 'BAKE!' button. On the right, the 'Input' panel contains a long string of numbers separated by '1's: 65183171165177165112318414811111951511122195188148182195112114811171951721117110414916311251. Below the input, the 'Output' panel displays the decoded result: ASGAMA{T0o_3z_X0R_y0u_Huh1?}. The interface also includes a 'STEP' indicator, an 'Auto Bake' checkbox, and a status bar at the bottom showing 'rec 28', '1ms', and 'LF (detected)'.

There you go, we found the flag:

ASGAMA{T0o_3z_X0R_y0u_Huh1?}