

Smiling ASCII CTF Writeup

This document is a walkthrough on one way to solve the **Smiling ASCII CTF** on **CTFLearn**. The objective is to explain how I was able to solve this CTF to my future self.

General Information

- *Difficulty:* **Medium**
- *Category:* Forensics (Steganography)
- *Link:* [Challenge - Smiling ASCII - CTFLearn - CTF Practice](#)

Introduction

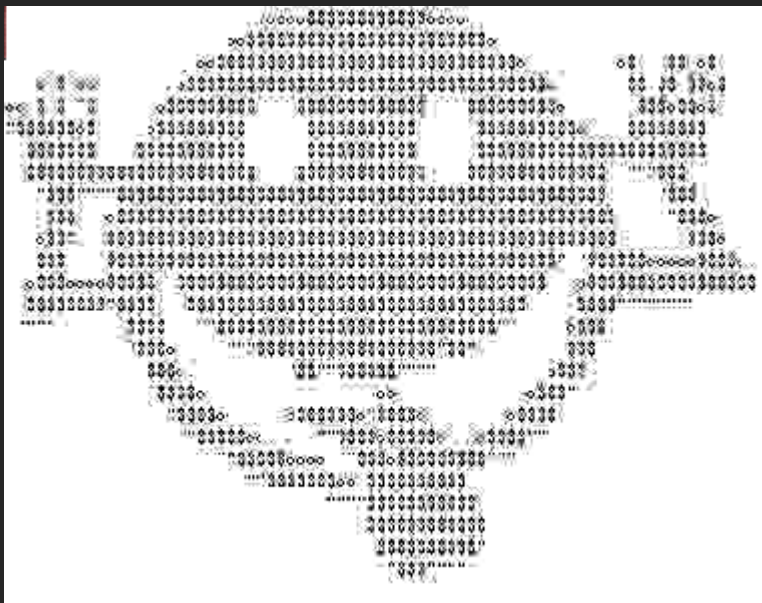
Smiling ASCII ✓

40 points **Medium**

Find the flag on the smiling face.

smiling.png

We're given a PNG file, which looks like this:



First and foremost, let's do a simple strings command on the file to see if there's anything hidden:

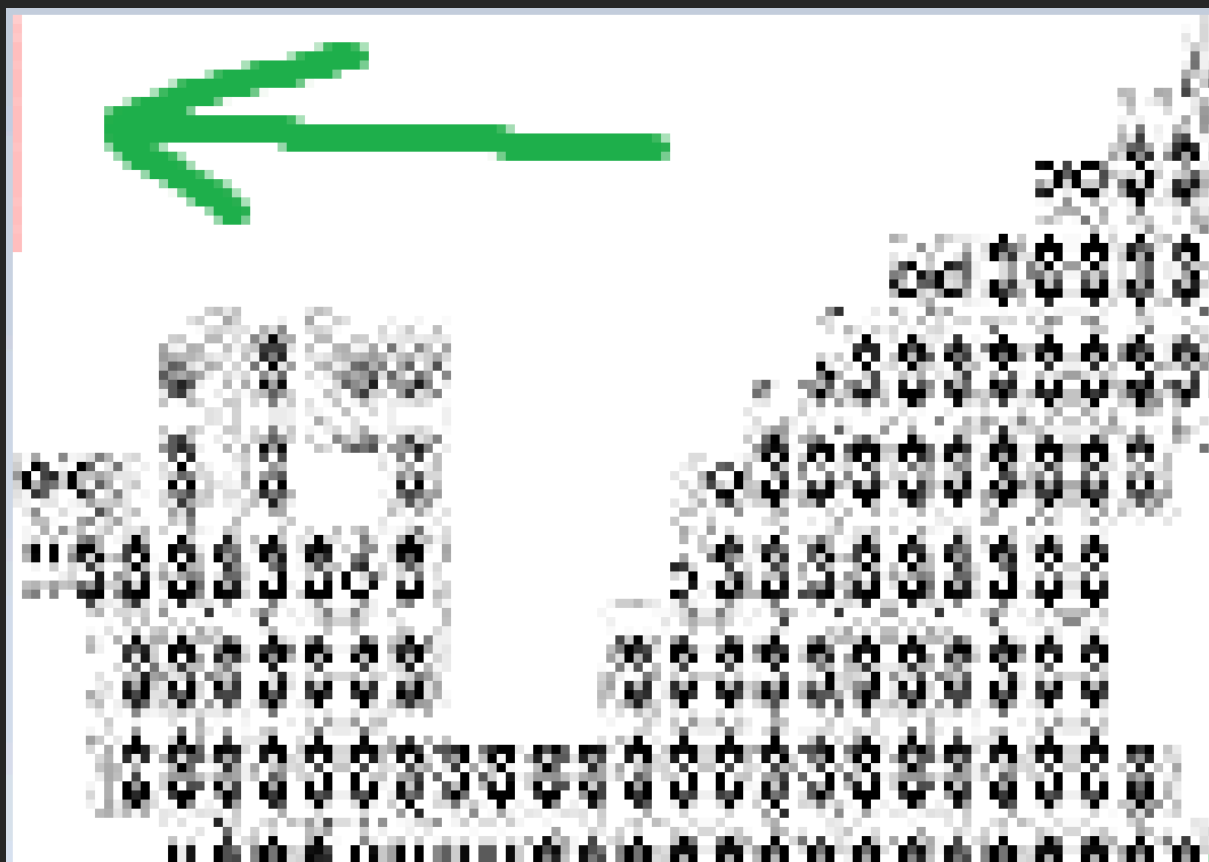
```
alexandre@vbox: ~/Downloads
$ strings smiling.png | tail
eddd:
eddd:
/###
_FFF
2222
eddd:
/###
, />I
IEND
RGlkIHlvdSBrbm93IHRobXQgcGl4ZWxzIGFyZSwgbGlrZSB0aGUgYXNjaWkgdGFibGUuIG51bWJl
cmVkaWZyb20gMCB0byAyNTU/Cg==
```

It looks like base64-encoded string was added at the end of the file

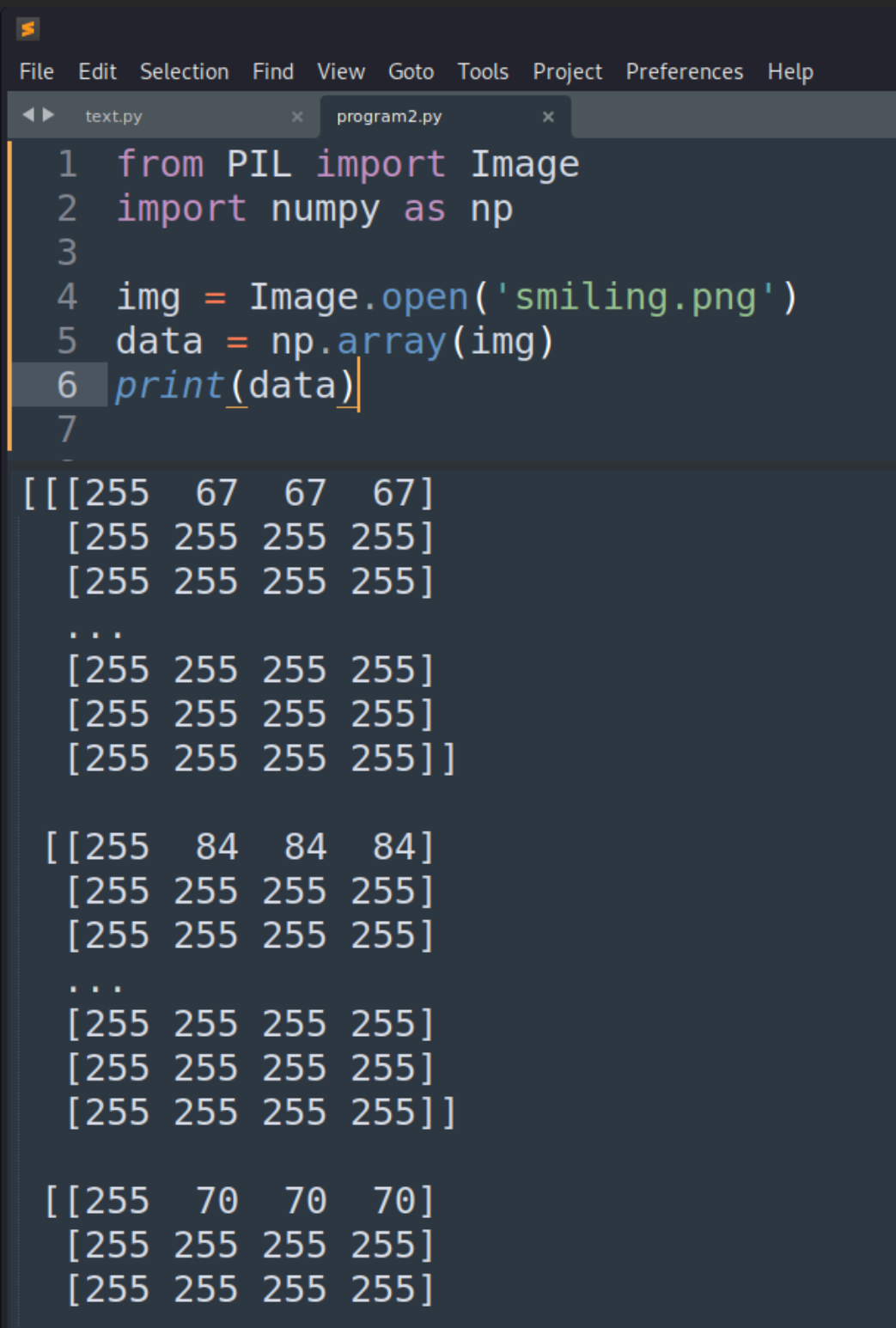
```
(alexandre@vbox)-[~/Downloads]
$ echo RGlKIHLvdSBrbm93IHRoYXQgcGl4ZWxzIGFyZSwgbGlrZSB0aGUgYXNjaWkgdGFibGU
sIG51bWJlcmVkIGZyb20gMCB0byAyNTU/Cg= | base64 --decode
Did you know that pixels are, like the ascii table, numbered from 0 to 255?
```

It looks like we're going to have to take a closer look at the pixels on the image

Effectively, upon closer look, red pixels can be seen at the top left of the image, which suggests that information has been added to the original image, perhaps the flag.



Using the PIL (for pillow) library in Python, we're able to convert the image into an array of numbers, each indicating the respective RGBA value of each pixel. The output confirms that the Green, Blue, and Alpha values of the first few pixels in the first row of the image are different to the others, perhaps their values are hiding a message ?

A screenshot of a Python IDE with a dark theme. The top menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. Below the menu bar, there are two tabs: 'text.py' and 'program2.py'. The 'program2.py' tab is active and shows the following Python code:

```
1 from PIL import Image
2 import numpy as np
3
4 img = Image.open('smiling.png')
5 data = np.array(img)
6 print(data)
7
```

The code is executed, and the output is displayed in a separate pane below the editor. The output shows a 3x4 array of values, representing the first three rows of the image's data. Each row starts with a 255 value (likely the Alpha channel), followed by three other values. The first row has values 67, 67, 67. The second and third rows have values 255, 255, 255. This pattern repeats for the next two rows, with the first row of each group having different values (84, 84, 84 and 70, 70, 70) and the subsequent rows having 255s.

```
[[255  67  67  67]
 [255 255 255 255]
 [255 255 255 255]
 ...
 [255 255 255 255]
 [255 255 255 255]
 [255 255 255 255]]

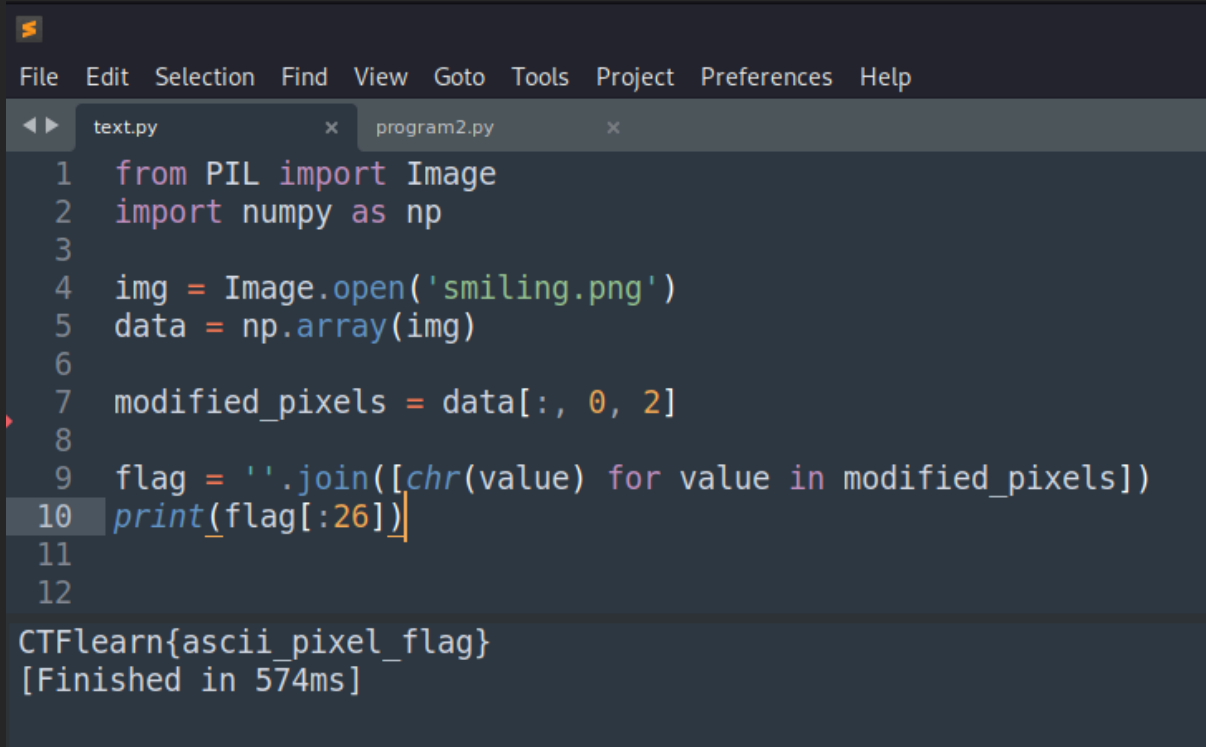
[[255  84  84  84]
 [255 255 255 255]
 [255 255 255 255]
 ...
 [255 255 255 255]
 [255 255 255 255]
 [255 255 255 255]]

[[255  70  70  70]
 [255 255 255 255]
 [255 255 255 255]]
```

Let's extract the blue value of each pixel in the row of the image (the same can be done for green and alpha values), we get the following array:

[illegible]

And there you have it, by converting the first 26 values of the array into ASCII characters, we find the following flag: **CTFlearn{ascii_pixel_flag}**.



```
File Edit Selection Find View Goto Tools Project Preferences Help
text.py x program2.py x
1 from PIL import Image
2 import numpy as np
3
4 img = Image.open('smiling.png')
5 data = np.array(img)
6
7 modified_pixels = data[:, 0, 2]
8
9 flag = ''.join([chr(value) for value in modified_pixels])
10 print(flag[:26])
11
12
CTFlearn{ascii_pixel_flag}
[Finished in 574ms]
```