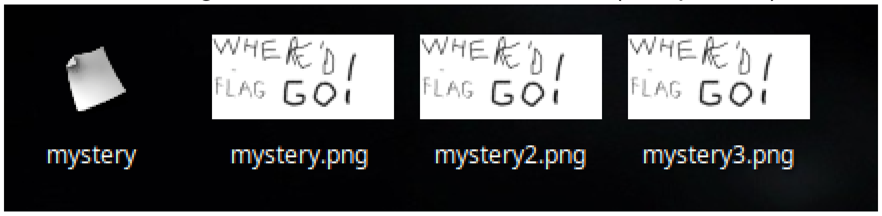
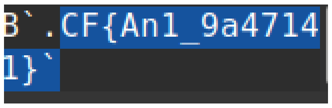
Mark Seliternikov
picoCTF - Investigative Reversing 1 **[350 points]**

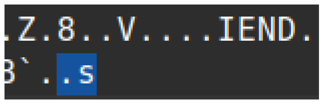
For this challenge we receive 4 files, 1 ELF file (compiled C) and 3 PNG files.



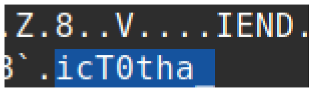
The challenge explained that both forensics and reversing are required to solve this challenge.
For that reason I have opened the PNG files in a hex editor to view them.
I know that a PNG file ends with the bytes 49 45 4E 44 AE 42 60 82 (IEND).



```
B`.CF{An1_9a4714
1}`
```



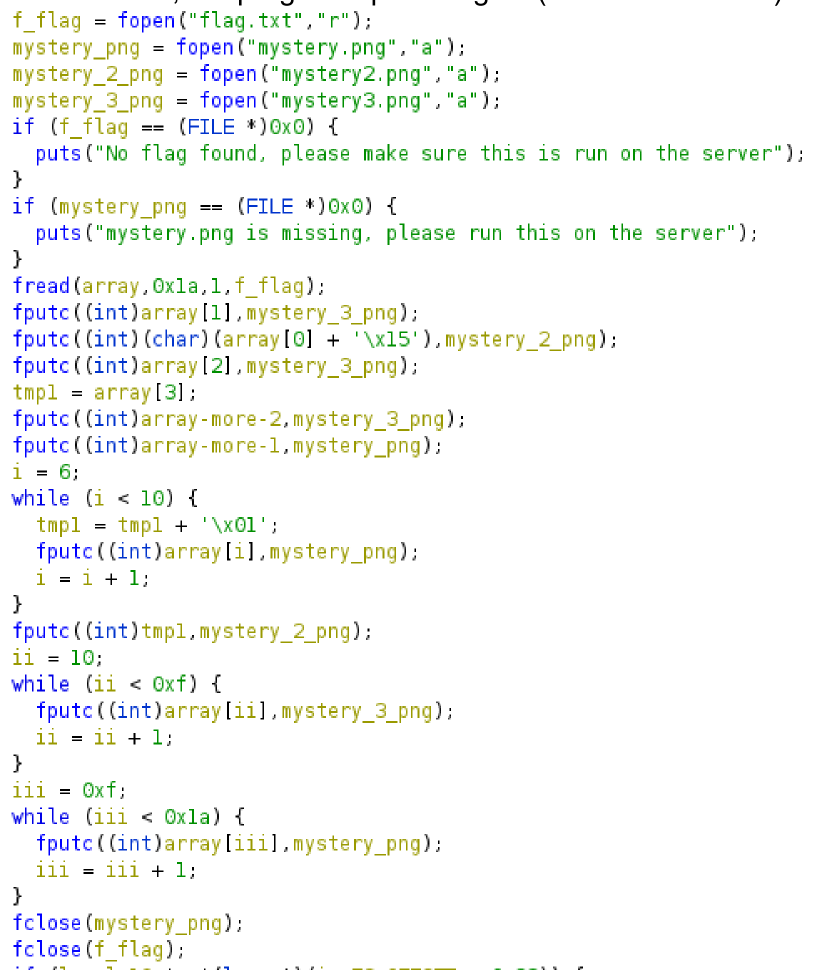
```
.Z.8..V....IEND.
B`..s
```



```
.Z.8..V....IEND.
B`.icT0tha_
```

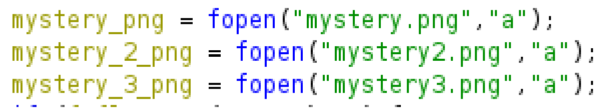
(mystery2 had special ASCII value so to ease on the solution I handled it in hex)

After finding parts that looked like they belong to the flag (a part from some missing letters like the p in pico)
I reversed the ELF file with ghidra to see what is it's purpose (most likely write the flag to the end of the image)
So after reversing I have changed some variable names to ease the reading.
As I found out, the program opens flag.txt (which i dont have) and writes it's contents to the images



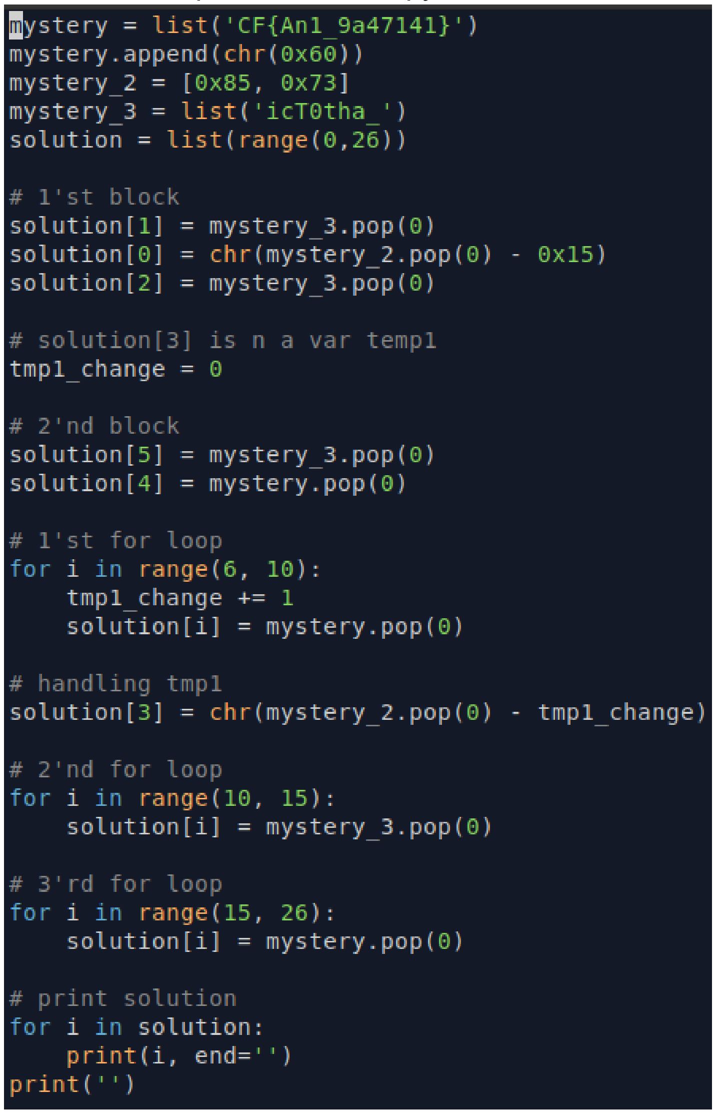
```
  f_flag = fopen("flag.txt","r");
  mystery_png = fopen("mystery.png","a");
  mystery_2_png = fopen("mystery2.png","a");
  mystery_3_png = fopen("mystery3.png","a");
  if (f_flag == (FILE *)0x0) {
    puts("No flag found, please make sure this is run on the server");
  }
  if (mystery_png == (FILE *)0x0) {
    puts("mystery.png is missing, please run this on the server");
  }
  fread(array,0x1a,1,f_flag);
  fputc((int)array[1],mystery_3_png);
  fputc((int)(char)(array[0] + '\x15'),mystery_2_png);
  fputc((int)array[2],mystery_3_png);
  tmp1 = array[3];
  fputc((int)array-more-2,mystery_3_png);
  fputc((int)array-more-1,mystery_png);
  i = 6;
  while (i < 10) {
    tmp1 = tmp1 + '\x01';
    fputc((int)array[i],mystery_png);
    i = i + 1;
  }
  fputc((int)tmp1,mystery_2_png);
  ii = 10;
  while (ii < 0xf) {
    fputc((int)array[ii],mystery_3_png);
    ii = ii + 1;
  }
  iii = 0xf;
  while (iii < 0x1a) {
    fputc((int)array[iii],mystery_png);
    iii = iii + 1;
  }
  fclose(mystery_png);
  fclose(f_flag);
```

As you can see it appends to the three PNG files



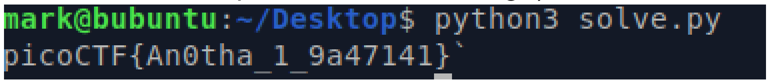
```
mystery_png = fopen("mystery.png","a");
mystery_2_png = fopen("mystery2.png","a");
mystery_3_png = fopen("mystery3.png","a");
```

And then there's a whole script that appends the values to the PNG files in a predetermined order.
So all I had to do is to print the appended part of the PNGs according to the program and get the contents of the original flag.txt.
Here is the script i've written in python



```python
mystery = list('CF{An1_9a47141}')
mystery.append(chr(0x60))
mystery_2 = [0x85, 0x73]
mystery_3 = list('icT0tha_')
solution = list(range(0,26))

# 1'st block
solution[1] = mystery_3.pop(0)
solution[0] = chr(mystery_2.pop(0) - 0x15)
solution[2] = mystery_3.pop(0)

# solution[3] is n a var temp1
tmp1_change = 0

# 2'nd block
solution[5] = mystery_3.pop(0)
solution[4] = mystery.pop(0)

# 1'st for loop
for i in range(6, 10):
    tmp1_change += 1
    solution[i] = mystery.pop(0)

# handling tmp1
solution[3] = chr(mystery_2.pop(0) - tmp1_change)

# 2'nd for loop
for i in range(10, 15):
    solution[i] = mystery_3.pop(0)

# 3'rd for loop
for i in range(15, 26):
    solution[i] = mystery.pop(0)

# print solution
for i in solution:
    print(i, end='')
print('')
```

Then I ran the script and I found the flag :)



```
mark@bubuntu:~/Desktop$ python3 solve.py
picoCTF{An0tha_1_9a47141}`
```