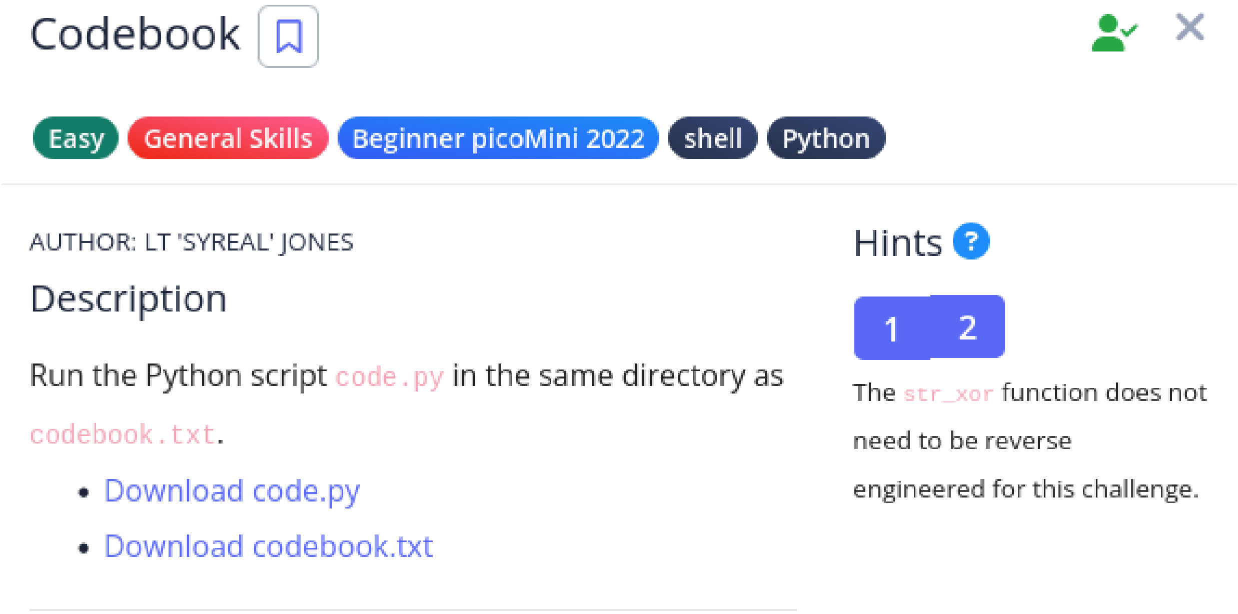
LinkedIn: Kelvin Kimotho



Codebook

Easy General Skills Beginner picoMini 2022 shell Python

AUTHOR: LT 'SYREAL' JONES

Description

Run the Python script `code.py` in the same directory as `codebook.txt`.

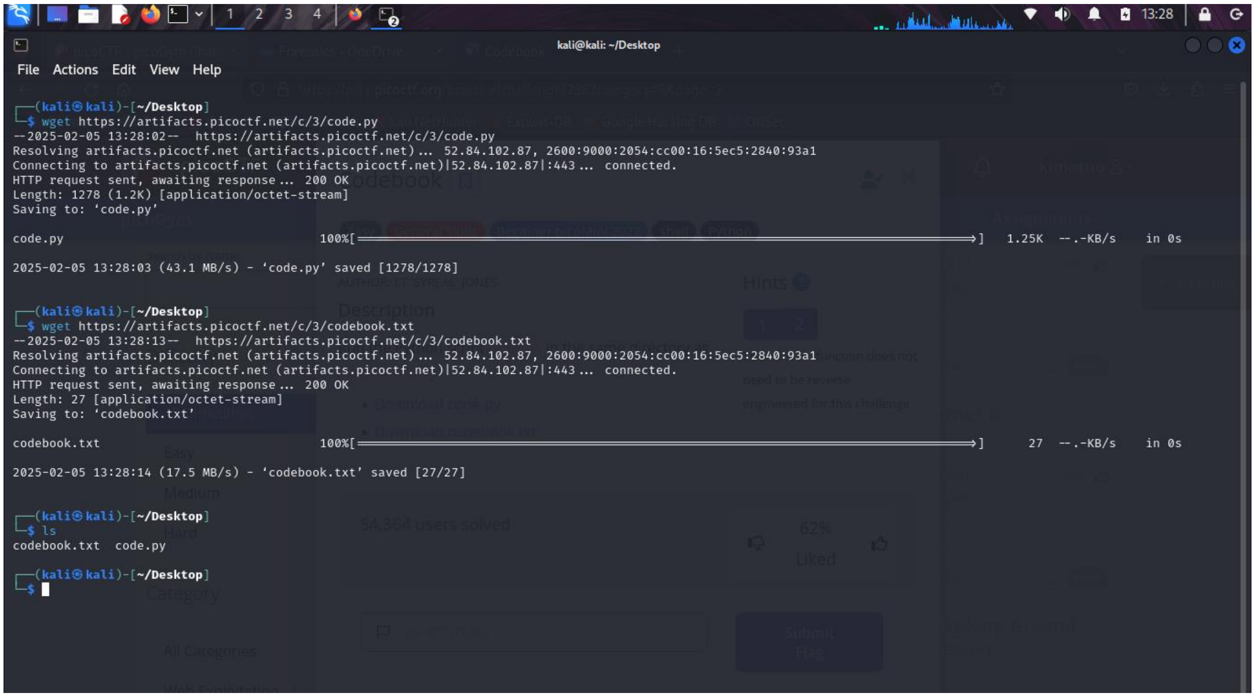
- Download code.py
- Download codebook.txt

Hints

1 2

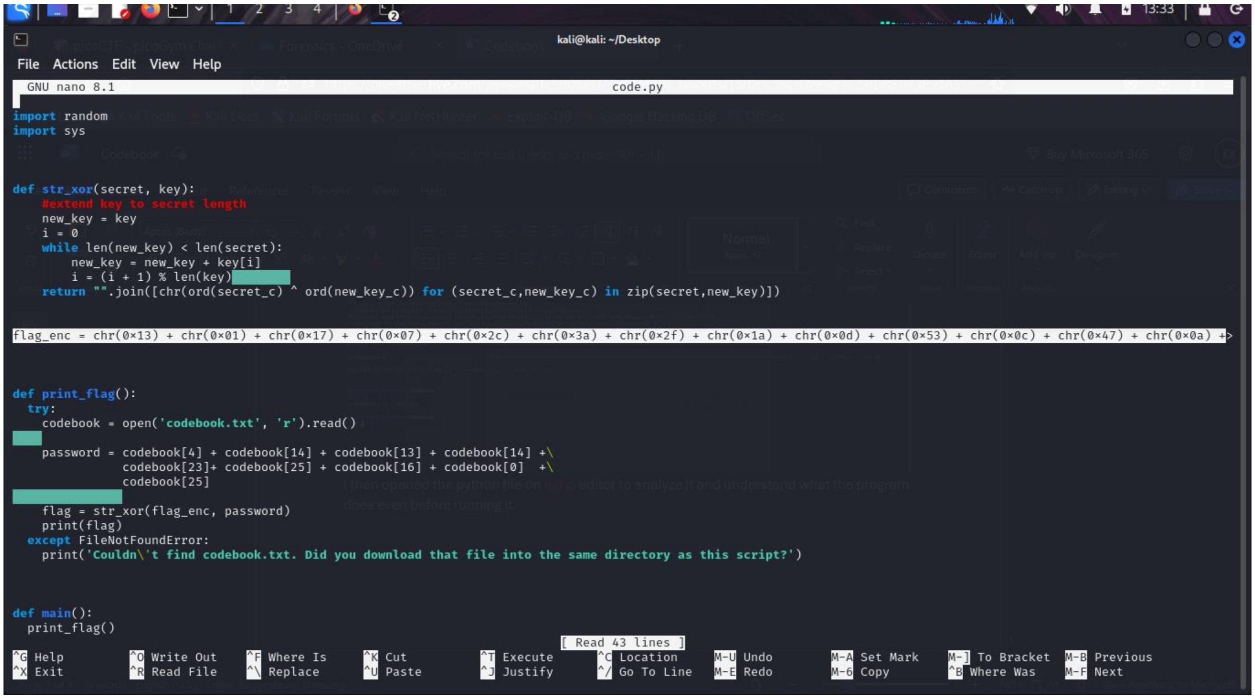
The `str_xor` function does not need to be reverse engineered for this challenge.

I downloaded the python and text files using wget tool.



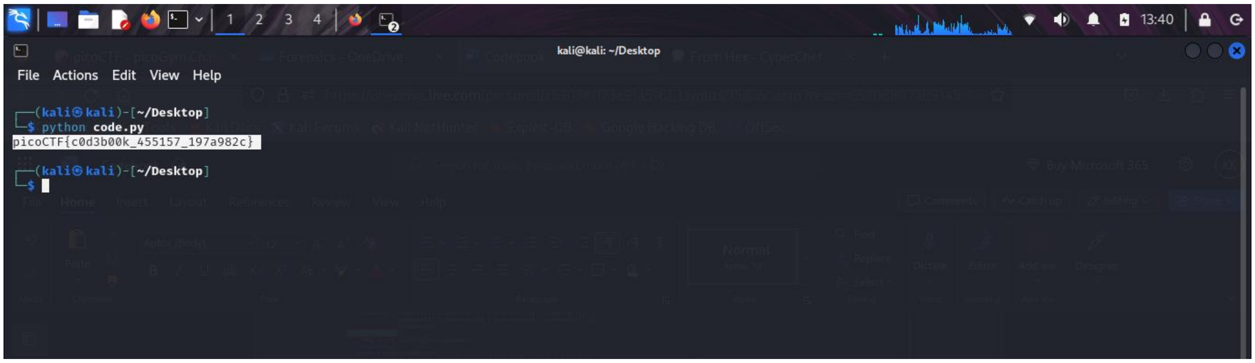
```
┌──(kali㉿kali)-[~/Desktop]
└─$ wget https://artifacts.picoctf.net/c/3/code.py
--2025-02-05 13:28:02--  https://artifacts.picoctf.net/c/3/code.py
Resolving artifacts.picoctf.net (artifacts.picoctf.net)... 52.84.102.87, 2600:9000:2054:cc00:16:5ec5:2840:93a1
Connecting to artifacts.picoctf.net (artifacts.picoctf.net)|52.84.102.87|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1278 (1.2K) [application/octet-stream]
Saving to: ‘code.py’

code.py                 100%[===================================================>]   1.25K  --.-KB/s    in 0s

2025-02-05 13:28:03 (43.1 MB/s) - ‘code.py’ saved [1278/1278]


┌──(kali㉿kali)-[~/Desktop]
└─$ wget https://artifacts.picoctf.net/c/3/codebook.txt
--2025-02-05 13:28:13--  https://artifacts.picoctf.net/c/3/codebook.txt
Resolving artifacts.picoctf.net (artifacts.picoctf.net)... 52.84.102.87, 2600:9000:2054:cc00:16:5ec5:2840:93a1
Connecting to artifacts.picoctf.net (artifacts.picoctf.net)|52.84.102.87|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 27 [application/octet-stream]
Saving to: ‘codebook.txt’

codebook.txt            100%[===================================================>]      27  --.-KB/s    in 0s

2025-02-05 13:28:14 (17.5 MB/s) - ‘codebook.txt’ saved [27/27]


┌──(kali㉿kali)-[~/Desktop]
└─$ ls
codebook.txt  code.py

┌──(kali㉿kali)-[~/Desktop]
└─$ █
```

I then opened the python file on nano editor to analyze it and understand what the program does even before running it. The flag is stored in variable as a hexadecimal encoding. The program reads the contents of the codebook.txt file where it captures various characters of the string in the codebook.txt file and forms a password which must enabled the program to decode the flag contents.



```
  GNU nano 8.1                                        code.py

import random
import sys


def str_xor(secret, key):
    #extend key to secret length
    new_key = key
    i = 0
    while len(new_key) < len(secret):
        new_key = new_key + key[i]
        i = (i + 1) % len(key)
    return "".join([chr(ord(secret_c) ^ ord(new_key_c)) for (secret_c,new_key_c) in zip(secret,new_key)])


flag_enc = chr(0x13) + chr(0x01) + chr(0x17) + chr(0x07) + chr(0x2c) + chr(0x3a) + chr(0x2f) + chr(0x1a) + chr(0x0d) + chr(0x53) + chr(0x0c) + chr(0x47) + chr(0x0a) +>



def print_flag():
  try:
    codebook = open('codebook.txt', 'r').read()

    password = codebook[4] + codebook[14] + codebook[13] + codebook[14] +\
               codebook[23]+ codebook[25] + codebook[16] + codebook[0]  +\
               codebook[25]

    flag = str_xor(flag_enc, password)
    print(flag)
  except FileNotFoundError:
    print('Couldn\'t find codebook.txt. Did you download that file into the same directory as this script?')



def main():
  print_flag()
                                                  [ Read 43 lines ]
^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket M-B Previous
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^B Where Was   M-F Next
```

I ran the program and the flag was printed out. Without the codebook.txt file, The decryption will not happen since the password required when decoding is compiled from some characters in the contents of the text file. The flag was picoCTF{c0d3b00k_455157_197a982c}.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ python code.py
picoCTF{c0d3b00k_455157_197a982c}

┌──(kali㉿kali)-[~/Desktop]
└─$ 
```