Writeup PicoCTF
Category Reverse Engineering

Created by

rainxmei

Daftar isi

[EASY]

Flag HUnters

Lyrics jump from verses to the refrain kind of like a subroutine call. There's a hidden refrain this program doesn't print by default. Can you get it to print it? There might be something in it for you.The program's source code can be downloaded here.
Additional details will be available after launching your challenge instance

Attachment: lyric-reader.py

Kita diberi file python dan kita diberi tahu bahwa ada hidden refrain yang tidak diprint oleh program kemungkinan hidden refrain tersebut adalah flag, kita coba lihat source codenya

lyric-reader.py

```
import re
import time


# Read in flag from file
flag = open('flag.txt', 'r').read()

secret_intro = \
'''Pico warriors rising, puzzles laid bare,
Solving each challenge with precision and flair.
With unity and skill, flags we deliver,
The ether's ours to conquer, '''\
+ flag + '\n'


song_flag_hunters = secret_intro +\
'''

[REFRAIN]
We're flag hunters in the ether, lighting up the grid,
No puzzle too dark, no challenge too hid.
With every exploit we trigger, every byte we decrypt,
We're chasing that victory, and we'll never quit.
CROWD (Singalong here!);
RETURN

[VERSE1]
Command line wizards, we're starting it right,
Spawning shells in the terminal, hacking all night.
Scripts and searches, grep through the void,
Every keystroke, we're a cypher's envoy.
Brute force the lock or craft that regex,
Flag on the horizon, what challenge is next?

REFRAIN;

Echoes in memory, packets in trace,
```

```
Digging through the remnants to uncover with haste.
Hex and headers, carving out clues,
Resurrect the hidden, it's forensics we choose.
Disk dumps and packet dumps, follow the trail,
Buried deep in the noise, but we will prevail.

REFRAIN;

Binary sorcerers, let's tear it apart,
Disassemble the code to reveal the dark heart.
From opcode to logic, tracing each line,
Emulate and break it, this key will be mine.
Debugging the maze, and I see through the deceit,
Patch it up right, and watch the lock release.

REFRAIN;

Ciphertext tumbling, breaking the spin,
Feistel or AES, we're destined to win.
Frequency, padding, primes on the run,
Vigenère, RSA, cracking them for fun.
Shift the letters, matrices fall,
Decrypt that flag and hear the ether call.

REFRAIN;

SQL injection, XSS flow,
Map the backend out, let the database show.
Inspecting each cookie, fiddler in the fight,
Capturing requests, push the payload just right.
HTML's secrets, backdoors unlocked,
In the world wide labyrinth, we're never lost.

REFRAIN;

Stack's overflowing, breaking the chain,
ROP gadget wizardry, ride it to fame.
Heap spray in silence, memory's plight,
Race the condition, crash it just right.
Shellcode ready, smashing the frame,
Control the instruction, flags call my name.

REFRAIN;

END;
'''

MAX_LINES = 100

def reader(song, startLabel):
  lip = 0
  start = 0
  refrain = 0
  refrain_return = 0
  finished = False

  # Get list of lyric lines
  song_lines = song.splitlines()

  # Find startLabel, refrain and refrain return
  for i in range(0, len(song_lines)):
    if song_lines[i] == startLabel:
      start = i + 1
    elif song_lines[i] == '[REFRAIN]':
      refrain = i + 1
    elif song_lines[i] == 'RETURN':
      refrain_return = i
```

```
    # Print lyrics
    line_count = 0
    lip = start
    while not finished and line_count < MAX_LINES:
      line_count += 1
      for line in song_lines[lip].split(';'):
        if line == '' and song_lines[lip] != '':
          continue
        if line == 'REFRAIN':
          song_lines[refrain_return] = 'RETURN ' + str(lip + 1)
          lip = refrain
        elif re.match(r"CROWD.*", line):
          crowd = input('Crowd: ')
          song_lines[lip] = 'Crowd: ' + crowd
          lip += 1
        elif re.match(r"RETURN [0-9]+", line):
          lip = int(line.split()[1])
        elif line == 'END':
          finished = True
        else:
          print(line, flush=True)
          time.sleep(0.5)
          lip += 1


reader(song_flag_hunters, '[VERSE1]')
```

Bagian yang tersembunyi itu termasuk flag tersebut, tetapi bagian tersebut tidak diprint coba kita cari apakah ada celah agar bisa print flag tersebut

```
        if line == 'REFRAIN':
          song_lines[refrain_return] = 'RETURN ' + str(lip + 1)
          lip = refrain
```

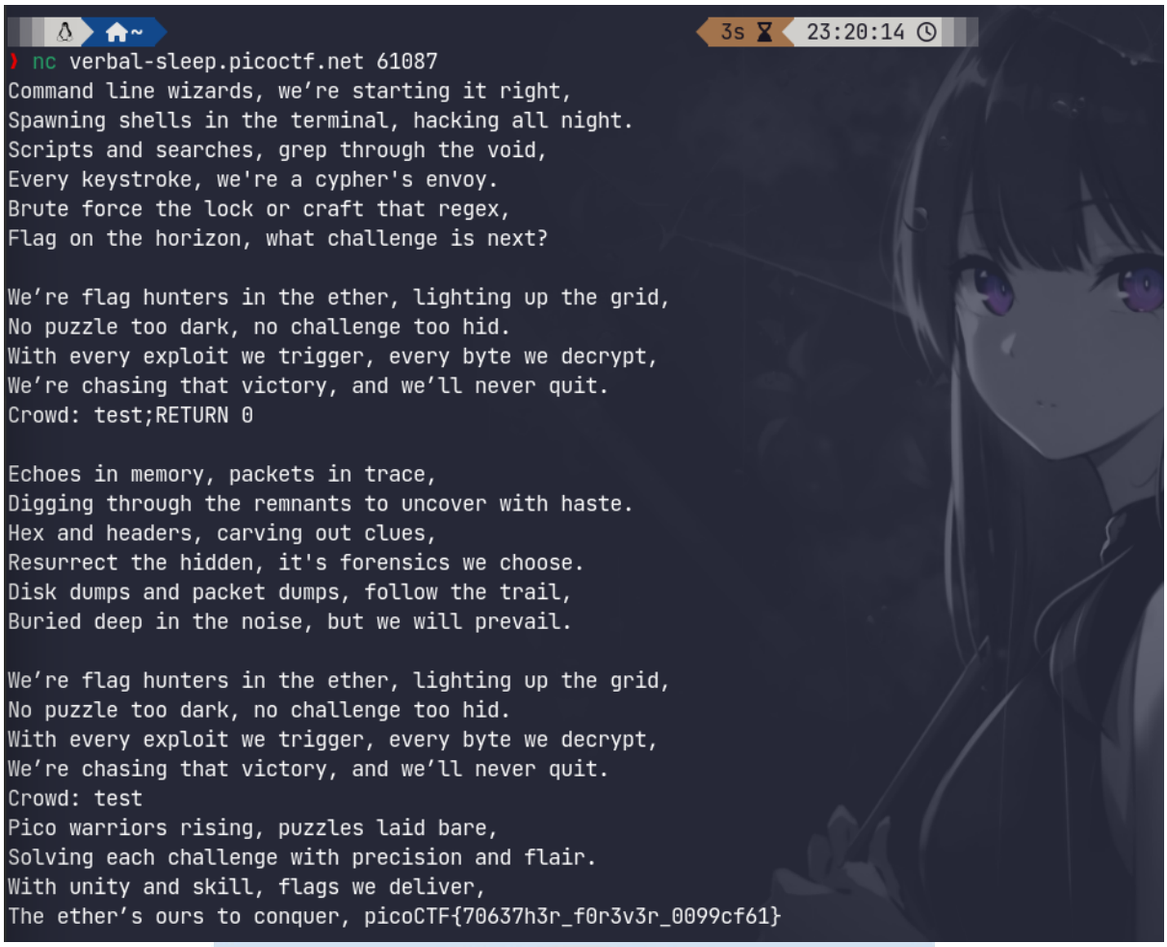
Sepertinya bagian ini dia setelah setalah print REFRAIN dia akan menyimpan baris terakhir ke baigan RETURN kita coba untuk RETURN 0 karena flag berada pada bagian awal

```
~                                                    INT X  23:09:43
❯ nc verbal-sleep.picoctf.net 65112
Command line wizards, we're starting it right,
Spawning shells in the terminal, hacking all night.
Scripts and searches, grep through the void,
Every keystroke, we're a cypher's envoy.
Brute force the lock or craft that regex,
Flag on the horizon, what challenge is next?

We're flag hunters in the ether, lighting up the grid,
No puzzle too dark, no challenge too hid.
With every exploit we trigger, every byte we decrypt,
We're chasing that victory, and we'll never quit.
Crowd: RETURN 0

Echoes in memory, packets in trace,
Digging through the remnants to uncover with haste.
Hex and headers, carving out clues,
Resurrect the hidden, it's forensics we choose.
Disk dumps and packet dumps, follow the trail,
Buried deep in the noise, but we will prevail.
```

Sepertinya RETURN 0-nya tidak tereksekusi karena dia tergabung dengan Crowd: coba kita pisah menggunakan semicolon karena program tersebut memisah berdasarkan semicolon



Horee kita sudah dapat flag tersebut

picoCTF{70637h3r_f0r3v3r_0099cf61}

Transformation
I wonder what this really is...enc ''.join([chr((ord(flag[i]) << 8) + ord(flag[i + 1])) for i in range(0, len(flag), 2)])
Attachment: env

Kita diberikan file env, sepertinya soal menyuruh kita untuk mendecode dari file env, kita coba lihat isi file enc

灩捯䍔䙻ㄶ形楴獟楮獴㌴摟潦弸形㝦㘲捡㕽

Coba kita coba bikin script untuk decode tulisan tersebut

Decode-enc.py

```python
#!/usr/bin/env python3
enc = "灩捯䍔䙻ㄶ形楴獟楮獴㌴摟潦弸形㝦㘲捡㕽"

decoded = ""

for c in enc:
    val = ord(c)

    high = val >> 8
    low  = val & 0xff

    decoded += chr(high)
    decoded += chr(low)

print(decoded)
```

```
~/a/pico-ctf/reverse-engineering/transformation    08:00:50
❯ ./decode-enc.py
picoCTF{16_bits_inst34d_of_8_b7f62ca5}
```

picoCTF{16_bits_inst34d_of_8_b7f62ca5}