WRITEUP IT FESTIVAL 2023

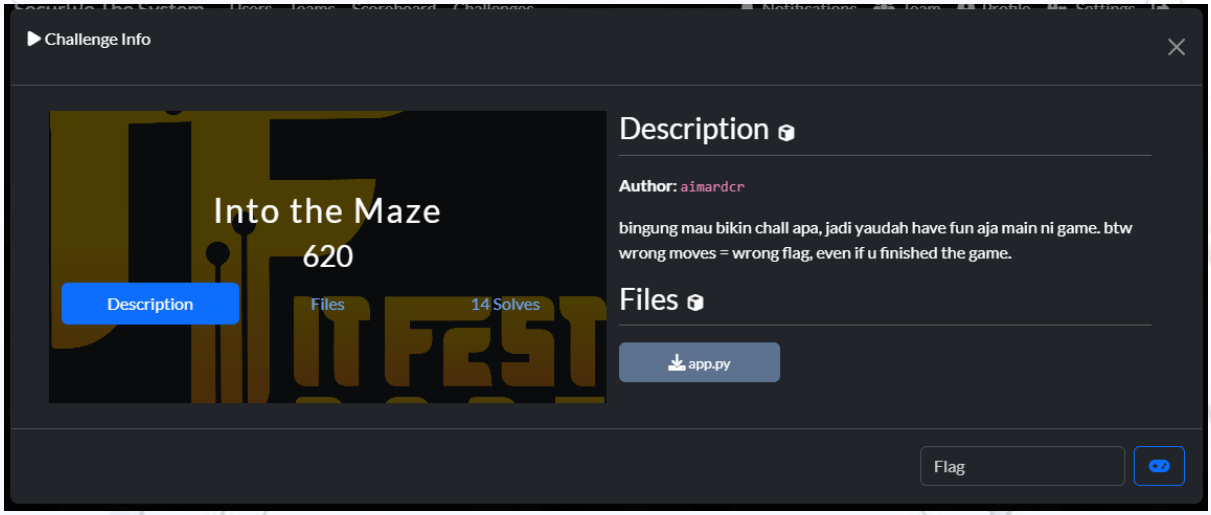
AJARIN DONG PUH SEPUH

CHRISTOPHER RALIN ANGGOMAN
FIKRI MUHAMMAD ABDILLAH
RUBEN ALBION YOBEL

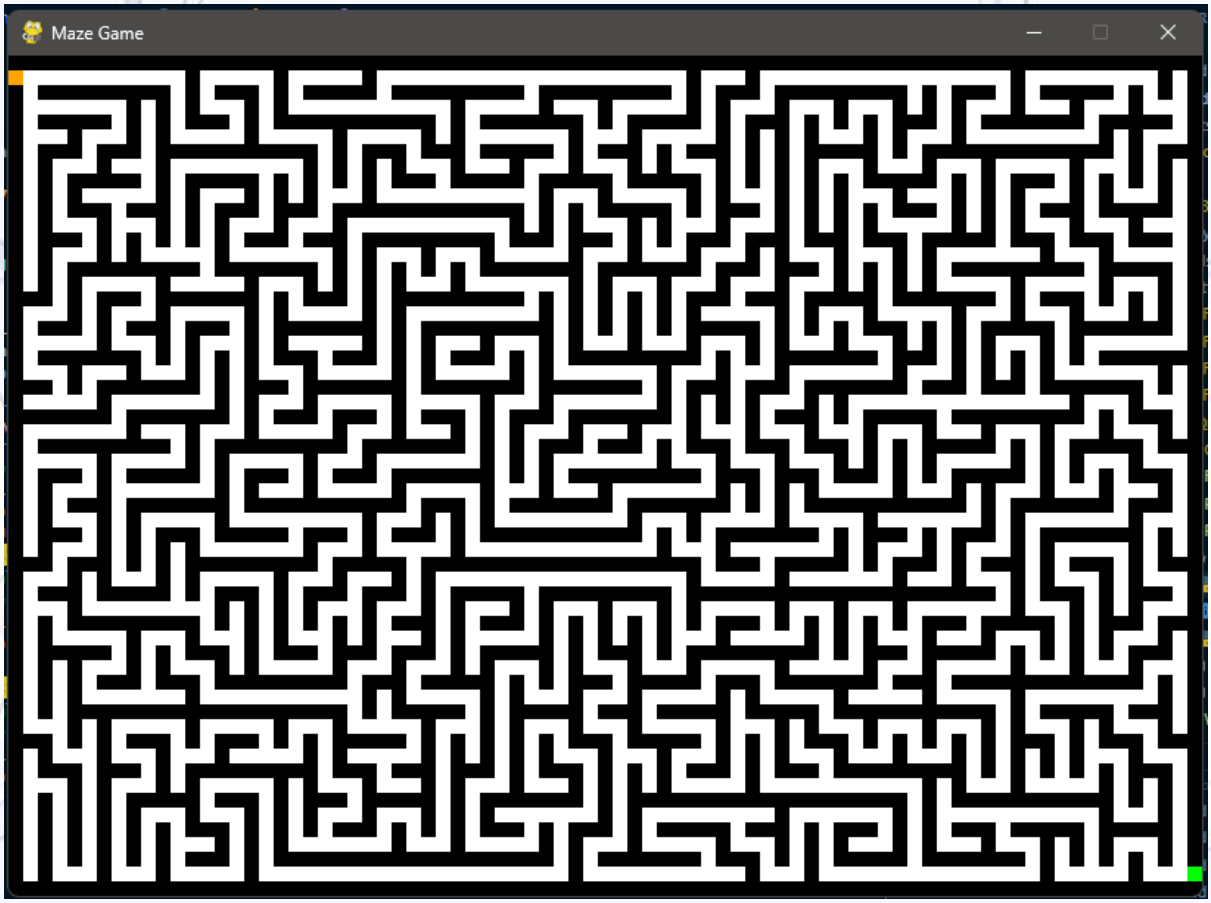
Daftar Isi

REVERSE ENGINEERING

Into The Maze



Kami mendapat file `app.py` yang isinya adalah sebuah game labirin (maze).



Dalam game tersebut kita akan disuruh untuk mencapai kotak hijau untuk mendapatkan flag, namun ketika mencapai kotak hijau, langkah tidak boleh salah dan harus mendapatkan rute terdekat, karena itu kami memodifikasi game asli agar bisa mendapatkan rute paling dekat, dan otomatis akan berjalan

Berikut modifikasi code kami:

```
import pygame
import sys

import hashlib
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

FLAG =
'0e0d9bc536bff99d227b6b26470de14d1784426798126874fb86f35144fac472e8cb8aa1ec32434
90706840421a5aa6c00824ef69048cb1a422bdb70c0157530'

class HackedGame:
    def __init__(self, width, height, maze_layout):
        self.width = width
        self.height = height
        self.screen = pygame.display.set_mode((width, height))
        pygame.display.set_caption("Maze Game")
        self.maze = Maze(maze_layout, width, height)
        self.player = Player(self.maze.cell_size)
        self.finded_route = self.find_route()
        self.key_solve = self.convert_route_to_moves(self.finded_route)
        self.is_winned = False

    def is_win(self):
        return self.player.x >= self.width - self.maze.cell_size and
self.player.y >= self.height - (self.maze.cell_size * 2)

    def get_possible_moves(self, pos):
        sys.setrecursionlimit(100000)
        x, y = pos
        moves = []
        if x > 0 and self.maze.layout[y][x - 1] == 0:
            moves.append((x - 1, y))
        if x < len(self.maze.layout[0]) - 1 and self.maze.layout[y][x + 1] == 0:
            moves.append((x + 1, y))
        if y > 0 and self.maze.layout[y - 1][x] == 0:
            moves.append((x, y - 1))
        if y < len(self.maze.layout) - 1 and self.maze.layout[y + 1][x] == 0:
            moves.append((x, y + 1))

        return moves

    def find_route(self, pos=(0, 1), route=[]):
        if pos[0] == len(self.maze.layout[0]) - 1 or pos[1] ==
len(self.maze.layout) - 1:
```

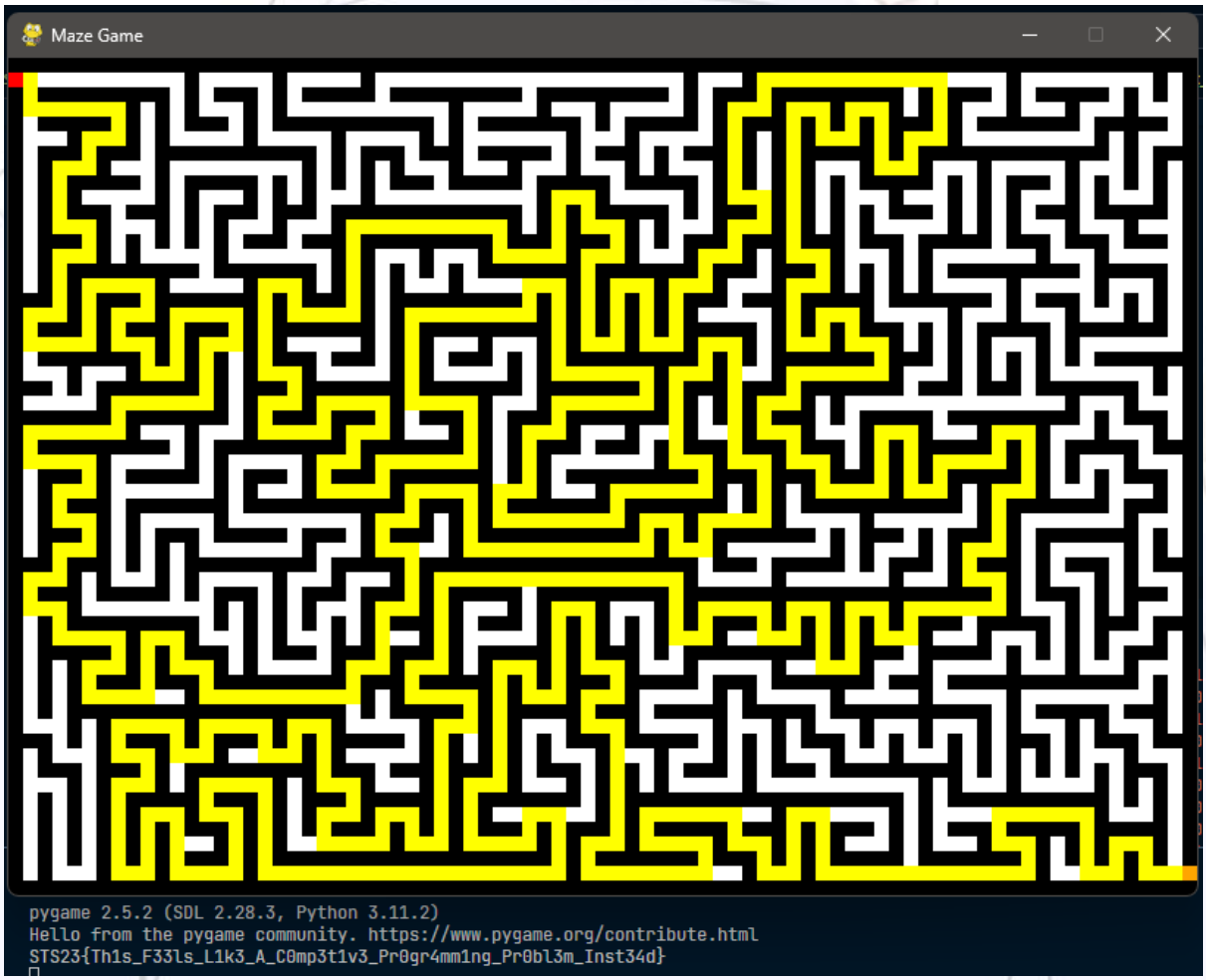
```
            return route

        for move in self.get_possible_moves(pos):
            if move not in route:
                route.append(move)
                if self.find_route(move, route):
                    return route
                route.pop()

        return False

    def convert_route_to_moves(self, route):
        moves = []
        x1, y1 = (0, 1)
        for i in range(len(route)):
            x2, y2 = route[i]

            if x1 == x2:
                if y1 < y2:
                    moves.append(pygame.K_DOWN)
                else:
                    moves.append(pygame.K_UP)
            else:
                if x1 < x2:
                    moves.append(pygame.K_RIGHT)
                else:
                    moves.append(pygame.K_LEFT)
            x1, y1 = x2, y2

        return moves

    def str_moves(self, moves):
        res = []
        for m in moves:
            if m == pygame.K_DOWN:
                res.append("DOWN")
            elif m == pygame.K_UP:
                res.append("UP")
            elif m == pygame.K_LEFT:
                res.append("LEFT")
            elif m == pygame.K_RIGHT:
                res.append("RIGHT")
        return res

    def run(self):
        running = True
        while running:
            event = pygame.event.get()
            while event or self.key_solve:
                if event:
```

```
                    evt = event.pop(0)
                    if evt.type == pygame.QUIT:
                        running = False
                    elif evt.type == pygame.KEYDOWN:
                        self.player.move(evt.key, self.maze)
                if self.key_solve:
                    self.player.move(self.key_solve.pop(0), self.maze)

            if not self.is_winned and self.is_win():
                key = hashlib.md5(str(self.player.state).encode()).digest()
                iv = hashlib.md5(str(self.player.x).encode() +
str(self.player.y).encode() + str(self.player.move_count).encode()).digest()

                try:
                    cipher = AES.new(key, AES.MODE_CBC, iv)
                    flag = unpad(cipher.decrypt(bytes.fromhex(FLAG)),
AES.block_size).decode()

                    print(flag)
                    self.is_winned = True
                except:
                    print("Invalid moves!")
                    running = False
                    break

            self.screen.fill(WHITE)
            self.maze.draw(self.screen)
            self.player.draw(self.screen)
            pygame.display.flip()

        pygame.quit()
        sys.exit()

class Maze:
    def __init__(self, layout, width, height):
        self.CELL_COLOR = {
            0: WHITE,
            1: BLACK,
            2: YELLOW
        }
        self.layout = layout
        self.cell_size = 10

    def draw(self, screen):
        for y, row in enumerate(self.layout):
            for x, cell in enumerate(row):
                rect = pygame.Rect(x * self.cell_size, y * self.cell_size,
self.cell_size, self.cell_size)
                color = self.CELL_COLOR[cell]
                pygame.draw.rect(screen, color, rect)
```

```
        pygame.draw.rect(screen, RED, (0, self.cell_size, self.cell_size,
self.cell_size))
        pygame.draw.rect(screen, GREEN, (len(self.layout[0]) * self.cell_size -
self.cell_size, len(self.layout) * self.cell_size - (self.cell_size * 2),
self.cell_size, self.cell_size))

    def is_valid_move(self, x, y):
        if x < 0 or y < 0 or x >= len(self.layout[0]) * self.cell_size or y >=
len(self.layout) * self.cell_size:
            return False
        return self.layout[y // self.cell_size][x // self.cell_size] in [0, 2]

class Player:
    def __init__(self, size):
        self.x = 0
        self.y = size
        self.size = size
        self.speed = size
        self.state = [0] * 32
        self.state_pos = 0
        self.move_count = 0

    def move(self, key, maze):
        if key == pygame.K_LEFT:
            new_x = self.x - self.speed
            if maze.is_valid_move(new_x, self.y):
                maze.layout[self.y // self.size][self.x // self.size] = 2
                self.x = new_x

            self.state_pos -= 1
            if self.state_pos < 0:
                self.state_pos = 0

        elif key == pygame.K_RIGHT:
            new_x = self.x + self.speed
            if maze.is_valid_move(new_x, self.y):
                maze.layout[self.y // self.size][self.x // self.size] = 2
                self.x = new_x

            self.state_pos += 1
            if self.state_pos >= len(self.state):
                self.state_pos = len(self.state) - 1
        elif key == pygame.K_UP:
            new_y = self.y - self.speed
            if maze.is_valid_move(self.x, new_y):
                maze.layout[self.y // self.size][self.x // self.size] = 2
                self.y = new_y

            self.state[self.state_pos] = 1
```

```
        elif key == pygame.K_DOWN:
            new_y = self.y + self.speed
            if maze.is_valid_move(self.x, new_y):
                maze.layout[self.y // self.size][self.x // self.size] = 2
                self.y = new_y

            self.state[self.state_pos] = 0

        self.move_count += 1

    def draw(self, screen):
        pygame.draw.rect(screen, ORANGE, (self.x, self.y, self.size, self.size))

pygame.init()

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
ORANGE = (255, 165, 0)
GREEN = (0, 255, 0)
RED = (255, 0, 0)
YELLOW = (255, 255, 0)

MAZE_LAYOUT = . . .

WIDTH, HEIGHT = 810, 570

game = HackedGame(WIDTH, HEIGHT, MAZE_LAYOUT)
game.run()
```

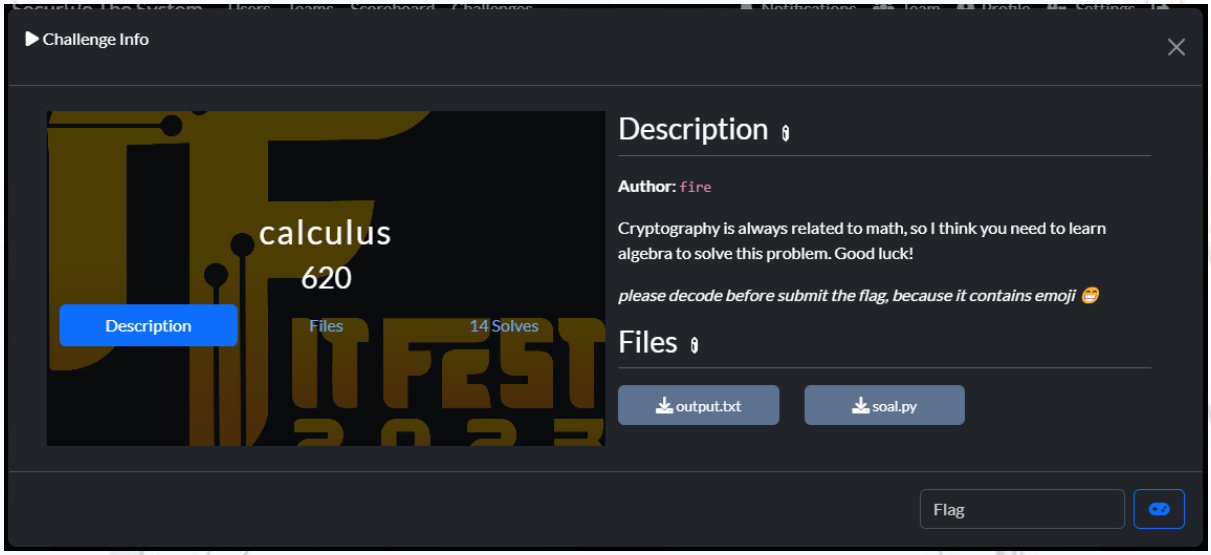
Setelah kami run program tersebut, kami pun mendapat flag nya:



FLAG: STS23{Th1s_F33ls_L1k3_A_C0mp3t1v3_Pr0gr4mm1ng_Pr0bl3m_Inst34d}

CRYPTOGRAPHY

calculus



Kami medapat 2 file `soal.py` & `output.txt`, dalam soal flag akan di masukkan ke dalam fungsi encrypt, dan hasil tersebut akan di taruh pada file `output.txt`.

Dalam fungsi encrypt, sebuah set di generate dengan isi 2 bilangan prima dengan besar 1024 bit, selanjutnya variable p akan di generate dengan bilangan prima dengan besar sama, yaitu 1024 bit. n dihasilkan dengan constant variable yaitu `2^3+2^4` dengan hasil `24`. selanjutnya 3 variabel akan dihasilkan (x, y z) seperti notasi di bawah ini:

$$
\begin{aligned}
x &= set_0 + set_1 \\
y &= set_0^2 + set_1^2 \\
z &= set_0 * set_1 + (p^2 \mod n)
\end{aligned}
$$

selanjutnya cipher dihasilkan dengan notasi di bawah ini:

$$c = (val \oplus (set_0^3 + set_1^3)) \cdot (x + y)$$

selanjutnya (x, z, c) akan dikembalikan (return).

Dengan penjelasan di atas, maka tujuan pertama adalah untuk mendapatkan nilai dalam set ini. Disini jika kita mengetahui hasil perkalian (mul) dari set kita dapat menggunakan rumus ‘quadratic formula’ seperti di bawah ini

let make set_0 as g and set_1 as h

$$
\begin{aligned}
x &= g + h \\
z &= g \cdot h \\
g &= x - h \\
z &= (x - h) \cdot h \\
z &= -h^2 + xh \\
h^2 - xh + z &= 0 \\
h &= \frac{-(-x) + \sqrt{(-x)^2 - 4z}}{2} \\
h &= \frac{x + \sqrt{x^2 - 4z}}{2}
\end{aligned}
$$

karena `z` diatas telah di tambah oleh `k`, kita harus mendapatkan nilai k, dan mengurangi z dengan k, lalu menggunakan formula diatas. karena `k` bernilai sangat kecil (0~23) kita dapat bruteforce nilai tersebut

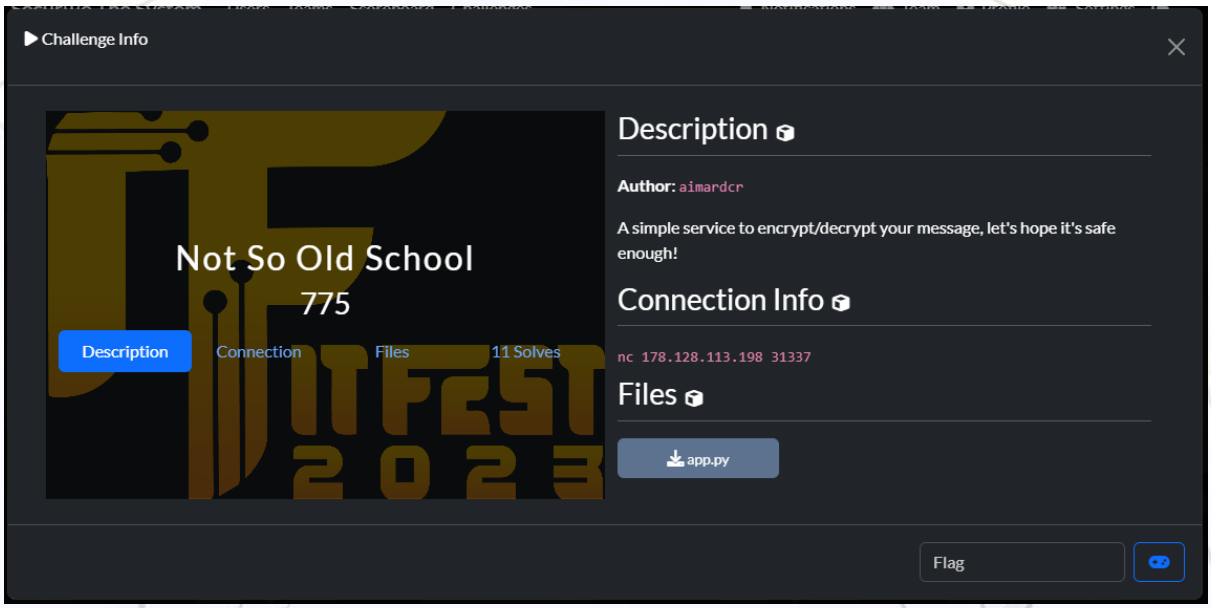
berikut adalah kode solvernya:

```python
from Crypto.Util.number import *
from decimal import Decimal, getcontext
getcontext().prec = 1000

def formula(x, z):
    return round((x + (Decimal(x**2 - 4*z).sqrt())) / 2)

x = . . .
z_p_k = . . .
c = . . .
c = int(c, 16)

for i in range(pow(2,3) + pow(2,4)):
    z = z_p_k - i
    p1 = formula(x, z)
    p2 = x - p1
    if p1 * p2 + i != z_p_k:
        continue
    y = pow(p1, 2) + pow(p2, 2)
    m = (c // (x + y))
    assert m * (x + y) == c
    m ^= (p1**3 + p2**3)
    print(long_to_bytes(m))
```

```
STS23{kalo_kamu_pake_z3_kamu_curang_woi_🗿🗿🗿}
```

FLAG: STS23{kalo_kamu_pake_z3_kamu_curang_woi_🗿🗿🗿}

Not So Old School



Kami diberikan file `app.py` yang merupakan source code dari netcat server, dalam fungsi utama kita dapat memilih 3 pilihan pada menu (“Get Encrypted Flag”, “Encrypt String”, dan ”Exit”). Program di awal generate key dengan cara random sepanjang 8 byte. key akan digunakan saat encrypt message (flag, input string). di dalam fungsi encrypt terlihat program sangat membingungkan untuk reversing key nya, namun saat diteliti, sebenarnya plaintext hanya di xor secara sederhana dengan `generated_xor_key` yang telah di generate menggunakan key asli, lalu hasil xor akan di gunakan untuk mendapatkan index sbox. dengan begitu kita hanya perlu mendapatkan index sbox dari cipher yang telah di generate dan menggunakan notasi di bawah ini:

$$c_{flag} = flag \oplus genkey$$
$$flag = c_{flag} \oplus genkey$$

$$c_{inp} = inp \oplus genkey$$
$$genkey = inp \oplus c_{inp}$$

$$flag = c_{flag} \oplus inp \oplus c_{inp}$$

berikut adalah solver nya:

```
from pwn import *

sbox = [98, 56, 7, 192, 121, 149, 107, 246, 120, 132, 191, 152, 229, 238, 94,
106, 176, 170, 161, 253, 145, 181, 237, 211, 219, 250, 131, 190, 158, 24, 126,
32, 79, 212, 244, 53, 60, 183, 83, 128, 162, 137, 15, 148, 50, 51, 166, 92, 171,
88, 44, 242, 69, 91, 101, 103, 175, 3, 82, 40, 245, 110, 34, 143, 248, 35, 109,
```

```
115, 227, 47, 140, 122, 193, 59, 39, 243, 208, 55, 165, 213, 224, 231, 96, 185,
151, 100, 105, 12, 66, 42, 160, 214, 205, 189, 130, 5, 147, 20, 236, 85, 142,
194, 2, 228, 124, 215, 14, 26, 240, 223, 154, 203, 54, 25, 141, 200, 8, 111,
177, 0, 75, 73, 204, 80, 230, 58, 112, 10, 52, 157, 116, 41, 4, 217, 18, 9, 174,
27, 226, 163, 36, 13, 167, 72, 241, 21, 186, 30, 87, 221, 168, 89, 239, 29, 178,
179, 249, 45, 195, 57, 95, 22, 180, 153, 129, 108, 201, 202, 63, 68, 64, 135,
207, 156, 133, 220, 11, 71, 6, 233, 232, 119, 173, 90, 102, 117, 136, 86, 247,
76, 234, 164, 172, 184, 78, 225, 125, 199, 46, 210, 216, 123, 31, 235, 182, 251,
38, 206, 139, 197, 159, 127, 150, 61, 16, 19, 28, 198, 93, 77, 49, 169, 1, 114,
134, 187, 188, 67, 113, 74, 218, 104, 254, 65, 196, 155, 144, 209, 37, 81, 70,
48, 43, 84, 138, 62, 17, 23, 222, 118, 146, 33, 99, 252, 97]

# nc 178.128.113.198 31337
HOST = "178.128.113.198"
POST = 31337
io = remote(HOST, POST)

def encrypt(s: bytes):
    io.sendlineafter(b"> ", b"2")
    io.sendlineafter(b": ", s)
    return io.recvline().strip()

def get_encrypted_flag():
    io.sendlineafter(b"> ", b"1")
    return io.recvline().strip()

def get_index(sbox, vals):
    res = []
    for val in vals:
        for i in range(len(sbox)):
            if sbox[i] == val:
                res.append(i)
                break
    return res

def check_valid_hex(s):
    for c in s:
        if c not in b"0123456789abcdef":
            return False
    return True

def main():
    global io
    while True:
        print("Getting encrypted flag...")
        enc_flag = get_encrypted_flag()
        if not check_valid_hex(enc_flag):
            print("Invalid hex, reconnecting...")
            io.close()
            io = remote(HOST, POST)
```

```
            continue
        print("Got encrypted flag!")
        enc_flag = get_index(sbox, bytes.fromhex(enc_flag.decode()))
        break
    inp = b"0" * len(enc_flag)
    enc_inp = get_index(sbox, bytes.fromhex(encrypt(inp).decode()))
    flag = xor(enc_flag, inp, enc_inp).decode()
    print(flag)

main()
```

berikut hasilnya:

```
[x] Opening connection to 178.128.113.198 on port 31337
[x] Opening connection to 178.128.113.198 on port 31337: Trying 178.128.113.198
[+] Opening connection to 178.128.113.198 on port 31337: Done
Getting encrypted flag...
Invalid hex, reconnecting...
[*] Closed connection to 178.128.113.198 port 31337
[x] Opening connection to 178.128.113.198 on port 31337
[x] Opening connection to 178.128.113.198 on port 31337: Trying 178.128.113.198
[+] Opening connection to 178.128.113.198 on port 31337: Done
Getting encrypted flag...
Got encrypted flag!
This was a very easy crypto chall, hope you didn't spent too much time into this chall.
BTW, Here's your flag: STS23{N3v3r_Us3_St4t1c_K3y}

[*] Closed connection to 178.128.113.198 port 31337
```

FLAG: STS23{N3v3r_Us3_St4t1c_K3y}