[2023 JBUCTF] misc

calculation_game

Write-Up

문제 개요

제공 파일 : calc_game.py

```python
from random import randint
from functools import wraps
import errno
import os
import signal

class TimeoutError(Exception):
    pass

def timeout(seconds=10, error_message=os.strerror(errno.ETIME)):
    def decorator(func):
        def _handle_timeout(signum, frame):
            raise TimeoutError(error_message)

        def wrapper(*args, **kwargs):
            signal.signal(signal.SIGALRM, _handle_timeout)
            signal.alarm(seconds)
            try:
                result = func(*args, **kwargs)
            finally:
                signal.alarm(0)
            return result

        return wraps(func)(wrapper)

    return decorator

def calc(coef, soul):
    result = 0
    for i in range(len(soul)):
        result += coef[i] * soul[i]
    return result

def swap(soul, i, j):
    soul[i] ^= soul[j]
    soul[j] ^= soul[i]
    soul[i] ^= soul[j]
    return soul

def printEqua(coef, soul):
    for i in range(len(soul)):
        if i == len(soul) - 1:
            print(f'({coef[i]} * {soul[i]}) = {calc(coef, soul)}')
            break
        print(f'({coef[i]} * {soul[i]})', end=' + ')
    print(f'The constant term is {calc(coef, soul)}.')

def initEqua():
    soul, coef = [0] * 5, [0] * 5
    const = randint(-15, 15)

    for i in range(len(soul)):
        soul[i] = randint(-30, 30)
        coef[i] = randint(-10, 10)
    return soul, coef, const

@timeout(5, '\n\nTIME OUT')
def run_game(soul, coef, const):
    for r in range(5):
        printEqua(coef, soul)
        print(f'The constant term we need is {const}. \n')

        while True:
            print(f'swap count : {5 - r}')
            try:
                i, j = map(int, input('Input index to swap (ex 1 3) : ').split())
                if i < 0 or i > 4 or j < 0 or j > 4:
                    raise IndexError
            except (IndexError, ValueError):
                print('Retry')
                continue
            except Exception as e:
                print(e)
                exit()
            break

        soul = swap(soul, i, j)

        if calc(coef, soul) == const:
            print(f'Flag is : {flag.decode()}')
            exit()

    print()
    printEqua(coef, soul)
    print('\nFailed :(')

if __name__ == '__main__':
    flag = open('/flag', 'rb').read()
    soul, coef, const = initEqua()
    run_game(soul, coef, const)
```

주어진 식에서 soul 변수의 값들의 위치를 변경할 수 있는 5번의 기회가 주어지며, 주어진 식의 계산 값이 문제에서 랜덤하게 (-15 ~ 15 사이의 정수) 정해진 값과 일치하면 flag를 출력해준다.

문제 풀이

일일이 계산하여 정해진 값이 나오게 하는 것은 힘들다.

swap 알고리즘을 자세히 보면 xor 연산을 이용하여 swap 하고 있다.

```
def swap(soul, i, j):
    soul[i] ^= soul[j]
    soul[j] ^= soul[i]
    soul[i] ^= soul[j]
    return soul
```

swap(soul, i, i) 가 호출되었다고 생각해보자.

그러면 soul[i] ^= soul[i]가 되면서 결과적으로는 soul[i] == 0 이 된다.

flag를 얻는 조건은 주어진 식의 계산 값이 문제에서 랜덤하게 (-15 ~ 15 사이의 정수) 정해진 값과 일치하는 것인데

랜덤하게 정해진 값이 0이 되도록 시도한 다음 swap에서 총 5번 같은 위치를 입력하면 soul의 모든 값이 0이 돼서 결과적으로는 식의 값이 0이 되게 된다.

(a*0 + b*0 + c*0 + d*0 + e*0 = 0)

따라서 flag를 얻는 조건인 주어진 식의 계산 값과 문제에서 랜덤하게 (-15 ~ 15 사이의 정수) 정해진 값이 일치하는 것인데 서로 0으로 일치하므로 flag를 출력한다.

exploit.py

```
1   from pwn import *
2
3   while True:
4       p = remote('172.17.0.2', 10004)
5       p.recvuntil(b'The constant term we need is ')
6       const = int(p.recvuntil(b'.')[:-1].decode())
7       if const != 0:
8           p.close()
9           continue
10      break
11  for i in range(5):
12      p.recvuntil(b'(ex 1 3) : ')
13      p.sendline(f'{i} {i}'.encode())
14
15  flag = p.recvline()
16  success(f'{flag.decode()}')
17
```

FLAG

scpCTF{0973dbab07dacff7660c520330ab54f2b9255bcf9a4b}