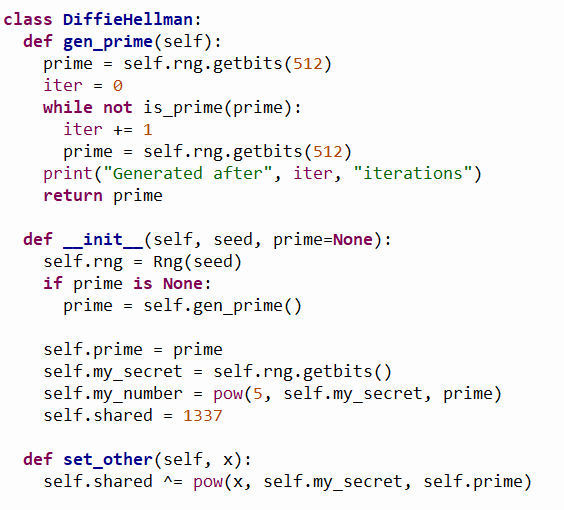
1. 문제



2. 과정

Bob과 Alice 간의 비트를 뒤집고 통신을 decrypt하라는 문제였다. 파일을 먼저 확인해봤는데 가짜 플래그 파일과 파이썬 코드를 확인할 수 있다. 파이썬 코드는 길어서 DiffieHellam클래스만 가져와봤다.



우선 파이썬 코드에서 플래그를 암호화하는 데 공유 대칭키를 사용하는데 이는 alice.shared를 찾으면 된다. (5^bob.my\_secret\*alice.my\_secret)

5^bob.my\_secret를 기반으로 (5^bob.my\_secret)

여기서 alice.my\_secret는 alice\_seed와 우리의 입력 값의 XOR으로 alice\_my\_secret의 값이 결정된다고 한다.

이제 플래그를 해독할 수 있도록 스크립트를 작성하면 되는데 다른 분이 쓰신 라업을 참고했다.

from Crypto.Util.number import long\_to\_bytes, bytes\_to\_long

import pwn

import base64

import re

import flag\_decrypt

import subprocess

def get\_bit(n, pos):

return (n >> pos) & 1

def set\_bit(n, pos):

return n | (1 << pos)

seed\_regex = re.compile(r'seed: (.\*)\n', re.MULTILINE)

gen\_regex = re.compile(r'Generated after (\d\*) iterations', re.MULTILINE)

bob\_regex = re.compile(r'bob number (\d\*)\n', re.MULTILINE)

iv\_regex = re.compile(r'bob .\*\n(.\*)\n', re.MULTILINE)

enc\_flag\_regex = re.compile(r'bob .\*\n.\*\n(.\*)\n', re.MULTILINE)

anno = False

remote = True

def send(flip):

payload = base64.b64encode(long\_to\_bytes(flip))

sh.sendline(payload)

output = sh.recvuntilS('bit-flip str:')

n\_iters = int(gen\_regex.search(output).group(1))

if anno:

seed = seed\_regex.search(output).group(1)

seed = bytes.fromhex(seed)

seed = bytes\_to\_long(seed)

else:

seed = None

return (n\_iters, seed)

def send\_get\_all(flip):

payload = base64.b64encode(long\_to\_bytes(flip))

sh.sendline(payload)

output = sh.recvuntilS('bit-flip str:')

n\_iters = int(gen\_regex.search(output).group(1))

if anno:

seed = seed\_regex.search(output).group(1)

seed = bytes.fromhex(seed)

seed = bytes\_to\_long(seed)

else:

seed = None

bob\_number = int(bob\_regex.search(output).group(1))

iv = iv\_regex.search(output).group(1)

iv = base64.b64decode(iv)

enc\_flag = enc\_flag\_regex.search(output).group(1)

enc\_flag = base64.b64decode(enc\_flag)

return {

'n\_iters': n\_iters,

'seed': seed,

'bob\_number': bob\_number,

'iv': iv,

'enc\_flag': enc\_flag

}

if remote:

sh = pwn.remote('bitflip1.hackable.software', 1337)

else:

if anno:

sh = pwn.process('./task\_anno.py')

else:

sh = pwn.process('./task.py')

if remote:

hashcash\_regex = r'.\* Proof of Work: (.\*)\n'

output = sh.recvline(hashcash\_regex).decode()

hashcash\_cmd = re.search(hashcash\_regex, output).group(1)

print('Running hashcash cmd:', hashcash\_cmd)

hashcash\_token = subprocess.check\_output(hashcash\_cmd, shell=True).decode().strip()

print('Got hashcash token:', hashcash\_token)

sh.sendline(hashcash\_token)

if anno:

output = sh.recvuntilS('bit-flip str:')

alice\_seed = re.search(r'alice\_seed: (.\*)\n', output).group(1)

alice\_seed = bytes.fromhex(alice\_seed)

alice\_seed = bytes\_to\_long(alice\_seed)

else:

print('Starting bit flipping')

output = sh.recvuntilS('bit-flip str:')

orig\_n\_iters = send(0)[0]

print('Original number of iters:', orig\_n\_iters)

guess = 0

for i in range(1, 128):

# 3 2 1 0

# x 1 1 0

des = set\_bit(0, i) - 2

flip = guess ^ des

(x\_n\_iters, x\_seed) = send(flip)

# 3 2 1 0

# x 0 0 0

flip = guess | set\_bit(0, i)

(y\_n\_iters, y\_seed) = send(flip)

if x\_n\_iters == y\_n\_iters + 1:

# It's zero

pass

else:

guess = set\_bit(guess, i)

print('.', end='', flush=True)

print()

guesses = [guess, guess + 1]

for guess in guesses:

response = send\_get\_all(0)

print(flag\_decrypt.get\_flag(response, guess))

b'DrgnS{T1min9\_4ttack\_f0r\_k3y\_generation}