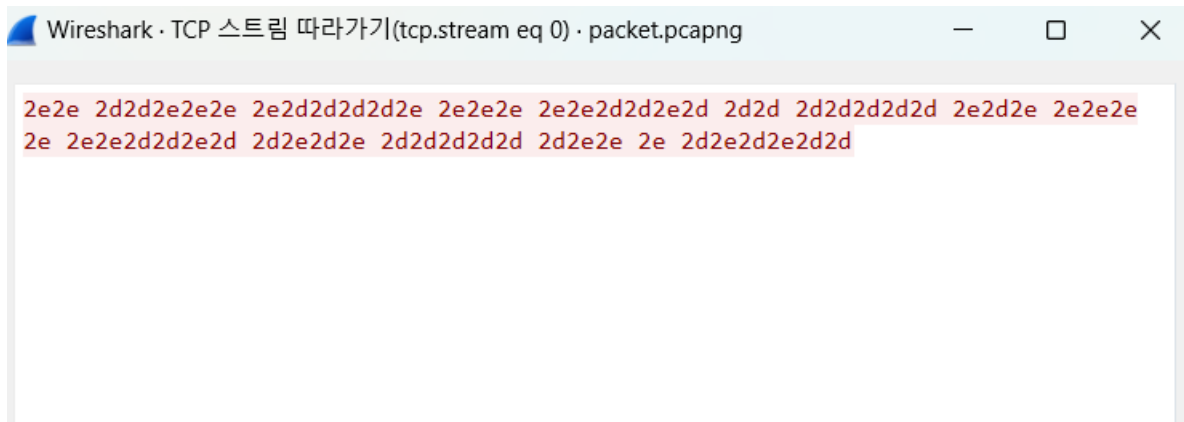


#1 돈돈돈 쓱쓱쓱 돈돈돈 (crpyto, forensic)



TCP 스트림 따라가기를 선택하면 이런 결과가 나타난다. 문제의 제목인 '돈돈돈 쓰쓰쓰 돈돈돈'은 어떤 곡에서 모스 부호를 의미하는 가사였다는 사실을 생각해냈다. 따라서 주어진 암호문을 모스 부호라 생각하고 풀어보았다. 먼저 2e를 모스부호의 '.'으로, 2d를 모스부호의 '-'으로 가정해보았다. 이 결과 '.. ---. .----.-.- - - - - - .,, ..-.- -.-, - - - - - - - ., -.-, —' 라는 결과를 얻을 수 있었고, 이를 문자로 변환하면 I7'S_MORSE_CODE! 이다. 따라서 플래그는 **3S{I7'S_MORSE_CODE!}** 임을 알 수 있다.

#2 Guess this! (crypto, misc)



검색을 통해 해당 암호가 Cistercian numerals임을 알 수 있었다. ([Cistercian numerals - Wikipedia](#))

(Cistercian digit generator (akosnikhazy.github.io))

이를 통해 해당 암호가 가리키는 숫자가 2930 2023 9108 7317 8184 1520 2638 8862 0380 5320
임을 알 수 있었으나 이후로 어떻게 해야 할지 모르겠다.

#3. Math-RSA (crypto)

파일 편집 보기



3으로 나누었을 때 2가 남고, 5로 나누었을 때 3이 남고,
7로 나누었을 때 2가 남는 정수는?

주어진 힌트는 다음과 같다. Hint.txt가 가리키는 가장 작은 정수는 23이다. 다음 파일인 rsa.txt를 보면, '?' = 3' 가 제시되어 있다. RSA 암호화는 소수 p와 q를 사용하기에 두 소수를 3과 23으로 가정하고 rsa.txt 파일의 숫자들을 복호화하는 코드를 작성했다.

```
from sympy import mod_inverse

# 주어진 소수와 암호문
p = 3
q = 23
n = p * q
phi_n = (p - 1) * (q - 1)

# 공개 지수
e = 65537

# 비밀 지수 d 계산
d = mod_inverse(e, phi_n)

def decrypt(ciphertext, d, n):
    return pow(ciphertext, d, n)

# 암호문
text1_1 = 1478962700725513601957534543632822994260624851747457593
text1_2 = 1553333156843419282597257064276240979923621010049562793
text2_1 = 9597936548531406843019430801598207447610652922253431793
text2_2 = 6048529312443343714111803981290901591081357034064453299
text3_1 = 956493083182816747924164716166355143422550221168846292
text3_2 = 8573715372353279168298618584725483148003210693982354317

# 복호화
decrypted_text1_1 = decrypt(text1_1, d, n)
decrypted_text1_2 = decrypt(text1_2, d, n)
decrypted_text2_1 = decrypt(text2_1, d, n)
decrypted_text2_2 = decrypt(text2_2, d, n)
decrypted_text3_1 = decrypt(text3_1, d, n)
decrypted_text3_2 = decrypt(text3_2, d, n)

print("Decrypted text1_1:", decrypted_text1_1)
print("Decrypted text1_2:", decrypted_text1_2)
print("Decrypted text2_1:", decrypted_text2_1)
print("Decrypted text2_2:", decrypted_text2_2)
print("Decrypted text3_1:", decrypted_text3_1)
print("Decrypted text3_2:", decrypted_text3_2)
```

공개지수 e를 가장 일반적인 숫자인 65537로 가정했을 때의 결과는 다음과 같다.

```
PS C:\Users\gram\Downloads\song> & C:/Users/gram/AppData/Local/Programs/Python/Python312/python.exe c:/Users/gram/Desktop/math.py
Decrypted text1_1: 29
Decrypted text1_2: 28
Decrypted text2_1: 68
Decrypted text2_2: 62
Decrypted text3_1: 2
Decrypted text3_2: 49
PS C:\Users\gram\Downloads\song>
```

여기에서 ASCII코드로 해독 등 여러 방법을 시도해봤는데 유의미한 결과가 나오지 않았다. 공개지수를 문제에서 주어진 숫자인 3이나 위에서 알아낸 숫자인 23을 해보는 등 여러 시도를 거쳤으나 이것 또한 유의미한 결과가 나오지 않았다.

#4. Royal Family (OSINT)



검색을 통해 해당 비행기가 에어버스 A400M 아틀라스임을 알 수 있었다. 문제의 제목과 이미지의 파일명이 royal family 라는 것에 기초해 해당 비행기 기종을 운용하는 왕립 공군을 찾아보니 영국의 왕립 공군(Royal Air Force, RAF)이 에어버스 A400M 아틀라스를 운용하고 있음을 알 수 있었다.

항공편 트래킹 사이트를 통해 RAF 가 운용하는 에어버스 A400M 아틀라스의 최근 기록들을 찾아볼 수 있었다. 이 중 문제의 조건에 맞는 기록은 다음과 같다.

RRR4140



LANDED 17d AGO

Copenhagen [CPH / EKCH]

Helsinki [HEL / EFHK]

Saturday, July 20 2024

12:48 CEST DELAYED




Saturday, July 20 2024

ON-TIME 15:14 EEST

Scheduled 12:15

Estimated 15:15

FLIGHT INFO	
AIRLINE/OPERATOR Royal Air Force	
DURATION 01h25m	DISTANCE 892 km
POSITION INFO	
LATITUDE 55.617	LONGITUDE 12.672
ALTITUDE -	GROUND SPEED 
AIRCRAFT INFO	
AIRCRAFT MODEL Airbus A400M Atlas C1	
REGISTRATION ZM403	MODE-S 43C5DD

FLIGHT ACTIVITY									
DATE	ORIGIN	STD	ATD	DESTINATION	STA	AIRCRAFT	DELAY	STATUS	DURATION
2024 20 Jul	Copenhagen (CPH/EKCH)	12:15 CEST	12:48 CEST	Helsinki (HEL/EFHK)	15:15 EEST	A400 (ZM403)		Landed 15:14 EEST	01h25m

이 정보에 따르면 플래그는 3S{a400_ZM403_01:26_13:49_15:15} 라 할 수 있겠지만 서버가 닫힌 관계로 답인지는 알 수 없다.

#5. what' s This Song

일단 노래를 암호화시킨 malware.py 파일의 코드를 분석해 보았을 때 해당 파일의 코드는 이 코드는 입력 텍스트 파일(input.txt)을 읽고, 각 문자를 16 진수로 인코딩한 다음, 이를 미리 섞은 16 진수 문자 집합으로 다시 매핑하여 결과를 출력 파일(output.txt)에 저장하며, 이 과정에서 난수를 사용하여 매핑을 무작위로 섞음으로써 인코딩 결과를 예측 불가능하게 만든다는 것을 알 수 있었다.

이를 복호화하기 위해 다음과 같은 코드를 짰다.

```
def get_seed(size):
    return int(os.urandom(size).hex(), 16)

def frequency_analysis(text):
    frequency = collections.Counter(text)
    most_common = frequency.most_common()
    return most_common

def decode_cipher(cipher_text, malware, crypto):
    output = bytearray()
    for i in range(0, len(cipher_text), 2):
        encoded_char = cipher_text[i:i+2]
        decoded_char = crypto[malware.index(encoded_char[0])] + crypto[malware.index(encoded_char[1])]
        output.extend(bytes.fromhex(decoded_char))
    return output

salt = get_seed(16)
random.seed(salt)

crypto = "fedcba9876543210"
malware = list(crypto)
random.shuffle(malware)
malware = ''.join(malware)

with open("output.txt", "r") as encrypted_file:
    encrypted_input = encrypted_file.read()

frequencies = frequency_analysis(encrypted_input)
print("빈도 분석:", frequencies)

most_common_encrypted = frequencies[0][0]
most_common_decrypted = 'e'
```

```
print(f"{most_common_encrypted} {most_common_decrypted}")

decrypted_output = decode_cipher(encrypted_input, malware, crypto)

with open("decrypted.txt", "wb") as decrypted_file:
    decrypted_file.write(decrypted_output)

print("복호화된 암호문:", decrypted_output.decode(errors='replace'))
```

문제에서 주어진 힌트인 빈도 분석과 ASCII 를 사용하며 다음과 같은 동작을 수행하는 코드를 작성하려 했다.

- 암호화된 텍스트와 원본 텍스트에서 가장 빈도가 높은 문자를 찾아 매핑을 추측
- 매핑을 통해 각 암호화된 문자 쌍을 다시 ASCII 문자로 변환

그러나 해당 코드를 통해 유의미한 결과가 나오지 않았다.