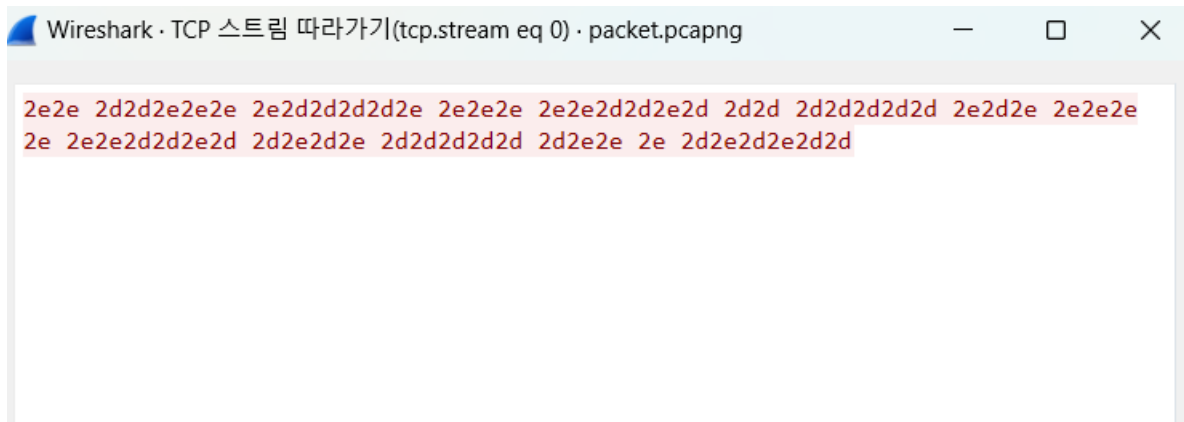


#1 돈돈돈 쓰쓰쓰 돈돈돈 (crpyto, forensic)



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.

#6. DUM DUM :p

▼ 오늘			
 dump.bin	2024-08-18 오후 1:47	BIN 파일	9KB

압축을 풀면 dump.bin 이라는 파일이 있다.

```
(kali@kali)-[~/Desktop]
$ binwalk dump.bin
```

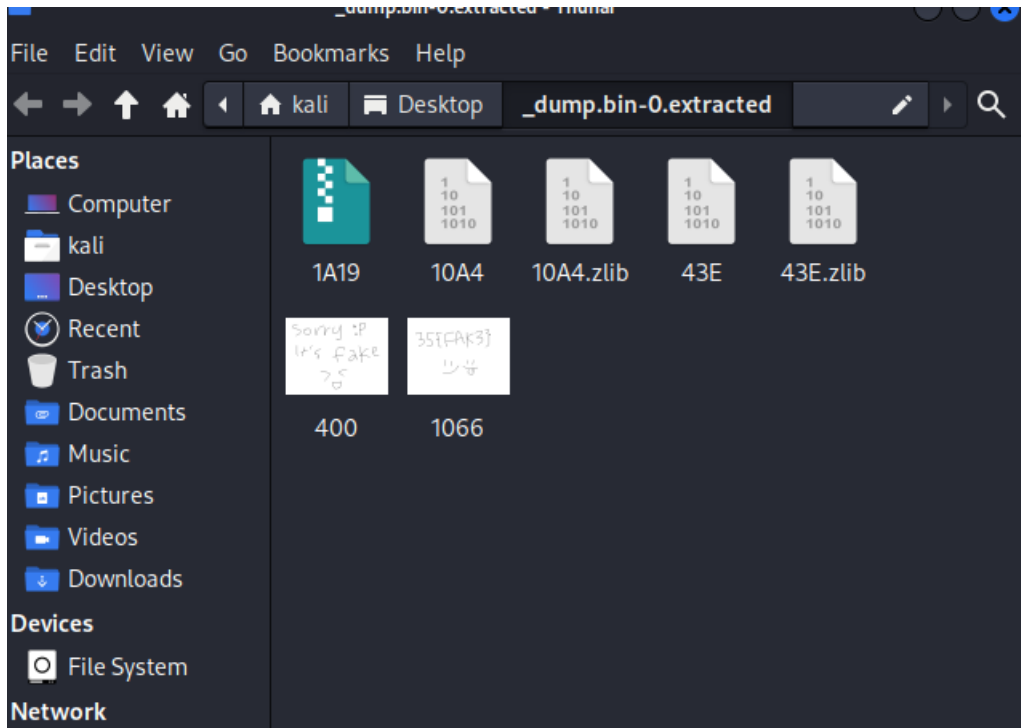
DECIMAL	HEXADECIMAL	DESCRIPTION
1024	0x400	PNG image, 400 x 300, 8-bit/color RGBA, non-interlaced
1086	0x43E	Zlib compressed data, default compression
4198	0x1066	PNG image, 400 x 300, 8-bit/color RGBA, non-interlaced
4260	0x10A4	Zlib compressed data, default compression
6681	0x1A19	7-zip archive data, version 0.3

이 파일을 칼리 리눅스에 가져가 파일 분석 및 데이터 추출 명령어인 binwalk 를 사용하여 열어보았다. 여러 이미지 파일, 압축 파일들이 있음을 확인할 수 있다.

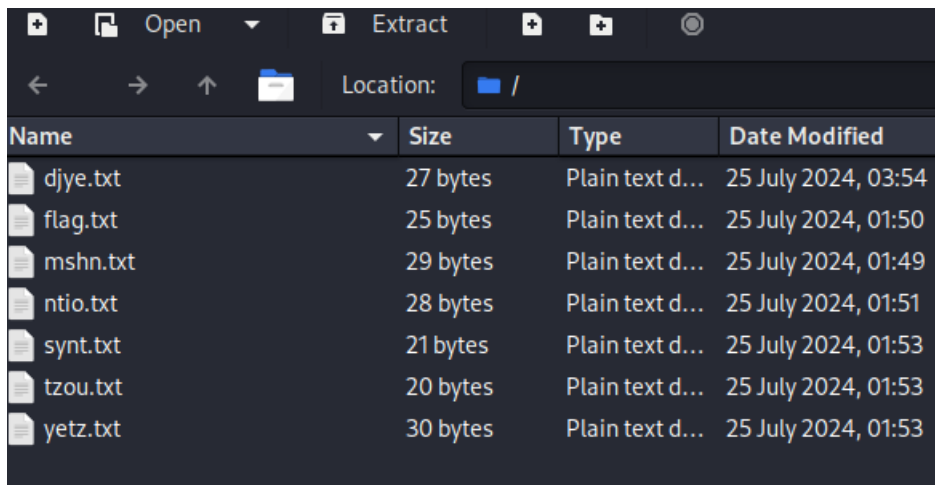
```
(kali@kali)-[~/Desktop]
$ binwalk --dd=".*" dump.bin
```

DECIMAL	HEXADECIMAL	DESCRIPTION
1024	0x400	PNG image, 400 x 300, 8-bit/color RGBA, non-interlaced
1086	0x43E	Zlib compressed data, default compression
4198	0x1066	PNG image, 400 x 300, 8-bit/color RGBA, non-interlaced
4260	0x10A4	Zlib compressed data, default compression
6681	0x1A19	7-zip archive data, version 0.3

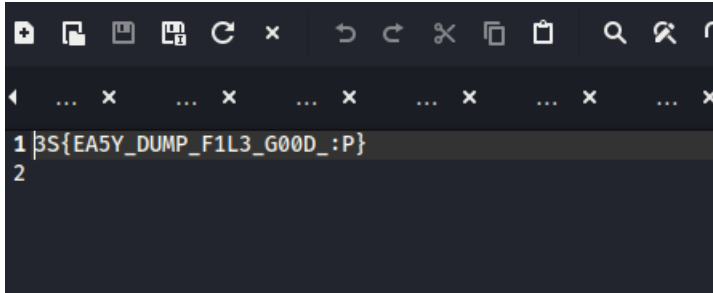
dump.bin 파일 내에서 발견된 모든 서명(즉, 모든 파일 유형이나 데이터 블록)에 대해 해당 데이터를 자동으로 추출하고 별도로 저장하는 명령어 binwalk -dd="*" 를 사용하여 칼리 리눅스의 바탕화면에 dump.bin 파일 안의 데이터들이 들어있는 파일을 생성하였다.



여기서 압축 파일을 다시 열어보기로 했다.




압축 파일 안에는 여러 텍스트 파일들이 있다.



이중 diye.txt 파일을 열어보면 **3S{EA5Y_DUMP_F1L3_G00D_:P}** 라는 플래그를 얻을 수 있다.

#7. Can solve without thinking

이름	수정한 날짜	유형	크기
▼ 오늘			
 main	2024-08-18 오후 2:21	파일	31,983KB

압축을 풀면 이런 파일이 나온다.

```
(kali@kali)-[~/Desktop]
$ ./main
pygame 2.6.0 (SDL 2.28.4, Python 3.10.12)
Hello from the pygame community. https://www.pygame.org/contribute.html
MESA-LOADER: failed to open zink: /usr/lib/dri/zink_dri.so: cannot open shared object file: No such file or directory (search paths /usr/lib/x86_64-linux-gnu/dri:\${ORIGIN}/dri:/usr/lib/dri, suffix _dri)
failed to load driver: zink
MESA-LOADER: failed to open swrast: /usr/lib/dri/swrast_dri.so: cannot open shared object file: No such file or directory (search paths /usr/lib/x86_64-linux-gnu/dri:\${ORIGIN}/dri:/usr/lib/dri, suffix _dri)
failed to load driver: swrast
X Error of failed request: BadValue (integer parameter out of range for operation)
Major opcode of failed request: 152 (GLX)
Minor opcode of failed request: 3 (X_GLXCreateContext)
Value in failed request: 0x0
Serial number of failed request: 166
Current serial number in output stream: 167
```

칼리 리눅스에서 main 파일을 열면 이렇게 나온다. 어떻게 할지 잘 모르겠어서 일단 나와 있는 링크인 <https://www.pygame.org/contribute.html> 에 접속해보았다.

CONTRIBUTE — WIKI

Hello,

welcome, and thanks for thinking about contributing.

New to contributing to Open Source Free Libre software? There is a draft of [Let's write a unit test!](#) which is a step by step guide on how to write your first unit test in python for pygame, which is very similar to how you would do it for other projects.

Want or need to [compile pygame](#) from source? [Head here](#) for platform specific instructions

Developer guide: or "How to hack on pygame": [Hacking](#)

Issues labeled "[good first issue](#)" are ones that should be easy for people to start with.

Other ways to talk to us: [info](#)






How to submit changes: [patchesandbugs](#)

Our github project page, where you can take a look at what we're doing on: <https://github.com/pygame/>

그러면 이런 페이지가 뜨고 깃허브 링크로 접속해보면 파이썬을 이용한 게임 프로젝트인 것 같은데 여기서 어떻게 플래그를 얻어내야 하는지는 잘 모르겠다.

#8. Sick xss

▼ 오늘

 app.py	2024-08-18 오후 4:37	Python 원본 파일	3KB
 flag.txt	2024-08-18 오후 4:37	텍스트 문서	1KB
 requirements.txt	2024-08-18 오후 4:37	텍스트 문서	1KB
 templates	2024-08-18 오후 4:37	파일 폴더	
 static	2024-08-18 오후 4:37	파일 폴더	

압축 파일을 열면 여러 파일들이 있다. 이중 app.py 라는 파일을 열어보았다. 웹페이지에 대한 파일로 보인다. 문제의 제목을 보았을 때 XSS 공격을 시도하면 플래그 값을 얻을 수 있을 것 같은데 코드로 여러 가지를 해보았지만 유의미한 결과를 얻을 수 없었다.

#9. CUTE-TIGER

링크를 따라 들어가보니 티스토리 게시물과 이 이미지 파일이 있었다.



먼저 파일을 헥스 에디터로 열어보았다.

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01
00000010 00 01 00 00 FF DB 00 43 00 04 04 04 04 04 04 04
00000020 04 04 04 06 06 05 06 06 08 07 07 07 07 08 0C 09
00000030 09 09 09 09 0C 13 0C 0E 0C 0C 0E 0C 13 11 14 10
00000040 0F 10 14 11 1E 17 15 15 17 1E 22 1D 1B 1D 22 2A
00000050 25 25 2A 34 32 34 44 44 5C FF DB 00 43 01 04 04
00000060 04 04 04 04 04 04 04 04 06 06 05 06 06 08 07 07
00000070 07 07 08 0C 09 09 09 09 09 0C 13 0C 0E 0C 0C 0E
00000080 0C 13 11 14 10 0F 10 14 11 1E 17 15 15 17 1E 22
00000090 1D 1B 1D 22 2A 25 25 2A 34 32 34 44 44 5C FF C0
000000A0 00 11 08 02 4D 01 FB 03 01 22 00 02 11 01 03 11
000000B0 01 FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00
000000C0 00 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09
000000D0 0A 0B FF C4 00 B5 10 00 02 01 03 03 02 04 03 05
000000E0 0F 04 04 00 00 01 0F 01 00 00 00 01 11 0F 10 01
```

시그니처도 올바르게 들어가 있고, 딱히 플래그라 할 부분은 찾을 수 없었다. 스테가노그래피 툴, 파일을 칼리 리눅스에 옮긴 뒤 binwalk 명령어 사용, 확장자를 txt 로 바꾸기 등 여러 방법을 시도해 봤지만 유의미한 결과를 얻을 수 없었다.

#10. PokWemoN

```
File Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)~[~/Desktop]
$ ./main
```

압축을 풀고 나온 main 파일을 칼리 리눅스에서 실행해보았다. 포켓몬 게임을 하는 프로그램인 것 같다.

```
Be the PokWemoN champion!
Choose your Pokwemon's type (0.Normal, 1.Fire, 2.Water, 3.Grass, 4.Electric,
5.Ice, 6.Fighting, 7.Poison, 8.Ground, 9.Flying, 10.Psychic, 11.Bug, 12.Rock,
13.Ghost, 14.Dragon, 15.Dark, 16.Steel, 17.Fairy): 14

Your PokWemoN type is Dragon

===== Battle 1 =====

===== Round 1 =====
Opponent Pokwemon(Dragon)'s current health: 10 | damage: 10
My Pokwemon(Dragon)'s current health: 30 | damage: 10

Choose to Attack(a), Basic Attack(b), or Defend(d): █
```

포켓몬 타입은 드래곤을 골랐다. 공격 / 기본 공격 / 방어를 선택하여 전투하는 방식인 것 같다.

```
===== Round 3 =====
Opponent Pokwemon(Rock)'s current health: 20 | damage: 10
My Pokwemon(Dragon)'s current health: 1 | damage: 10

Choose to Attack(a), Basic Attack(b), or Defend(d): a
Opponent Pokwemon attacked my Pokwemon! Damage: 10
My Pokwemon attacked the opponent Pokwemon! Damage: 10

== Health Bars ==
Opponent Pokwemon(Rock): =====
My Pokwemon(Dragon): █

My Pokwemon has fainted! Opponent Pokwemon wins.
```

전투는 턴제 전투인 기존 포켓몬과 비슷한 듯하다. 하지만 이 게임에서 어떻게 플래그를 얻어내야 하는지는 잘 모르겠다. 또한 게임의 구조 상 모두 이기고 플래그를 얻어내는 방법도 불가능할 듯하다.

#11. lucky_cook

```
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)-[~/Desktop]
$ ./lucky_cook
Let's Make Your Own FOOD!
You should make food start with "Y"!
But, this is Unlucky word.I'll give you a chance.
Try with 'D'.
Donut
Donut
Dim sum
Dim sum
dumpling
dumpling
donut
donut
```

압축을 풀고 안의 lucky_cook 파일을 실행해보면 이런 화면이 나온다. D로 시작하는 음식을 입력하라는 것 같은데 아무리 입력해보도 같은 입력 결과를 출력하기만 한다. 여기서 뭘 해야하는지는 잘 모르겠다.

#12. Brob

```
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)-[~/Desktop]
$ ./brob
$*****A*****
```

먼저 다운받은 파일을 칼리 리눅스에서 열어보았다. 의미를 모르겠는 문자들만이 나온다. 포너블 문제라서 관련 툴을 찾아봤지만 잘 모르겠다. 포너블 문제를 풀려면 관련 지식이 더 필요할 것 같다.

#13. Untitled

일단 file.php 파일을 열어보았다.

```
1 <?php
2 $flag = "3S{**REDACTED**}";
3 ?>
```

플래그 값을 정의하는 코드이다. flag 값은 REDACTED 로 감춰져 있음을 알 수 있다.

```
<?php

if(preg_match("/^.*(flag|etc|test|passwd|group|shadow|php|ini|sql|index).*$\/im",$_GET['test'])){
    echo "hi";
}else if($_GET['test'] === "phpinfo"){
    phpinfo();
}else{
    include "../".str_replace(" ", "",$_GET['test']).".php";
    echo $flag;
}

?>
```

Index.php 파일을 열어보았다. 이 코드는 사용자가 GET 요청으로 전달한 매개변수 test 에 따라 동작을 달리하는 웹 애플리케이션의 일부분이다. preg_match 함수는 정규 표현식을 사용하여 특정 문자열이 매개변수 test 에 포함되어 있는지를 확인한다.

"/^.*(flag|etc|test|passwd|group|shadow|php|ini|sql|index).*\$\/im"는 다음과 같은 패턴을 찾는다: flag, etc, test, passwd, group, shadow, php, ini, sql, index 중 하나라도 포함되어 있다면 조건이 참이 된다. i 플래그는 대소문자를 구분하지 않음을 의미하고, m 플래그는 여러 줄에 걸쳐 일치 여부를 확인한다.

만약 이 패턴이 test 매개변수 내에서 발견되면, "hi"를 출력한다.

Else if 블록은 test 매개변수의 값이 문자열 "phpinfo"와 정확히 일치하면 phpinfo() 함수를 호출한다.

test 매개변수가 위의 조건들에 일치하지 않는 경우, else 블록이 실행된다.

이렇게 코드를 분석해봤지만 여기서 더 어떻게 해야하는지는 잘 모르겠다.

#14. Find Spade

```
[+] ♠ == UNICODE(0x2660)

[?] A + 0x25eb == ♠
[?] 0x2674 - B == ♠
[?] 0x26d1 - C == ♠
[?] (D * 0x72) - 0xf10 == ♠
[?] 0x25 * (E - B) + 0x195e == ♠
[?] 0x200 * (F >> 1) + 0x1460 == ♠
[?] 0x75 * (G << 4) - 0x34720 == ♠
[?] C + H + 0x258e == ♠
[?] (I + 3) ^ 0x2635 == ♠
[?] (J ^ A) + 0x2644 == ♠

[+] Flag: 3S{ASCII(ABCDEFGHJIJ)}
```

압축을 풀고 실행 프로그램을 실행하면 이렇게 나온다. A 부터 까지의 숫자나 문자를 모두 알아낸 다음 아스키 코드로 다시 치환하여 플래그를 알아내야 할 것 같다.

$$A + 0x25eb == 0x2660$$

$$A = 0x2660 - 0x25eb = 0x0175 = 373$$

$$0x2674 - B == 0x2660$$

$$B = 0x2674 - 0x2660 = 0x0014 = 20$$

$$0x26d1 - C == 0x2660$$

$$C = 0x26d1 - 0x2660 = 0x0071 = 113$$

$$(D * 0x72) - 0xf10 == 0x2660$$

$$D * 0x72 = 0x2660 + 0xf10 = 0x3570$$

$$0x25 * (E - B) + 0x195e == 0x2660$$

$$0x25 * (E - 20) + 0x195e = 0x2660$$

$$0x25 * (E - 20) = 0x2660 - 0x195e = 0x0062 = 98$$

$0x200 * (F \gg 1) + 0x1460 == 0x2660$

$0x200 * (F \gg 1) = 0x2660 - 0x1460 = 0x1200 = 4608$

$F = 18$

$0x75 * (G \ll 4) - 0x34720 == 0x2660$

$0x75 * (G \ll 4) = 0x2660 + 0x34720 = 0x36d80 = 224384$

$C + H + 0x258e == 0x2660$

이렇게 계속해서 계산해보았지만 플래그에 맞는 값이 잘 나오지 않는 것 같다. 계속해서 다른 라이트업과 다른 값만 나온다.

#15. Python Executor

```
from flask import Flask, request
import os, subprocess, string

app = Flask(__name__)
app.config["SECRET_KEY"] = os.urandom(32)

@app.route("/", methods=["GET"])
def index():
    base_command = ["python"]
    file = list(request.args.keys())

    bannlist = []
    for i in string.ascii_lowercase:
        bannlist += ["-"+i+" "]

    base_command += file
    base_command[1] += ".py"

    My_Injection_filter = ['$','{','}','|','&',';','\n','!','?','=','*','`',' ','"', 'flag', 'system']+bannlist

    for i in My_Injection_filter:
        if i in ' '.join(base_command):
            return "NOPE! XD"+i, 500

    if not os.path.exists(base_command[1]):
        return "?", 404

    res = subprocess.run(base_command, capture_output=True, text=True)

    return res.stdout[:5]

app.run(host="0.0.0.0", port=9999)
```

먼저 코드를 분석해보자. 이 Flask 웹 애플리케이션은 사용자가 GET 요청으로 전달한 파일 이름에 해당하는 Python 스크립트를 서버에서 실행하고, 그 출력의 처음 5 글자를 반환하는 역할을 한다. 코드 내에서 명령어 인젝션을 방지하기 위해 일부 문자와 플래그를 필터링하며, 요청된 Python 파일이 존재하지 않으면 404 오류를 반환한다.

이렇게 했지만 이 다음부터는 어떻게 할지는 잘 모르겠다.

#16. BrokenHearted

일단 파일에 들어가있는 hint.txt를 열어보니 파일 카빙을 해야하는 문제인 것 같다. 찾아보니 파일 카빙(File Carving)은 디지털 포렌식에서 사용되는 기술로, 손상되거나 파일 시스템 정보가 손실된 데이터 저장 장치에서 특정 파일이나 데이터를 복구하는 방법을 말한다고 한다.



들어있는 CTF.jpg 는 다음과 같은 파일이다. 이 파일을 칼리 리눅스에 옮겨 파일 카빙 과정을 거치려 했으나 자꾸 오류 메시지가 떠 실패했다.

#17. Wat ch??

먼저 압축을 풀어 나온 apps.sw 폴더를 칼리 리눅스에 옮겨 하려고 문제를 풀려고 했으나 오류 메시지가 떴다. 그래서 폴더를 vscode 에서 열기로 했다.

```
8 2023/10/29 22:38:18:447 __stopAfterNSeconds
9 2023/10/29 22:38:18:447 setRequest Send response for com.samsung.w-reminder-appcon
0 2023/10/29 22:38:25:339 _onStopWorker
1 2023/10/29 22:38:25:397 _checkAppVersion version=1.6.31, updateVersion=1.6.31
2 2023/10/29 22:38:25:525 __unsetTimerForGearNotiAck
3 2023/10/29 22:38:25:525 __unsetTimerForUpdateGearAck
4 2023/10/29 22:45:39:581 getOriginOfLaunchingFromCurrentRequestBT off gear change r
5 2023/10/29 22:45:39:581 start
6 2023/10/29 22:45:39:594 __onStartWipcAgent
7 2023/10/29 22:45:39:594 __stopAfterNSeconds
8 2023/10/29 22:45:39:594 getOriginOfLaunchingFromCurrentRequestBT off gear change r
9 2023/10/29 22:45:39:595 __onReceiveBtOffGearChangeRequest
0 2023/10/29 22:45:39:595 __sendUpdateInd
1 2023/10/29 22:45:39:618 __stopAfterNSeconds
2 2023/10/29 22:45:39:618 setRequest Send response for com.samsung.w-reminder-widgett
3 2023/10/29 22:45:47:332 _onStopWorker
4 2023/10/29 22:45:47:383 _checkAppVersion version=1.6.31, updateVersion=1.6.31
5 2023/10/29 22:45:47:494 __unsetTimerForGearNotiAck
6 2023/10/29 22:45:47:495 __unsetTimerForUpdateGearAck
7 2023/10/29 22:45:48:764 getOriginOfLaunchingFromCurrentRequestBT off gear change r
8 2023/10/29 22:45:48:764 start
9 2023/10/29 22:45:48:888 __onStartWipcAgent
0 2023/10/29 22:45:48:888 __stopAfterNSeconds
1 2023/10/29 22:45:48:888 getOriginOfLaunchingFromCurrentRequestBT off gear change r
2 2023/10/29 22:45:48:888 __onReceiveBtOffGearChangeRequest
3 2023/10/29 22:45:48:889 __sendUpdateInd
4 2023/10/29 22:45:48:965 __stopAfterNSeconds
5 2023/10/29 22:45:48:965 setRequest Send response for com.samsung.w-reminder-appcon
6 2023/10/29 22:45:56:332 _onStopWorker
7 2023/10/29 22:45:56:397 _checkAppVersion version=1.6.31, updateVersion=1.6.31
8 2023/10/29 22:45:56:521 __unsetTimerForGearNotiAck
9 2023/10/29 22:45:56:521 __unsetTimerForUpdateGearAck
0
```

리마인더 관련 폴더인 com.samsung.w-reminder 안의 reminderConsumerLog 파일에서 3 번째 리마인더 작성 시간을 알아낼 수 있었다.

```
com.samsung.w-music-player > data > ≡ local_queue_data
1 ia/Sounds/Over the Horizon.mp3", "id": "ca7a55d6-978
```

음악 파일명은 com.samsung.w-music-player 에 들어가서 local_queue_data 파일에서 over the horizon.mp3 라는 음악 파일명을 찾을 수 있었다.

여기까진 알아내었지만 4 번째 질문과 5 번째 질문의 답은 잘 모르겠다.

#18. Minjuns_travel_2

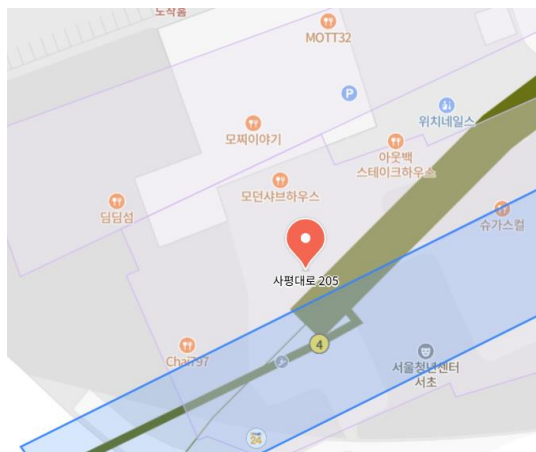
민준이가 술술 입대 날짜가 다가와 준비를 하러 본가에 내려
가려 합니다. 가는데 오래 걸리니 무엇을 먹어야 되겠어요.

(37.503,127.0037)

어느 음식점에 갈지 맞춰보세요!

(띄어쓰기 X, 영어 대문자)

첫 번째 pdf 다. 숫자들은 당연히 좌표일 것이라 생각하고 지도앱에 숫자를 입력했다.

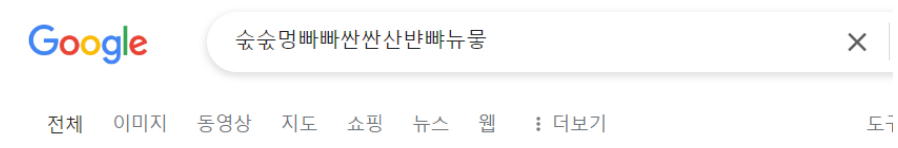


여러 식당들이 나온다. 여러 식당들의 이름을 입력해보다가 DIMDIMSUM 을 입력하니 2 번째 pdf
가 열린다.

딴섬은 정말 맛있었습니다!(FLAG = "3S{thank_"})

드디어 본가에 도착해서 택시를 기다리고 있는데 어느 외국
인이 저한테 무엇을 물어보네요

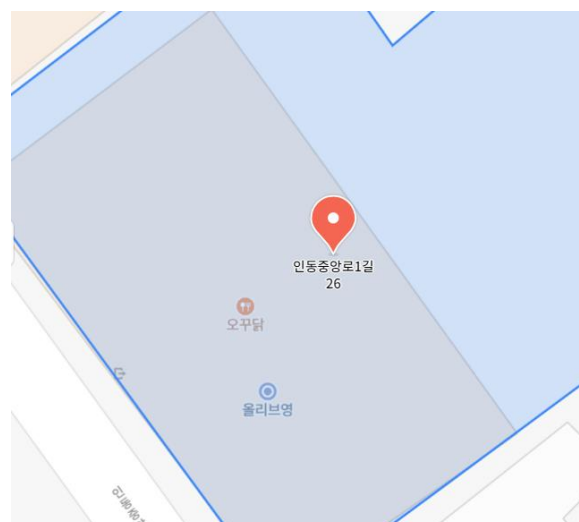
슌슌똥똥쌔쌔산산반똥뉴똥
똥똥따또똥똥똥똥똥똥
슌똥똥똥똥똥똥똥똥똥
똥똥똥똥똥똥똥똥똥똥
똥똥똥똥똥똥똥똥똥똥
슌똥똥똥똥똥똥똥똥똥
쌔똥똥똥똥똥똥똥똥
똥똥똥똥똥똥똥똥똥
똥똥똥똥똥똥똥똥똥



구글링을 통해 정체불명의 문자열은 '아희' 라는 프로그래밍 언어이며, 문제의 문자열은 해당 언어로 구구단을 쓴 것임을 알 수 있었다. 그래서 'GUGUDAN' 이 암호임을 알 수 있다.









왜 나한테 구구단 코드를 말하지?? 이상한 사람이네요
빨리 도망갑시다(FLAG = "you_")
지금 시간 오후 2시 반, 시간이 많이 남아 집 가는 길에 군
대에서 꼭 필요한 선크림을 사러가야겠습니다.
마침 근처에 있네요! 완전 럭키비키잖아~
(36.1082, 128.4188)
그러한 피부 관련 제품들을 모르다보니 사람들이 많이 쓰는
것을 사야겠네요.
오! 마음에 쏙 드는 제품을 찾았어요
이것은 어느 브랜드일까요?
(띄어쓰기 X, 한글)

다시 지도에 해당 좌표를 입력한다.



올리브영 공식 사이트에서 선크림 제품들을 찾아보기로 했다.

선케어에서 많이 본 상품이에요

 <p>[선착순 팩키트증정] 브링그린 알로에99%... 13,600원- 12,200원 -</p> <p>세일 증정 오늘드림</p>	 <p>[8월 올영픽/1+] 에스트라 더마UV365 장벽수분... 31,000원 28,900원</p> <p>세일 증정 오늘드림</p>	 <p>[규진PICK/파데프리 선크림/8월 올영픽]아누아... 28,000원 19,500원</p> <p>세일 오늘드림</p>	 <p>[8월 올영픽][무기자차/ 콜라보키링증정] 닥터지... 29,000원 26,150원</p> <p>세일 쿠폰 증정 오늘드림</p>
 <p>[구달X연작귀][1+기획] 구달 맑은 여성초 진정 수... 20,000원 17,500원</p> <p>세일 증정 오늘드림</p>	 <p>식물나라 워터프루프 선 스프레이 1+ 기획 14,800원 13,900원</p> <p>세일 증정 오늘드림</p>	 <p>[최혜선 PICK/등센] 라운드랩 자작나무 수분... 25,000원 17,500원</p> <p>세일 증정 오늘드림</p>	 <p>[1등선스틱/노세범] AHC 마스터즈 에어리치 선스틱... 21,000원 17,900원</p> <p>세일 오늘드림</p>

해당 랭킹에서 하나씩 입력하다 보니 암호가 '식물나라' 를 그대로 영타로 친 것임을 알 수 있었다.

오! 저랑 마음이 잘 통하셨네요 (FLAG = "for_")

마블 영화를 아주 좋아하는 데 이번에 보는 영화가 마지막이
gett네요.. (; '^)

영화를 다 보고 친구들과 약속을 잡으려 합니다.

언제로 시간을 알려주면 좋을까요?

(숫자 4자리, ex) 오후 3시 23분 -> 1523)

이전 pdf 에서의 시간이 2 시 30 분이었으니 선크림을 사는 데에 걸린 시간 + 제일 최근에 개봉한
마블 영화의 러닝타임으로 구하면 된다. 가장 최근에 개봉한 마블 영화인 <데드풀과 울버린>의
러닝타임은 2 시간 7 분이다. 선크림을 사는 데에 걸린 시간을 정확히 알아낼 수 없어 1 분부터 쪽
입력해보니 암호는 오후 5 시 2 분, 즉 1702 임을 알 수 있었다.

눈썰미가 좋으시네요! (FLAG = "participating_")

친구와 노는데 뭔가 숨기는 게 있는 것 같습니다.

핸드폰 비번을 해킹해서 알아봅시다!



핸드폰 배경화면에 포켓몬들이 있네요!

뭔가 힌트가 있을 것 같은데.. 흠

포켓몬과 관련된 숫자는 포켓몬들에게 붙어있는 고유의 도감 번호들이다. 화면에 나와있는 각 포켓몬들의 도감 번호는 타부자고 1000, 체리꼬 421, 코코리 231, 가디안 282, 레시라무 643, 레지스틸 379 이다. 이 번호들을 모두 더하면 암호가 2956 임을 알 수 있다.

오 포켓몬을 잘 아시는군요!

포너블 포켓몬 문제도 풀으셨겠죠?(FLAG = "this_")



메모장을 열어보니 이렇게 있네요...

구글링을 해보니 프로그래밍 언어 중 Brainfuck 이라는 언어였다.

```
ahh crimescene is funny. I watch this tv show when i make this  
pdf. harumane 3 munjae mandeum. so tired. soju jom geuman  
mukgo sip a. anyway this munjae is funny? jiguem no halmal but  
i want big code. so i say amumal. thank you|
```

디코딩하면 이런 메시지가 나온다. 암호로 CRIMESCENE 을 입력하였다.

크라임씬! 정답입니다~ 여러분들 좋아하시나요?

(FLAG = "3sctf_wjfeoahtEofuakwcnj}")

모든 문제를 다 푸셨습니다 축하드려요!

여러분들의 보안 인생 언제나 잘 풀리셨으면 좋겠습니다.

수고하셨습니다

나는 2년동안 잊어지지만...

지금까지 얻은 모든 플래그 값들을 합쳐

3S{thank_you_for_participating_this_3sctf_wjfeoahtEofuakwcnj} 가 플래그임을 알아낼 수 있다.