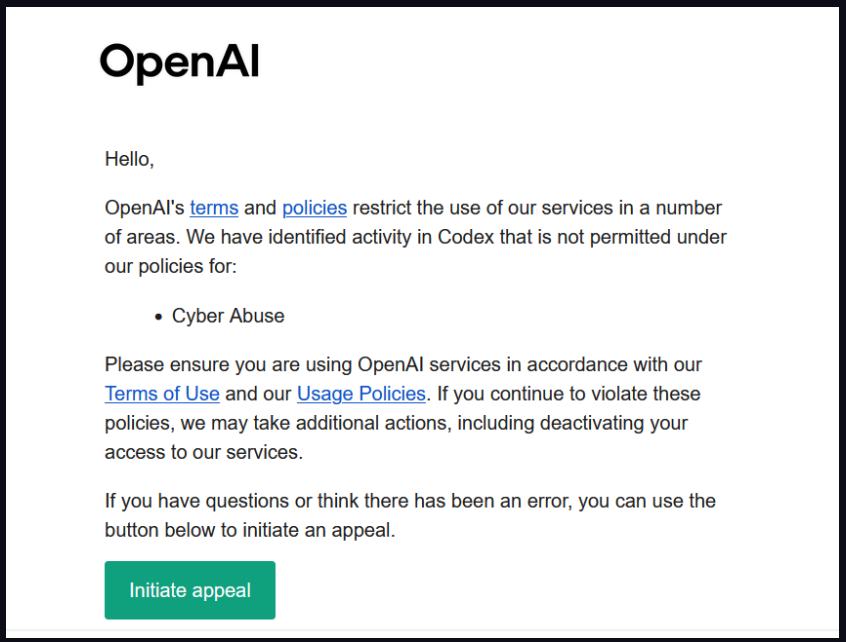


Write Up Quals

ITFest 2026

Team **HiB - kalo pesertanya no ai kira-kira probsetnya no ai jg ngga ya?**



OpenAI

Hello,

OpenAI's terms and policies restrict the use of our services in a number of areas. We have identified activity in Codex that is not permitted under our policies for:

- Cyber Abuse

Please ensure you are using OpenAI services in accordance with our Terms of Use and our Usage Policies. If you continue to violate these policies, we may take additional actions, including deactivating your access to our services.

If you have questions or think there has been an error, you can use the button below to initiate an appeal.

Initiate appeal

2byte . AriqArdian . zenc

Table of Contents

Table of Contents **1**
Cryptography 2
Clockwork (119 Points) 2
Prime Era (198 Points) 16
Nguyen++ (368 Points) 29
Digital Forensic 42
petani durr (105 Points) 42
durja otw magang (220 Points) 🔥🔥🔥 49
Traitor (247 Points) 53
Miscellaneous 58
Kingdom of Catofel (102 Points) 58
Mr. Durrr the Traveler (166 Points) 61
The Murder of Alfred Cuthbert (107 Points) 63
Reverse Engineering **65**
rekcehcglaf (166 Points) 65
Web Exploitation **74**
antinrml (104 Points) 74

Cryptography

Clockwork (119 Points)

Every signature is unique, but the backup signer keeps the same rhythm after boot. Help Mr. Dur study these samples and recover something.

Author: Kae

Jadi pada challenge ini kita diberikan sebuah file chall.py dengan isi berikut.

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

from Crypto.Util.number import isPrime
import json
import math
import os
import sys

with open("flag.txt","rb") as f:
    FLAG = f.read().rstrip()

MIN_BITS = 3700
MAX_BITS = 4096
MAX_INPUT_BYTES = 32 * 1024

def send(data):
    print(json.dumps(data), flush=True)

def get_primes(count):
    primes = []
    number = 2

    while len(primes) < count:
        if all(number % p != 0 for p in primes if p * p <= number):
            primes.append(number)

        number = 3 if number == 2 else number + 2

    return primes

def miller_rabin(n, base):
    if n < 2 or n % 2 == 0:
        return n == 2
```

```
    d = n - 1
    s = 0

    while d % 2 == 0:
        d //= 2
        s += 1

    x = pow(base, d, n)

    if x == 1 or x == n - 1:
        return True

    for _ in range(s - 1):
        x = pow(x, 2, n)

        if x == n - 1:
            return True

    return False


def parse_number(value, name):
    text = str(value)

    if not text.isascii() or not text.isdigit():
        raise ValueError(f"{name} must be a positive decimal integer")

    if len(text) > len(str(1 << MAX_BITS)):
        raise ValueError(f"{name} is too large")

    return int(text)


def main():
    print(f"Hi, welcome to ITFest 2026")
    print(f"goodluck create your own RSA")

    e = 65537
    bases = get_primes(150)
    message = int.from_bytes(FLAG, "big")

    line = sys.stdin.buffer.readline(MAX_INPUT_BYTES + 1)

    if not line:
        send({"status": "error", "error": "no submission received"})
        return

    if len(line) > MAX_INPUT_BYTES:
        send({"status": "error", "error": "submission is too large"})
        return
```

```
    try:
        data = json.loads(line)

        if not isinstance(data, dict):
            raise ValueError("submission must be a JSON object")

        n = parse_number(data.get("n"), "n")
        raw_factors = data.get("factors")

        if not isinstance(raw_factors, list) or len(raw_factors) != 3:
            raise ValueError("factors must contain exactly three integers")

        factors = [
            parse_number(value, f"factors[{i}]")
            for i, value in enumerate(raw_factors)
        ]

        if not MIN_BITS <= n.bit_length() <= MAX_BITS:
            raise ValueError(f"n must be between {MIN_BITS} and {MAX_BITS} bits")
        if n % 2 == 0:
            raise ValueError("n must be odd")
        if len(set(factors)) != 3 or min(factors) <= 1:
            raise ValueError("factors must be distinct and greater than one")
        if math.prod(factors) != n:
            raise ValueError("factors do not multiply to n")
        if not all(isPrime(factors) for factors in factors):
            raise ValueError("all factors must be prime")
        if not all(miller_rabin(n, base) for base in bases):
            raise ValueError("n failed Miller-Rabin")

        lambda_n = math.lcm(*(factor - 1 for factor in factors))

        if math.gcd(e, lambda_n) != 1:
            raise ValueError("e is not invertible modulo lambda")

        if message >= n:
            raise ValueError("n is too small for the flag")

        send({
            "status": "ok",
            "n": n,
            "e": e,
            "c": pow(message, e, n),
        })

    except (ValueError, TypeError) as error:
        send({"status": "error", "error": str(error)})

if __name__ == "__main__":
```

```
    main()
```

Maka disini saya mencoba memahami isi dari kode tersebut bagaimana cara kode tersebut bekerja.

```
def send(data):
    print(json.dumps(data), flush=True)

def get_primes(count):
    primes = []
    number = 2

    while len(primes) < count:
        if all(number % p != 0 for p in primes if p * p <= number):
            primes.append(number)

        number = 3 if number == 2 else number + 2

    return primes

def miller_rabin(n, base):
    if n < 2 or n % 2 == 0:
        return n == 2

    d = n - 1
    s = 0

    while d % 2 == 0:
        d //= 2
        s += 1

    x = pow(base, d, n)

    if x == 1 or x == n - 1:
        return True

    for _ in range(s - 1):
        x = pow(x, 2, n)

        if x == n - 1:
            return True

    return False


def parse_number(value, name):
    text = str(value)

    if not text.isascii() or not text.isdigit():
```

```
        raise ValueError(f"{name} must be a positive decimal integer")

    if len(text) > len(str(1 << MAX_BITS)):
        raise ValueError(f"{name} is too large")

    return int(text)
```

di fungsi get primes itu sama aja nyari bilangan prima cuma menggunakan trial division , lalu selanjutnya disini ada fungsi miller_rabin yaitu untuk mengecek apakah n kemungkinan prima. Jika n < 2 tidak prima , n == 2 prima kalo genap ngga prima, Menghitung x = base ^ d mod n. Jika x == 1 atau x == n - 1, kemungkinan besar prima = true , Melakukan squaring berulang (x = x^2 mod n) sebanyak s-1 kali; jika di suatu titik x == n-1, return True. Jika tidak ada kondisi di atas terpenuhi, n composite (bukan prima) untuk basis ini → return False. dan terakhir ada fungsi parse_number yaitu untuk memastikan value bilangan bulat positif.

Lalu disini selanjutnya saya mencoba memahami kode selanjutnya.

```
def main():
    print(f"Hi, welcome to ITFest 2026")
    print(f"goodluck create your own RSA")

    e = 65537
    bases = get_primes(150)
    message = int.from_bytes(FLAG, "big")

    line = sys.stdin.buffer.readline(MAX_INPUT_BYTES + 1)

    if not line:
        send({"status": "error", "error": "no submission received"})
        return

    if len(line) > MAX_INPUT_BYTES:
        send({"status": "error", "error": "submission is too large"})
        return

    try:
        data = json.loads(line)

        if not isinstance(data, dict):
            raise ValueError("submission must be a JSON object")

        n = parse_number(data.get("n"), "n")
        raw_factors = data.get("factors")

        if not isinstance(raw_factors, list) or len(raw_factors) != 3:
            raise ValueError("factors must contain exactly three integers")
```

```
        factors = [
            parse_number(value, f"factors[{i}]")
            for i, value in enumerate(raw_factors)
        ]

        if not MIN_BITS <= n.bit_length() <= MAX_BITS:
            raise ValueError(f"n must be between {MIN_BITS} and {MAX_BITS} bits")
        if n % 2 == 0:
            raise ValueError("n must be odd")
        if len(set(factors)) != 3 or min(factors) <= 1:
            raise ValueError("factors must be distinct and greater than one")
        if math.prod(factors) != n:
            raise ValueError("factors do not multiply to n")
        if not all(isPrime(factors) for factors in factors):
            raise ValueError("all factors must be prime")
        if not all(miller_rabin(n, base) for base in bases):
            raise ValueError("n failed Miller-Rabin")

        lambda_n = math.lcm(*(factor - 1 for factor in factors))

        if math.gcd(e, lambda_n) != 1:
            raise ValueError("e is not invertible modulo lambda")

        if message >= n:
            raise ValueError("n is too small for the flag")

        send({
            "status": "ok",
            "n": n,
            "e": e,
            "c": pow(message, e, n),
        })

    except (ValueError, TypeError) as error:
        send({"status": "error", "error": str(error)})
```

Nah jadi pada kode ini cara kerja challenge ini adalah, parsing & validasi input => Validasi terhadap n dan factors => Validasi nilai RSA yang riil => Jika validasi sukses maka akan menghasilkan sebuah flag. Jadi intinya itu kita cari 3 bilangan prima yang berbeda tetapi hasil kalinya (n,komposit,ganjil) lolos terhadap dari miller_rabin di 150 basis berbeda, karena ada 3 faktor jadi kita bisa menggunakan Carmichael number.

Jadi langkah solver nya adalah : 1. Menggunakan Carmichael number karena kemungkinan besar lolos dalam tahap test Miller_rabin untuk hampir semua basis yang co-prime dengannya. 2. Kriteria Korselt karena Bilangan n (hasil kali prima berbeda) adalah carmichael number jika

dan hanya jika setiap faktor prima p dari n. 3. Chernick Formula : yaitu sebuah rumus untuk membuat Carmichael number dari 3 prima, berikut bentuk dari Chernick Formula

```
p = 6k + 1
q = 12k + 1
r = 18k + 1
```

Jika k dipih , sehingga p,q,r semuanya prima, maka n = p*q*r otomatis adalah Carmichael number.

jadi bentuk lebih jelasnya : iterasi nilai k dengan mengkuti MIN_BITS dan MAX_BITS dari server, setiap k hitung p = 6k + 1 , q = 12k + 1 dan r = 18k + 2 , lalu cek ketiga primanya , lalu hitung n = p*q*r , Hitung lambda_n = lcm(p-1, q-1, r-1), Lalu cek apakah gcd(65537, lambda_n) == 1 jika tidak cari nilai k yang lain, jika semua sudah cocok maka kirim n dan p,q,r ke server dalam format json.

dan berikut solver script nya.

```
#!/usr/bin/env python3
"""
Solver untuk challenge PrimeEra (ITFest 2026) - RSA Carmichael Number
yang lolos Miller-Rabin di 150 basis sekaligus.

=== KONSEP ===
Chernick-style Carmichael number TIDAK CUKUP: ia hanya lolos tes Fermat,
tapi gagal di tes Miller-Rabin (yang mengecek akar kuadrat dari 1 di
sepanjang proses kuadrat berulang).

Untuk benar-benar lolos Miller-Rabin di banyak basis sekaligus, kita pakai
metode ARNAULT (1995) "Constructing Carmichael Numbers which are Strong
Pseudoprimes to Several Bases". Referensi implementasi konsep (SageMath):
https://www.kevincao.xyz/posts/generating-strong-pseudoprimes
(port murni Python di bawah, tanpa SageMath)

Ide inti untuk n = p1 * p2 * p3 (h=3 prima berbeda):
1. Pilih k1=1, k2, k3 -- bilangan GANJIL, coprime terhadap semua basis.
   Definisikan p_i = k_i * (p1 - 1) + 1  untuk i=2,3.
   Karena k_i ganjil, v_2(p_i - 1) = v_2(p1 - 1) untuk semua i --> s (2-adic
   valuation) otomatis SAMA untuk ketiga prima. Ini kunci supaya proses
   squaring Miller-Rabin "sinkron" antar faktor.
2. Untuk tiap basis a, kita PAKSA Legendre symbol (a/p_i) = -1 untuk SEMUA i.
   Ini menjamin a^d mod tiap p_i berujung di -1 pada langkah yang sama,
   sehingga via CRT, a^d mod n juga -1 (yaitu x == n-1, syarat lolos di
   fungsi miller_rabin server).
3. Syarat tambahan supaya m_i = (prod_j!=i p_j - 1)/(p_i - 1) integer
   (dibutuhkan supaya n Carmichael/Korselt-valid):
```

```
       p1 ≡ -k3^-1 (mod k2)
       p1 ≡ -k2^-1 (mod k3)

Kondisi Legendre di atas diterjemahkan (via quadratic reciprocity) menjadi
syarat p_i mod 4a untuk tiap basis a -- lalu ditarik balik ke syarat p1 mod 4a
(karena p_i adalah fungsi linear dari p1). Gabungkan SEMUA syarat (150 basis
+ 2 syarat k2,k3) via CRT --> dapat p1 ≡ r (mod M).
Lalu p1 = r + i*M untuk i tertentu (dipilih supaya ukuran bit p1 pas),
digeser sampai p1, p2, p3 semuanya prima.
"""

from pwn import *
import json
import math
import random
import multiprocessing as mp

from sympy import prime as nth_prime, isprime as sympy_isprime, randprime
try:
    from sympy.functions.combinatorial.numbers import jacobi_symbol
except ImportError:
    from sympy.ntheory.residue_ntheory import jacobi_symbol

# ==== Primality test cepat: gmpy2 kalau ada, fallback sympy ====
try:
    import gmpy2

    def is_prime(x):
        return bool(gmpy2.is_prime(int(x), 25))

    HAVE_GMPY2 = True
except ImportError:
    is_prime = sympy_isprime
    HAVE_GMPY2 = False


# ==== Konfigurasi ====
HOST = "chall.itfestmicroipb.web.id"
PORT = 30846  # NOTE: cek ulang port tiap kali instance baru dibuka
LOCAL_BINARY = None

MIN_BITS = 3700
MAX_BITS = 4096
E = 65537
NUM_BASES = 150
NUM_WORKERS = mp.cpu_count()
SIEVE_LIMIT = 100_000


def get_bases(n):
```

```
    """150 basis pertama, identik dengan get_primes(150) di server."""
    return [int(nth_prime(i)) for i in range(1, n + 1)]


BASES = get_bases(NUM_BASES)


def egcd(a, b):
    old_r, r = a, b
    old_s, s = 1, 0
    old_t, t = 0, 1
    while r != 0:
        q = old_r // r
        old_r, r = r, old_r - q * r
        old_s, s = s, old_s - q * s
        old_t, t = t, old_t - q * t
    return old_r, old_s, old_t  # (gcd, x, y) dengan a*x + b*y = gcd


def crt2(r1, m1, r2, m2):
    """Gabungkan x=r1(mod m1), x=r2(mod m2) -> (r, lcm) atau None kalau tidak
konsisten."""
    g, x, _ = egcd(m1, m2)
    if (r2 - r1) % g != 0:
        return None
    lcm = m1 // g * m2
    tmp = ((r2 - r1) // g * x) % (m2 // g)
    r = (r1 + m1 * tmp) % lcm
    return r, lcm


def compute_S(a):
    """S(a) = { r mod 4a ganjil : Legendre(a, r) == -1 } (via Jacobi symbol,
    berlaku untuk r prima berkat quadratic reciprocity - periodik mod 4a)."""
    return [r for r in range(3, 4 * a + 1, 2) if jacobi_symbol(a, r) == -1]


log.info("Menghitung S(a) untuk semua 150 basis (sekali saja, di-cache)...")
S_CACHE = {a: compute_S(a) for a in BASES}
log.info("Selesai precompute S(a).")


def sieve_of_eratosthenes(limit):
    sieve = bytearray([1]) * (limit + 1)
    sieve[0] = sieve[1] = 0
    for i in range(2, int(limit ** 0.5) + 1):
        if sieve[i]:
            sieve[i * i:: i] = bytearray(len(sieve[i * i:: i]))
    return [i for i, v in enumerate(sieve) if v]
```

```
SMALL_PRIMES = sieve_of_eratosthenes(SIEVE_LIMIT)
# buang basis dari daftar prima kecil (tidak perlu dites lagi, k2/k3 sudah dijamin
coprime terhadapnya)
SMALL_PRIMES_SET_BASES = set(BASES)


def passes_small_prime_filter(*nums):
    for sp in SMALL_PRIMES:
        for x in nums:
            if x % sp == 0 and x != sp:
                return False
    return True


def pick_ks(bit_size=11):
    """k1=1 tetap; k2,k3 = prima acak berbeda, LEBIH BESAR dari basis terbesar
    (supaya otomatis coprime terhadap semua basis -> invers mod 4a selalu ada)."""
    lower = BASES[-1] + 1
    lo = max(lower, 1 << (bit_size - 1))
    hi = max(lo + 1, 1 << bit_size)
    while True:
        k2 = int(randprime(lo, hi))
        k3 = int(randprime(lo, hi))
        if k2 != k3:
            return [1, k2, k3]


def build_subsets(ks):
    """Untuk tiap basis a, hitung subset kandidat residu p1 mod 4a yang
    memenuhi syarat Legendre(a,p_i)=-1 untuk SEMUA i sekaligus."""
    subsets = {}
    for a in BASES:
        sa = S_CACHE[a]
        inter = None
        mod = 4 * a
        for k in ks:
            try:
                kinv = pow(k, -1, mod)
            except ValueError:
                return None  # k tidak invertible mod 4a -> pilihan k2,k3 gagal
            mapped = {(kinv * (s + k - 1)) % mod for s in sa}
            inter = mapped if inter is None else (inter & mapped)
            if not inter:
                break
        if not inter:
            return None
        subsets[a] = list(inter)
    return subsets
```

```python
def find_p1_residue(ks, max_backtrack_nodes=2_000_000):
    """Cari p1 ≡ r (mod M) yang memenuhi SEMUA syarat (150 basis + syarat k2,k3).
    Return (r, M) atau None kalau gagal (coba ks lain)."""
    k1, k2, k3 = ks

    subsets = build_subsets(ks)
    if subsets is None:
        return None

    # syarat awal supaya m_i integer (h=3, lihat referensi Arnault/Kevin Cao)
    try:
        r0 = (-pow(k3, -1, k2)) % k2
        r1 = (-pow(k2, -1, k3)) % k3
    except ValueError:
        return None
    merged = crt2(r0, k2, r1, k3)
    if merged is None:
        return None
    base_r, base_m = merged

    # heuristik: proses basis dengan jumlah kandidat PALING SEDIKIT dulu (fail fast)
    order = sorted(BASES, key=lambda a: len(subsets[a]))

    nodes = [0]

    def backtrack(i, r, m):
        if nodes[0] > max_backtrack_nodes:
            return None
        if i == len(order):
            return r, m
        a = order[i]
        cands = subsets[a][:]
        random.shuffle(cands)
        mod_a = 4 * a
        for cand in cands:
            nodes[0] += 1
            if nodes[0] > max_backtrack_nodes:
                return None
            merged = crt2(r, m, cand, mod_a)
            if merged is None:
                continue
            new_r, new_m = merged
            result = backtrack(i + 1, new_r, new_m)
            if result is not None:
                return result
        return None

    return backtrack(0, base_r, base_m)
```

```
def search_primes_worker(args):
    """Diberi (r, M, k2, k3, start_i, step, min_bits, max_bits), cari i sedemikian
    rupa p1=r+i*M, p2=k2*(p1-1)+1, p3=k3*(p1-1)+1 semuanya prima & n dlm rentang
bit."""
    r, M, k2, k3, start_i, step, min_bits, max_bits, worker_id, report_every = args
    i = start_i
    tried = 0
    while True:
        tried += 1
        p1 = r + i * M
        p2 = k2 * (p1 - 1) + 1
        p3 = k3 * (p1 - 1) + 1

        if passes_small_prime_filter(p1, p2, p3):
            if is_prime(p1) and is_prime(p2) and is_prime(p3):
                n = p1 * p2 * p3
                bl = n.bit_length()
                if min_bits <= bl <= max_bits:
                    lambda_n = math.lcm(p1 - 1, p2 - 1, p3 - 1)
                    if math.gcd(E, lambda_n) == 1:
                        return (p1, p2, p3, n, lambda_n, i, tried)
        i += step

        if worker_id == 0 and tried % report_every == 0:
            log.info(f"[worker0] ~{tried * NUM_WORKERS} percobaan, i~{i}")


def find_carmichael_strong_pseudoprime():
    attempt = 0
    while True:
        attempt += 1
        ks = pick_ks(bit_size=11)
        log.info(f"[percobaan {attempt}] mencoba k2={ks[1]}, k3={ks[2]}")

        result = find_p1_residue(ks)
        if result is None:
            log.info("  -> gagal membangun kombinasi CRT, coba k2/k3 lain")
            continue

        r, M = result
        log.success(f"  -> ketemu p1 ≡ r (mod M), M bit_length = {M.bit_length()}")

        k1, k2, k3 = ks

        # target ukuran bit p1: pilih di tengah2 rentang yang mungkin
        # n ~ k2*k3*p1^3  ->  log2(p1) ~ (target_n_bits - log2(k2) - log2(k3)) / 3
        target_n_bits = (MIN_BITS + MAX_BITS) // 2
        target_p1_bits = (target_n_bits - k2.bit_length() - k3.bit_length()) // 3
```

```
        if target_p1_bits <= M.bit_length():
            # M sudah cukup besar sendiri, mulai dari i kecil (0,1,2,...)
            start_i_center = 0
        else:
            extra_bits = target_p1_bits - M.bit_length()
            start_i_center = 1 << extra_bits

        log.info(f"  -> target p1 ~{target_p1_bits} bit, mulai pencarian
i~{start_i_center}")

        tasks = [
            (r, M, k2, k3, start_i_center + w, NUM_WORKERS, MIN_BITS, MAX_BITS, w, 300)
            for w in range(NUM_WORKERS)
        ]

        with mp.Pool(NUM_WORKERS) as pool:
            result_holder = {}

            def on_result(res):
                if res is not None:
                    result_holder["result"] = res
                    pool.terminate()

            for task in tasks:
                pool.apply_async(search_primes_worker, (task,), callback=on_result)

            pool.close()
            pool.join()

        if "result" in result_holder:
            p1, p2, p3, n, lambda_n, i, tried = result_holder["result"]
            log.success(f"DITEMUKAN! i={i}")
            log.info(f"p1={p1}")
            log.info(f"p2={p2}")
            log.info(f"p3={p3}")
            log.info(f"n bit_length = {n.bit_length()}")
            return p1, p2, p3, n, lambda_n

        log.info("  -> tidak ketemu dalam batas waktu, coba k2/k3 lain")


def main():
    log.info(f"gmpy2 tersedia: {HAVE_GMPY2}")
    log.info(f"jumlah worker: {NUM_WORKERS}")

    p1, p2, p3, n, lambda_n = find_carmichael_strong_pseudoprime()

    d = pow(E, -1, lambda_n)

    if LOCAL_BINARY:
```

```
        io = process(LOCAL_BINARY)
    else:
        io = remote(HOST, PORT)

    io.recvline()
    io.recvline()

    submission = {
        "n": n,
        "factors": [p1, p2, p3],
    }
    io.sendline(json.dumps(submission).encode())

    response_line = io.recvline()
    log.info(f"Response: {response_line}")
    result = json.loads(response_line)

    if result.get("status") != "ok":
        log.failure(f"Server menolak submission: {result.get('error')}")
        io.close()
        return

    c = result["c"]
    assert result["n"] == n and result["e"] == E

    m = pow(c, d, n)
    flag_bytes = m.to_bytes((m.bit_length() + 7) // 8, "big")
    log.success(f"FLAG: {flag_bytes}")

    io.close()


if __name__ == "__main__":
    main()
```

```
71552241634420092999768650914730732859028107, "e": 65537, "c": 22978904487556163738452556610257928041311445331573321828962143921323286562700893864
06481545063125060241395044879733053973127635802103396204591128714193285467394923116661544032025557483073037580521841407720732632504302046545390028
62030339044642101123289936747997388016411182279205537668299263775591749331108492510435867568223093257973972163448569901780388771868988477036150333
62279945637863793564651275430628422642362165335151667781965067405703402932943841327621798455508556960271214397997046724813924627840244483774444599
05496317493023142958058239452120320552733259847719477335102100347963511573053409586968069660154665145439636914452729952941613482979142739026182333
98460923491780982057362392392519958359289227126197344616872673815153477615386377914089472499310864426953091051691103654627525961056653057375645466
34266025597858003592032651563491475102122504734401175405604061848324532246746786738736759012223133923973344632780876339435705617440258713868011534
39088152802099416484229907932308488395785319633099547580295663867836824665368208875425518082304025807861093108659495834314930526235948019914007873
28110282069016854639160604325120530593064157201491533281891019648416}\n'
[+] FLAG: b'ITFest2026{THReE_priMes_7Old_oNe_57r0Ng_lIe_54e395f48169}'
[*] Closed connection to chall.itfestmicroinb.web.id port 30846
```

FLAG: ITFest2026{THReE_priMes_7Old_oNe_57r0Ng_lIe_54e395f48169}

AI Transparency : https://claude.ai/share/aea88f47-1f26-4c0a-8bd4-59367c355e51

Prime Era (198 Points)

```
Really? I can create my own RSA?

Author: Kae
```

Jadi pada challenge ini kita diberikan sebuah file chall.py dengan isi berikut.

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

import hashlib
import hmac
import json
import secrets
import sys

P = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F
N = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
G = (
    0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798,
    0x483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8,
)

LOW_BITS = 128
WEAK_SAMPLES = 4
DECOY_SAMPLES = 2

def point_add(left, right):
    if left is None:
        return right

    if right is None:
        return left

    x1, y1 = left
    x2, y2 = right

    if x1 == x2 and (y1 + y2) % P == 0:
        return None

    if left == right:
        slope = (3 * x1 * x1) * pow(2 * y1, -1, P)
    else:
        slope = (y2 - y1) * pow(x2 - x1, -1, P)
```

```
    slope %= P
    x3 = (slope * slope - x1 - x2) % P

    return x3, (slope * (x1 - x3) - y1) % P

def scalar_mult(value, point):
    result = None
    addend = point
    while value:
        if value & 1:
            result = point_add(result, addend)
        addend = point_add(addend, addend)
        value >>= 1
    return result

def compress(point):
    x, y = point
    return (b"\x02" if y % 2 == 0 else b"\x03").hex() + x.to_bytes(32, "big").hex()

def sign(message, private_key, nonce):
    digest = hashlib.sha256(message).digest()
    h = int.from_bytes(digest, "big")
    r = scalar_mult(nonce, G)[0] % N
    s = (pow(nonce, -1, N) * (h + r * private_key)) % N
    if r == 0 or s == 0:
        raise RuntimeError

    return {
        "message": message.decode("ascii"),
        "r": f"{r:064x}",
        "s": f"{s:064x}",
    }

class BackupSignerRng:
    def __init__(self):
        self.boot_state = secrets.randbits(256 - LOW_BITS)

    def next_nonce(self):
        while True:
            nonce = (self.boot_state << LOW_BITS) | secrets.randbits(LOW_BITS)
            if 1 <= nonce < N:
                return nonce

def derive_key(private_key):
    material = private_key.to_bytes(32, "big")
    return hashlib.sha256(b"ITFest2026|Clockwork|" + material).digest()

def encrypt_flag(flag, private_key):
    key = derive_key(private_key)
    nonce = secrets.token_bytes(16)
```

```
    stream = bytearray()
    counter = 0
    while len(stream) < len(flag):
        stream.extend(hashlib.sha256(b"stream|" + key + nonce + counter.to_bytes(4,
"big")).digest())
        counter += 1
    ciphertext = bytes(a ^ b for a, b in zip(flag, stream))
    tag = hmac.new(key, b"tag|" + nonce + ciphertext, hashlib.sha256).digest()[:16]
    return {"nonce": nonce.hex(), "ciphertext": ciphertext.hex(), "tag": tag.hex()}

def main():
    with open("flag.txt","rb") as f:
        FLAG = f.read().rstrip()

    private_key = secrets.randbelow(N - 1) + 1
    public_key = compress(scalar_mult(private_key, G))
    flawed_rng = BackupSignerRng()

    messages = [f"approve request {secrets.token_hex(8)}".encode("ascii") for _ in
range(WEAK_SAMPLES + DECOY_SAMPLES)]
    samples = []
    for message in messages[:WEAK_SAMPLES]:
        sample = sign(message, private_key, flawed_rng.next_nonce())
        sample["ticket"] = secrets.token_hex(4)
        samples.append(sample)

    for message in messages[WEAK_SAMPLES:]:
        nonce = secrets.randbelow(N - 1) + 1
        sample = sign(message, private_key, nonce)
        sample["ticket"] = secrets.token_hex(4)
        samples.append(sample)

    secrets.SystemRandom().shuffle(samples)

    result = {
        "challenge": "Clockwork",
        "curve": "secp256k1",
        "public_key_compressed": public_key,
        "samples": samples,
        "encrypted_flag": encrypt_flag(FLAG, private_key),
    }
    json.dump(result, sys.stdout, indent=2)
    sys.stdout.write("\n")

if __name__ == "__main__":
    main()
```

umm ya ini challenge ECC disini tetapi seperti nya disini digabungkan dengan rng maka disini saya langsung mencoba memahami isi kode tersebut.

```
LOW_BITS = 128
WEAK_SAMPLES = 4
DECOY_SAMPLES = 2

def point_add(left, right):
    if left is None:
        return right

    if right is None:
        return left

    x1, y1 = left
    x2, y2 = right

    if x1 == x2 and (y1 + y2) % P == 0:
        return None

    if left == right:
        slope = (3 * x1 * x1) * pow(2 * y1, -1, P)
    else:
        slope = (y2 - y1) * pow(x2 - x1, -1, P)

    slope %= P
    x3 = (slope * slope - x1 - x2) % P

    return x3, (slope * (x1 - x3) - y1) % P

def scalar_mult(value, point):
    result = None
    addend = point
    while value:
        if value & 1:
            result = point_add(result, addend)
        addend = point_add(addend, addend)
        value >>= 1
    return result

def compress(point):
    x, y = point
    return (b"\x02" if y % 2 == 0 else b"\x03").hex() + x.to_bytes(32, "big").hex()

def sign(message, private_key, nonce):
    digest = hashlib.sha256(message).digest()
    h = int.from_bytes(digest, "big")
    r = scalar_mult(nonce, G)[0] % N
    s = (pow(nonce, -1, N) * (h + r * private_key)) % N
    if r == 0 or s == 0:
        raise RuntimeError

    return {
```

```
        "message": message.decode("ascii"),
        "r": f"{r:064x}",
        "s": f"{s:064x}",
    }
```

Jadi di challenge ini tipe nya adalah ECDSA di kode tersebut terdapat sebuah fungsi sign (signature) pakai koordinat affine, tapi ada variable LOW_BITS, WEAK_SAMPLES dan DECOY_SAMPLES seperti nya ini tipe attack nya ke Lattice, disini ada fungsi point_add dan scalar_mult ini memang normal ada di chall tipe ECC , lalu ada fungsi compress yaitu encode public key , menggunakan prefix 02/03 berdasarkan parity nilai Y, untuk bagian fungsi sign ini standar dari ECDSA, lalu untuk bagian LOW_BITS = 128 server memberikan 128 bit bawah dari nonce k dari setiap signature, karena N berukuran 256 berarti setengahnya sudah bocor. WEAK_SAMPLE = 4 : terdapat 4 signature yang menggunakan nilai nonce yang bocor , dan DECOY_SAMPLES = ya intinya ada 2 signature itu cuma tambahana atau palsu gitu seperti arti dari decoy itu sendiri.

Maka disini saya lanjut untuk memahami kode selanjutnya.

```
class BackupSignerRng:
    def __init__(self):
        self.boot_state = secrets.randbits(256 - LOW_BITS)

    def next_nonce(self):
        while True:
            nonce = (self.boot_state << LOW_BITS) | secrets.randbits(LOW_BITS)
            if 1 <= nonce < N:
                return nonce

def derive_key(private_key):
    material = private_key.to_bytes(32, "big")
    return hashlib.sha256(b"ITFest2026|Clockwork|" + material).digest()

def encrypt_flag(flag, private_key):
    key = derive_key(private_key)
    nonce = secrets.token_bytes(16)
    stream = bytearray()
    counter = 0
    while len(stream) < len(flag):
        stream.extend(hashlib.sha256(b"stream|" + key + nonce + counter.to_bytes(4,
"big")).digest())
        counter += 1
    ciphertext = bytes(a ^ b for a, b in zip(flag, stream))
    tag = hmac.new(key, b"tag|" + nonce + ciphertext, hashlib.sha256).digest()[:16]
    return {"nonce": nonce.hex(), "ciphertext": ciphertext.hex(), "tag": tag.hex()}
```

```
def main():
    with open("flag.txt","rb") as f:
        FLAG = f.read().rstrip()

    private_key = secrets.randbelow(N - 1) + 1
    public_key = compress(scalar_mult(private_key, G))
    flawed_rng = BackupSignerRng()

    messages = [f"approve request {secrets.token_hex(8)}".encode("ascii") for _ in
range(WEAK_SAMPLES + DECOY_SAMPLES)]
    samples = []
    for message in messages[:WEAK_SAMPLES]:
        sample = sign(message, private_key, flawed_rng.next_nonce())
        sample["ticket"] = secrets.token_hex(4)
        samples.append(sample)

    for message in messages[WEAK_SAMPLES:]:
        nonce = secrets.randbelow(N - 1) + 1
        sample = sign(message, private_key, nonce)
        sample["ticket"] = secrets.token_hex(4)
        samples.append(sample)

    secrets.SystemRandom().shuffle(samples)

    result = {
        "challenge": "Clockwork",
        "curve": "secp256k1",
        "public_key_compressed": public_key,
        "samples": samples,
        "encrypted_flag": encrypt_flag(FLAG, private_key),
    }
    json.dump(result, sys.stdout, indent=2)
    sys.stdout.write("\n")
```

Setelah dipahami jadi kode ini bekerja dengan cara , boot_state adalah secret berukuran 128bit dibuat sekali saja, setiap nonce k dibentuk : k = (boot_state << 128) | random_lower_128. berarti : 128 bit bagian atas nilainya identik setiap nonce dihasilkan dari RNG , sedangkan 128 bit bawah bersifat acak, jadi Vuln dari challenge ini adalah predictable nonce , key nya di encrypt dengan SHA256(“ITFest2026|Clockwork|” + private_key_bytes). Lalu yang terakhir dari challenge ini jalankan maka akan mengeluarkan output json yang menghasilkan : challenge, curve, public_key_compressed, samples dan encrypted_flag.

Dan disini untuk langkah solver nya adalah

1. Filter decoy leaknya : kita dapat 6 sample , tapi hanya 4 sample yang memiliki nilai nonce bocor, jadi harus mencoba semua subset berisi 4 dari 6 sample dengan total 15 kombinasi. Setiap subset, lalu jalankan lattice attack, Hanya subset yang benar saja akan menghasilkan private key yang cocok dengan public key.
2. Modif persamaan nya : Untuk sample lemah i, nonce nya adalah:

   k_i = U * 2^128 + l_i

   U =128bit bagian atas (boot_state), l_i = 128 bit bagian bawah yang acak.

   Dari ECDSA kita dapatkan menjadi :

   s_i * k_i ≡ h_i + r_i * priv (mod N)

   Subtitusi k_i:

   s_i * (U * 2^128 + l_i) ≡ h_i + r_i * priv (mod N)

   Susun ulang menjadi persamaan modular linear:

   (s_i * 2^128) * U + s_i * l_i - r_i * priv ≡ h_i (mod N)

   Disini nilai yang masih belum diketahui adalah U, l_i dan priv karena U dan l_i ini relatif kecil dengan nilai $\leq$ 2^128 dan punya 4 persamaan maka bisa menggunakan Hidden Number Problem

3. Lattice (LLL) : membuat basis Lattice di mana vektor pender bisa mempresentasikan nilai-nilai yang belum diketahui tersebut.
4. Verification & decrypt : cek apakah scalar_mult(priv, G) cocok dengan compressed public key yang diberikan. Kalo cocok berarti nemu subset yang cocok, pakai derive_key(priv) untuk dapetin kunci AES nya , terakhir tinggal decrypt ciphertext menggunakan stream SHA256, kemudian verifikasi tag HMAC nya.

Dan berikut solver scriptnya.

```
#!/usr/bin/env python3
from pwn import *
import json
import subprocess
```

```
import tempfile
import os
import sys
import hashlib
import hmac
import re

# ---------- pure Python decryption (independent of Sage) ----------
def derive_key(private_key):
    material = private_key.to_bytes(32, "big")
    return hashlib.sha256(b"ITFest2026|Clockwork|" + material).digest()

def decrypt_flag(encrypted_flag, private_key):
    key = derive_key(private_key)
    nonce = bytes.fromhex(encrypted_flag["nonce"])
    ciphertext = bytes.fromhex(encrypted_flag["ciphertext"])
    tag = bytes.fromhex(encrypted_flag["tag"])

    expected_tag = hmac.new(key, b"tag|" + nonce + ciphertext,
hashlib.sha256).digest()[:16]
    if not hmac.compare_digest(tag, expected_tag):
        raise ValueError("HMAC tag mismatch")

    stream = bytearray()
    counter = 0
    while len(stream) < len(ciphertext):
        stream.extend(hashlib.sha256(b"stream|" + key + nonce + counter.to_bytes(4,
"big")).digest())
        counter += 1
    return bytes(a ^ b for a, b in zip(ciphertext, stream))

# ---------- embedded Sage solver (unchanged, still has XOR bug but we ignore)
----------
SAGE_SOLVER = r'''
import json
import hashlib
import hmac
import sys
from itertools import combinations, permutations

# ---------- secp256k1 ----------
P = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F
N = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
Gx = 0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798
Gy = 0x483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8
G = (Gx, Gy)
LOW_BITS = 128
S = 2**LOW_BITS
C = 2**256
```

```
# ---------- ECC ----------
def point_add(left, right):
    if left is None: return right
    if right is None: return left
    x1, y1 = left; x2, y2 = right
    if x1 == x2 and (y1 + y2) % P == 0: return None
    if left == right:
        slope = (3 * x1 * x1) * pow(2 * y1, -1, P)
    else:
        slope = (y2 - y1) * pow(x2 - x1, -1, P)
    slope %= P
    x3 = (slope * slope - x1 - x2) % P
    return x3, (slope * (x1 - x3) - y1) % P

def scalar_mult(value, point):
    result = None
    addend = point
    while value:
        if value & 1:
            result = point_add(result, addend)
        addend = point_add(addend, addend)
        value >>= 1
    return result

def compress(point):
    x, y = point
    return (b"\x02" if y % 2 == 0 else b"\x03").hex() + x.to_bytes(32, "big").hex()

# ---------- Challenge helpers ----------
def derive_key(private_key):
    material = private_key.to_bytes(32, "big")
    return hashlib.sha256(b"ITFest2026|Clockwork|" + material).digest()

def decrypt_flag(encrypted_flag, private_key):
    key = derive_key(private_key)
    nonce = bytes.fromhex(encrypted_flag["nonce"])
    ciphertext = bytes.fromhex(encrypted_flag["ciphertext"])
    tag = bytes.fromhex(encrypted_flag["tag"])

    expected_tag = hmac.new(key, b"tag|" + nonce + ciphertext,
hashlib.sha256).digest()[:16]
    if not hmac.compare_digest(tag, expected_tag):
        raise ValueError("HMAC tag mismatch")

    stream = bytearray()
    counter = 0
    while len(stream) < len(ciphertext):
        stream.extend(hashlib.sha256(b"stream|" + key + nonce + counter.to_bytes(4,
"big")).digest())
        counter += 1
```

```
    # XOR decryption
    plaintext = bytes(a ^ b for a, b in zip(ciphertext, stream))
    return plaintext

# ---------- Lattice attack ----------
def recover_private_key(samples_subset, public_key_compressed):
    ref = samples_subset[0]
    h0 = int.from_bytes(hashlib.sha256(ref["message"].encode()).digest(), "big")
    r0 = int(ref["r"], 16)
    s0 = int(ref["s"], 16)

    rows = []
    for i in range(1, 4):
        smp = samples_subset[i]
        hi = int.from_bytes(hashlib.sha256(smp["message"].encode()).digest(), "big")
        ri = int(smp["r"], 16)
        si = int(smp["s"], 16)

        A = ((r0 * si - ri * s0) * S) % N
        B = (-ri * s0) % N
        C = (r0 * si) % N
        D = (r0 * hi - ri * h0) % N

        row = [0]*9
        row[0] = A
        row[1] = B
        row[1 + i] = C
        row[5 + (i-1)] = -N
        row[8] = -D
        rows.append(row)

    M = matrix(ZZ, rows)
    K = M.right_kernel().basis_matrix()
    K = matrix(ZZ, K.rows())

    for r in range(K.nrows()):
        for c in range(5):
            K[r, c] *= S
        K[r, 8] *= C

    L = None
    try:
        L = K.LLL()
    except:
        try:
            L = K.BKZ(block_size=20)
        except:
            pass
    if L is None:
```

```
        return None

    for v in L:
        if v[8] == C or v[8] == -C:
            if v[8] == -C:
                v = -v
            if all(v[c] % S == 0 for c in range(5)):
                a = v[0] // S
                l0 = v[1] // S
                try:
                    inv_r0 = pow(r0, -1, N)
                    priv = (s0 * (a * S + l0) - h0) % N
                    priv = (priv * inv_r0) % N
                except ValueError:
                    continue
                if compress(scalar_mult(priv, G)) == public_key_compressed:
                    return priv
    return None

# ---------- Main ----------
def solve_challenge_from_file(json_file):
    with open(json_file, "r") as f:
        data = json.load(f)

    public_key = data["public_key_compressed"]
    samples = data["samples"]
    encrypted_flag = data["encrypted_flag"]

    for combo in combinations(range(6), 4):
        subset = [samples[i] for i in combo]
        for perm in permutations(subset):
            priv = recover_private_key(list(perm), public_key)
            if priv is not None:
                print(f"[Sage] Found private key: {priv:x}")
                flag = decrypt_flag(encrypted_flag, priv)
                print(f"[Sage] Flag: {flag.decode()}")
                return
    print("[Sage] Private key not found.")

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: sage solver.sage <json_file>")
        sys.exit(1)
    solve_challenge_from_file(sys.argv[1])
'''

# ---------- Main: connect, run Sage, then decrypt with pure Python ----------
def main():
    log.info("Connecting to remote server...")
    host = "chall.itfestmicroipb.web.id"
```

```
    port = 31502
    io = remote(host, port)

    log.info("Receiving data...")
    output = io.recvall().decode()
    io.close()

    try:
        data = json.loads(output)
    except json.JSONDecodeError:
        log.error("Failed to parse JSON from server")
        print(output)
        sys.exit(1)

    log.info("Received challenge data.")

    # Write JSON to temp file
    with tempfile.NamedTemporaryFile(mode='w', suffix='.json', delete=False) as f:
        json.dump(data, f)
        json_path = f.name

    # Write Sage solver to temp file (with XOR fix but we'll ignore)
    with tempfile.NamedTemporaryFile(mode='w', suffix='.sage', delete=False) as f:
        f.write(SAGE_SOLVER)
        solver_path = f.name

    log.info("Running Sage lattice solver...")
    proc = subprocess.Popen(['sage', solver_path, json_path],
                            stdout=subprocess.PIPE, stderr=subprocess.PIPE,
text=True)
    stdout, stderr = proc.communicate()

    # Print Sage output
    print(stdout)
    if stderr:
        print("[Sage stderr]", stderr, file=sys.stderr)

    # Parse private key from stdout
    # Look for "Found private key: <hex>"
    match = re.search(r'Found private key: ([0-9a-fA-F]+)', stdout)
    if match:
        priv_hex = match.group(1)
        private_key = int(priv_hex, 16)
        log.info(f"Recovered private key: {priv_hex}")
        # Decrypt using pure Python (correct XOR)
        flag = decrypt_flag(data["encrypted_flag"], private_key)
        print(f"Flag: {flag.decode()}")
    else:
        log.error("Private key not found in Sage output.")
```

```
    # Cleanup
    os.unlink(json_path)
    os.unlink(solver_path)
    log.info("Done.")

if __name__ == "__main__":
    main()
```

```
[*] Received challenge data.
[*] Running Sage lattice solver...
[Sage] Found private key: 8904e695a7df334f430136efb051e21a8ba6bf9280cc30208b3386768e4f1065

[Sage stderr] Traceback (most recent call last):
  File "/tmp/tmpgio_tskr.sage.py", line 164, in <module>
    solve_challenge_from_file(sys.argv[_sage_const_1 ])
  File "/tmp/tmpgio_tskr.sage.py", line 155, in solve_challenge_from_file
    flag = decrypt_flag(encrypted_flag, priv)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/tmp/tmpgio_tskr.sage.py", line 73, in decrypt_flag
    plaintext = bytes(a ** b for a, b in zip(ciphertext, stream))
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: bytes must be in range(0, 256)

[*] Recovered private key: 8904e695a7df334f430136efb051e21a8ba6bf9280cc30208b3386768e4f1065
Flag: ITFest2026{HAlF_7hE_nONC3_5tUCK_On_7HE_s4M3_T1Ck_8bdaeac6b3c6}
[*] Done.
```

FLAG: ITFest2026{HAlF_7hE_nONC3_5tUCK_On_7HE_s4M3_T1Ck_8bdaeac6b3c6}

AI Transparency : https://chat.deepseek.com/share/fye8yhaao9yh513r9a

Nguyen++ (368 Points)

```
RSA? EC? RSA & EC? Good old days.

Author: ji4xuu

Hint:
The curve is not defined over a field.
pikirlah sama ko orang homomorfisma yg memetakan ke ZmodN
```

Jadi pada challenge ini kita diberikan 2 script yaitu chall.sage dan instance.py maka disini saya langsung mencoba menganalisis dari isi masing masing kode tersebut.

```
def send(s):
    print(s, end="", flush=True)


def bytes_to_long(data):
    return int.from_bytes(data, "big")


def read_limited(prompt, max_chars):
    value = input(prompt).strip()
    if not value or len(value) > max_chars:
        raise ValueError("invalid input length")
    return value


with open(INSTANCE_PATH) as f:
    instance = json.load(f)


def encrypt_flag():
    with open("flag.txt", "rb") as flag_file:
        flag = flag_file.read(257)
    if not flag or len(flag) > 256:
        raise RuntimeError("dynamic flag is unavailable")
    key = hashlib.sha256(str(instance["target_x"]).encode()).digest()[:16]
    cipher = AES.new(key, AES.MODE_CBC, b"\0" * 16)
    return cipher.encrypt(pad(flag, 16))


def format_points(points):
    return repr([(int(x), int(y)) for x, y in points])


def unlock_archive():
```

```
    magic = os.urandom(17)
    magic_num = bytes_to_long(magic)

    send(f"Magic: {magic.hex()}\n")
    send("Could you use it to do encryption as hash?\n")

    n = int(read_limited("N:> ", 1230))
    e = int(read_limited("E:> ", 1230))
    data = bytes.fromhex(read_limited("data:> ", 1024))

    if n <= 0 or e < 0 or not data:
        return False

    h = bytes_to_long(hashlib.sha384(data).digest())
    m = bytes_to_long(data)
    return n >> (384 - 17 * 8) == magic_num and pow(m, e, n) == h
```

Inti nya fungsi di kode ini seperti biasa send untuk mencetak teks tanpa karakter baris baru, bytes_to_long juga itu cuma convert dari bytes ke angka integer, read_limited ini Membaca input dari pengguna dengan memiliki batasan panjang karakter. Lalu d=ada fungsi format_points untuk : Mengubah daftar koordinat matematika (x, y) menjadi representasi *string*. fungsi unlock_archive : Inisialisasi Magic Number: Program menghasilkan 17 byte data acak (magic) dan mengubahnya menjadi integer (magic_num). Server kemudian memberikan nilai *hex* dari *magic* ini kepada pengguna.Input Pengguna: Pengguna ditantang ("*Could you use it to do encryption as hash?*") dan diminta untuk memasukkan tiga nilai:

- N (Modulus)
- E (Eksponen)
- data (Pesan / Data masukan dalam bentuk heksadesimal)

Kondisi Kemenangan (Verifikasi):
Program menghitung dua hal dari data:

- h = SHA-384(data) diubah ke integer
- m = data (diubah ke integer)

Agar fungsi me-return True (berhasil membuka arsip), dua kondisi matematika ini harus dipenuhi:

- Kondisi 1: N >> (384 - 17 x 8) == magic_num
  Ini adalah operasi *bit-shift*. Nilai 384 - 136 = 248 .Ini berarti ketika N digeser ke kanan sebanyak 248 bit, hasilnya harus persis sama dengan magic_num. 17 byte pertama (Most Significant Bytes) dari N harus sama dengan magic, dan N dianggap sebagai angka 384-bit (48 byte).

- Kondisi 2: m^E pmod N == h
  Ini adalah rumus standar enkripsi RSA, di mana ciphertext-nya diwajibkan sama persis dengan *hash* SHA-384 dari *plaintext*-nya sendiri.

Lalu selanjutnya disini saya analisis dari kode berikut.

```
unlocked = False

send("nguyen++ archive\n")
send("Some points are archived behind old RSA magic.\n\n")

while True:
    send("1. get_gift\n")
    send("2. get_sample_points\n")
    send("3. unlock_archive\n")
    send("4. get_points\n")
    send("5. encrypt_flag\n")
    send("6. quit\n")
    send("> ")

    choice = input().strip()

    if choice == "1":
        send(f"gift = {instance['gift']}\n\n")

    elif choice == "2":
        points = instance["points"][:16]
        send(f"shown = {len(points)}\n")
        send(f"points = {format_points(points)}\n\n")

    elif choice == "3":
        unlocked = unlock_archive()
        if unlocked:
            send("archive unlocked\n\n")
        else:
            send("try harder!!!\n\n")

    elif choice == "4":
        if unlocked:
            points = instance["points"]
            send(f"shown = {len(points)}\n")
            send(f"points = {format_points(points)}\n\n")
        else:
            send("archive locked\n\n")

    elif choice == "5":
        send(f"target_coeffs = {instance['target_coeffs']}\n")
        send(f"flagct = {encrypt_flag()!r}\n\n")
```

```
    elif choice == "6":
        send("bye\n")
        break
```

Program menampilkan 6 pilihan menu yang terus berulang hingga pemain memilih keluar.

1. get_gift: Seharus nya gift disini adalah nilai dari order kurva

2. get_sample_points: Menampilkan 16 titik pertama dari daftar instance["points"]. Titik-titik ini (koordinat x, y) atau data sampel untuk persamaan polinomial.

3. unlock_archive: Ini memanggil fungsi unlock_archive() yang kita bahas sebelumnya (teka-teki memanipulasi N, E, dan data agar cocok dengan RSA dan nilai *hash*).

- Jika pemain berhasil maka menjadi True dan muncul tulisan "archive unlocked".
- Jika gagal, muncul "try harder!!!".

4. get_points: Ini adalah hadiah dari solve no 3.

- Jika arsip sudah terbuka (unlocked == True), program akan mencetak seluruh daftar titik koordinat yang disembunyikan.
- Jika belum terbuka, program menolak dengan pesan "archive locked". Seluruh titik ini pasti sangat krusial untuk menyelesaikan teka-teki tahap akhir.

5. encrypt_flag

1. target_coeffs: Koefisien target
2. flagct: Teks tersandi (*ciphertext*) dari flag.txt yang dienkripsi dengan algoritma AES dari fungsi encrypt_flag() sebelumnya.

Lalu selanjut nya disini saya menganalisis dari kode chall.sage

```
def point_sum(points, zero):
    acc = zero
    for point in points:
        acc += point
    return acc


p = getPrime(384)
q = getPrime(384)
n = p * q

Gx = bytes_to_long(os.urandom(32))
Gy = bytes_to_long(os.urandom(32))
a = randint(1, n**2)
b = (Gy**2 - (Gx**3 + a * Gx)) % (n**2)
```

```
E_n = EllipticCurve(Zmod(n), [a, b])
gift = int(E_n.change_ring(GF(p)).order() * E_n.change_ring(GF(q)).order())
assert gift * E_n(Gx, Gy) == E_n(0, 1, 0)

E = EllipticCurve(Zmod(n**2), [a, b])
base = E(Gx, Gy)
hidden = [randint(1, n) * base for _ in range(16)]
coeff_rows = [list(os.urandom(16)) for _ in range(64)]

coeff_cols = list(zip(*coeff_rows))
assert len(set(coeff_cols)) == 16
ordered = sorted(zip(coeff_cols, hidden), key=lambda x: x[0])
coeff_cols = [col for col, point in ordered]
hidden = [point for col, point in ordered]
coeff_rows = [list(row) for row in zip(*coeff_cols)]
```

Fungsi dari point_sum disini adalah untuk menjumlahkan sekumpulan titik pada kurva eliptik. Lalu disini menghasilkan p dan 1 384 bits yang berarti N nya menajdi 768 bits, dan di challenge ini pembuatan kurva eliptik nya secara terbalik jadi membuat koordinat acak dari G_x dan G_y lalu memilih nilai a acak. lalu disini. Variabel gift adalah total jumlah titik yang ada di kurva tersebut (disebut sebagai Orde Kurva / Curve Order). Karena modulusnya adalah bilangan komposit n = p x q, berdasarkan Chinese Remainder Theorem, orde kurva atas Z_n adalah hasil kali orde kurva atas Galois Field F_p dan F_q. Lalu yang terakhir ada pembentukan matriks : Membuat matriks ukuran 64 baris x 16 kolom (coeff_rows) berisi byte acak. Matriks ditranspos menjadi 16 kolom x 64 baris (coeff_cols). Lalu menggabungkan 16 kolom tersebut dengan 16 titik hidden (memasangkannya). ordered: Pasangan ini kemudian diurutkan (di-sort). Langkah ini mengacak urutan titik hidden dengan cara yang mengikuti data acak coeff_cols. setelah diurutkan, data dipisahkan kembali menjadi coeff_cols, matriks dikembalikan ke bentuk baris coeff_rows, dan hidden sekarang telah berubah urutannya.

Lalu disini saya menganalisis kode selanjutnya :

```
points = [
    point_sum([coef * point for coef, point in zip(row, hidden)], E(0, 1, 0))
    for row in coeff_rows
]

target_coeffs = [3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59]
target = point_sum(
    [coef * point for coef, point in zip(target_coeffs, hidden)],
    E(0, 1, 0),
)
```

```
def serialize_point(P):
    if P.is_zero():
        return ["0", "0"]
    # Untuk titik affine di atas ring, cukup ambil P[0] dan P[1]
    return [str(int(P[0])), str(int(P[1]))]

instance = {
    "gift": str(gift),
    "points": [serialize_point(P) for P in points],
    "target_coeffs": [int(x) for x in target_coeffs],
    "target_x": str(int(target[0])) if not target.is_zero() else "0",
}
```

fungsi points disini adalah : Setiap titik dihitung dengan operasi kombinasi linear. Program mengambil satu baris koefisien acak (row) dari coeff_rows, mengalikannya satu per satu dengan 16 titik hidden, lalu menjumlahkan semuanya menjadi satu titik menggunakan fungsi point_sum. Lalu untuk bagian target disini adalah : menghitung titik target dengan cara: melakukan kombinasi linear dari 16 titik rahasia (hidden), tetapi kali ini menggunakan target_coeffs sebagai pengalinya. Lalu untuk instance disini adalah yang dijadikan output dari chall.sage dalam bentuk format json.

Lalu disini untuk langkah solve nya adalah :

1. Membuat N dan E fake di mana N memuat magic bytes dan rumus eksponensialnya menghasilkan nilai hash yang valid (membuka arsip untuk mendapatkan seluruh 64 points).
2. Lalu menggunakan nilai gift (orde grup) untuk memfaktorkan N menjadi p dan q
3. Discrete log :Karena kurva menggunakan modulus N^2, dan N sudah difaktorkan, pemain dapat memetakan titik-titik kurva ke grup Z_n dan menggunakan p-adic-lifting.
4. Setelah titik-titik diubah menjadi angka skalar (integer) di atas modulo N, masalah pencarian hidden poin berubah menjadi Sistem Persamaan Linear biasa
5. Setelah vektor hidden berhasil direkonstruksi, kalikan dengan deret bilangan prima (target_coeffs) untuk mendapatkan target.
6. Ambil koordinat-X dari target, jadikan kunci AES, dan decrypt flag

Lalu karena ada Hint 2 maka menjadi berikut : Kurva Eliptik pada modulo N^2 memiliki sebuah "subgrup" rahasia. Jika kita memiliki sebuah titik di subgrup ini, operasi

penjumlahan titik kurva tersebut bentuknya memiliki struktur yang persis sama dengan operasi penjumlahan angka biasa di modulo N ($\mathbb{Z}_N$). Jadi, homomorfisma adalah sebuah jembatan yang memetakan: Titik Geometri (Kurva) => Angka Skalar Biasa Z_N , dan karena ada jebakan di bagian LLL nya yaitu matriks yang kita temukan memiliki *sign/angka* yang berbeda dari aslinya, saat kita mengurutkannya (actual_cols.sort()), urutannya menjadi hancur berantakan dibandingkan urutan yang ada di server! Akibatnya, titik hidden_points tertukar, target_x menjadi salah, dan AES gagal didekripsi. Maka disini menggunakan Integer Linear Programming (ILP)

Dan berikut solver scriptnya.

```
import hashlib
import random
import sys
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
from pwn import remote, context
from sage.numerical.mip import MixedIntegerLinearProgram

context.log_level = 'error'

# ==========================================
# FASE 1: Interaksi Server & Ultra-Fast RSA Bypass
# ==========================================

print("[*] Menghubungkan ke server...")
r = remote("chall.itfestmicroipb.web.id", 30160)

r.sendlineafter(b"> ", b"3")
r.recvuntil(b"Magic: ")
magic_hex = r.recvline().strip().decode()
magic_num = int(magic_hex, 16)

print(f"[*] Received Magic Number: {hex(magic_num)}")

def get_smooth_prime(magic):
    print("[*] Generating ultra-fast smooth prime N (Metode A*B+1)...")
    P = magic << 248
    small_primes = list(primes(1000))

    while True:
        A = 1
        while A.bit_length() < 220:
            A *= random.choice(small_primes)

        X = P // A + 1
```

```
        for i in range(30000):
            B = X + i
            N = A * B + 1

            if pow(2, N - 1, N) == 1:
                if is_prime(N):
                    try:
                        facs = factor(B)
                        if max([p for p, e in facs]) < 2**34:
                            return N
                    except:
                        pass

N = get_smooth_prime(magic_num)

print("[*] Generating Finite Field and finding Primitive Root...")
F = GF(N)
g = F.multiplicative_generator()

print("[*] Selecting payload and calculating Discrete Logarithm...")
k_pow = 1
while True:
    if gcd(k_pow, N - 1) == 1:
        m_val = int(g**k_pow)
        m_bytes = m_val.to_bytes((m_val.bit_length() + 7) // 8, 'big')
        h = int.from_bytes(hashlib.sha384(m_bytes).digest(), 'big')

        if h < N and m_val > 1:
            E = F(h).log(F(m_val))
            break
    k_pow += 1

print("[*] Mengirim payload RSA ke server...")
N_int, E_int = int(N), int(E)
r.sendlineafter(b"N:> ", str(N_int).encode())
r.sendlineafter(b"E:> ", str(E_int).encode())
r.sendlineafter(b"data:> ", m_bytes.hex().encode())

resp = r.recv(4096, timeout=5)
if b"archive unlocked" not in resp:
    print("[-] GAGAL: Server menolak jawaban kita.")
    sys.exit(1)

print("[+] Archive successfully unlocked! Extracting secret points...")

# ==========================================
# PENGUMPULAN DATA
# ==========================================
r.sendline(b"1")
```

```
r.recvuntil(b"gift = ")
gift = int(r.recvline().strip())

r.sendlineafter(b"> ", b"4")
r.recvuntil(b"points = ")
points_str = r.recvline().strip().decode()
points_list = eval(points_str)
pts = [(int(x), int(y)) for x, y in points_list]

r.sendlineafter(b"> ", b"5")
r.recvuntil(b"target_coeffs = ")
target_coeffs = eval(r.recvline().strip().decode())
r.recvuntil(b"flagct = ")
flagct = eval(r.recvline().strip().decode())

r.sendlineafter(b"> ", b"6")
r.close()

# ========================================
# FASE 2: Smart's Attack (P-adic Logarithm di EC Paillier)
# ========================================
print("[*] Recovering secret modulus N^2 from points...")
D = 0
for i in range(len(pts) - 2):
    X1, Y1 = pts[i]
    X2, Y2 = pts[i+1]
    X3, Y3 = pts[i+2]

    A1 = Y1**2 - Y2**2 - X1**3 + X2**3
    B1 = X1 - X2
    A2 = Y2**2 - Y3**2 - X2**3 + X3**3
    B2 = X2 - X3

    delta = A1 * B2 - A2 * B1
    D = gcd(D, delta)

for p in primes(1000):
    while D > 0 and D % p == 0 and not is_square(D):
        D //= p

assert D > 0 and is_square(D), "N2 must be a perfect square!"
N2 = D
n = isqrt(N2)
inv = inverse_mod(X1 - X2, N2)
a = ((Y1**2 - Y2**2 - X1**3 + X2**3) * inv) % N2

def double(P):
    X, Y, Z = P
    if Z == 0: return (0, 1, 0)
    W = (a*Z**2 + 3*X**2) % N2
```

```
    S = (Y*Z) % N2
    B = (X*Y*S) % N2
    H = (W**2 - 8*B) % N2
    X3 = (2*H*S) % N2
    Y3 = (W*(4*B - H) - 8*Y**2*S**2) % N2
    Z3 = (8*S**3) % N2
    return (X3, Y3, Z3)

def add(P, Q):
    if P[2] == 0: return Q
    if Q[2] == 0: return P
    X1, Y1, Z1 = P; X2, Y2, Z2 = Q
    U1 = (Y2*Z1) % N2; U2 = (Y1*Z2) % N2
    V1 = (X2*Z1) % N2; V2 = (X1*Z2) % N2
    if V1 == V2:
        if U1 == U2: return double(P)
        return (0, 1, 0)
    U = (U1 - U2) % N2; V = (V1 - V2) % N2; W = (Z1*Z2) % N2
    A = (U**2*W - V**3 - 2*V**2*V2) % N2
    X3 = (V*A) % N2
    Y3 = (U*(V**2*V2 - A) - V**3*U2) % N2
    Z3 = (V**3*W) % N2
    return (X3, Y3, Z3)

def mul(k, P):
    R = (0, 1, 0); T = P
    for bit in bin(k)[2:][::-1]:
        if bit == '1': R = add(R, T)
        T = double(T)
    return R

print("[*] Applying P-adic Homomorphism map to Z_N...")
L_vals = []
for X, Y in pts:
    P = (X, Y, 1)
    Q = mul(gift, P)
    Xq, Yq, Zq = Q
    val = (Xq * inverse_mod(Yq, N2)) % N2
    L_vals.append((val // n) % n)

# ==========================================
# FASE 3: ILP Lattice Recovery (Mengalahkan LLL Trap)
# ==========================================
print("[*] Running LLL to get base lattice...")
M = matrix(ZZ, 65, 65)
for i in range(64):
    M[i, i] = 1
    M[i, 64] = L_vals[i]
M[64, 64] = n
```

```
red = M.LLL()
null_vecs = [list(row[:64]) for row in red if row[64] == 0]
V = matrix(ZZ, null_vecs)
B_mat = V.right_kernel().matrix().LLL()

print("[*] Extracting EXACT original random bytes matrix using ILP Polytope...")
found_cols = set()

# Mencari 16 matriks urandom asli yang dikurung di dalam rentang [0, 255]
while len(found_cols) < 16:
    p = MixedIntegerLinearProgram(maximization=True)
    u = p.new_variable(integer=True)

    for i in range(16):
        p.set_min(u[i], -100)
        p.set_max(u[i], 100)

    for j in range(64):
        expr = sum(u[i] * int(B_mat[i, j]) for i in range(16))
        p.add_constraint(expr, min=0, max=255)

    # Pastikan bukan vektor nol
    sum_expr = sum(sum(u[i] * int(B_mat[i, j]) for i in range(16)) for j in
range(64))
    p.add_constraint(sum_expr, min=1000)

    # Objektif acak agar ILP memantul ke semua sisi polihedron dan menemukan 16
vektor berbeda
    obj = sum(int(randint(-1000, 1000)) * u[i] for i in range(16))
    p.set_objective(obj)

    try:
        p.solve()
        vec = tuple(int(round(p.get_values(u[i]))) for i in range(16))
        col = tuple(int(sum(vec[i] * B_mat[i, j] for i in range(16))) for j in
range(64))

        if col not in found_cols:
            found_cols.add(col)
            sys.stdout.write(f"\r    -> Collected {len(found_cols)}/16 valid urandom
columns...")
            sys.stdout.flush()
    except:
        pass

print("\n[+] ILP Extraction 100% Successful!")
actual_cols = list(found_cols)
actual_cols.sort() # SEKARANG sort-nya dijamin 100% sama dengan server!
C = matrix(ZZ, actual_cols).transpose()
```

```
curve_order = n * gift
idx = None
C_sub_inv = None

print("[*] Searching for invertible submatrix modulo curve order...")
rows = list(range(64))
for _ in range(50000):
    sample = random.sample(rows, 16)
    M_sub = C[sample, :]
    det = M_sub.det()
    if det != 0 and gcd(det, curve_order) == 1:
        idx = sample
        C_sub_inv = M_sub.change_ring(Zmod(curve_order)).inverse()
        break

if C_sub_inv is None:
    raise Exception("[-] Gagal menemukan matriks yang invertible.")

print("[*] Reconstructing explicit hidden points...")
hidden_points = []
for i in range(16):
    h_i = (0, 1, 0)
    for j in range(16):
        weight = int(C_sub_inv[i, j])
        if weight == 0: continue
        P = (pts[idx[j]][0], pts[idx[j]][1], 1)
        term = mul(weight, P)
        h_i = add(h_i, term)
    hidden_points.append(h_i)

print("[*] Computing final target point...")
target_P = (0, 1, 0)
for i in range(16):
    weight = int(target_coeffs[i])
    if weight == 0: continue
    term = mul(weight, hidden_points[i])
    target_P = add(target_P, term)

Xq, Yq, Zq = target_P

if Zq == 0:
    target_x = 0
else:
    target_x = int((Xq * inverse_mod(Zq, N2)) % N2)

print(f"[+] Final Accurate Target X: {target_x}")

# Decrypt Flag
key = hashlib.sha256(str(target_x).encode()).digest()[:16]
cipher = AES.new(key, AES.MODE_CBC, b"\0" * 16)
```

```python
flag = unpad(cipher.decrypt(flagct), 16).decode()

print(f"\n[🚀] FLAG: {flag}")
```

```
[*] Extracting EXACT original random bytes matrix using ILP Polytope...
    -> Collected 16/16 valid urandom columns...
[+] ILP Extraction 100% Successful!
[*] Searching for invertible submatrix modulo curve order...
[*] Reconstructing explicit hidden points...
[*] Computing final target point...
[+] Final Accurate Target X: 95903556421698341316061371115716430167461446362071778599506275278013123539428008973167857020400498424413419811324120
43870300627322055140615578410623044369676161766851830400129040877188715813827757941833178123571986820486864257388187274268969729415952734838428
06267765089566669129904105376992819492225123801002535456562228963455989467679551767209614358412532175428524373134480918676671387571653165320745545
53553877040652711644454154263198480151048761776739 95

[🚀] FLAG: ITFest26{n6UyEnlUS_sTernlus_p1US_pIUs_arcHlVe_77223818aa9f}
```

FLAG: ITFest26{n6UyEnlUS_sTernlus_p1US_pIUs_arcHlVe_77223818aa9f}

AI Transparency : https://share.gemini.google/aCCYpIr6SXc5

Digital Forensic

petani durr (105 Points)

Bang durrr ingin menjadi petani modern, jadi dia ingin membuat hydroponic dirumahnya. Ia belajar sistem hydroponic menggunakan handphonenya. Tetapi ada apa dengan handphone bang durrr ini?

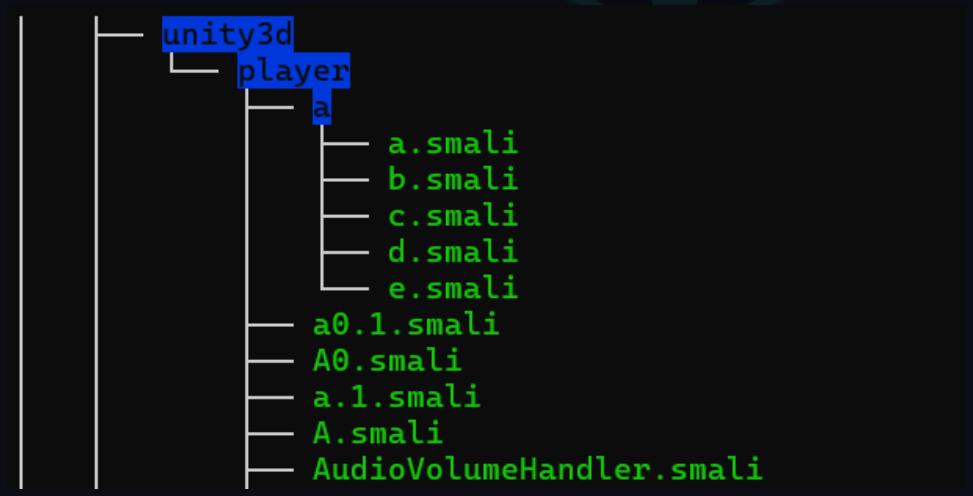
Password: LzShMlOxqKWlpvOmLJ5aVUOuoaI0LJ4=

Author: Fbrina

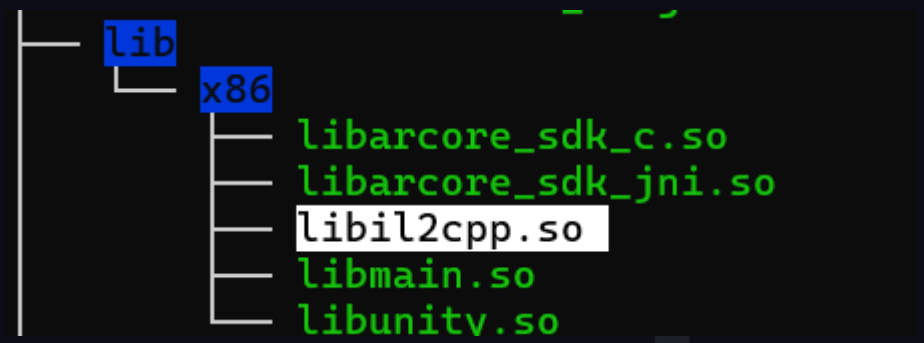
Oke ini dapat chall android dan archive dari internal stroagenya. Cek satu - satu di Downloads/ ada file .apk dan file yang kena enkrip

```
2026-08-07 10:31:52 ....A       374160              sdcard/Download/Hidroponik.pdf.vect
2026-08-07 10:31:57 ....A       485712              sdcard/Download/Kartu Benih.jpg.vect
2026-08-07 10:31:52 ....A       476816              sdcard/Download/Kartu Panen.jpg.vect
2026-08-07 10:31:57 ....A        29504              sdcard/Download/Laporan_Hidroponik.docx.vect
2026-08-07 10:31:57 ....A       257616              sdcard/Download/Laporan_Hidroponik.pdf.vect
2026-08-07 10:31:57 ....A     56365496              sdcard/Download/ViroPonicAR.apk
2026-08-07 10:31:52 ....A        13328              sdcard/Pictures/.thumbnails/13.jpg.vect
2026-08-07 10:31:52 ....A        12272              sdcard/Pictures/.thumbnails/14.jpg.vect
2026-08-07 10:31:52 ....A        13344              sdcard/Pictures/.thumbnails/23.jpg.vect
2026-08-07 10:31:52 ....A        12288              sdcard/Pictures/.thumbnails/24.jpg.vect
------------------- ----- ------------ ------------  ------------------------
2026-08-07 10:33:03          230213779     48100128  31 files, 24 folders
```

Sebenarnya flow dari solving nya cukup simple, decompile .apk -> dapet key -> decrypt. But the shitty part is kenapa ada dua encryption routine: file kecil sama file besar 😡. Oke masuk ke analisis .apk nya. Dari output apktool d ViroPonicAR.apk ini adalah aplikasi android yang dibikin dari Unity (woi ndak pernah megang Unity)



Well saya nyari function buat encrypt sama C2 nya ngga ketemu cuma dari apktool doang. Abis tu saya tree . trus lempar GPT biar tau mana yang harus saya analisa. Turns out ternyata .apk ini dibuild dari Unity IL2CPP.



This looks promising ya, karena encryption routine-nya ngga didapet dari .smali. So, kata gpt kita bisa dapat pake il2cppdumper. Enaknya udh ada yg bikin tools tersebut secara online https://il2cppdumper.com/. Jadi tinggal masukin libil2cpp.so sama assets/bin/Data/Managed/Metadata/global-metadata.dat. Hasilnya adalah kita dapat dump.cs. CTRL + F AES_KEY nemu region dari encryption routinenya

dump.cs

AES_KEY

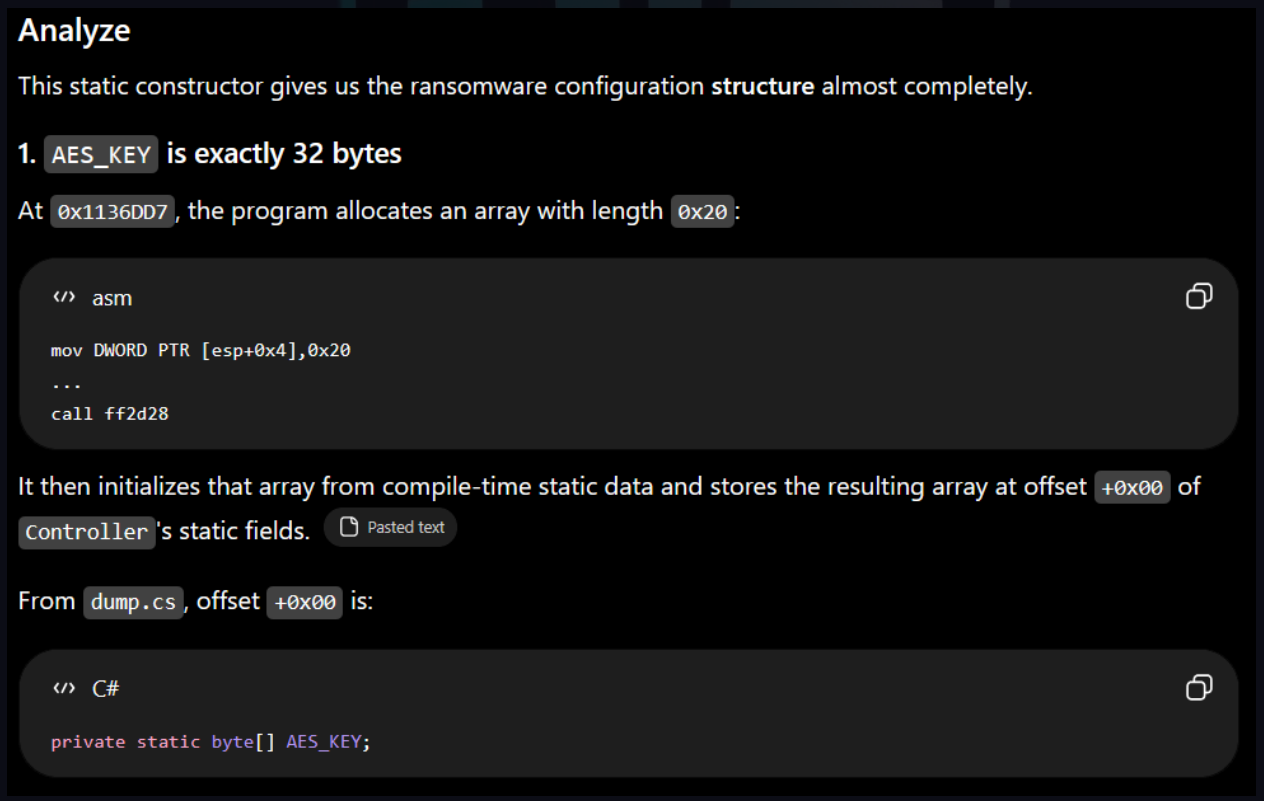
```
470060  // Namespace:
470061  public class Controller : MonoBehaviour // TypeDefIndex: 7998
470062  {
470063      // Fields
470064      private static byte[] AES_KEY; // 0x0
470065      private static readonly string C2_SERVER; // 0x4
470066      private static readonly string FLAG_ENDPOINT; // 0x8
470067      private static readonly string EXFIL_ENDPOINT; // 0xC
470068      private static readonly long SIZE_THRESHOLD; // 0x10
470069      private static readonly int CHUNK_COUNT; // 0x18
470070      private static string FLAG_STRING; // 0x1C
470071      private static readonly string[] TARGET_FOLDERS; // 0x20
470072      private static readonly string[] TARGET_EXTENSIONS; // 0x24
470073
```

Another shitty part here, dll namespace nya ngga ke detect jadi ngga bisa langsung import ke ILSpy (cok masih suruh nguli offset). Let GPT cook, dan katanya kode buat encryption ada di

function ExecuteRansomware di offset 0x11312E5. Dan juga ditemukan function - function encryption lain ya.



Ya sudah lah, karena ghidra saya lag, suruh gpt pakekan objdump dari offset-offsetnya ini. Dari 0x1136c76 - 0x113712e, ditemukan kalo di 0x1136DD7 program alokasi array dengan panjang 0x20 atau 32 bytes (tipikal aes-256) dan ini nyimpen data dari offset +0x00 yang mana adalah AES_KEY.



Terus karena cuma ada 1 32-byte blob ini bisa jadi adalah keynya, untuk offsetnya ada di 0x492AD8. Langsung aja dump dari global-metadata.dat.

```
// Namespace:
[CompilerGenerated]
internal sealed class <PrivateImplementationDetails> // TypeDefIndex: 8118
{
   // Fields
   internal static readonly <PrivateImplementationDetails>.__StaticArrayInitTypeSize=32
30C69175FD99650A82A8C1426D86DF069CC4F585CF0C6E34F7025729DFB45F00
/*Metadata offset 0x492AD8*/; // 0x0
```

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/petani/sdcard/D
data]
└─$ xxd -g 1 -s 0x492ad8 -l 32 global-metadata.dat
00492ad8: 53 61 6d 70 6c 65 4b 65 79 46 6f 72 43 54 46 47  SampleKeyForCTFG
00492ae8: 61 6d 65 41 45 53 32 35 36 42 69 74 73 21 21 21  ameAES256Bits!!!
```

Sesuai dump.cs lagi, kita objdump offset dari EncryptFile(). Dan dapatlah routine dari EncryptFIle() yaitu encrypt pake key yang ditemukan sebelumnya, padding pkcs#7, trus IV yang di-random tapi dimasukin ke last 16 bytes dari encrypted file. But this all is for only small file < 131072.

Therefore the on-disk layout is:

```
+---------------------------+----------------+
|       AES ciphertext      |       IV       |
|       variable size       |    16 bytes    |
+---------------------------+----------------+
                            ^
                     last 16 bytes
```

So for an encrypted small file:

```
IV         = encrypted_file[-16:]
ciphertext = encrypted_file[:-16]
```

Nah karena cuma ada 1 file doang yang size nya dibawah 131072 jadinya kita pindah ke EncryptLargeFile (sumpah 1 doang napa sih). Mana lebih susah lagi, intinya split 4 chunks -> encrypt kyk small file = encrypted chunk -> buat tiap chunk IV baru then encrypted chunk -> trus digabung gitu deh.

```
└─ size > 128 KiB
     │
     ├─ split file into 4 chunks
     │
     ├─ UTF8(FLAG_STRING)
     │     │
     │     └─ AES-256-CBC-PKCS7
     │          random IV
     │
     │          encryptedFlag || flagIV
     │
     ├─ for each of 4 chunks:
     │     └─ AES-256-CBC-PKCS7
     │          fresh IV
     │          encryptedChunk
     │
     ├─ concatenate encrypted chunks
     ├─ incorporate encrypted flag blob
     └─ write output
          │
          └─ delete original
```

Singkat cerita, GPT pick file Hidroponik.pdf.vect buat di decrypt duluan. Dia nyuruh buat paste last 64 bytes buat predict IV dari chunk yang random ini.

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/petani/sdcard/Download]
└─$ xxd -g 1 -s -64 -l 64 Hidroponik.pdf.vect
0005b550: 32 d2 0b 1e d8 72 c0 1d e4 1f 2f 35 bd fd 76 c0  2....r..../5..v.
0005b560: f7 3b 0e 2b 24 71 4f 1c cd 2c e9 0f 37 09 14 62  .;.+$qO..,..7..b
0005b570: 7d f1 8f 04 75 c2 19 80 ae 97 5e 9d 6b 81 01 b5  }...u.....^.k...
0005b580: 1d 86 9e 39 c2 53 b5 4c fe 99 e7 8b 0e e5 63 33  ...9.S.L......c3
```

Dari last 64 bytes, berhasil recover sebagian tapi encrypted bytes yang disuplai ternyata kurang jadi tambahin lagi 16 bytes, total yang di carve ada 80 bytes.

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/petani/sdcard/Download]
└─$ xxd -g 1 -s -80 -l 80 Hidroponik.pdf.vect
0005b540: 7f e1 54 62 f8 07 80 25 3c 1b 1d 3f 19 aa 7a 1d  ..Tb...%<..?..z.
0005b550: 32 d2 0b 1e d8 72 c0 1d e4 1f 2f 35 bd fd 76 c0  2....r..../5..v.
0005b560: f7 3b 0e 2b 24 71 4f 1c cd 2c e9 0f 37 09 14 62  .;.+$qO..,..7..b
0005b570: 7d f1 8f 04 75 c2 19 80 ae 97 5e 9d 6b 81 01 b5  }...u.....^.k...
0005b580: 1d 86 9e 39 c2 53 b5 4c fe 99 e7 8b 0e e5 63 33  ...9.S.L......c3
```

Jadinya IV bisa diambil dari data[-32:-16] , and here is the decryption script. And luckily the flag is in Hidroponik.pdf

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

import argparse
from pathlib import Path
```

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad


KEY = b"SampleKeyForCTFGameAES256Bits!!!"
BLOCK_SIZE = 16


def decrypt_flag(path: Path):
    data = path.read_bytes()

    if len(data) < 80:
        raise ValueError("File is too small to use the large .vect format")

    # Large-file tail:
    #
    # [ ... file data ... ]
    # [ encrypted FLAG : 48 bytes ]
    # [ FLAG IV        : 16 bytes ]
    # [ chunk 3 IV     : 16 bytes ]

    flag_ciphertext = data[-80:-32]
    flag_iv = data[-32:-16]
    last_chunk_iv = data[-16:]

    print(f"[+] File          : {path}")
    print(f"[+] Size          : {len(data)} bytes")
    print(f"[+] Flag ciphertext: {flag_ciphertext.hex()}")
    print(f"[+] Flag IV       : {flag_iv.hex()}")
    print(f"[+] Chunk 3 IV    : {last_chunk_iv.hex()}")

    cipher = AES.new(
        KEY,
        AES.MODE_CBC,
        flag_iv
    )

    padded = cipher.decrypt(flag_ciphertext)

    try:
        plaintext = unpad(
            padded,
            BLOCK_SIZE,
            style="pkcs7"
        )
    except ValueError:
        print("[-] Invalid PKCS#7 padding")
```

```
        print(f"[-] Raw plaintext: {padded.hex()}")
        return

    print(f"[+] Raw flag bytes : {plaintext!r}")

    try:
        flag = plaintext.decode("utf-8")
    except UnicodeDecodeError:
        flag = plaintext.decode("utf-8", errors="replace")

    print()
    print(f"[+] FLAG: {flag}")


def main():
    parser = argparse.ArgumentParser(
        description="Extract embedded flag from large .vect files"
    )

    parser.add_argument(
        "file",
        help="Large encrypted .vect file"
    )

    args = parser.parse_args()

    path = Path(args.file)

    if not path.is_file():
        raise SystemExit(f"[-] File not found: {path}")

    decrypt_flag(path)


if __name__ == "__main__":
    main()
```

FLAG: ITFest26{Yeyyyy_s0lv3d_w4ktuny4_m3n4n4m!}

AI Proof :

https://chatgpt.com/share/6a76bc6d-8170-83ec-95d5-cdbb0b342df3

durja otw magang (220 Points)

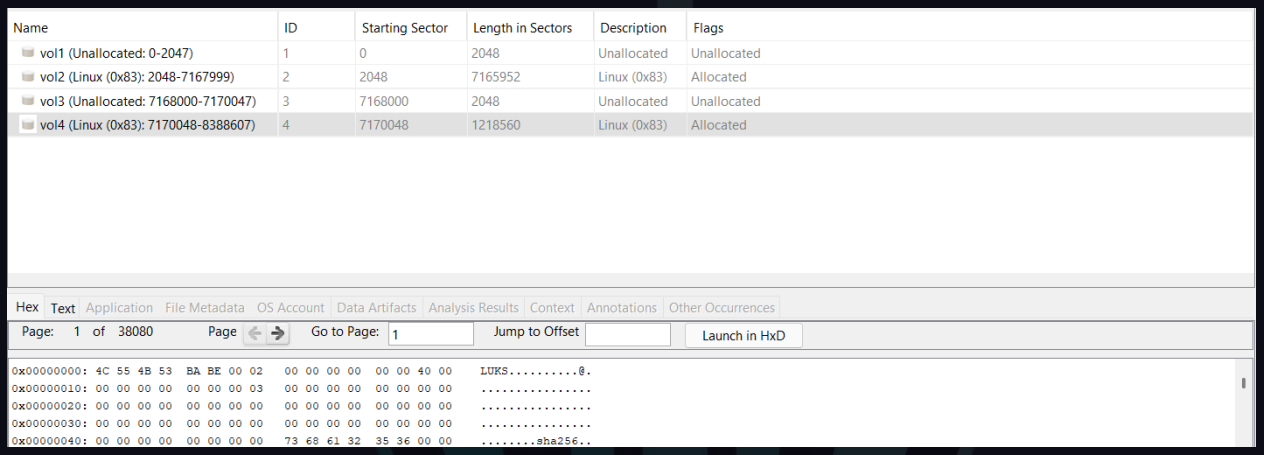
bang dur otw magang tapi laptopnya kena hack, gimana nyak?

Password: 53599bce04a968bd

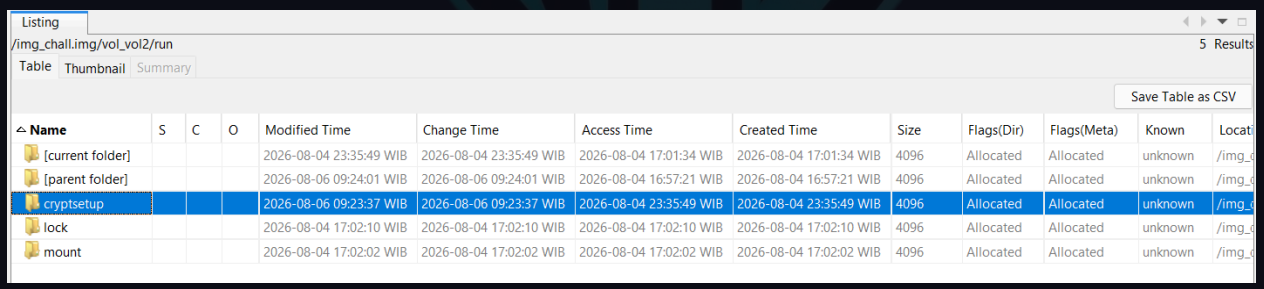
P.S: nggak tanggung jawab kalo ngerjainnya nggak di safe environment

Author: asburg

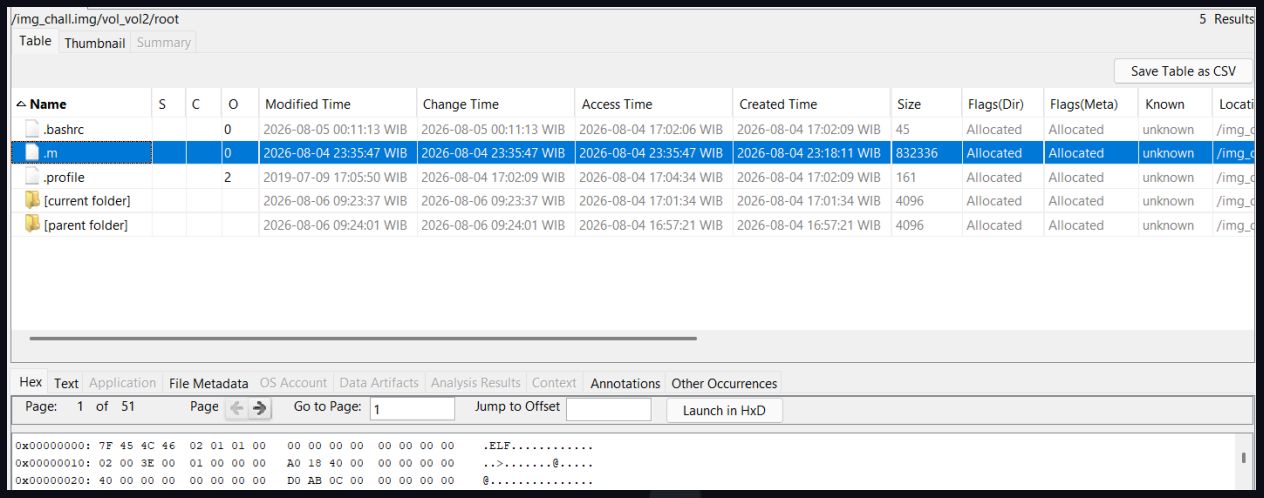
Kali ini dikasih file .img. Load ke autopsy ada dua partisi, partisi pertama valid linux fs dan partisi kedua pas dicek kok ada LUKS header.



Dan bener aja pas dicek di partisi 1 ada cryptsetup yang ke install



Next step, tinggal nyari passphrase yang buat nge encrypt partisi ini (tidak semudah itu aowkawokawokwao). Check ke home dir kosong, tapi cek ke root dir ada executable (here we go again).



Wayahe gpt beraksi, intinya that ELF nge invoke command echo -n '%s' | cryptsetup luksFormat %s -q --key-file=- 2>/dev/null. Nah, this %s is the hardest to find.

Analyze

This new function confirms that `FUN_0044b900` is **formatting infrastructure**, not part of the LUKS-key derivation itself.

The strongest evidence comes from its callers:

C

```
FUN_0044b900(
    auStack_238,
    0x200,
    1,
    0x200,
    "echo -n '%s' | cryptsetup luksFormat %s -q --key-file=- 2>/dev/null",
    &DAT_004ca4a0,
    "/dev/sdb2"
);
```

Oke buat passphrase nya itu dibuat dari DAT_0049a201 xor with DAT_0049a1e1. Recover staticly dari data blob di ELF doesn't give any correct passphrase. The correct passphrase can be return by nguli di syslog ya karena dari sana bisa nemu "bekas" eksekusi dari .m.

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/durja]
└─$ sudo grep -aEin 'sdb|stolen|device-mapper|dm-[0-9]+|EXT4-fs|/tmp/.mnt' /mnt/chall-root/var/log/syslog /mnt/chall-root/var/log/kern.log 2>/dev/null
/mnt/chall-root/var/log/syslog:973:◆◆H◆I◆H◆MH◆◆tg1◆◆fDH◆◆H9◆tTH◆◆H◆◆H◆|t◆◆'◆◆◆L◆◆◆◆      ◆◆I◆L◆I◆H◆◆t◆1◆◆
                                                                                                                H◆◆H9◆t◆H◆◆H◆◆I◆
|t◆◆◆◆◆◆H◆◆◆     ◆◆I◆◆◆◆◆L◆◆◆    ◆◆I◆DM◆I◆
                                                $H◆◆t◆1◆◆@H◆◆H9◆◆x◆◆◆H◆◆H◆◆I◆|t◆◆◆◆◆◆b ◆◆I◆M◆tI◆H◆◆t◆1◆◆f◆H◆◆H9◆t◆H◆◆H◆◆I◆|t◆◆G
◆◆◆◆◆H◆=]^H◆◆tOU1◆SH◆◆H◆/H◆◆t◆H◆◆H◆◆◆H◆◆u◆H◆=!^H◆H◆◆◆u◆H◆[]◆◆◆◆◆◆H◆H◆◆%02x/tmp/.mnt/dev/sdb2/homecp -r %s/* %s/ 2>/dev/nullumount %s 2>/dev/nullrm -rf /home/* 2>/dev/null0123456789abcdefecho -n '%s' | cryptsetup luksFormat %s -q --key-file=- 2>/dev/nullecho -n '%s' | cryptsetup open %s stolen --key-file=- 2>/dev/nulldd if=/dev/zero of=/dev/mapper/stolen bs=1M 2>/dev/nullmkfs.ext4 -F -L %s /dev/mapper/stolen 2>/dev/nullmount /dev/mapper/stolen %s 2>/dev/nullcryptsetup close stolen 2>/dev/null◆;Bl◆◆MG
```

Dengan script ini kita bisa dump region dari key-array nya

```
from pathlib import Path

p = Path("/mnt/chall-root/var/log/syslog")
data = p.read_bytes()

sig = b"%02x\x00/tmp/.mnt\x00/dev/sdb2\x00/home\x00"

off = data.find(sig)

print(f"signature offset = {off}")
if off < 0:
    raise SystemExit("signature not found")

print(f"signature offset hex = 0x{off:x}")

start = off + 0x1d0
chunk = data[start:start+0x70]

for i in range(0, len(chunk), 16):
    row = chunk[i:i+16]
    print(f"+0x{0x1d0+i:03x}: " + " ".join(f"{b:02x}" for b in row))
```

Dari situ, kita tau bahwa blob data di ELF .m itu beda dengan yang di syslog menandakan itu udah dimodif. Buat recover passphrasenya sama aja, yaitu DAT_0049a201 xor with DAT_0049a1e1.

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/durja]
└─$ python3 k.py
signature offset = 631001
signature offset hex = 0x9a0d9
+0x1d0: 00 00 00 00 00 00 00 00 00 00 00 00 ec 3b 42 6c
+0x1e0: aa c6 4d 47 0a 22 24 58 0b 55 dc da 0c b1 9b d3
+0x1f0: 76 50 be 8f 9d ed fe f6 3b bf 27 75 92 ea ed c6
+0x200: 7a b6 c0 0d e7 1d 7a 89 c2 cc 58 b4 65 d1 21 fb
+0x210: 5c b6 2c 9e d4 1f 96 19 45 99 14 26 e6 79 f6 ff
+0x220: d6 79 f6 ff 07 7a f6 ff 1e 7a f6 ff ce 79 f6 ff
+0x230: 3c 7a f6 ff 51 7a f6 ff 6d 7a f6 ff 62 7a f6 ff
```

Hasilnya adalah 7ed1afaad0708d4aed3f5ed1c999846e6960ba282ae6921149f268ef7e263353. Langsung aja decrypt LUKS nya dan baca flagnya

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/durja]
└─$ printf '%s' \
'7ed1afaad0708d4aed3f5ed1c999846e6960ba282ae6921149f268ef7e263353' \
| sudo cryptsetup open \
    --readonly \
    /dev/loop0p2 recovered \
    --key-file=-
[sudo] password for byte:

┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/durja]
└─$ sudo mount -o ro,noload /dev/mapper/recovered /mnt/recovered

┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/durja]
└─$ tree /mnt/recovered
/mnt/recovered
├── durja
│   └── Documents
│       └── Kerja
│           └── catatan_magang.txt
└── lost+found  [error opening dir]

5 directories, 1 file

┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/durja]
└─$ cat /mnt/recovered/durja/Documents/Kerja/catatan_magang.txt
Catatan Harian Magang - PT Nusantara Teknologi
==============================================
```

FLAG:

ITFest2026{sec_eng_bravo_4th_place_lmitd_top_1_region_cbc_rev_role_olivia_big_congratulations_top_solver}

AI Proof :

https://chatgpt.com/share/6a76ad98-a3cc-83ec-a397-afa143ab4ba1

https://claude.ai/share/215cc64a-5904-407e-9ed6-3a4d648e80ba

Traitor (247 Points)

Sensitive internal company data has been leaked. Investigators suspect an employee is behind it. Our IT team has taken a forensic image of the suspect's personal work device, believed to have been used to send the data somewhere holding a large amount of information.

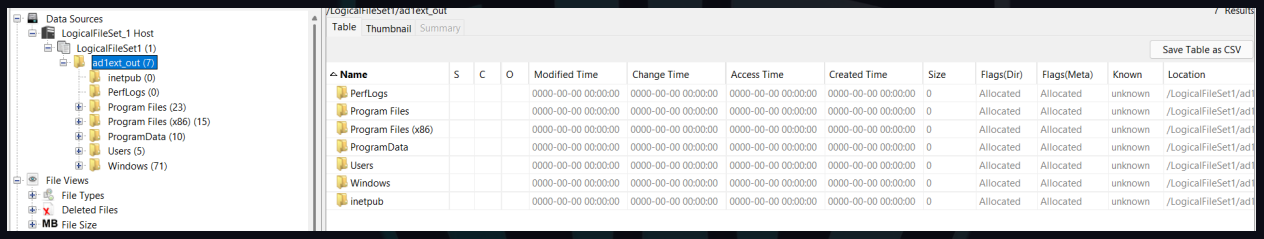
Can you uncover the trail of evidence and reconstruct what happened?

Password: 6a698600-a5dc-83ec-9ab3-0ea9d1c1f24a

Author: Farel

Aren't you paying attention to the time?

Dikasih .ad1, dari hint suruh perhatiin ts. Karena di mount sama ftk imager kurang jelas buat MAC ts nya, jadi saya pake ad1extract -> mount ke autopsy.

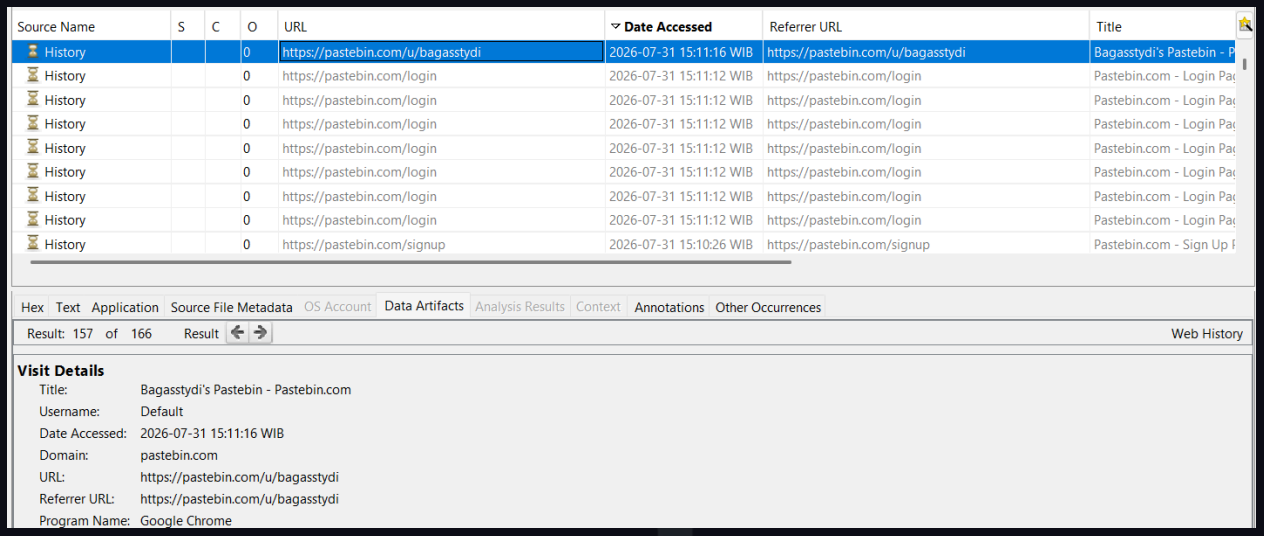


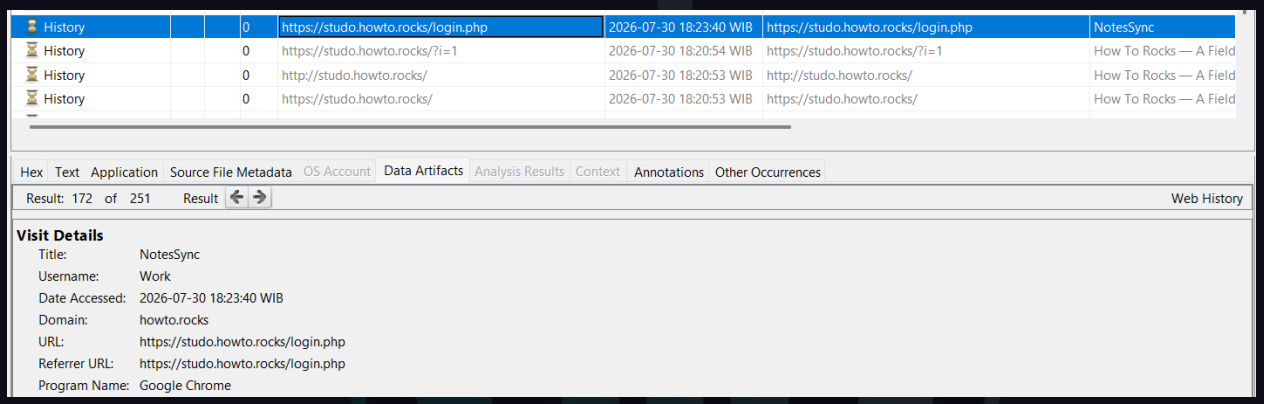
Langsung aja pake timeline tools dari autopsy. Dari timeline akhir-akhir itu banyak web access, sesuai sama deskripsi yang katanya ada internal data yg ke leak.

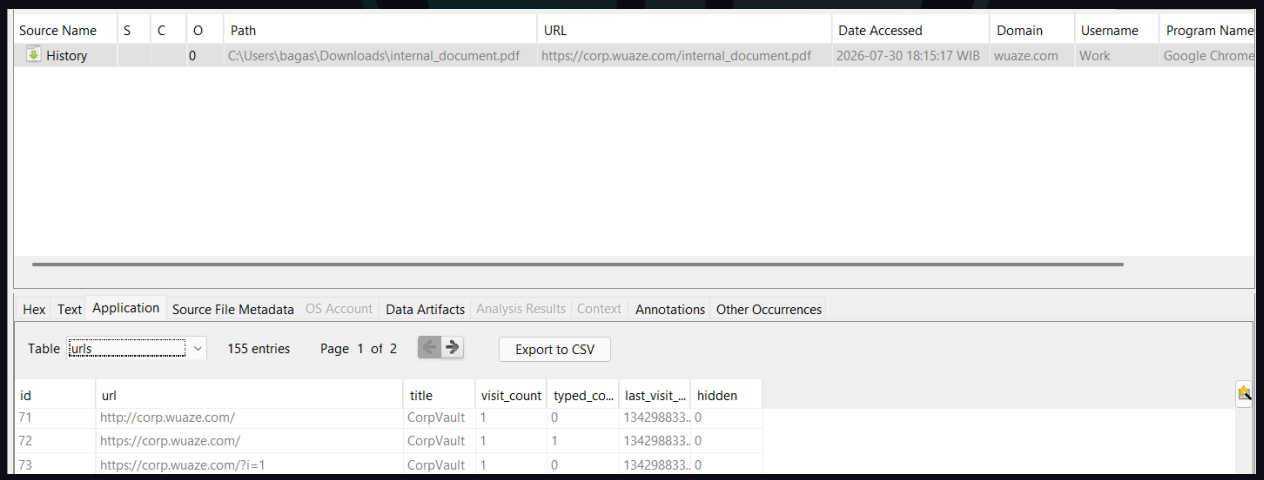


Dari history, dapet lah beberapa kandidat yaitu pastebin.com, studo.howto.rocks, sama corp.wuaze.com.

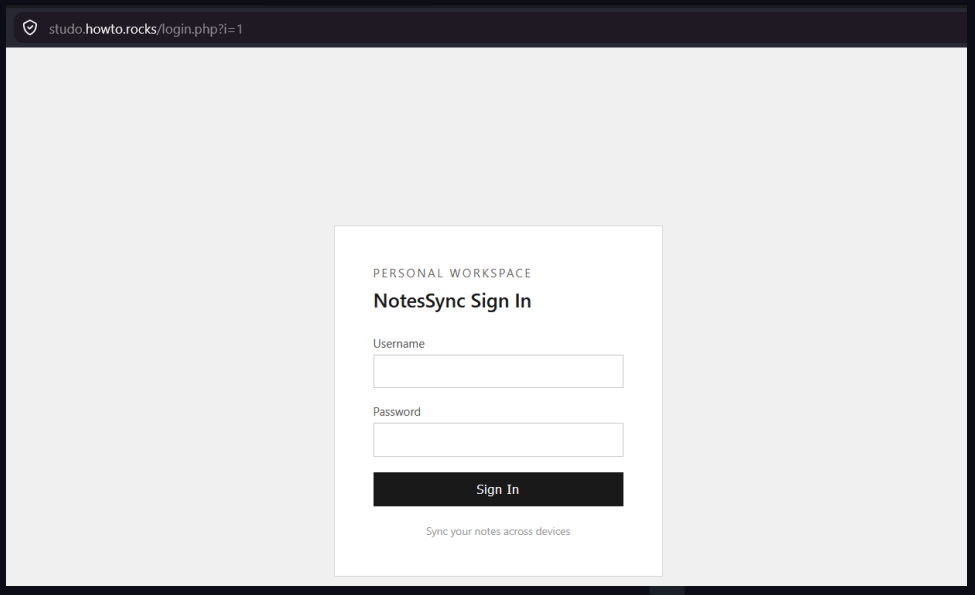
Presented by @hackitbraw_ub



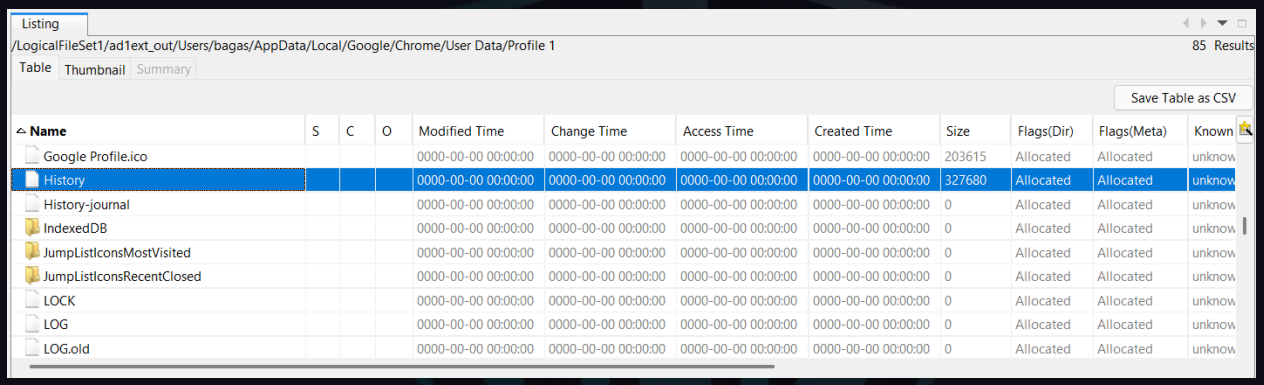




Nah saya kira itu decrypt Login Data trus masuk pake creds dari pastebin.com but not. Jadi disini saya akses ke dua web tadi dan yang bisa ngikutin flow history itu cuma yang studo.howto.rocks. But we need the creds.



The tricky part is assuming this is from Default chrome user but isn't. If you follow the file path bakal ngarah ke Profile 1.



Buat decrypt Login Data kita perlu password computer user. Hal ini bisa direcovery dengan DPAPImk2john dan kita tinggal supply sid dan masterkeynya then bf with rockyou. ref https://www.hackthebox.com/blog/seized-ca-ctf-2022-forensics-writeup (also from TECHFEST 2026, mksh mas owen)

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ DPAPImk2john --sid="S-1-5-21-2854916940-654886857-3489945081-1001" --masterkey="f0f96789-a673-479a-83ee-5a10a81adee2
" --context="local" > hash.txt

┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ john hash.txt -wordlist=~/wordlist/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (DPAPImk, DPAPI masterkey file v1 and v2 [SHA1/MD4 PBKDF2-(SHA1/SHA512)-DPAPI-variant 3DES/AES256
 256/256 AVX2 8x])
No password hashes left to crack (see FAQ)

┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ john hash.txt --show
?:123

1 password hash cracked, 0 left
```

Actually next we can use mimikatz, but i prefer pypykatz. The first step is creating the prekey then decrypt the masterkey with that prekey.

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ pypykatz dpapi prekey password S-1-5-21-2854916940-654886857-3489945081-1001 123 -o prekey.txt

┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ pypykatz dpapi masterkey f0f96789-a673-479a-83ee-5a10a81adee2 prekey.txt -o masterkey.txt

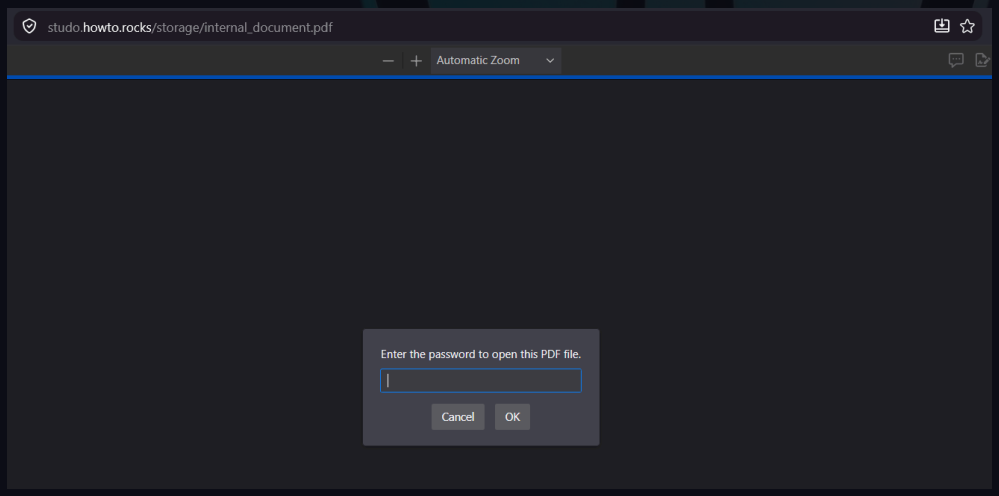
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ cat prekey.txt
6bcfd13800f67f5cbe59488bda0a0f0b16968372
141c385b2fc43f4466391fd3e1b823ddbc2aad64
a2fd0d7fa8b217d09380b886fd9627c8711e456b
0d5399508427ce79556cda71918020c1e8d15b53

┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ cat masterkey.txt
{
    "backupkeys": {},
    "masterkeys": {
        "f0f96789-a673-479a-83ee-5a10a81adee2": "68bfec7a4f6ede084343bbb4070159d4864e4754c8f6108f6755976e6371eeb3006d9d5
d82d9c01bb8592f31f82a47752b800c5fb965c8fc5fbfb2983bcea164"
    }
}
```

Then we can continue by dumping the Local State and Login Data of Profile 1. Then we can decrypt that Login Data and get the creds

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ pypykatz dpapi chrome masterkey.txt "Local State" --logindata "Login Data"
file: Login Data user: bagas pass: b'Admin123#' url: https://corp.wuaze.com/
file: Login Data user: bagas pass: b'6a698600-a5dc-83ec-9ab3-0ea9d2c2f24a' url: https://studo.howto.rocks/login.php
```

Login to website and we get the document. But it's protected.



Download it then bf the password with pdf2john and hashcat

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ pdf2john internal_document.pdf > pdf.hash

┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ grep -o '\$pdf\$.*' pdf.hash > hash.txt

┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ hashcat -m 10500 hash.txt ~/wordlist/rockyou.txt
hashcat (v7.1.2) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, SPIR-V, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
====================================================================================================================================================
* Device #01: cpu-haswell-12th Gen Intel(R) Core(TM) i5-12500H, 2221/4443 MB (1024 MB allocatable), 16MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 32
Minimum salt length supported by kernel: 0
Maximum salt length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.
      For more information, see https://hashcat.net/faq/potfile

Started: Sun Aug  9 02:26:03 2026
Stopped: Sun Aug  9 02:26:04 2026

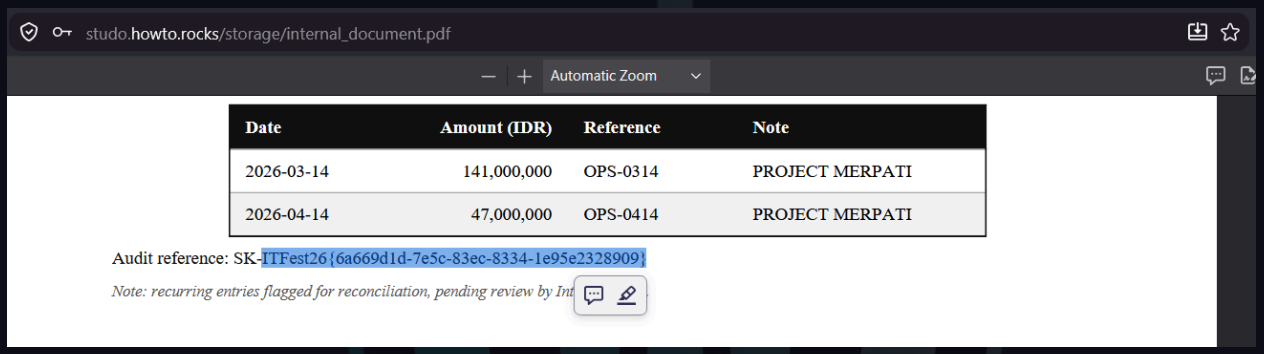
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/foren/chall]
└─$ hashcat hash.txt --show
Hash-mode was not specified with -m. Attempting to auto-detect hash mode.
The following mode was auto-detected as the only one matching your input hash:

10500 | PDF 1.4 - 1.6 (Acrobat 5 - 8) | Document

NOTE: Auto-detect is best effort. The correct hash-mode is NOT guaranteed!
Do NOT report auto-detect issues unless you are certain of the hash type.

$pdf$4*4*128*-1028*1*16*55d2e6cc78343fbf1114237515476ddb*32*1dfaba46b44297e0e247db63730ccb4a0021446990b9e4114071a4d9104984c1*32*1ecbadfcbb525de04038dd474117d355faaeffafd1deb54d27005dbfed30ddef:malachite
```

Then we get the flag



FLAG: ITFest26{6a669d1d-7e5c-83ec-8334-1e95e2328909}

AI Proof :

https://chatgpt.com/share/6a76db99-2ffc-83ec-b54e-742ef4136b76

https://claude.ai/share/b5119151-d29e-4ed7-8fbe-72783b204794

Miscellaneous

Kingdom of Catofel (102 Points)

Greeting, young adventurers! Welcome to Kingdom of Catofel. Looking for a legendary flag, I guess?

An elf, realizing your high potential, decided to lead you to a magic chest with the flag inside. But the chest, since it contains the most valuable treasure, needs a key which is as magical as the chest.

But, to gain the key, a strange rule exists. Adventurer needs to visit places in the right order.

Most adventurers failed because they didn't pay attention for the clues throughout the journey. But one day, a group led by Rubillion Charis found the clues and made the right order. Unfortunately, they died before reaching the chest. Something seemed to attack them while they were sleeping at the forest near a temple, and Rubillion's journal was found near her body.

In order to acquire the magic key, the elf gave you some relics:

Adventurer's Handbook
Rubillion's Journal
These relics could help you to unlock the chest and claim the flag.

Good luck, adventurers. Beware of the kaiju. Happy traveling!

FLAG FORMAT: ITFest26{...}

Author: nuby

Alamak blm apa-apa udah dikasih light novel. Dari dist.zip isinya adalah file pdf, protected 7z, dan file txt.

```
   Date      Time    Attr         Size   Compressed  Name
------------------- ----- ------------ ------------  ------------------------
2026-08-05 15:10:24 .....     11530690     11530690  magic_chest.7z
2026-08-05 15:10:24 .....         2575         1251  Rubillion's journal.txt
2026-08-05 15:10:24 .....     23615291     23493889  adventurer's handbook.pdf
------------------- ----- ------------ ------------  ------------------------
2026-08-05 15:10:24           35148556     35025830  3 files
```

Baca file txt nya adalah jurnal dari perjalanan adventurernya.

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/misc]
└─$ cat Rubillion\'s\ journal.txt
Day 17

After long days of journey, we finally made it close to this kingdom. Kingdom of Catofel. A kingd
cripts. They said that this place holds such priceless treasure, and they call it "flag". We don'
ag? Or pirate flag? But we're so intended to find out.

Day 18

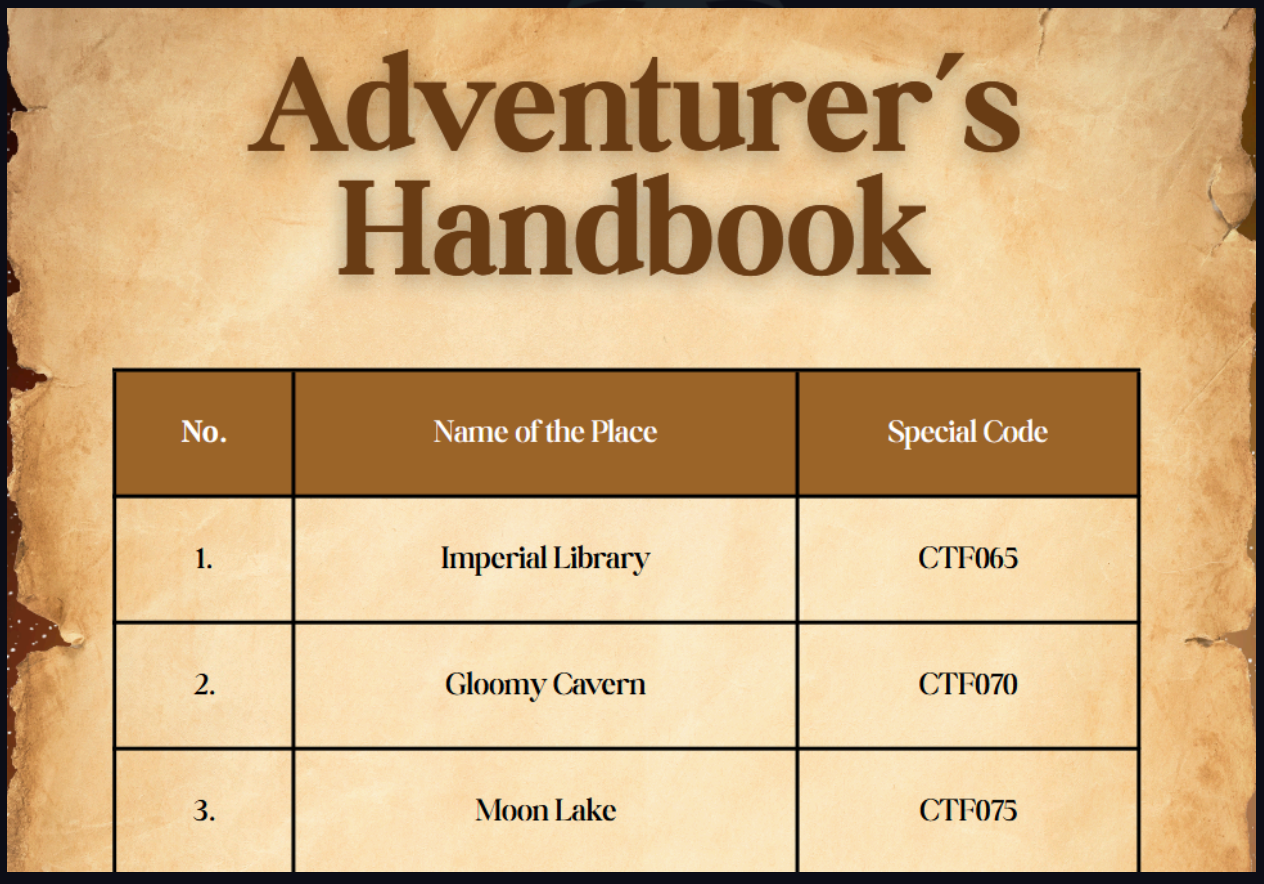
We headed to the South Gate. Two guards standing in front of it with their big axe crossing each

"We are here to look for the legendary flag", I said as I handed them a permission letter.

The guard read the letter, nodded, and opened the gate for us. We walked in.

Day 19
```

Sesuai dengan deskripsinya kita harus ngurutin path dari adventurer menurut jurnal ini. Nah di file pdf ada kyk kode tiap lokasinya jadi tinggal di mapping aja nanti.



Adventurer's Handbook

No.	Name of the Place	Special Code
1.	Imperial Library	CTF065
2.	Gloomy Cavern	CTF070
3.	Moon Lake	CTF075

Mapping place ke special code, yang kebentuk adalah CTF077 CTF065 CTF071 CTF073 CTF067. Langsung masukkan gpt karena saya ndak tau maksudnya, ternyata cuma ascii dan passwordnya adalah MAGIC.

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/misc]
└─$ 7z l magic_chest.7z -pMAGIC

7-Zip 26.00 (x64) : Copyright (c) 1999-2026 Igor Pavlov : 2026-02-12
 64-bit locale=en_US.UTF-8 Threads:16 OPEN_MAX:10240, ASM

Scanning the drive for archives:
1 file, 11530690 bytes (11 MiB)

Listing archive: magic_chest.7z

--
Path = magic_chest.7z
Type = 7z
Physical Size = 11530690
Headers Size = 210
Method = LZMA2:12m 7zAES
Solid = -
Blocks = 1

   Date      Time    Attr         Size   Compressed  Name
------------------- ----- ------------ ------------  ------------------------
2026-08-05 14:14:06 ....A     11656650     11530480  flag.pdf
------------------- ----- ------------ ------------  ------------------------
2026-08-05 14:14:06           11656650     11530480  1 files
```

FLAG: ITFest26{c0n9r4tul4t10n5}

AI Proof :

https://chatgpt.com/share/6a76c4d1-8610-83ec-ad78-d69cefe66e05

Mr. Durrr the Traveler (166 Points)

> Mr. Durrr recovered an old archive of photographs from years of wandering. He remembers leaving a small personal note inside one of them, but no longer remembers which. Can you find it?
>
> Author: Faizylux

Oke ini ala-ala zip bomb gitu isinya foto yang sangat banyak.

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/misc]
└─$ 7z l dist-memorize.7z | tail
2026-07-26 21:49:31 ....A        15360              memorize/photos/64/99/92.jpg
2026-07-26 21:49:32 ....A        15360              memorize/photos/64/99/93.jpg
2026-07-26 21:49:32 ....A        15360              memorize/photos/64/99/94.jpg
2026-07-26 21:49:31 ....A        15360              memorize/photos/64/99/95.jpg
2026-07-26 21:49:33 ....A        15360              memorize/photos/64/99/96.jpg
2026-07-26 21:49:32 ....A        15360              memorize/photos/64/99/97.jpg
2026-07-26 21:49:32 ....A        15360              memorize/photos/64/99/98.jpg
2026-07-26 21:49:32 ....A        15360              memorize/photos/64/99/99.jpg
------------------- ----- ------------ ------------  ------------------------
2026-07-26 21:53:43         9984000000     12866984  650000 files, 6567 folders
```

Wandering in internet, nemu tools ini https://github.com/yo-wotop/7z-tools tapi setelah di run ternyata error.

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/misc]
└─$ python3 7z-tools/src/7zsteg.py dist-memorize.7z
Traceback (most recent call last):
  File "/mnt/c/Users/ACER/Downloads/CTF/itfest26/misc/7z-tools/src/7zsteg.py", line 124, in <module>
    main()
    ~~~~^^
  File "/mnt/c/Users/ACER/Downloads/CTF/itfest26/misc/7z-tools/src/7zsteg.py", line 35, in main
    all_files = os.listdir(folder)
FileNotFoundError: [Errno 2] No such file or directory: '/mnt/c/Users/ACER/Downloads/CTF/itfest26/misc\\'
```

Suruh benerin gpt lalu kasih konteks challnya, dia kasih solusi buat nampilin CRC (simpelnya checksum buat file di dalam file) dari tiap file karena banyak yang sizenya sama dan buat ngecek apakah ada file-file yang duplikat.

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/misc]
└─$ 7z l -slt dist-memorize.7z | awk '
/^Path = / {
    p=substr($0,8)
}
/^CRC = / && p ~ /\.jpg$/ {
    print substr($0,7), p
}' > jpg-crc.txt
```

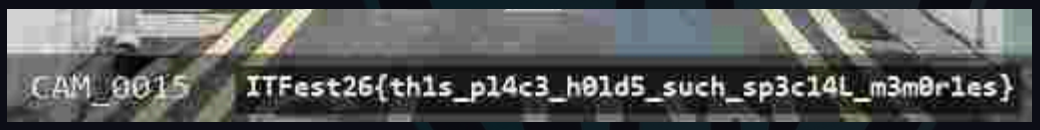
Lalu hitung CRC mana yang paling jarang muncul. Dan ditemukan ada 1 CRC dan kalo kita grep hasilnya adalah file memorize/photos/55/73/42.jpg

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/misc]
└─$ cut -d' ' -f1 jpg-crc.txt \
  | sort \
  | uniq -c \
  | sort -n \
  | head -30
      1 BA8059DB
  40315 A359901C
  40364 8C9AF29A
  40393 C3277BA5
  40485 58D5D311
  40578 947ED0E1
  40609 BBC7B40B
  40647 D06EE5E7
  40649 3DB1354C
  40664 CC87A2C1
  40688 01A44C35
  40691 7341D7A0
  40696 6387B6F8
  40735 6CCE2886
  40771 5B7EED10
  40854 B595F1F1
  40860 3CED6342

┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/misc]
└─$ grep '^BA8059DB ' jpg-crc.txt
BA8059DB memorize/photos/55/73/42.jpg
```

Lalu extract file itu aja dan buka jpg nya

```
┌──(byte㉿byuda)-[/mnt/c/Users/ACER/Downloads/CTF/itfest26/misc]
└─$ 7z x dist-memorize.7z \
  -so 'memorize/photos/55/73/42.jpg' \
  > suspect.jpg
```



FLAG: ITFest26{th1s_pl4c3_h0ld5_such_sp3c14L_m3m0r1es}

AI Proof :

https://chatgpt.com/share/6a76ee91-7f40-83ec-81e5-a0173055c818

The Murder of Alfred Cuthbert (107 Points)

POLICE REPORT:

Victim: Alfred Cuthbert (25y)
Crime Scene: Room 512, Crimson Red Apartment
Dead Body Found: Thursday, 23 July 2026 | 02:53 PM
Estimated Time of Death: Wednesday, 22 July 2026 | 04:21 PM

CRIMSON RED APARTMENT SECURITY NOTICE:

Only the lobby camera was operational. Cameras installed on residential floors, including the 5th floor, were out of service due to an unresolved maintenance issue.

INVESTIGATION NOTES:

Someone hired an assassin to commit the murder.
Find both the assassin and the person who hired them.

FLAG FORMAT: ITFest26{AssassinName_PersonWhoHired}

Example:

Assassin: Caroline W
Person who Hired: Brandon C
Flag: ITFest26{Caroline_Brandon}

Author: nuby

Sebenarnya ini nyoba nyoba semua path investigations sih hehe. nc ke service yang mencurigakan adalah si Oliver karena cuma dia yang berhubungan dengan kamar korban pada saat itu.

```
CASE FILE #02

Name: Oliver Scott
Age: 35
Occupation: Delivery driver in Roswatch Express

Background:
- Knows every street and building
- No criminal record
- Can visit any building without suspicion

Officer's Notes:
"Witnesses reported seeing him entering the victim's apartment one day before the body has discovered. But the security and residents claimed that almost ev
eryday he visited the apartment to deliver some packages for residents, including the victim."
```

Dari chat log juga aneh, seperti ada kode rahasia.

```
--- BeeTalk ---

--- Mr. Custo's Roomchat ---
[19/07/2026 05:40 PM]

Mr. Custo: "I need my package delivered ASAP."

Oliver Scott: "Again? May I know your package ID?"

Mr. Custo: "B00512AC. It's a Cherry Bomb toys for my CRAZY friend."

Oliver Scott: "Got it. Please wait until July 24th at the latest. The package will be delivered."

Mr. Custo: "Thanks."


--- Unknown's Roomchat ---
[22/07/2026 04:10 PM]

Unknown: "Hi. I need my package with the ID B10485GA to be delivered before 5 PM. It supposed to be here today at 2."

Oliver Scott: "Your package is on the way. Sorry for your inconvenience."

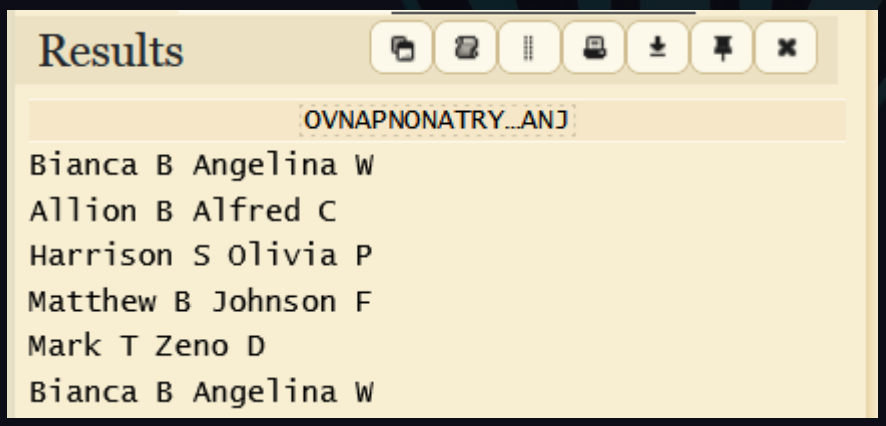
Unknown: "Okay."
```

Investigasi nih orang coba satu - satu pathnya dan dapat di alur 3 -> 1 -> 1 -> E2415 -> 1. Dan kita dapat history dari deliverynya si Oliver.

```
You received delivery_history.pdf

-------------------------------------------------
| Name       | Receiver      | Status           |
-------------------------------------------------
| Ovnapn O   | Natryvan J    | Delivered        |
| Nyyvba O   | Nyserq P      | Delivered        |
| Uneevfba F | Byvivn C      | Ready to Deliver |
| Znggurj O  | Wbuafba S     | Ready to Deliver |
| Znex G     | Mrab Q        | Waiting Payment  |
-------------------------------------------------
```

Pake https://www.dcode.fr/cipher-identifier dapet ternyata ini di ROT13. Decrypt we got siapa person who hired nya.



FLAG: ITFest26{Oliver_Allion}

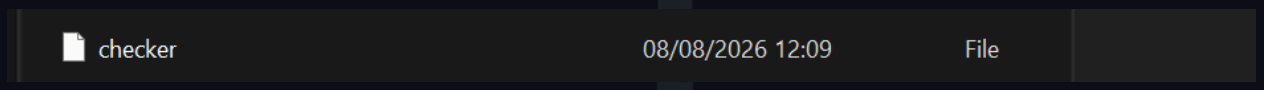
Reverse Engineering

rekcehcglaf (166 Points)

```
well..... you know what it is

Author: Durrr
```

Jadi pada challenge ini kita diberikan sebuah binary file yaitu checker



Jika dirun file tersebut maka menghasilkan output tersebut.

```
AriqArdian : /mnt/c/U/ariq/Doc/IT/Rev feat/exe-testing → ./checker
glaf: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Error: failed to initialize validation module

Caused by:
    expected 1 imports, found 0
```

Maka disini saya langsung sj membuka ghidra dan di ghidra terdapat sebuah fungsi entry dan isi berikut.

```
1
2  void processEntry entry(undefined8 param_1,undefined8 param_2)
3
4  {
5    undefined1 auStack_8 [8];
6
7    __libc_start_main(FUN_00264990,param_2,&stack0x00000008,0,0,param_1,auStack_8);
8    do {
9                      /* WARNING: Do nothing block with infinite loop */
10   } while( true );
11 }
```

Disini saya langsung saja ke fungsi yang ada yaitu FUN_00264990 dan ketika di buka isi nya

terdapat banyak kode dan function.

Decompile: FUN_00264990 - (checker2)

```
1
2  /* WARNING: Globals starting with '_' overlap smaller symbols at the same address */
3
4  code * FUN_00264990(undefined8 param_1,undefined8 param_2)
5
6  {
7    uint uVar1;
8    long lVar2;
9    int *piVar3;
10   int iVar4;
11   int iVar5;
12   long lVar6;
13   char *pcVar7;
14   __sighandler_t p_Var8;
15   ulong uVar9;
16   pthread_t __th;
17   long lVar10;
18   undefined8 *puVar11;
19   int *piVar12;
20   int *piVar13;
21   ulong uVar14;
22   code *pcVar15;
23   code *pcVar16;
24   size_t *__stacksize;
25   long in_FS_OFFSET;
26   bool bVar17;
27   undefined1 local_e1;
28   size_t local_e0;
29   undefined8 *local_d8;
30   int local_cc;
31   undefined1 local_c8 [56];
32   ulong uStack_90;
33   ulong local_88;
34   ulong uStack_80;
35   ulong local_78;
36   ulong uStack_70;
37   ulong local_68;
38   ulong uStack_60;
39   ulong local_58;
40   ulong uStack_50;
41   ulong local_48;
42   undefined8 uStack_40;
43   _func_5327 *local_38;
44
```

Dan disini cukup lama process nya disini saya mencoba membuka semua isi function yang ada dalam kode ini :(. Tapi dari semua function yang di akses ada function yang decompile nya lamaaaaaaaaaaaaaaa bangeeeettttt jadi kemungkinan file tersebut menyimpan kode yang sangat panjang.

```
    uStack_40._0_4_ = 0x8000004;
    local_c8._0_8_ = FUN_0047ea70;
    sigaction(7,(sigaction *)local_c8,(sigaction *)0x0);
  }
  _DAT_00651bc0 = param_2;
  puVar11 = (undefined8 *)FUN_0025a97b(FUN_0025e0b9);
  if (puVar11 != (undefined8 *)0x0) {
    local_c8._0_8_ = &local_d8;
    local_c8._8_8_ = FUN_0026777c;
    local_d8 = puVar11;
    local_e0 = FUN_0047ede0(&DAT_00651c30,&DAT_0015aa2c,local_c8);
    if (local_e0 != 0) {
      FUN_00454020(&local_e0);
```

Yaitu FUN_0025e0b9 dan ketika berhasil di decompile dan yap hasilnya berikut.

```
3673    }
3674    goto LAB_00261301;
3675  code_r0x002613f8:
3676    FUN_0047f370(&DAT_0018db6d);
3677    FUN_0047f9e0();
3678    lVar16 = FUN_004819d0(&DAT_00651080);
3679    if (lVar16 != 0) {
3680      puVar37 = (undefined8 *)FUN_004dd1dd(lVar16);
3681      ptVar21 = local_f30;
3682      goto LAB_0026164b;
3683    }
3684    ptVar19 = (timespec *)(ulong)((int)ptVar19 + 1);
3685    goto LAB_0026137d;
3686  LAB_002634c9:
3687    pcVar47 = &DAT_00189dd7;
3688    ppuVar52 = (undefined **)&DAT_00641420;
3689    ptStack_f50 = (timespec *)0x38;
3690  LAB_00262af9:
3691                      /* WARNING: Subroutine does not return */
3692    FUN_0026be20(pcVar47,ptStack_f50,ppuVar52);
3693  }
3694
```

Crazy 3000 lines 💀, Maka disini saya lempar AI untuk analisis dan hasilnya adalah.

Program membaca dari stdin (read(0, ptVar19, ...)) ke dalam buffer yang dapat membesar, baris demi baris dengan memisahkan data berdasarkan '\n'. Hal ini menunjukkan bahwa program kemungkinan sedang membaca sebuah skrip atau program.

- Program mengatur opsi konfigurasi Cranelift "enable_verifier" dan "opt_level" — ini menunjukkan bahwa engine Wasmtime sedang dikonfigurasi secara programatis.
- Setelah itu, program mengompilasi modul (FUN_0054cd5c, FUN_00537b4b kemungkinan merupakan Module::new), membuat instance dari modul tersebut, kemudian mencari export atau fungsi berdasarkan namanya ("instance", lalu melakukan pemanggilan terhadapnya).
- Terdapat string yang ditulis langsung di dalam binary seperti "cannot create a subtype of a final supertype" serta berbagai pesan panic dari Cranelift/Wasmtime. Semuanya merupakan string internal library bukan logika challenge yang sebenarnya.

Berdasarkan hal tersebut, dugaan saya: binary ini menerima bytecode WASM atau teks WAT sebagai input melalui stdin, kemudian menjalankannya menggunakan interpreter Wasmtime yang tertanam di dalam binary logika challenge yang sebenarnya kemungkinan berada pada host function yang diekspos kepada modul WASM, atau pada cara program memproses output yang dihasilkan.

Oke selanjut nya disini saya jalankan : strings binary | grep -i flag (siapa tau ada) dan ya hasil nya hanya strings acak tidak ada apa apa.

Lalu disini saya menjalankan strings -t -x checker | grep -i glaf karena di output program ketika di jalankan itu glaf bukan flag , dan hasil nya adalah berikut.

```
AriqArdian : /mnt/c/U/ariq/Doc/IT/Rev feat/exe-testing +1 !1 ?278 → strings -t x checker | grep -i glaf
  660ae assertion failed: !store.0.async_support()must use `grow_async` if an async resource limiter is usedtype mismatch with parameterstype mism
atch with resultsmust use `call_async` with async storesassertion failed: supertype.comes_from_same_engine(engine) (resultscannot create a subtype
 of a final supertypecannot fail without a supertypeinternal error: entered unreachable code: We should not have returned to the host with an Unwi
ndToWasm stateinternal error: entered unreachable code: We should not have gotten an error with no unwind stateassertion failed: ty.comes_from_sam
e_engine(engine)assertion failed: async_state.current_suspend.is_none()assertion failed: async_state.current_future_cx.is_none()cross-`Engine` ins
tantiation is not currently supportedassertion failed: core::ptr::eq(&**store.instance(context.instance).env_module(), module)glaf:
```

Lalu disini selanjutnya saya menjalankan command berikut xxd -s 0x660ae -l 200 checker dan strings -t x checker | grep -A 3 -B 3 '^\s*660ae'

```
AriqArdian : /mnt/c/U/ariq/Doc/IT/Rev feat/exe-testing +1 !1 ?278 → # dump raw bytes around that offset in the binary to see the string boundaries
xxd -s 0x660ae -l 200 checker
000660ae: 6173 7365 7274 696f 6e20 6661 696c 6564  assertion failed
000660be: 3a20 2173 746f 7265 2e30 2e61 7379 6e63  : !store.0.async
000660ce: 5f73 7570 706f 7274 2829 6d75 7374 2075  _support()must u
000660de: 7365 2060 6772 6f77 5f61 7379 6e63 6020  se `grow_async`
000660ee: 6966 2061 6e20 6173 796e 6320 7265 736f  if an async reso
000660fe: 7572 6365 206c 696d 6974 6572 2069 7320  urce limiter is
0006610e: 7573 6564 7479 7065 206d 6973 6d61 7463  usedtype mismatc
0006611e: 6820 7769 7468 2070 6172 616d 6574 6572  h with parameter
0006612e: 7374 7970 6520 6d69 736d 6174 6368 2077  stype mismatch w
0006613e: 6974 6820 7265 7375 6c74 736d 7573 7420  ith resultsmust
0006614e: 7573 6520 6063 616c 6c5f 6173 796e 6360  use `call_async`
0006615e: 2077 6974 6820 6173 796e 6320 7374 6f72   with async stor
0006616e: 6573 6173 7365 7274                      esassert
AriqArdian : /mnt/c/U/ariq/Doc/IT/Rev feat/exe-testing +1 !1 ?278 → strings -t x checker | grep -A 3 -B 3 '^\s*660ae'
  66029 "stack should always be aligned to
  6604e tableLog requires too much memory : unsupported
  6607e This mode cannot generate an uncompressed block
  660ae assertion failed: !store.0.async_support()must use `grow_async` if an async resource limiter is usedtype mismatch with parameterstype mism
atch with resultsmust use `call_async` with async storesassertion failed: supertype.comes_from_same_engine(engine) (resultscannot create a subtype
 of a final supertypecannot fail without a supertypeinternal error: entered unreachable code: We should not have returned to the host with an Unwi
ndToWasm stateinternal error: entered unreachable code: We should not have gotten an error with no unwind stateassertion failed: ty.comes_from_sam
e_engine(engine)assertion failed: async_state.current_suspend.is_none()assertion failed: async_state.current_future_cx.is_none()cross-`Engine` ins
tantiation is not currently supportedassertion failed: core::ptr::eq(&**store.instance(context.instance).env_module(), module)glaf:
  66429 host_xorshift32
  66456 memory
  6645f check
```

Lalu disini selanjutnya yaitu dump bytes untuk menemukan stringsnya

```
AriqArdian : /mnt/c/U/ariq/Doc/IT/Rev feat/exe-testing +1 !1 ?278 → objdump -s -j .rodata checker | grep -B2 -A2 -i 676c6166   # "glaf" in hex = 6
7 6c 61 66
AriqArdian : /mnt/c/U/ariq/Doc/IT/Rev feat/exe-testing +1 !1 ?278 → xxd -s 0x66429 -l 400 checker
00066429: 686f 7374 5f78 6f72 7368 6966 7433 3200  host_xorshift32.
00066439: 0003 0605 0001 0200 0305 0301 0011 0609  ................
00066449: 017f 0141 8080 c000 0b07 2003 066d 656d  ...A...... ..mem
00066459: 6f72 7902 0005 6368 6563 6b00 030b 6d61  ory...check...ma
00066469: 696e 7465 6e61 6e63 6500 040a c20b 059c  intenance.......
00066479: 0501 0e7f 2380 8080 8000 41c0 006b 2201  ....#.....A..k".
00066489: 2480 8080 8000 4100 2102 41a6 b285 8a03  $.....A.!.A.....
00066499: 2103 4100 2104 0340 2002 41d8 0020 0241  !.A.!..@ .A.. .A
000664a9: d800 4b1b 2105 2004 4101 7221 0620 0341  ..K.!. .A.r!. .A
000664b9: 1876 2107 0240 0340 2002 2208 2005 460d  .v!..@.@ ."". .F.
000664c9: 0141 0021 0920 0141 0041 c000 fc0b 0020  .A.!. .A.A.....
000664d9: 0841 016a 2102 4100 210a 0240 0340 2009  .A.j!.A.!..@.@ .
000664e9: 210b 200a 2209 41ff 0346 0d01 2009 2d00  !. ."..A..F.. .-.
000664f9: 8080 c080 0021 0c20 0941 026a 210a 2009  .....!. .A.j!. .
00066509: 2d00 8180 c080 0021 0902 4002 4002 4002  -......!..@.@.@.
00066519: 4002 4002 4002 4002 4002 4002 4002 4002  @.@.@.@.@.@.@.@.
00066529: 4002 4002 4020 0c41 7f6a 0e0a 0102 0000  @.@.@ .A.j......
00066539: 0000 0000 0305 000b 200c 4177 6a41 ff01  ........ .AwjA..
00066549: 7141 fa01 490d 0e20 0b41 0249 0d0e 200b  qA..I.. .A.I.. .
00066559: 417f 6a21 0920 0b41 114f 0d03 2001 2009  A.j!. .A.O.. . .
00066569: 4102 746a 2802 0021 0d20 0120 0b41 0274  A.tj(..!. . .A.t
00066579: 6a41 786a 220b 2802 0021 0e20 0c41 7d6a  jAxj".(..!. .A}j
00066589: 0e06 0506 0708 0a09 050b 200b 410f 4b0d  .......... .A.K.
00066599: 0d20 0120 0b41 0274 6a20 0020 0820 096c  . . .A.tj . . .l
000665a9: 2007 6a41 d800 706a 2d00 0036 0200 200b   .jA..pj-..6.. .
```

ini adalah bytecode WASM mentah itu sendiri, yang ditanam secara langsung di dalam .rodata. Perhatikan polanya:

- host_xorshift32, memory, check, maintenance ini merupakan nama import/export di dalam bagian name/export dari sebuah modul WASM.
- Byte setelahnya (0e7f 2380808080004...) merupakan opcode biner WASM: 0x23 = global.get, 0x41 = i32.const, 0x6b = i32.sub, 0x24 = global.set, 0x20 = local.get, 0x6a = i32.add, 0x46 = i32.eq, dan seterusnya. Ini merupakan prolog fungsi standar (global.get $stack_pointer; i32.const 0x40; i32.sub; ...) WASM hasil kompilasi yang

sepenuhnya normal, kemungkinan berasal dari C atau Rust dengan target wasm32-unknown-unknown.

Jadi, modul WASM guest yang sebenarnya yang hampir pasti berisi logika pengecekan flag ditanam sebagai blob biner di dalam executable checker. Program host, yaitu seluruh mekanisme Wasmtime/Cranelift yang sebelumnya, hanya memuat dan mengeksekusi modul tersebut.

Lalu disini selanjutnya saja menjalankan command berikut.

```
python3 - <<'PY'
data = open("checker", "rb").read()

magic = b"\x00asm\x01\x00\x00\x00"

offset = 0

while True:
   idx = data.find(magic, offset)
   if idx == -1:
      break

   print(f"WASM candidate: offset={idx:#x} ({idx})")
   print(data[idx:idx+32].hex(" "))

   offset = idx + 1
PY
```

```
python3 - <<'PY'
data = open("checker", "rb").read()
start = 0x66404

blob = data[start:]
open("candidate.wasm", "wb").write(blob)

print(f"[+] start : {start:#x}")
print(f"[+] size  : {len(blob)} bytes")
PY

wasm-validate candidate.wasm
wasm-objdump -h candidate.wasm
```

```
AriqArdian : /mnt/c/U/ariq/Doc/IT/Rev feat/exe-testing +1 !1 ?278 → python3 - <<'PY'
data = open("checker", "rb").read()

magic = b"\x00asm\x01\x00\x00\x00"

offset = 0

while True:
    idx = data.find(magic, offset)
    if idx == -1:
        break

    print(f"WASM candidate: offset={idx:#x} ({idx})")
    print(data[idx:idx+32].hex(" "))

    offset = idx + 1
PY
WASM candidate: offset=0x66404 (418820)
00 61 73 6d 01 00 00 00 01 13 04 60 01 7f 01 7f 60 01 7f 00 60 02 7f 7f 01 7f 60 00 00 02 17 01
AriqArdian : /mnt/c/U/ariq/Doc/IT/Rev feat/exe-testing +1 !1 ?278 →
AriqArdian : /mnt/c/U/ariq/Doc/IT/Rev feat/exe-testing +1 !1 ?278 → python3 - <<'PY'
data = open("checker", "rb").read()
start = 0x66404

blob = data[start:]
open("candidate.wasm", "wb").write(blob)

print(f"[+] start : {start:#x}")
print(f"[+] size  : {len(blob)} bytes")
PY

wasm-validate candidate.wasm
wasm-objdump -h candidate.wasm
[+] start : 0x66404
[+] size  : 5149060 bytes
0004bf8: error: invalid section code: 108

candidate.wasm: file format wasm 0x1
0004bf8: error: invalid section code: 108
0004c69: error: invalid section code: 105
0004cda: error: invalid section code: 84
0004d44: error: invalid section code: 110
0004db9: error: invalid section code: 114
0004dec: error: invalid section code: 49
0004e22: error: invalid section code: 120
0004e91: error: invalid section code: 113
0004f01: error: invalid section code: 110
0004f72: error: invalid section code: 111
0004fe2: error: invalid section code: 112
0005049: error: invalid section code: 116
```

Dan disini file wasm berhasil terbuat maka selanjut nya disini membuat file nya menjadi clean wasm dengan command berikut.

```
data = Path("checker").read_bytes()

start = 0x66404
size  = 0xA8A

wasm = data[start:start+size]

Path("clean.wasm").write_bytes(wasm)

print(f"[+] start : {start:#x}")
print(f"[+] end   : {start+size:#x}")
print(f"[+] size  : {len(wasm)}")
print("[+] wrote clean.wasm")
PY

wasm-validate clean.wasm
wasm-objdump -x clean.wasm
```

Maka masih rusak disini maka ubah kebentuk .wat dengan command berikut. wasm2wat clean.wasm -o clean.wat dan setelah dapat logic dari clean.wat. Dan sudah terlihat logic dari wat nya yang berisi berikut.

stage pertama checker dapat kita balik secara langsung.

Stage pertama melakukan: table[i] XOR (19*i - 91)

input[i] + ((17*i) XOR 35)

karena result XOR di-OR ke accumulator, dan accumulator harus menjadi 0 agar lolos.

Jadi: input[i] = (table[i] XOR ((19*i - 91) & 0xff)) - (((17*i) XOR 35) & 0xff) mod 256

Table-nya berada di: memory + 1049132 dan berasal dari data segment yang sudah ada di WAT.

Karena masih punya clean.wasm, buat script Node.js supaya WASM sendiri yang menginisialisasi memory, lalu dump table-nya. Maka berikut untuk solver script nya.

```
const fs = require("fs");

let rng = 0x12345678 >>> 0;

function xorshift32(x) {
    x >>>= 0;
    x ^= (x << 13) >>> 0;
    x ^= x >>> 17;
    x ^= (x << 5) >>> 0;
    return x >>> 0;
}

const imports = {
    env: {
        host_xorshift32: (x) => {
            return xorshift32(x) | 0;
        }
    }
};

(async () => {
    const wasm = fs.readFileSync("clean.wasm");
    const { instance } = await WebAssembly.instantiate(wasm, imports);

    const mem = new Uint8Array(instance.exports.memory.buffer);
```

```
    const TABLE1 = 1049132;

    let flag = [];

    for (let i = 0; i < 88; i++) {
        const t = mem[TABLE1 + i];

        const a = (19 * i - 91) & 0xff;
        const b = ((17 * i) ^ 35) & 0xff;

        const c = t ^ a;

        flag.push((c - b) & 0xff);
    }

    const buf = Buffer.from(flag);

    console.log("[+] raw hex:");
    console.log(buf.toString("hex"));

    console.log("\n[+] candidate:");
    console.log(buf.toString("latin1"));

    console.log("\n[+] bytes:");
    console.log([...buf]);
})();
```

```
AriqArdian : /mnt/c/U/ariq/Doc/IT/Rev feat/exe-testing +1 !1 ?278 → node dump.js
[+] raw hex:
49544665737432367b73307272795f6630725f666c3467634833636b33725f346734316e5f315f7072306d3173335f746831735f773372335f7468335f6c3473745f666c3467634833
636b33725f316e5f3174663373747d

[+] candidate:
ITFest26{s0rry_f0r_fl4gcH3ck3r_4g41n_1_pr0m1s3_th1s_w3r3_th3_l4st_fl4gcH3ck3r_1n_1tf3st}

[+] bytes:
[
   73,  84,  70, 101, 115, 116,  50,  54, 123, 115,  48, 114,
  114, 121,  95, 102,  48, 114,  95, 102, 108,  52, 103,  99,
   72,  51,  99, 107,  51, 114,  95,  52, 103,  52,  49, 110,
   95,  49,  95, 112, 114,  48, 109,  49, 115,  51,  95, 116,
  104,  49, 115,  95, 119,  51, 114,  51,  95, 116, 104,  51,
   95, 108,  52, 115, 116,  95, 102, 108,  52, 103,  99,  72,
   51,  99, 107,  51, 114,  95,  49, 110,  95,  49, 116, 102,
   51, 115, 116, 125
]
```

FLAG:

ITFest26{s0rry_f0r_fl4gcH3ck3r_4g41n_1_pr0m1s3_th1s_w3r3_th3_l4st_fl4gcH3ck3r_1n_1tf3st}

AI Transparency : https://claude.ai/share/03ae6a42-2fe9-4c72-b33b-e11420ef8740 https://chatgpt.com/share/6a76f879-d4dc-83ec-a71c-805d02931bca

Web Exploitation

antinrml (104 Points)

ANTINRML adalah pasar beat hipdut bawah tanah — tempat producer ketemu pembeli. Sekilas tampilannya kayak toko online biasa: ada login, katalog beat, keranjang, riwayat pembelian, dan profil. Kelihatan seperti aplikasi CRUD e-commerce standar.

Kamu bantuin seorang rapper upcoming dari Citayem yang lagi seret duit. Dia pengen banget si SECRET BEATS — ANTINRML Exclusive, tapi harganya jauh di atas saldo yang dia punya.

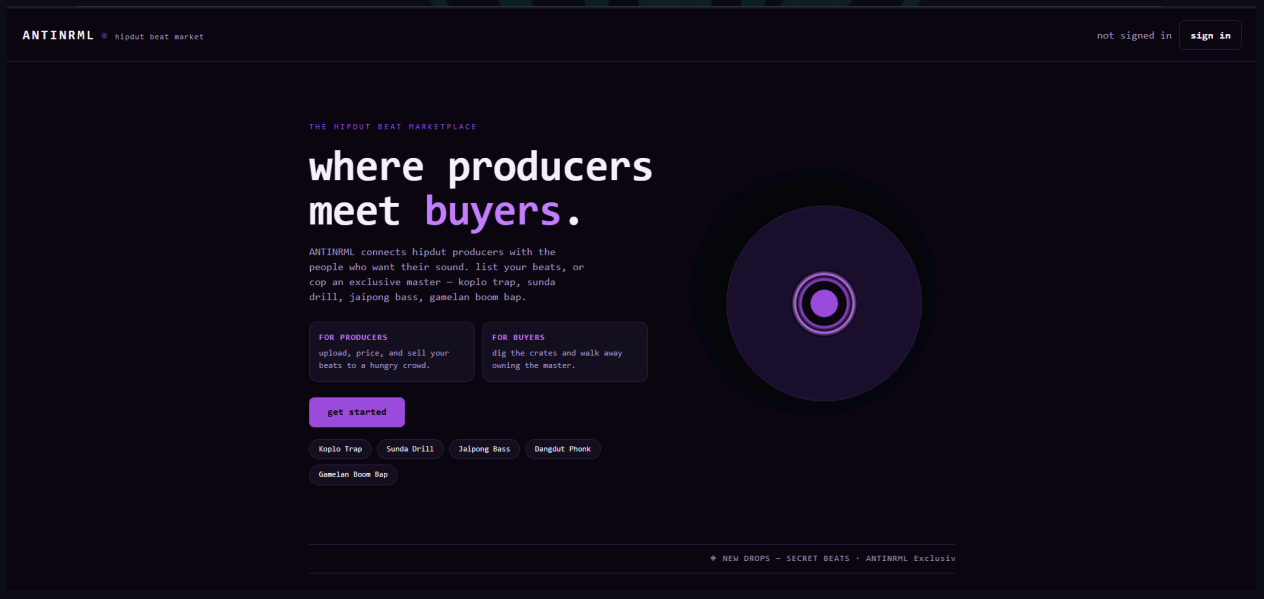
Di suatu tempat di dalam sistem ini, master dari beat eksklusif itu disimpan. Ambil.

Daftar akun sendiri untuk memulai

Author: UTARA

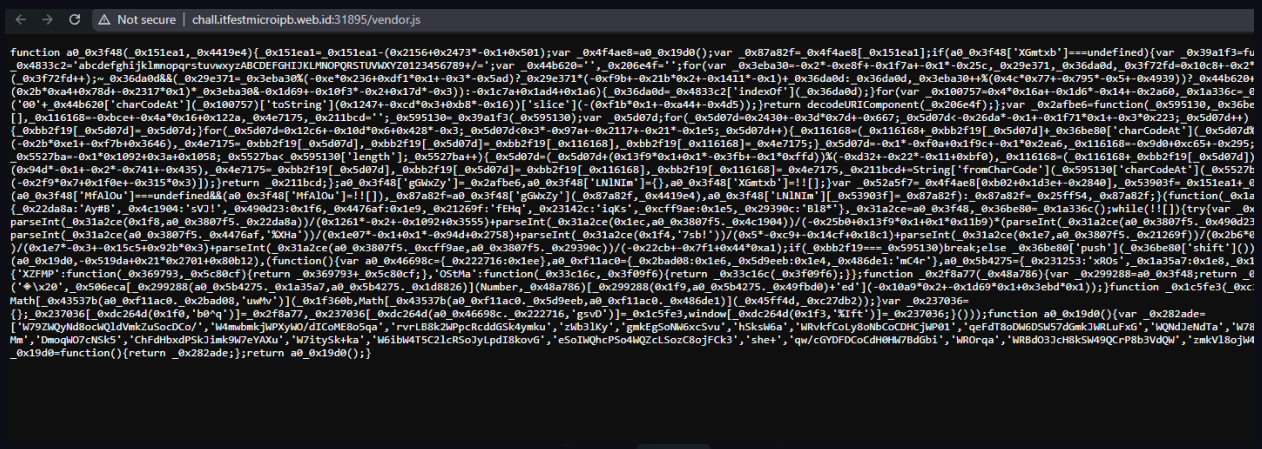
~Evidence history AI sudah ada di submission flag

Pada challenge web ini, diberikan sebuah web dengan tampilan seperti ini dan tidak diberikan attachment apa pun, which means chall ini blackbox dan menurut saya chall ini lebih mengarah ke reverse engineering.



Pada web tsb, saya dapat melakukan registrasi akun. Objective-nya adalah gimana caranya saya dapat checkout item “**SECRET BEATS – ANTINRML Exclusive” dengan credit yang diberikan sejak awal, yakni 50.00.**

Jika menelurusuri source yang ada di client-side, ada beberapa file JavaScript yang di-obfuscate



Pada awalnya saya mencoba untuk decode dengan JS deobfuscator biasa, seperti https://obf-io.deobfuscate.io/, namun masih tidak readable.

```
function a0_0x3f48(_0x151ea1, _0x4419e4) {
  _0x151ea1 = _0x151ea1 - 484;
  var _0x4f4ae8 = a0_0x19d0();
  var _0x87a82f = _0x4f4ae8[_0x151ea1];
  if (a0_0x3f48.XGmtxb === undefined) {
    var _0x39a1f3 = function (_0x332297) {
      var _0x44b620 = '';
      var _0x206e4f = '';
      var _0x3eba30 = 0;
      var _0x29e371;
      var _0x36da0d;
      for (var _0x3f72fd = 0; _0x36da0d =
_0x332297.charAt(_0x3f72fd++); ~_0x36da0d && (_0x29e371 = _0x3eba30 %
4 ? _0x29e371 * 64 + _0x36da0d : _0x36da0d, _0x3eba30++ % 4) ?
_0x44b620 += String.fromCharCode(255 & _0x29e371 >> (-2 * _0x3eba30 &
6)) : 0) {
        _0x36da0d =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/='.ind
exOf(_0x36da0d);
      }
      var _0x100757 = 0;
      for (var _0x1a336c = _0x44b620.length; _0x100757 < _0x1a336c;
_0x100757++) {
        _0x206e4f += '%' + ('00' +
_0x44b620.charCodeAt(_0x100757).toString(16)).slice(-2);
      }
      return decodeURIComponent(_0x206e4f);
    };
    var _0x2afbe6 = function (_0x595130, _0x36be80) {
      var _0xbb2f19 = [];
      var _0x116168 = 0;
      var _0x4e7175;
      var _0x211bcd = '';
      _0x595130 = _0x39a1f3(_0x595130);
      var _0x5d07d;
      for (_0x5d07d = 0; _0x5d07d < 256; _0x5d07d++) {
        _0xbb2f19[_0x5d07d] = _0x5d07d;
      }
      for (_0x5d07d = 0; _0x5d07d < 256; _0x5d07d++) {
        _0x116168 = (_0x116168 + _0xbb2f19[_0x5d07d] +
_0x36be80.charCodeAt(_0x5d07d % _0x36be80.length)) % 256;
        _0x4e7175 = _0xbb2f19[_0x5d07d];
        _0xbb2f19[_0x5d07d] = _0xbb2f19[_0x116168];
```

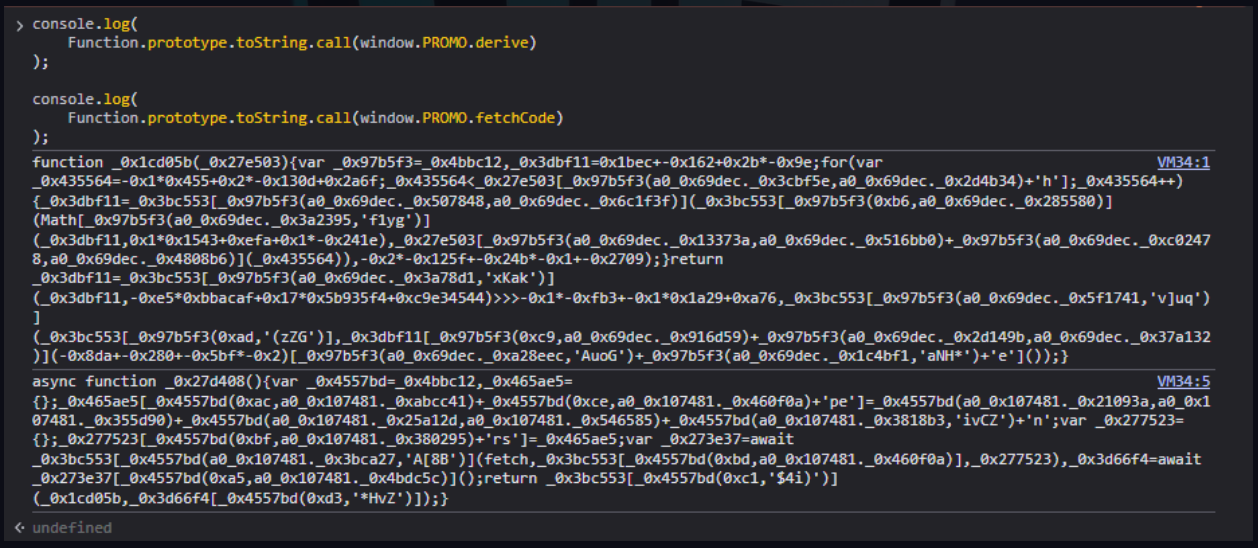
Maka dari itu saya mencoba konsul ke GPT untuk find the best way to reverse code tadi. Terlebih lagi ini bukan js obfuscation biasa karena melibatkan arr rotation dan sebagainya.

Singkat cerita, akhirnya saya paham logic dari obfuscationnya ini. Pokoknya kurang lebih kek gini:

Proses reverse engineering pada file JavaScript ini tidak dilakukan secara manual (string-by-string decoding) karena struktur obfuscation yg cukup kompleks. Obfuscator ini memanfaatkan bbrp lapisan proteksi:

- Array Rotation (IIFE Inisialisasi): Begitu file dimuat, sebuah IIFE (Immediately Invoked Function Expression) langsung mengeksekusi loop untuk memutar (rotate) urutan array string global hingga kondisi checksum numerik tertentu terpenuhi.
- Dynamic Decoder Routine (a0_0x...): String plaintext tidak disimpan dalam bentuk teks biasa maupun Base64 sederhana, melainkan diakses melalui fungsi dekoder dinamis yang menerima parameter offset heksadesimal dan decryption key (menggunakan algoritma berbasis RC4/XOR dengan caching internal).
- Dynamic Property Binding: Objek dan method ga di-assign secara eksplisit (kayak window.PROMO), melainkan menggunakan computed property names dari hasil dekoder dinamis (misalnya window[decoder(0xaf, "...") + 'y']).

Nah ini yang menurut saya butuh bantuan AI dalam mempelajari logic dari obfuscator-nya. Alhasil, dari beberapa



Script di bawah ini saya pakai untuk mendapatkan hasil decode dari file promo.js

analyze.js

```
const fs = require("fs");
const vm = require("vm");
const path = require("path");

const TARGET = process.argv[2];
```

```
if (!TARGET) {
    console.error("Usage: node analyze.js <file.js>");
    process.exit(1);
}

const source = fs.readFileSync(path.resolve(TARGET), "utf8");

// Fake Storage & DOM
const storage = new Map();
const localStorage = {
    getItem: (k) => storage.get(k) ?? null,
    setItem: (k, v) => storage.set(k, String(v)),
    removeItem: (k) => storage.delete(k),
    clear: () => storage.clear(),
    key: (i) => [...storage.keys()][i] ?? null,
    get length() { return storage.size; }
};

const fakeDocument = {
    cookie: "",
    createElement: (t) => ({ tagName: String(t).toUpperCase(),
style: {}, setAttribute(){}, getAttribute(){ return null; },
appendChild(){}, remove(){}, addEventListener(){},
removeEventListener(){} }),
    querySelector: () => null,
    querySelectorAll: () => [],
    getElementById: () => null,
    addEventListener: () => {},
    removeEventListener: () => {}
};

async function fakeFetch(url, options = {}) {
    console.log(`\n[FETCH INTERCEPTED] ${options.method || 'GET'}
-> ${url}`);
    if (options.body) console.log("[FETCH BODY]", options.body);

    return {
        ok: true, status: 200, statusText: "OK",
        headers: { get: (h) => h?.toLowerCase() ===
"content-type" ? "application/json" : null },
        async text() { return "TEST-SEED"; },
        async json() { return { seed: "TEST-SEED" }; },
        async arrayBuffer() { return new ArrayBuffer(0); }
    };
}

const windowTarget = {};
const window = new Proxy(windowTarget, {
    get: (t, p, r) => p === "window" ? r : Reflect.get(t, p, r),
```

```
    set: (t, p, v, r) => Reflect.set(t, p, v, r)
});

window.fetch = fakeFetch;
window.localStorage = localStorage;
window.document = fakeDocument;
window.console = console;

const sandbox = {
    window, document: fakeDocument, localStorage, fetch:
fakeFetch, console,
    setTimeout: () => 0, setInterval: () => 0, clearTimeout: ()
=> {}, clearInterval: () => {},
    alert: () => {}, confirm: () => false, prompt: () => null,
    location: { href: "http://sandbox.invalid/", hostname:
"sandbox.invalid", pathname: "/", search: "", hash: "" },
    navigator: { userAgent: "Node.js Obfuscation Analysis
Sandbox" }
};

sandbox.globalThis = sandbox;
sandbox.self = window;
window.window = window;
window.self = window;
window.globalThis = sandbox;

(async () => {
    const context = vm.createContext(sandbox);

    try {
        const script = new vm.Script(source, { filename: TARGET
});
        script.runInContext(context, { timeout: 5000 });
        console.log("[+] Script berhasil dimuat.");
    } catch (err) {
        console.error("\n[!] Script execution error:", err.stack
|| err);
        return;
    }

    // Hooking Decoder dengan Deduplikasi Log
    const seenDecodes = new Set();

    for (const key of Object.keys(context)) {
        if (typeof context[key] === "function" &&
(key.startsWith("a0_") || key.startsWith("_0x"))) {
            try {
                const testRes = context[key]();
                if (Array.isArray(testRes)) continue;
            } catch (e) {
```

```
            }

            const originalDecoder = context[key];
            context[key] = function (...args) {
                const result = originalDecoder.apply(this, args);

                const logKey =
`${key}:${args.join(",")}:${result}`;
                if (!seenDecodes.has(logKey)) {
                    seenDecodes.add(logKey);
                    console.log(`[DECODE] ${key}(${args.map(x =>
typeof x === 'string' ? JSON.stringify(x) : '0x' +
Number(x).toString(16)).join(", ")}) =>
${JSON.stringify(result)}`);
                }
                return result;
            };
            Object.assign(context[key], originalDecoder);
        }
    }

    // Eksekusi Fungsi PROMO
    const PROMO = windowTarget.PROMO;
    if (PROMO) {
        if (typeof PROMO.derive === "function") {
            console.log("\n[+] Running
PROMO.derive('TEST-SEED')...");
            try {
                const deriveRes = PROMO.derive("TEST-SEED");
                console.log("[+] Derive Result:", deriveRes);
            } catch (e) { console.error("[-] Derive Error:",
e.message); }
        }

        if (typeof PROMO.fetchCode === "function") {
            console.log("\n[+] Running PROMO.fetchCode()...");
            try {
                const fetchRes = await PROMO.fetchCode();

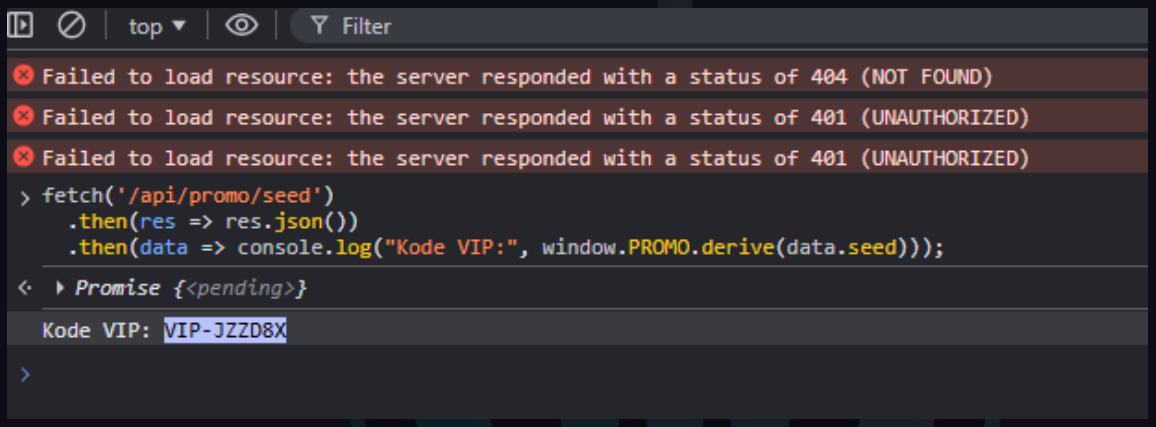
console.log("\n============================================");
                console.log("[+] FINAL RESULT
PROMO.fetchCode():", fetchRes);

console.log("============================================");
            } catch (e) { console.error("[-] FetchCode Error:",
e.message); }
        }
    }
})();
```

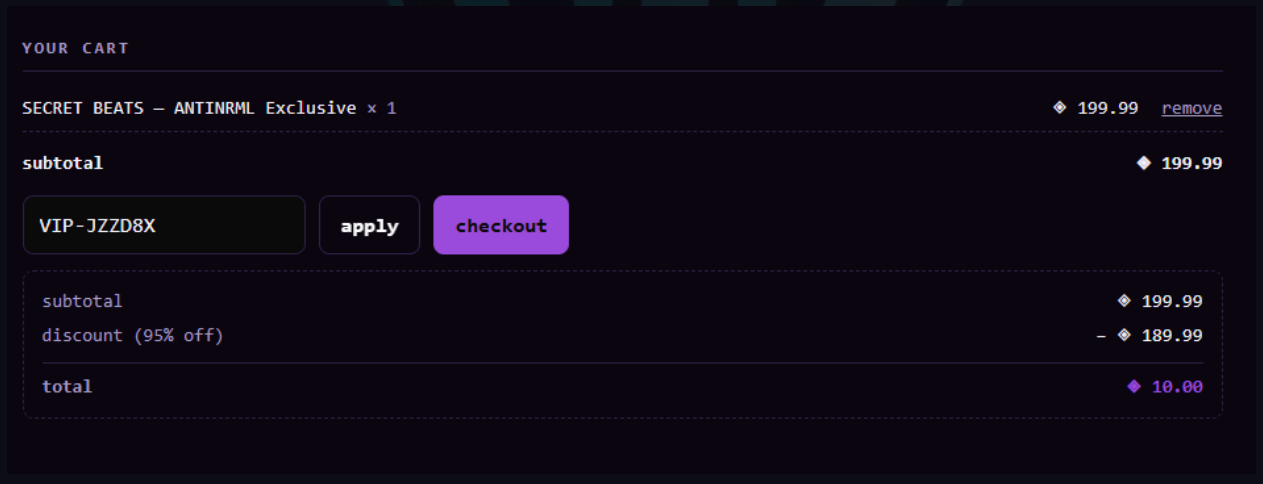
Hasilnya adalah saya mendapatkan endpoint api yang berisi PROMO KEY yang dibuat ketika sebuah akun login or regist.

```
[FETCH INTERCEPTED] GET -> /api/promo/seed
[DECODE] a0_0x5b04(0xa5, "DTIx") => "json"
[DECODE] a0_0x5b04(0xc1, "$4i)") => "Cwbyi"
[DECODE] a0_0x5b04(0xd3, "*HvZ") => "seed"
```

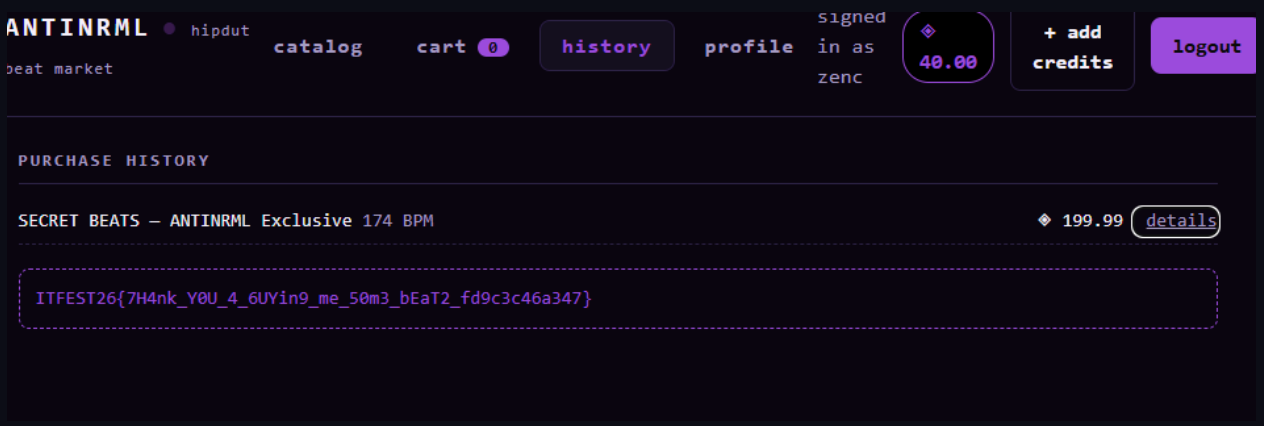
Berarti, sebenarnya tinggal tembak endpoint tersebut dan fetch PROMO KEY nya. karena saya malas buat solver, saya fetch saja melalui console di browser.



Lalu apply kode tersebut untuk memberli **SECRET BEATS — ANTINRML Exclusive.**



Flag-nya ada di sana



FLAG: ITFEST26{7H4nk_Y0U_4_6UYin9_me_50m3_bEaT2_fd9c3c46a347}