PiPi

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Tue Feb 18 2025

PiPi Audit Report

1 Executive Summary

1.1 Project Information

Description	Stable coin and meme coin
Type	DeFi
Auditors	MoveBit
Timeline	Thu Dec 12 2024 - Tue Feb 18 2025
Languages	Move
Platform	Movement
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Copariot-Labs/PicWe-core-contracts
Commits	a946f2d0d34c91de1340a86749a37168b07fe30c 2f573728cd2d13bf900eb42288bc1db91153046f 2dd806a96bdcf0fc73f22224eeb67028f97f8f1c d43cd84feae98fb3fc930f69d08fa99fe55601ad

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
MOV	move/Move.toml	b0cf420d4c9af9a0f2b26ba5eb3b3f0c7e7746eb
WEU	move/sources/weusd.move	b6bd0e593a1eeaa11c05be5c866a0eb3c41c09d2
PIP	move/sources/pipi.move	1bd2bb07b5559714eaa976a35ff5a9c9b7fffd10
POR	move/sources/price_oracle.move	7365b64c993698a7e1462198a71202b6a2ab5b1e
AIR	move/sources/airdrop.move	6eda8e993229d89d25bb2359320f4263b2069970
WOP	move/sources/weusd_operations.move	d2b517c84c8d32de050f64e1a27b53fe02c3bc52

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	8	6	2
Informational	2	1	1
Minor	2	2	0
Medium	1	1	0
Major	3	2	1
Critical	0	0	0

1.4 MoveBit Audit Breakdown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Formal Verification(Optional)

Perform formal verification for key functions with the Move Prover.

(4) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by PicWe to identify any potential issues and vulnerabilities in the source code of the PiPi smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 8 issues of varying severity, listed below.

ID	Title	Severity	Status
PIP-1	The Calculation of `actual_mint_amount` may Overflow	Major	Acknowledged
PIP-2	Dependence on Centralized Storage Service for `IMAGE_URL`	Informational	Fixed
POR-1	Single-step Ownership Transfer Can be Dangerous	Medium	Fixed
WEU-1	Missing Access Control for `mint()`	Major	Fixed
WEU-2	Change Public `deposit()` to Internal `deposit()`	Minor	Fixed
WOP-1	The Calculation Results in an Overflow	Major	Fixed
WOP-2	A Bad Actor may Use a Small Amount to Redeem and Avoid Paying the Fee	Minor	Fixed
WOP-3	Unused Parameter in the Function `get_token_price()`	Informational	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the PiPi Smart Contract :
Admin

- `start_airdrop` : Allows the admin to initiate an airdrop by setting the total airdrop amount, token metadata address, and total supply of `PiPi` . This function checks that the caller is the authorized admin (`signer::address_of(admin) == @picwe`) and ensures that an airdrop is not already active. Updates the `AirdropState` and emits an `AirdropStarted` event.
- `set_fee_ratio` : Allows the admin (`@picwe`) to update the `fee_ratio` for the system.
- `set_move_ratio` : Allows the admin (`@picwe`) to update the MOVE token ratio used for minting `WeUSD` .
- `set_min_mint_amount` : Allows the admin (@picwe) to set the minimum WeUSD mint amount.
- `set_fee_recipient` : Allows the admin (@picwe) to update the recipient address for collected fees.
- `initiate_admin_transfer` : Allows the current admin to transfer administrative privileges to a new address(pending_admin).
- `confirm_admin_transfer` : After pending_admin confirms, admin is changed successfully.
- `update_price` : Allows the admin to update the price details in the `OracleStore` .
- `set_price_valid_interval` : Allows the admin to update the price validity interval in the `OracleStore` .
- `init_module` :admin initializes `PiPi` meme token, including `name` , `symbol` , `decimals` , `icon` , `URL` .

User

- `mintWeUSD` : Enables a user to mint WeUSD tokens by depositing MOVE tokens and stablecoins.
- `redeemWeUSD` : Allows a user to redeem their `WeUSD` tokens for MOVE and stablecoins based on the current token price and reserves.
- `mint_weusd_for` : Allows a user to mint tokens for other addresses.

- `claim_airdrop`: Enables a user to claim their share of an active airdrop. Validates that the airdrop is active, the user has not already claimed, and that the user is eligible (has minted `PiPi` before the snapshot time). Calculates the user's token share based on their `PiPi` holdings and total airdrop supply, transfers the tokens, and emits an `AirdropClaimed` event.

4 Findings

PIP-1 The Calculation of `actual_mint_amount` may Overflow

Severity: Major

Status: Acknowledged

Code Location:

move/sources/pipi.move#149

Descriptions:

In the `mint()` function, the protocol calculates the `actual_mint_amount` as follows:

`let actual_mint_amount = (base_amount * mint_amount) / INITIAL_MINT_AMOUNT`.

The calculation `base_amount * mint_amount` may cause an overflow.

Suggestion:

It is recommended to change the type to `u256` to prevent this issue.

PIP-2 Dependence on Centralized Storage Service for `IMAGE_URL`

Severity: Informational

Status: Fixed

Code Location:

move/sources/pipi.move#54

Descriptions:

The current implementation uses a centralized storage service for the `IMAGE_URL` constant:

```
fun init_module(admin: &signer) {
    let constructor_ref = &object::create_named_object(admin, ASSET_SYMBOL);
    primary_fungible_store::create_primary_store_enabled_fungible_asset(
      constructor_ref,
      option::none(),
      utf8(b"PIPI Coin"), /* name */
      utf8(ASSET_SYMBOL), /* symbol */
      6, /* decimals */
      utf8(b"http://example.com/favicon.ico"), /* icon */
      utf8(b"http://example.com") /* project */
    );
```

Suggestion:

It is redommended to use decentralized storage solutions such as IPFS or Arweave.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

POR-1 Single-step Ownership Transfer Can be Dangerous

Severity: Medium

Status: Fixed

Code Location:

move/sources/price_oracle.move#137-144

Descriptions:

Single-step ownership transfer means that if a wrong address was passed when transferring ownership or admin rights it can mean that role is lost forever. If the admin permissions are given to the wrong address within this function, it will cause irreparable damage to the contract. The `set_admin()` function directly transfers the admin role to a new address, which poses the above-mentioned risks.

```
// Admin functions
public entry fun set_admin(
    sender: &signer,
    new_admin: address
) acquires OracleStore {
    let store = borrow_global_mut<OracleStore>(@picwe);
    assert!(signer::address_of(sender) == store.admin, ENOT_AUTHORIZED);
    store.admin = new_admin;
}
```

Suggestion:

It is recommended to use a two-step ownership transfer process.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

WEU-1 Missing Access Control for `mint()`

Severity: Major

Status: Fixed

Code Location:

move/sources/weusd.move#122-130

Descriptions:

In the `mint()` function, the protocol calls `authorized_borrow_refs()` to retrieve `managed_fungible_asset`, and then mints assets to the user.

```
    // :!:>mint
    /// Mint as the owner of metadata object.
    public entry fun mint(admin: &signer, to: address, amount: u64) acquires
ManagedFungibleAsset {
        let asset = get_metadata();
        let managed_fungible_asset = authorized_borrow_refs(admin, asset);
        let to_wallet = primary_fungible_store::ensure_primary_store_exists(to, asset);
        let fa = fungible_asset::mint(&managed_fungible_asset.mint_ref, amount);
        fungible_asset::deposit_with_ref(
            &managed_fungible_asset.transfer_ref, to_wallet, fa
        );
    }
```

However, the protocol does not verify the owner when calling `authorized_borrow_refs()`, allowing any user to invoke this function to mint assets.

The `transfer()` and `burn()` functions have the same issue.

```
    inline fun authorized_borrow_refs(
        owner: &signer, asset: Object<Metadata>
    ): &ManagedFungibleAsset acquires ManagedFungibleAsset {
        // assert!(
        //     object::is_owner(asset, signer::address_of(owner)),
        //     error::permission_denied(ENOT_OWNER)
        // );
```

```
    borrow_global<ManagedFungibleAsset>(object::object_address(&asset))
}
```

Suggestion:

It is recommended to add access control.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

WEU-2 Change Public `deposit()` to Internal `deposit()`

Severity: Minor

Status: Fixed

Code Location:

move/sources/weusd.move#104-109

Descriptions:

In the `transfer()` function, the protocol calls the `deposit()` function to credit the `to_wallet` with FA, passing the `transfer_ref` parameter. However, `deposit()` is a public function. It is recommended to change it to an internal function.

```
public fun deposit<T: key>(
    store: Object<T>, fa: FungibleAsset, transfer_ref: &TransferRef
) acquires State {
    assert_not_paused();
    fungible_asset::deposit_with_ref(transfer_ref, store, fa);
}
```

Suggestion:

It is recommended to modify the `deposit()` function to be an internal function.

Resolution:

This issue has been fixed. The client removed the function.

WOP-1 The Calculation Results in an Overflow

Severity: Major

Status: Fixed

Code Location:

move/sources/weusd_operations.move#151

Descriptions:

The `convert_value_to_amount()` function is used to convert a value to an amount based on the token price and decimals. In the function, if `price_expo_negative` is true, the return value is calculated as follows:

```
(scaled_value * ((math128::pow(10, (price_expo as u128))) as u64)) / price_value
```

The decimals of `scaled_value` and `price_expo` are both 8. If `scaled_value` is large, the multiplication may cause an overflow. This could lead to incorrect calculations or potential errors in the system. The same issue exists in the following code. The same issues exist in the flowing code:

```
let redemption_value = if (total_pool_value >= total_weusd_supply) {
        // If fully collateralized, redeem at 1:1
        weusd_amount
    } else {
        // If undercollateralized, redeem proportionally
        (weusd_amount * total_pool_value) / total_weusd_supply
    };
```

airdrop.move:

```
// Calculate user's share
    let user_token_amount = (pipi_amount * airdrop_state.total_airdrop_amount) /
airdrop_state.total_pipi_supply;
```

launchpad.move:

```
let total_cost = (price * amount) / 100000000; // price is in 1e8
    let total_cost2_round = (price * amount * 100) / 100000000;
```

Suggestion:

It is recommended to convert the value to `u128` type.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

WOP-2 A Bad Actor may Use a Small Amount to Redeem and Avoid Paying the Fee

Severity: Minor

Status: Fixed

Code Location:

move/sources/weusd_operations.move#340

Descriptions:

In the `redeemWeUSD()` function, the code calculates `stablecoin_fee` and `move_fee` for the `fee_recipient`.

```
let stablecoin_fee = (stablecoin_withdraw_amount * mint_state.fee_ratio) / 10000;
    let actual_stablecoin_amount = stablecoin_withdraw_amount - stablecoin_fee;
    let fee_recipient = mint_state.fee_recipient;
```

```
let move_fee = (move_withdraw_amount * mint_state.fee_ratio) / 10000;
    let actual_move_amount = move_withdraw_amount - move_fee;
```

However, the protocol does not verify whether the fee is greater than 0. A bad actor may use a small amount to redeem and avoid paying the fee.

Suggestion:

It is recommended to ensure that the fee is greater than 0 if `stablecoin_withdraw_amount` and `move_withdraw_amount` are greater than 0.

Resolution:

This can be fixed by checking `stablecoin_fee`

```
assert!(stablecoin_fee > 0 && move_fee > 0, E_INSUFFICIENT_FEE);
```

WOP-3 Unused Parameter in the Function `get_token_price()`

Severity: Informational

Status: Acknowledged

Code Location:

move/sources/weusd_operations.move#123

Descriptions:

The `get_token_price()` function is used to retrieve the token price from the selected oracle. However, the parameter `_pyth_price_update` in the function is unused and does not serve any purpose.

```
    // Get token price from selected oracle
    fun get_token_price(_user: &signer, _price_feed_id: vector<u8>, _pyth_price_update:
vector<vector<u8>>): (u64, u8, bool) {
        // let pyth_price = pyth_oracle::get_price(user, price_feed_id, pyth_price_update);
        // (
        //     pyth_oracle::get_price_value(&pyth_price),
        //     pyth_oracle::get_price_expo(&pyth_price),
        //     pyth_oracle::is_price_expo_negative(&pyth_price)
        // )
        let price = price_oracle::get_price();
        (
            price_oracle::get_price_value(&price),
            price_oracle::get_price_expo(&price),
            price_oracle::is_price_expo_negative(&price)
        )
    }
```

Suggestion:

It is recommended to remove this parameter.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.