Dipcoin

Audit Report

MOVEBIT

contact@bitslab.xyz

https://twitter.com/movebit_

Wed Aug 13 2025

Dipcoin Audit Report

1 Executive Summary

1.1 Project Information

Description	A smart contract is designed to support Spot AMM (Automated Market Maker) swaps and perpetual contracts, enabling users to trade directly on the blockchain with high throughput and low latency.
Type	DEX
Auditors	MoveBit
Timeline	Fri Aug 01 2025 - Wed Aug 13 2025
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/dipcoinlab/dipcoin-amm
Commits	a65b036f4db46c4a4d736c65500ae068ce9ada6b e831b242e265b25ad0b9368a323cb4523ebd854d

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MAT	sources/math.move	5e78a120e82dc6aa52ac1e7fddac590e197e49fc
ROU	sources/router.move	5c485459e5bab453e5f70c30baf756a1917a21cd
EVE	sources/event.move	668b09b1f945634b52e065eadec029078f9d7de7
COR	sources/core.move	b60dd6911442293ad525bf759c585d18bf292fd3
COM	sources/comparator.move	2481b46e06c921ac9d0e8caaef0a3a43790bb4d6
CON1	sources/control.move	51b349993d0b38dec579e5f108a0acc7909b4133
MAN	sources/manage.move	40fa66be6742cdf4684c9f25328a0aeb78a070a2

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	3	3	0
Informational	1	1	0
Minor	1	1	0
Medium	1	1	0
Major	0	0	0
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 Dipcoin Team to identify any potential issues and vulnerabilities in the source code of the Dipcoin 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 3 issues of varying severity, listed below.

ID	Title	Severity	Status
CON-1	Repeat Settings will Cause Unnecessary Gas Consumption	Informational	Fixed
COR-1	Incorrect `min_add_liquidity_lp_amount` Check Does Not Match Settings	Medium	Fixed
MAN-1	Incorrect Verification Resulted in Temporary Asset Freezing	Minor	Fixed

3 Participant Process

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

I. Core Module Description

1. **control.move**
    - Admin permission control entry
    - Pool creation and management
    - Protocol fee control
2. **router.move**
    - Add/remove liquidity
    - User trading entry
    - Token exchange routing
3. **core.move**
    - Core trading logic implementation
    - Liquidity calculation
    - Fee handling
4. **manage.move**
    - Global configuration management
    - Pool status management
    - Permission verification
5. **math.move**
    - AMM mathematical calculations

- Slippage protection
- Liquidity calculation

6. **comparator.move**
    - Token sorting comparison
    - LP token naming

II. Key Admin Permission Methods

1. create Trading Pool (create_pool_by_admin_cap)

```
sequenceDiagram
    participant Admin
    participant Control
    participant Manage
    participant Comparator

    Admin->>Control: create_pool_by_admin_cap<X,Y>(AdminCap)
    Control->>Manage: check_version
    Control->>Manage: check_is_normal
    Control->>Comparator: is_order<X,Y>
    alt X < Y
        Control->>Manage: create_pool<X,Y>
    else Y < X
        Control->>Manage: create_pool<Y,X>
    end
    Manage-->>Control: (pool_address, lp_name, fee_rate)
    Control->>Control: emit(CreatedPoolEvent)
```

2. Emergency Pause/Resume (pause/resume)

```
sequenceDiagram
    participant Admin
    participant Control
    participant Manage
```

```
    Admin->>Control: pause(AdminCap)
    Control->>Manage: check_version
    Control->>Manage: has_paused
    Control->>Manage: pause
    Note over Manage: Set has_paused = true

    Admin->>Control: resume(AdminCap)
    Control->>Manage: check_version
    Control->>Manage: has_paused
    Control->>Manage: resume
    Note over Manage: Set has_paused = false
```

3. Fee Rate Control (control_protocol_fee_switch/modify_fee_rate)

```
sequenceDiagram
    participant Admin
    participant Control
    participant Manage

    Admin->>Control: control_protocol_fee_switch(AdminCap, is_open)
    Control->>Manage: check_version
    Control->>Manage: check_is_normal
    Control->>Manage: control_protocol_fee_switch
    Note over Manage: Update protocol fee status

    Admin->>Control: modify_fee_rate<X,Y>(AdminCap, new_rate)
    Control->>Manage: check_version
    Control->>Manage: check_is_normal
    Control->>Manage: modify_fee_rate
    Note over Manage: Update pool fee rate
```

III. Trading Related Methods

1. Add Liquidity (add_liquidity)

```
sequenceDiagram
    participant User
```

```
    participant Router
    participant Core
    participant Math
    participant Manage

    User->>Router: add_liquidity<X,Y>
    Router->>Manage: check_version
    Router->>Manage: check_is_normal
    Router->>Core: add_liquidity
    Core->>Math: calc_optimal_coin_values
    Note over Math: Calculate optimal addition ratio
    Core->>Math: sqrt(optimal_x * optimal_y)
    Note over Math: Calculate initial liquidity for first addition
    Core->>Manage: increase_supply
    Note over Core: Mint LP tokens
    Core-->>Router: (LP tokens, actual_x, actual_y)
    Router->>Router: emit(AddLiquidityEvent)
```

2. Remove Liquidity (remove_liquidity)

```
sequenceDiagram
    participant User
    participant Router
    participant Core
    participant Math
    participant Manage

    User->>Router: remove_liquidity<X,Y>
    Router->>Manage: check_version
    Router->>Manage: check_is_normal
    Router->>Core: remove_liquidity
    Core->>Math: mul_div calculate return ratio
    Core->>Manage: decrease_supply
    Note over Core: Burn LP tokens
    Core-->>Router: (coin_x, coin_y)
    Router->>Router: emit(RemoveLiquidityEvent)
```

3. Token Swap (swap)

I. Exact Input Swaps

1. swap_exact_x_to_y - Swap Fixed Amount of X for Y

```
sequenceDiagram
    participant User
    participant Router
    participant Core
    participant Math
    participant Manage

    User->>Router: swap_exact_x_to_y<X,Y>(coin_x_in, value_y_out_min)
    Router->>Manage: check_version
    Router->>Manage: check_is_normal
    Router->>Core: swap_exact_x_to_y

    Core->>Math: get_amount_out
    Note over Math: amount_out = (reserve_out * amount_in * (10000 - fee_rate))<br/>/
(reserve_in * 10000 + amount_in * (10000 - fee_rate))

    Core->>Core: assert!(value_y_out >= value_y_out_min)
    Note over Core: Check slippage protection

    alt is_open_protocol_fee
        Core->>Math: get_fee_to_team
        Note over Math: coin_x_fee = value_x_in * fee_rate / (FEE_SCALE * 5)
        Core->>Manage: join fee balance
    end

    Core->>Manage: join remaining X to pool
    Core->>Manage: take Y from pool

    Core->>Math: assert_lp_value_is_increased
    Note over Math: Verify (new_x * new_y) >= (old_x * old_y)

    Core-->>Router: (value_x_in, value_y_out, fee_x, fee_y, coin_y_out)
    Router->>Router: emit(SwapEvent)
```

2. swap_exact_y_to_x - Swap Fixed Amount of Y for X

```
sequenceDiagram
    participant User
    participant Router
    participant Core
    participant Math
    participant Manage

    User->>Router: swap_exact_y_to_x<X,Y>(coin_y_in, value_x_out_min)
    Router->>Manage: check_version
    Router->>Manage: check_is_normal
    Router->>Core: swap_exact_y_to_x

    Core->>Math: get_amount_out
    Note over Math: amount_out = (reserve_x * amount_in * (10000 - fee_rate))<br/>/
(reserve_y * 10000 + amount_in * (10000 - fee_rate))

    Core->>Core: assert!(value_x_out >= value_x_out_min)
    Note over Core: Check slippage protection

    alt is_open_protocol_fee
        Core->>Math: get_fee_to_team
        Note over Math: coin_y_fee = value_y_in * fee_rate / (FEE_SCALE * 5)
        Core->>Manage: join fee balance
    end

    Core->>Manage: join remaining Y to pool
    Core->>Manage: take X from pool

    Core->>Math: assert_lp_value_is_increased
    Note over Math: Verify (new_x * new_y) >= (old_x * old_y)

    Core-->>Router: (value_y_in, value_x_out, fee_x, fee_y, coin_x_out)
    Router->>Router: emit(SwapEvent)
```

II. Exact Output Swaps

1. swap_x_to_exact_y - Use X to Swap for Fixed Amount of Y

```
sequenceDiagram
    participant User
    participant Router
    participant Core
    participant Math
    participant Manage

    User->>Router: swap_x_to_exact_y<X,Y>(coin_x_in, value_y_out)
    Router->>Manage: check_version
    Router->>Manage: check_is_normal
    Router->>Core: swap_x_to_exact_y

    Core->>Math: get_amount_in
    Note over Math: amount_in = (reserve_in * amount_out * 10000)<br/>/ ((reserve_out - amount_out) * (10000 - fee_rate))

    Core->>Core: assert!(value_x_in <= value_x_max)
    Note over Core: Check maximum input limit

    alt value_x_in < value_x_max
        Core->>Core: create excess_x_coin
        Note over Core: Return excess X tokens
    else
        Core->>Core: create zero coin
    end

    alt is_open_protocol_fee
        Core->>Math: get_fee_to_team
        Note over Math: coin_x_fee = value_x_in * fee_rate / (FEE_SCALE * 5)
        Core->>Manage: join fee balance
    end

    Core->>Manage: join remaining X to pool
    Core->>Manage: take Y from pool

    Core->>Math: assert_lp_value_is_increased
    Note over Math: Verify (new_x * new_y) >= (old_x * old_y)
```

```
    Core-->>Router: (value_x_in, value_y_out, fee_x, fee_y, coin_y_out, excess_x_coin)
    Router->>Router: emit(SwapEvent)
```

2. swap_y_to_exact_x - Use Y to Swap for Fixed Amount of X

```
sequenceDiagram
    participant User
    participant Router
    participant Core
    participant Math
    participant Manage

    User->>Router: swap_y_to_exact_x<X,Y>(coin_y_in, value_x_out)
    Router->>Manage: check_version
    Router->>Manage: check_is_normal
    Router->>Core: swap_y_to_exact_x

    Core->>Math: get_amount_in
    Note over Math: amount_in = (reserve_in * amount_out * 10000)<br/>/ ((reserve_out - amount_out) * (10000 - fee_rate))

    Core->>Core: assert!(value_y_in <= value_y_max)
    Note over Core: Check maximum input limit

    alt value_y_in < value_y_max
        Core->>Core: create excess_y_coin
        Note over Core: Return excess Y tokens
    else
        Core->>Core: create zero coin
    end

    alt is_open_protocol_fee
        Core->>Math: get_fee_to_team
        Note over Math: coin_y_fee = value_y_in * fee_rate / (FEE_SCALE * 5)
        Core->>Manage: join fee balance
    end

    Core->>Manage: join remaining Y to pool
```

```
Core->>Manage: take X from pool

Core->>Math: assert_lp_value_is_increased
Note over Math: Verify (new_x * new_y) >= (old_x * old_y)

Core-->>Router: (value_y_in, value_x_out, fee_x, fee_y, coin_x_out, excess_y_coin)
Router->>Router: emit(SwapEvent)
```

Swap Additional Notes: Main differences between the two methods

1. **Exact Input Swaps**
    - Fixed input amount
    - Need to specify minimum output amount (slippage protection)
    - Calculation uses get_amount_out
    - No need to return excess tokens
2. **Exact Output Swaps**
    - Fixed output amount
    - Need to specify maximum input amount
    - Calculation uses get_amount_in
    - Need to handle and return excess input tokens
3. Trading Checkpoints
4. **Mathematical Calculation Security**

```
// get_amount_out calculation
amount_out = (reserve_out * amount_in * (10000 - fee_rate))
        / (reserve_in * 10000 + amount_in * (10000 - fee_rate))

// get_amount_in calculation
amount_in = (reserve_in * amount_out * 10000)
        / ((reserve_out - amount_out) * (10000 - fee_rate))
```

5. **Slippage Protection**
    - Exact Input: Check output is not below minimum
    - Exact Output: Check input does not exceed maximum
6. **K Value Verification**

```
assert!(
    (old_reserve_x * old_reserve_y) <= (new_reserve_x * new_reserve_y),
    EIncorrectSwap
)
```

7. **Fee Handling**
    - Protocol fee calculation (20% of fee)
    - LP fee remains in pool
    - Return excess tokens

IV. Key Security Point Checks

1. **Permission Control**
    - AdminCap permission verification
    - Emergency pause mechanism
    - Version control check
2. **Mathematical Calculation Security**
    - Overflow protection
    - Slippage limits
    - Minimum liquidity restrictions
3. **Rate Limitations**

- Maximum rate limit (1%)
- Fixed protocol fee ratio (20%)
- Rate precision control

4. **Transaction Protection**
   - K value verification
   - Minimum output protection
   - Zero value checks
5. **Asset Security**
   - Balance checks
   - Return excess tokens
   - LP token minting/burning

V. Constant Configuration

```
// Fee related
const DEFAULT_FEE_RATE: u64 = 30;    // Default trading fee rate 0.3%
const MAX_FEE_RATE: u64 = 100;       // Maximum fee rate 1%
const FEE_SCALE: u64 = 10000;        // Fee precision

// Liquidity related
const MINIMAL_LIQUIDITY: u64 = 1000; // Minimum liquidity requirement

// Value limits
const U64_MAX: u64 = 18446744073709551615; // u64 maximum value
```

VI. Event Monitoring

1. **CreatedPoolEvent**

- pool_address
- lp_name
- creator
- fee_rate

2. **AddLiquidityEvent**
    - lp_name
    - coin_x_val
    - coin_y_val
    - lp_val
3. **RemoveLiquidityEvent**
    - lp_name
    - coin_x_val
    - coin_y_val
    - lp_val
4. **SwapEvent**
    - lp_name
    - value_x_in/out
    - value_y_in/out
    - fee_x/y
5. **WithdrewEvent**
    - lp_name
    - fee_coin_x
    - fee_coin_y

4 Findings

CON-1 Repeat Settings will Cause Unnecessary Gas Consumption

Severity: Informational

Status: Fixed

Code Location:

sources/control.move#143-178

Descriptions:

In these functions: `control_protocol_fee_switch` , `modify_fee_rate` , `modify_min_add_liquidity_lp_amount`

```
manage::control_protocol_fee_switch(global, is_open_protocol_fee);
```

```
manage::modify_fee_rate<X, Y>(pool, new_fee_rate);
```

```
manage::modify_min_add_liquidity_lp_amount<X, Y>(pool, min_add_liquidity_lp_amount);
```

If the new value is the same as the current value, it isn't very sensible to update it.

Suggestion:

Add a check before updating the value; it can reduce unnecessary gas consumption.

Resolution:

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

COR-1 Incorrect `min_add_liquidity_lp_amount` Check Does Not Match Settings

Severity: Medium

Status: Fixed

Code Location:

sources/core.move#60-63

Descriptions:

In the `add_liquidity` function, we have a check to ensure that `provided_liq` is greater than or equal to `pool.min_add_liquidity_lp_amount`, but if `lp_supply == 0`, the value of `provided_liq` is reduced by `MINIMAL_LIQUIDITY`. In this case, the logic may need to check `initial_liq >= manage::get_min_add_liquidity_lp_amount(pool)` instead of the value of subtracting `MINIMAL_LIQUIDITY` from `provided_liq`.

Suggestion:

Change the check to:

```
    let provided_liq = if (lp_supply == 0) {
+    assert!(initial_liq >= manage::get_min_add_liquidity_lp_amount(pool),
ELessThanMinAddLiquidityLpAmount);
        ...

    } else {
        let x_liq = (lp_supply as u128) * (optimal_coin_x as u128) / (coin_x_reserve as u128);
            let y_liq = (lp_supply as u128) * (optimal_coin_y as u128) / (coin_y_reserve as u128);
            if (x_liq < y_liq) {
                assert!(x_liq < (U64_MAX as u128), EU64Overflow);
+                assert!(x_liq >= manage::get_min_add_liquidity_lp_amount(pool),
ELessThanMinAddLiquidityLpAmount);
                (x_liq as u64)
            } else {
                assert!(y_liq < (U64_MAX as u128), EU64Overflow);
+                assert!(y_liq >= manage::get_min_add_liquidity_lp_amount(pool),
```

```
ELessThanMinAddLiquidityLpAmount);
            (y_liq as u64)
        }
    }
        ...

-       assert!(provided_liq >= manage::get_min_add_liquidity_lp_amount(pool),
ELessThanMinLPAmount);
```

Resolution:

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

MAN-1 Incorrect Verification Resulted in Temporary Asset Freezing

Severity: Minor

Status: Fixed

Code Location:

sources/manage.move#106-115

Descriptions:

In the `withdraw` function

```
assert!(bal_x_fee > 0 && bal_y_fee > 0, EZeroAmount);
```

If the `bal_x_fee` or `bal_y_fee` is 0, it will cause the transaction to fail, which may not be the expected behavior. Maybe we just ensure one of them is greater than 0.

Suggestion:

Change the check to:

```
assert!(bal_x_fee > 0 || bal_y_fee > 0, EZeroAmount);
```

Resolution:

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

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.