MOVEBIT
Securing the Move Ecosystem

Transit Finance
Audit Report

https://twitter.com/movebit_

contact@movebit.xyz

Transit Finance Audit Report

1 Executive Summary

1.1 Project Information

Type	Swap Aggregator
Auditors	MoveBit
Timeline	2022-11-28 to 2022-12-7
Languages	Move
Methods	Architecture Review, Unit Testing, Formal Verification, Manual Review
Source Code	Repository: https://github.com/Transit-Finance/transit-aptos-core-v1 Last Reviewed Commit: 93330d22441e583b7d349480120af1d6fa28ae2b

1.2 Issue Statistic

Item	Count	Fixed	Pending
Total	7	6	1
Minor	1	1	
Medium	6	5	1
Major			
Critical			

1.3 Issue Level

- **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.

1.4 Issue Status

- **Fixed:** The issue has been resolved.
- **Pending:** The issue has been acknowledged by the code owner, but has not yet been resolved. The code owner may take action to fix it in the future.

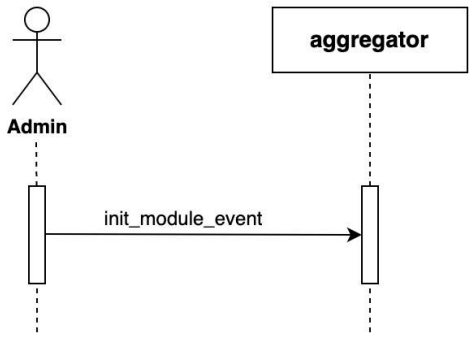
2 Summary of Findings

Transit Finance is a cross-chain swap platform that integrates DEXs, aggregate transactions, and one-stop cross-chain. Without certification, users can complete decentralized transactions in real-time and instantly swap assets across networks supported by Transit Finance. Our team mainly focused on reviewing the Code Security and normative, then conducted code running tests and business logic security tests on the test net, Our team has been in close contact with the developing team for the past few days. As a result, our team found a total of 7 issues and plans to address them remaining.

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

(1) Admin

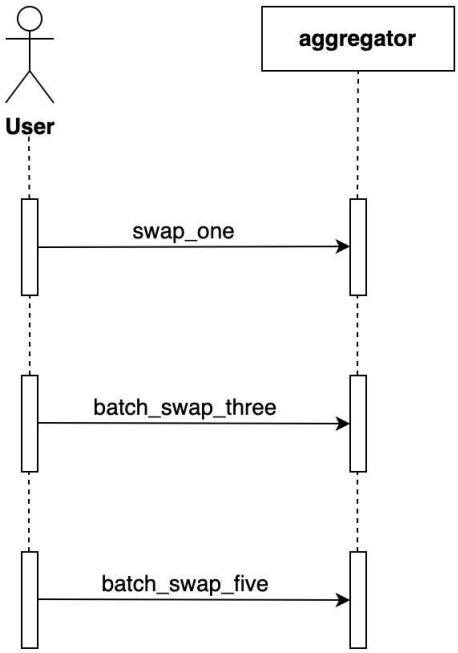
- Initialize the event of the module.



(2) User

- Swap by one type of token pair.
- Swap by three types of token pairs.

- Swap by five types of token pairs.



3 MoveBit Audit BreakDown

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

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- 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

4 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", and that 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

Refer to **Appendix 1** for code scope.

(3) Formal Verification

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, and they should actively cooperate (which may include 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 time.

5 Findings

5.1 The definition of the variable should be placed where it is used

Severity: Minor
Status: Fixed
Descriptions: The function `get_intermediate_out_from_dexs` in the `aggregator` module defines `let amount_in_value = coin::value(&x_in);` at the beginning, but the `amount_in_value` is used only under the `dex_type == AUX_DEX` condition.

Code Location: tp_contract/aggregator/aptos/aptosAggregatorV1/aggregator.move, line 82.

aggregator.move

```
fun get_intermediate_out_from_dexs<X, Y, E>(
        sender: &signer,
        dex_type: u64,
        pool_type: u64,
        is_x_to_y: bool,
        x_in: coin::Coin<X>
    ): coin::Coin<Y> {
        let amount_in_value = coin::value(&x_in);
        let (x_out_opt, y_out) = if (dex_type == LIQUIDSWAP_DEX) {
            use liquidswap::router_v2;
            let y_out =router_v2::swap_exact_coin_for_coin<X, Y, E>(x_in, 0);
            (option::none(), y_out)
        } else if (dex_type == APTOSWAP_DEX) {
            use Aptoswap::pool;
            if (is_x_to_y) {
                let y_out = pool::swap_x_to_y_direct<X, Y>(x_in);
                (option::none(), y_out)
            }
            else {
                let y_out = pool::swap_y_to_x_direct<Y, X>(x_in);
                (option::none(), y_out)
            }
            ...
    }
```

Suggestion: Move the `let amount_in_value = coin::value(&x_in);` into the `dex_type == AUX_DEX` condition.

5.2 Function visibility issue

Severity: Medium

Status: Fixed

Descriptions: The function of the `emit_swap_event` in the `aggregator` module is to emit the swap event. The visibility is public, anyone can call it and it will cause event resource pollution and affect the event record.

Code Location: tp_contract/aggregator/aptos/aptosAggregatorV1/aggregator.move, line 47.

aggregator.move

```
public fun emit_swap_event<X, Y>(
    trader:address,
    channel:u64,
    input_amount:u64,
    output_amount: u64
) acquires EventStore {
    let event_store = borrow_global_mut<EventStore>(@transit_aggregator);
    event::emit_event<SwapEvent>(
        &mut event_store.swap_events,
        SwapEvent {
            trader,
            channel,
            x_type_info: type_of<coin::Coin<X>>(),
            y_type_info: type_of<coin::Coin<Y>>(),
            input_amount,
            output_amount,
        },
    );
}
```

Suggestion: Change function visibility to private.

5.3 Excessive reliance on external dex contract calls and there is no way to control or suspend external dex

Severity: Medium

Status: Fixed

Descriptions: The implementation of the swap function in the contract has the problem of over-reliance on external contracts (multiple external dex contracts are called in the function `get_intermediate_out_from_dexs`), and there are no security measures in this contract. If the external swap contract has a security problem or becomes a malicious contract, the transaction will not be suspended, resulting in the loss of user benefits.

Code Location: tp_contract/aggregator/aptos/aptosAggregatorV1/aggregator.move, line 75.

aggregator.move

```
fun get_intermediate_out_from_dexs<X, Y, E>(
        sender: &signer,
        dex_type: u64,
        pool_type: u64,
        is_x_to_y: bool,
        x_in: coin::Coin<X>
    ): coin::Coin<Y> {
        let amount_in_value = coin::value(&x_in);
        let (x_out_opt, y_out) = if (dex_type == LIQUIDSWAP_DEX) {
            use liquidswap::router_v2;
            let y_out =router_v2::swap_exact_coin_for_coin<X, Y, E>(x_in, 0);
            (option::none(), y_out)
        } else if (dex_type == APTOSWAP_DEX) {
            use Aptoswap::pool;
            if (is_x_to_y) {
                let y_out = pool::swap_x_to_y_direct<X, Y>(x_in);
                (option::none(), y_out)
            }
            else {
                let y_out = pool::swap_y_to_x_direct<Y, X>(x_in);
                (option::none(), y_out)
            }
        } else if (dex_type == PANCAKE_DEX){
            use pancake::router;
            let y_out = router::swap_exact_x_to_y_direct_external<X, Y>(x_in);
            (option::none(), y_out)
        } else if (dex_type == ANIMESWAP_DEX) {
            use SwapDeployer::AnimeSwapPoolV1;
            let y_out =AnimeSwapPoolV1::swap_coins_for_coins<X, Y>(x_in);
            (option::none(), y_out)
        } else if (dex_type == AUX_DEX) {
            if (pool_type == AUX_TYPE_AMM){
                use aux::amm;
                let y_out = coin::zero<Y>();
                amm::swap_exact_coin_for_coin_mut(
                    @transit_aggregator,
                    &mut x_in,
                    &mut y_out,
                    amount_in_value,
                    0,
                    false,
                    0,
                    0
                );

                (option::some(x_in),y_out)
            } else if (pool_type == AUX_TYPE_MARKET){
                use aux::clob_market;
                let y_out = coin::zero<Y>();
                if (is_x_to_y){
                    clob_market::place_market_order_mut<X, Y>(
                        @transit_aggregator,
```

```
                        &mut x_in,
                        &mut y_out,
                        false,
                        102,// IMMEDIATE_OR_CANCEL in aux::router,
                        0,
                        amount_in_value,
                        0
                    );
                } else {
                    abort ERR_UNSUPPORTED
                };
                (option::some(x_in),y_out)
            } else {
                abort ERR_UNKNOWN_POOL_TYPE
            }
        }  else if (dex_type == CETUS_DEX) {
            use cetus_amm::amm_router;
            let y_out = amm_router::swap<X, Y>(@transit_aggregator, x_in);
            (option::none(),y_out)
        } else {
            abort ERR_UNKNOWN_DEX
        };
        check_and_deposit_opt(sender, x_out_opt);
        y_out
    }
```

Suggestion: Add safety measures to suspend or remove a DEX from the `transit_aggregator` contract. If any DEX has risks, it can be suspended or removed from the `transit_aggregator` contract.

5.4 Code readability needs to be improved in the get_intermediate_out_from_dexs function

Severity: Medium

Status: Fixed

Descriptions: The code readability of the `get_intermediate_out_from_dexs` function in the `aggregator` module is poor, the dex swap logic of six different branches can be split into six functions to improve the readability of the code.

Code Location: tp_contract/aggregator/aptos/aptosAggregatorV1/aggregator.move, line 75-151.

aggregator.move

```
fun get_intermediate_out_from_dexs<X, Y, E>(
        sender: &signer,
        dex_type: u64,
        pool_type: u64,
        is_x_to_y: bool,
        x_in: coin::Coin<X>
    ): coin::Coin<Y> {
        let amount_in_value = coin::value(&x_in);
        let (x_out_opt, y_out) = if (dex_type == LIQUIDSWAP_DEX) {
            use liquidswap::router_v2;
            let y_out =router_v2::swap_exact_coin_for_coin<X, Y, E>(x_in, 0);
            (option::none(), y_out)
        } else if (dex_type == APTOSWAP_DEX) {
            use Aptoswap::pool;
            if (is_x_to_y) {
                let y_out = pool::swap_x_to_y_direct<X, Y>(x_in);
                (option::none(), y_out)
            }
            else {
                let y_out = pool::swap_y_to_x_direct<Y, X>(x_in);
                (option::none(), y_out)
            }
        } else if (dex_type == PANCAKE_DEX){
            use pancake::router;
            let y_out = router::swap_exact_x_to_y_direct_external<X, Y>(x_in);
            (option::none(), y_out)
        } else if (dex_type == ANIMESWAP_DEX) {
            use SwapDeployer::AnimeSwapPoolV1;
            let y_out =AnimeSwapPoolV1::swap_coins_for_coins<X, Y>(x_in);
            (option::none(), y_out)
        } else if (dex_type == AUX_DEX) {
            if (pool_type == AUX_TYPE_AMM){
                use aux::amm;
                let y_out = coin::zero<Y>();
                amm::swap_exact_coin_for_coin_mut(
                    @transit_aggregator,
                    &mut x_in,
                    &mut y_out,
                    amount_in_value,
                    0,
                    false,
                    0,
                    0
                );

                (option::some(x_in),y_out)
            } else if (pool_type == AUX_TYPE_MARKET){
                use aux::clob_market;
                let y_out = coin::zero<Y>();
                if (is_x_to_y){
                    clob_market::place_market_order_mut<X, Y>(
                        @transit_aggregator,
```

```
                        &mut x_in,
                        &mut y_out,
                        false,
                        102,// IMMEDIATE_OR_CANCEL in aux::router,
                        0,
                        amount_in_value,
                        0
                    );
                } else {
                    abort ERR_UNSUPPORTED
                };
                (option::some(x_in),y_out)
            } else {
                abort ERR_UNKNOWN_POOL_TYPE
            }
        }  else if (dex_type == CETUS_DEX) {
            use cetus_amm::amm_router;
            let y_out = amm_router::swap<X, Y>(@transit_aggregator, x_in);
            (option::none(),y_out)
        } else {
            abort ERR_UNKNOWN_DEX
        };
        check_and_deposit_opt(sender, x_out_opt);
        y_out
    }
```

Suggestion: The logic at different branches of the function is split into a single function, and the branch in the function is changed to a call to the function corresponding to the branch.

Modify the code as follows.

aggregator.move

```
fun liquid_swap<X, Y, E>(x_in: coin::Coin<X>): coin::Coin<Y> {
    ...
    }

fun apto_swap<X, Y>(x_in: coin::Coin<X>): coin::Coin<Y> {
    ...
    }

    ...

fun get_intermediate_out_from_dexs<X, Y, E>(
        sender: &signer,
        dex_type: u64,
        pool_type: u64,
        is_x_to_y: bool,
        x_in: coin::Coin<X>
    ): coin::Coin<Y> {
        let (x_out_opt, y_out) = if (dex_type == LIQUIDSWAP_DEX) {
            let y_out = liquid_swap<X, Y, E>(x_in);
            (option::none(), y_out)
        } else if (dex_type == APTOSWAP_DEX) {
            let y_out = apto_swap<X, Y>(is_x_to_y, x_in);
            (option::none(), y_out)
        ...
    }
```

5.5 Common code should be encapsulated as a function to be called

Severity: Medium

Status: Fixed

Descriptions: The `batch_swap_five` and `batch_swap_three` functions in the `aggregator` module have roughly the same code except for the number of type parameters.

Code Location: tp_contract/aggregator/aptos/aptosAggregatorV1/aggregator.move, line 244, 325.

aggregator.move

```
public entry fun batch_swap_three<
        X, OutCoin,
        Y0, Z0, E01, E02, E03,
        Y1, Z1, E11, E12, E13,
        Y2, Z2, E21, E22, E23,
    >(
        sender: &signer,
        channel: u64,
        batch_num: u64,
        num_steps_vec: vector<u64>,
        first_dex_type_vec: vector<u64>,
        first_pool_type_vec: vector<u64>,
        first_is_x_to_y_vec: vector<bool>,
        second_dex_type_vec: vector<u64>,
        second_pool_type_vec: vector<u64>,
        second_is_x_to_y_vec: vector<bool>,
        third_dex_type_vec: vector<u64>,
        third_pool_type_vec: vector<u64>,
        third_is_x_to_y_vec: vector<bool>,
        x_in_vec: vector<u64>,
        m_min_out: u64,
    ) acquires EventStore {
        ...
    }
public entry fun batch_swap_five<
        X, OutCoin,
        Y0, Z0, E01, E02, E03,
        Y1, Z1, E11, E12, E13,
        Y2, Z2, E21, E22, E23,
        Y3, Z3, E31, E32, E33,
        Y4, Z4, E41, E42, E43,
    >(
        sender: &signer,
        channel: u64,
        batch_num: u64,
        num_steps_vec: vector<u64>,
        first_dex_type_vec: vector<u64>,
        first_pool_type_vec: vector<u64>,
        first_is_x_to_y_vec: vector<bool>,
        second_dex_type_vec: vector<u64>,
        second_pool_type_vec: vector<u64>,
        second_is_x_to_y_vec: vector<bool>,
        third_dex_type_vec: vector<u64>,
        third_pool_type_vec: vector<u64>,
        third_is_x_to_y_vec: vector<bool>,
        x_in_vec: vector<u64>,
        m_min_out: u64,
    ) acquires EventStore {
        ...
    }
```

Suggestion: Modify the common code as a function, and then call it in two functions. It will improve code reuse and maintainability.

5.6 The business logic structure is too complex

Severity: Medium
Status: Pending
Descriptions: The `batch_swap_five` function has twenty-seven type parameters and fifteen function parameters. It is inconvenient for code maintenance, user command line execution, and function call, the gas consumption also will be higher.

Code Location: tp_contract/aggregator/aptos/aptosAggregatorV1/aggregator.move, line 325.

aggregator.move

```
public entry fun batch_swap_five<
        X, OutCoin,
        Y0, Z0, E01, E02, E03,
        Y1, Z1, E11, E12, E13,
        Y2, Z2, E21, E22, E23,
        Y3, Z3, E31, E32, E33,
        Y4, Z4, E41, E42, E43,
    >(
        sender: &signer,
        channel: u64,
        batch_num: u64,
        num_steps_vec: vector<u64>,
        first_dex_type_vec: vector<u64>,
        first_pool_type_vec: vector<u64>,
        first_is_x_to_y_vec: vector<bool>,
        second_dex_type_vec: vector<u64>,
        second_pool_type_vec: vector<u64>,
        second_is_x_to_y_vec: vector<bool>,
        third_dex_type_vec: vector<u64>,
        third_pool_type_vec: vector<u64>,
        third_is_x_to_y_vec: vector<bool>,
        x_in_vec: vector<u64>,
        m_min_out: u64,
    ) acquires EventStore {
        ...
    }
```

Suggestion: We suggest that sort out business logic again, and optimizing transaction steps.

5.7 Deploy smart contract without multi-sig

Severity: Medium
Status: Fixed
Descriptions: The smart contract is not deployed under a multi-sig account. Operations

performed with multiple signatures will provide greater security. Even if the loss of a single private key will not allow an attacker to gain access to the contract. Multiple trusted parties must approve the update at the same time, otherwise, it will not work.

Suggestion: Use a multi-sig account for the smart contract when deploying.

Appendix 1 - Files in Scope

The following are the SHA1 hashes of the last reviewed files:

Files	SHA-1 Hash
transit-aptos-core-v1/contract/sources/aggregator.move	1a784615274a5cde4c9484fad4f487606a60a9df
transit-aptos-core-v1/contract/scripts/swap.move	7ca602229116e2f73f92e33fb322ad469388a26d
transit-aptos-core-v1/contract/exchange/cetus/sources/amm_router.move	ce868c3fa23b0f321b2d3d391c3aa00b656b6a8d
transit-aptos-core-v1/contract/exchange/animeswap/sources/swap.move	49224f7c4ac275088e8cea68690fe85bda270cc7
transit-aptos-core-v1/contract/exchange/aux/sources/clob_market.move	234275ca3cc4257f2ef3422b7515df7b034bf32d
ransit-aptos-core-v1/contract/exchange/aux/sources/amm.move	bce5e785cae3cdd28aaa4ee486f5727086c03a95
transit-aptos-core-v1/contract/exchange/aptoswap/sources/pool.move	2ae35067b4e01efd965caa1509fabf65450058ea
transit-aptos-core-v1/contract/Move.toml	f73ad368d98e8636d9a25840177915fac49b8b6e
transit-aptos-core-v1/contract/exchange/cetus/Move.toml	4937ffda6e75021bae2c00e502400590514109b2
transit-aptos-core-v1/contract/exchange/animeswap/Move.toml	b783720c75a85ff4877d57e85fad27bfd23571ad
transit-aptos-core-v1/contract/exchange/aux/Move.toml	7673e123d70d442a567f39a66f236e3e1db0ad8c
transit-aptos-core-v1/contract/exchange/aptoswap/Move.toml	014a96dbe89a9c7ef8863178e99ce630f276857b

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.

;

MOVEBIT

Securing the Move Ecosystem

https://twitter.com/movebit_

contact@movebit.xyz