MOVEBIT
Securing the Move Ecosystem

Wisp Swap Smart Contract **Audit Report**

https://twitter.com/movebit_

contact@movebit.xyz

Wisp Swap Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	An AMM DEX on Sui.
Type	DEX
Auditors	MoveBit
Timeline	Apr 26th, 2023 – May 9th, 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/wispswap-labs/wispswap-smart-contracts
Commits	dcce940eef30d0825ad67d693211b70bb9fb239d 2341df5e6181eac89ff02584812e90af6e57f635 f9bd2d8b72465686d38bdf9f0ad38d4d60145fd8

1.2 Files in Scope

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

ID	Files	SHA-1 Hash
PUT	sources/exchange/pool_utils.move	b677d133902033e85d81f7c33247f49e98ba131f
WSP	sources/exchange/wisp_pool.move	efbec6cc4f8663979db5940b8128c4d36baa4143
WSR	sources/exchange/wisp_router.move	968d54e6eabab033b3ea0bca47e9bd728452ee52

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	9	7	2
Informational	1	1	
Minor	3	3	
Medium	1		1
Major	4	3	1
Critical			

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

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 **WispSwap** to identify any potential issues and vulnerabilities in the source code of the **WispSwap Exchange** 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 have identified **9** issues of varying severity, listed below.

ID	Title	Severity	Status
WSP-01	Incorrect Comment for Function `process_swap_exact_output`	Informational	Fixed
WSP-02	Incorrect Protocol Fee Handling	Major	Fixed
WSR-03	Incorrect Condition Check in `swap_exact_output_doublehop_` Function	Major	Fixed
WSP-04	Discussion for `process_swap_exact_output`	Minor	Fixed
WSP-05	Lack of Validation for `output_amount` in `get_input_amount`	Minor	Fixed
WSP-06	Missing Emit Events	Minor	Fixed
WSP-07	Lack of Verification of Reserve in Pool	Major	Acknowledged
WSP-08	Missing Check for Liquidity Greater than 0	Major	Fixed
WSP-09	Function Visibility Error	Medium	Acknowledged

3 Participant Process

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

Creator

- Creators can change the `fee_to` address through `set_fee_to_()`.

User

- User can create any liquidity pool through `create_pool()`.
- User can add liquidity to any existing pool through `add_liquidity()`, `zap_in_first()` and `zap_in_second()`.
- User can remove its liquidity from a pool through `remove_liquidity()`.
- User can swap tokens for other tokens as it needs through `swap_exact_first_to_second()`, `swap_exact_second_to_first()`, `swap_first_to_exact_second()`, `swap_second_to_exact_first()`, `process_swap_exact_input()` and `process_swap_exact_output()`.

4 Findings

WSP-01 Incorrect Comment for Function `process_swap_exact_output`

Severity: Informational

Status: Fixed
Code Location: sources/exchange/wisp_pool.move, L581
Descriptions: The function is used to swap for an exact output amount, so the comment is incorrect.
Suggestion: It is recommended to update the Swapping exact input to Swapping to get the exact output.

Resolution: The client has followed our suggestion and fixed the issue.

WSP-02 Incorrect Protocol Fee Handling

Severity: Major

Status: Fixed
Code Location: sources/exchange/wisp_pool.move, L625, L641
Descriptions: Problem reproduction:

1. When initializing, if `fee_to` is set to 0x0, the condition on line 640 will be triggered, and `pool.k_last` will be set to 0;
2. Once `k_last` is set to 0, the condition on line 625 will not be triggered;
3. The conditions on lines 282 and 335 that `fee_amount > 0` will not be triggered, and `k_last` will never be updated again;
4. Even if `fee_to` is set to a valid address afterward, the fee for this pool will never be obtained.

Suggestion: Use `fee_to != @0x0` as the condition to update `k_last` on lines 282 and 335.

Resolution: The client has followed our suggestion and fixed the issue.

WSR-03 Incorrect Condition Check in `swap_exact_output_doublehop_` Function

Severity: Major

Status: Fixed

Code Location: sources/exchange/wisp_router.move, L401, L425

Descriptions:

1. On line 401, the wrong use of the sorting result of the `second pool` as the condition, it should be `comparator::is_smaller_than(&sort_first_pool)`;
2. On line 425, the wrong use of the sorting result of the `first pool` as the condition, it should be `comparator::is_smaller_than(&sort_second_pool)`.

Suggestion: It's recommended to modify the code to the correct conditions.

Resolution: The client has followed our suggestion and fixed the issue.

WSP-04 Discussion for `process_swap_exact_output`

Severity: Minor

Status: Fixed

Code Location: sources/exchange/wisp_pool.move, L582

Descriptions: When calculating the input amount for a given output quantity, there is a risk of precision loss, which may result in a lower amount of tokens being deducted than expected. This could cause harm to users who are adding liquidity.

Suggestion: We would like to know if it aligns with the design.

Resolution: The client has modified the logic and fixed the issue.

WSP-05 Lack of Validation for `output_amount` in `get_input_amount`

Severity: Minor

Status: Fixed
Code Location: sources/exchange/wisp_pool.move, L725
Descriptions: In `get_input_amount`, if the `output_amount` is greater than the maximum reserve, it will revert.
Suggestion: It is recommended to add a check for the `output_amount`.

Resolution: The client has followed our suggestion and fixed the issue.

WSP-06 Missing Emit Events

Severity: Minor

Status: Fixed
Code Location: exchange/wisp_pool.move, L646
Descriptions: The `set_fee_to` method is used to change the address of the fee recipient. However, since this is a critical piece of information, it should trigger a log when changed.
Suggestion: It is recommended to emit events when executing sensitive actions.

Resolution: The client has followed our suggestion and fixed the issue.

WSP-07 Lack of Verification of Reserve in Pool

Severity: Major

Status: Acknowledged
Code Location: exchange/wisp_pool.move, L225
Descriptions: When adding liquidity, there is a lack of verification of the reserve in the pool. If there is no `first_reserve` and `second_reserve` in the current pool, according to the current code logic, it will never be possible to successfully add liquidity.
Suggestion: Add the liquidity calculation formula when the reserve is empty.

Resolution: The client decided not to fix it since that case is handled in the `create_pool` method and a `MINIMUM_LIQUIDITY` is burned to make sure the reserves won't be empty.

WSP-08 Missing Check for Liquidity Greater Than 0

Severity: Major

Status: Fixed
Code Location: exchange/wisp_pool.move, L282
Descriptions: When adding liquidity, you should check whether the final added liquidity is greater than 0. If it is equal to 0, it will cause user asset loss.
Suggestion: The liquidity added by adding an assert check is greater than 0.

Resolution: The client has followed our suggestion and fixed the issue.

WSP-09 Function Visibility Error

Severity: Medium

Status: Acknowledged
Code Location: exchange/wisp_pool.move, L567, L590
Descriptions: The functions `process_swap_exact_input()` and `process_swap_exact_output()` do not provide parameters to limit the minimum number of output tokens.
Suggestion: Change it to a private function.

Resolution: The client is aware of this problem, and there is a comment in front of the function to remind users to use it with caution.

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

MOVEBIT

Securing the Move Ecosystem

https://twitter.com/movebit_

contact@movebit.xyz