iZUMi Finance

Smart Contract

Audit Report

07/13/2023

✉ contact@scalebit.xyz   https://twitter.com/scalebit_

ScaleBit

iZUMi Audit Report

1 Executive Summary

1.1 Project Information

Description	Discretized Liquidity AMM
Type	AMM
Auditors	ScaleBit
Timeline	July 3, 2023 – July 11, 2023
Languages	Solidity
Platform	zkSync
Methods	Architecture Review, Unit Testing, Manual Review, Static analysis
Source Code	https://github.com/izumiFinance/iZiSwap-core https://github.com/izumiFinance/iZiSwap-periphery
Commits	392f52725e44e20a5a7931aaae47c1b015ebd634 d1aa577edca18e17e0dc099801304a2aa3c0d9c7

1.2 Files in Scope

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

ID	Files	SHA-1 Hash
FLH	contracts/flash.sol	86e27eb0111f24c47303d543dd37a532f7b2b68b
FAC	contracts/iZiSwapFactory.sol	361e52a8d2a094f9b7eb68cf5c26ee8626ea1cdc

POL	contracts/iZiSwapPool.sol	200e0caa376e658d83f35baba2c13502396fafe8
LOR	contracts/limitOrder.sol	4e76e6841e0dc0f42da866a933fa68b770dc55d4
LQT	contracts/liquidity.sol	86e27eb0111f24c47303d543dd37a532f7b2b68b
SXY	contracts/swapX2Y.sol	589fcc506951e6cbcfa7eaaa0a986041985193e7
SYX	contracts/swapY2X.sol	589fcc506951e6cbcfa7eaaa0a986041985193e7
LOM	contracts/LimitOrderManager.sol	b9a693121ea59e1d6905835347e4d1f40364091f
LOW	contracts/LimitOrderWithSwapManager.sol	e500c8a4a4f1830b72bade61fc74d8105609a686
LQM	contracts/LiquidityManager.sol	a345ea2db1da968b6330c2c78d5bb956d3689846
QTR	contracts/Quoter.sol	24ecfb2d67ab201fb57d46c492d7f5e2b347951e
QWL	contracts/QuoterWithLim.sol	94b012ef1a49ab75a46026c69fe779338eeb10e0
SWP	contracts/Swap.sol	d303d9ccad7b9420ecd184313a06e3dceab17ed7
AMH	contracts/libraries/AmountMath.sol	efcbcc474ecbc0c8bbb979bee07c39a580edc256
CVT	contracts/libraries/Converter.sol	d683475146fe345c9edd2127dd4a618df405eb75
LOD	contracts/libraries/LimitOrder.sol	8513def919d149d0dd7adab4ea013ce615e656ec
LQY	contracts/libraries/Liquidity.sol	d683475146fe345c9edd2127dd4a618df405eb75

LPM	contracts/libraries/LogPowMath.sol	d683475146fe345c9edd2127dd4a618df405eb75
MMM	contracts/libraries/MaxMinMath.sol	d683475146fe345c9edd2127dd4a618df405eb75
MDM	contracts/libraries/MulDivMath.sol	d683475146fe345c9edd2127dd4a618df405eb75
ORA	contracts/libraries/Oracle.sol	d683475146fe345c9edd2127dd4a618df405eb75
OOE	contracts/libraries/OrderOrEndpoint.sol	d683475146fe345c9edd2127dd4a618df405eb75
POT	contracts/libraries/Point.sol	5336041b58d832e5c5be5c7393c46dda69f37ccd
PBM	contracts/libraries/PointBitmap.sol	61df0aad8ad2ebe3ff47647378a79d47b52a980d
STE	contracts/libraries/State.sol	5336041b58d832e5c5be5c7393c46dda69f37ccd
SCE	contracts/libraries/SwapCache.sol	7d321baefb1b99a7fc43433a8f89c2831a979950
SMH	contracts/libraries/SwapMathX2Y.sol	61df0aad8ad2ebe3ff47647378a79d47b52a980d
SMD	contracts/libraries/SwapMathX2YDesire.sol	61df0aad8ad2ebe3ff47647378a79d47b52a980d
SMY	contracts/libraries/SwapMathY2X.sol	efcbcc474ecbc0c8bbb979bee07c39a580edc256

SXD	contracts/libraries/SwapMathY2XDesire.sol	d683475146fe345c9edd2127dd4a618df405eb75
TTF	contracts/libraries/TokenTransfer.sol	d683475146fe345c9edd2127dd4a618df405eb75
TPW	contracts/libraries/TwoPower.sol	d683475146fe345c9edd2127dd4a618df405eb75
UER	contracts/libraries/UserEarn.sol	48279428f57b0b8f4434dd192c967433aa60f987
AMT	contracts/libraries/AmountMath.sol	86d376d9b5bbd10a9c091145b5e45c2018c0026c
BLB	contracts/libraries/BytesLib.sol	86d376d9b5bbd10a9c091145b5e45c2018c0026c
CVT	contracts/libraries/Converter.sol	6db8d5ce692e18a3a890c745dfd015fd116dab7f
LMO	contracts/libraries/LimOrder.sol	c4dc0b8b874aa08c3d1fab230ebd8c5a0fdd706e
LCQ	contracts/libraries/LimOrderCircularQueue.sol	86d376d9b5bbd10a9c091145b5e45c2018c0026c
LPM	contracts/libraries/LogPowMath.sol	86d376d9b5bbd10a9c091145b5e45c2018c0026c
MMH	contracts/libraries/MintMath.sol	994bd251d2bb4c07bcc204af0ecc651d76a54bc3
MDM	contracts/libraries/MulDivMath.sol	86d376d9b5bbd10a9c091145b5e45c2018c0026c
PTH	contracts/libraries/Path.sol	86d376d9b5bbd10a9c091145b5e45c2018c0026c

TPR	contracts/libraries/TwoPower.sol	86d376d9b5bbd10a9c091145b5e45c2018c0026c

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	2		2
Informational			
Minor			
Medium	2		2
Major			
Critical			

1.4 ScaleBit Audit BreakDown

ScaleBit 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
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power

- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

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) 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 **iZUMi Finance** to identify any potential issues and vulnerabilities in the source code of the **iZiswap** 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 **2** issues of varying severity, listed below.

ID	Title	Severity	Status
SWP-01	Steal Fees From Protocol	Medium	Acknowledged
QTR-02	No deadline control for swapping	Medium	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the `iZiswap` SmartContract:

Owner

- Owner can modify charge receiver through `modifyChargeReceiver()` .
- Owner can modify the default fee charge percent through `modifyDefaultFeeChargePercent()` .
- Owner can enable fee amount through `enableFeeAmount()` .

User

- User can Create a limit order for recipient through `newLimOrder()` .
- User can update a limit order to claim earned tokens as much as possible through `updateOrder()` .
- User can decrease amount of selling-token of a limit order through `decLimOrder()` .
- User can collect earned or decreased token from a limit order through `collectLimOrder()` .
- User can cancel a limit order through `cancel()` .
- User can collect earned token from an limit order through `collect()` .
- User can perform the exchange of a single asset through `swapDesireSingle()` .
- User can swap asset through `swapAmountSingle()` .
- User can create a pool through `createPool()` .

- User can add a new liquidity through `mint()` .
- User can burn a generated nft through `burn()` .
- User can add liquidity to a existing nft through `addLiquidity()` .
- User can decrease liquidity from a nft through `decLiquidity()` .
- User can collect fee gained of token withdrawed from nft through `collect()` .
- User can swap given amount of target token through `swapDesire()` .
- User can swap given amount of input token through `swapAmount()` .
- User can swap tokenY for tokenX through `swapY2X()` .
- User can swap tokenY for tokenX , given user's desired amount of tokenX ,through `swapY2XDesireX()` .
- User can swap tokenX for tokenY through `swapX2Y()` .
- User can swap tokenX for tokenY, given amount of tokenY user desires through `swapX2YDesireY()` .

4 Findings

SWP-01 Steal Fees From Protocol

Severity: Medium

Status: Acknowledged

Code Location: contracts/Swap.sol#L178-L194

Descriptions: The `Swap.swapDesire()` function is used to handle a swap operation, ensuring that the user pays an acceptable amount and receives the desired amount of tokens.

This function could be vulnerable to a type of frontrunning attack known as a "sandwich attack". In this scenario, a bad actor monitors the mempool (the pool of pending transactions) for swap transactions.

When they identify a suitable swap transaction, they can take advantage of the information provided by the current point and high point parameters in the swap. The attacker first adds liquidity to the pool, which could potentially influence the price of the tokens being swapped.

Once the original swap transaction is executed and the tokens are exchanged, the attacker then removes their added liquidity from the pool.

By doing so, they may earn fees from the swap transaction that they wouldn't have otherwise received.

QTR-02 No Deadline Control for Swapping

Severity: Medium

Status: Acknowledged

Code Location: /contracts/Quoter.sol#L183-L208

Descriptions: The token swap function in the Quotrer and QuoterWithLim contracts does not include a deadline check. This omission can allow transactions to be executed in unfavorable conditions or be maliciously exploited, especially in the context of Miner Extractable Value (MEV).

Users might unknowingly perform trades that are disadvantageous to them if market conditions change dramatically after the transaction has been broadcasted but before it is included in a block. Additionally, miners or any privileged entities could manipulate the order of transactions to benefit from user trades. Both scenarios could lead to a loss of funds for users.

```
function swapY2X(
        address tokenX,
        address tokenY,
        uint24 fee,
        uint128 amount,
        int24 highPt
    ) public returns (uint256 amountX, int24 finalPoint) {
        require(tokenX < tokenY, "x<y");
        address poolAddr = pool(tokenX, tokenY, fee);
        try
            IiZiSwapPool(poolAddr).swapY2X(
                address(this), amount, highPt,
                abi.encodePacked(tokenY, fee, tokenX)
            )
        {} catch (bytes memory reason) {
            (amountX, finalPoint) = parseRevertReason(reason);
        }
    }
```

Suggestion: Implement a deadline check in the swap function.

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.