#HashEx
BLOCKCHAIN SECURITY

Swapline

smart contracts final audit report

October 2023

hashex.org

contact@hashex.org

Contents

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed. HashEx owns all copyright rights to the text, images, photographs, and other content provided in the following document. When using or sharing partly or in full, third parties must provide a direct link to the original document mentioning the author (hashex.org).

2. Overview

HashEx was commissioned by the Swapline team to perform an audit of their smart contract. The audit was conducted between 01-09-2023 and 27-09-2023. The code is a fork of traderjoe-xyz/joe-v2 repository which was audited by Paladin Blockchain Security after the commit 7f71d0. According to the Paladin report the original code has no critical, high or medium security unresolved issues.

The purpose of this audit was to achieve the following:

- Identify potential security issues introduced with updates to the original smart contracts
- Formally check the logic behind the updates in the given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

The code is available at @metropolis-exchange/lb-v2 and was audited after the 3601c1a commit.

Update: the Swapline team has responded to this report. The updated code is located in the same repository after the bc981fc commit.

2.1 Summary

Project name	Swapline
URL	https://swapline.com/
Platform	Fantom Network, Polygon Network, Optimism, Polygon Zkevm, Base
Language	Solidity

Centralization level	Medium
Centralization risk	High

2.2 Contracts

Name	Address
LBFactory	
LBPair	
LBQuoter	
LBRouter	
LBSwaplineQuoter	
LBToken	
Encoded	
LiquidityConfiguration	
PackedUint128Math	
SafeCast	
SampleMath	
TreeMath	
AddressHelper	
BinHelper	
Clone	

- Constants
- FeeHelper
- ImmutableClone
- OracleHelper
- PairParameterHelper
- PendingOwnable
- PriceHelper
- ReentrancyGuard
- TokenHelper

3. Project centralization risks

The project owner can change an implementation for pair contract, update parameters for new pairs, update fees parameters for existing pairs, manage whitelisted assets for quoting, disable selected pairs from routing.

4. Found issues

Low	1 (33%)
Info	2 (67%)

C25. LBSwaplineQuoter

ID	Severity	Title	Status
C25I8f	Info	Unnecessary code	Resolved
C25I90	Info	Not implemented safe cast fix in legacy code	Resolved

C2c. TreeMath

ID	Severity	Title	Status
C2cI8e	Low	Not implemented fix	Resolved

5. Contracts

C21. LBFactory

Overview

A factory contract for Liquidity Book pairs creation. No changes were made to the original contract.

C22. LBPair

Overview

The Liquidity Book Pair is the contract for the DEX. Has low level functions for minting liquidity tokens, performing swaps, flashloans. The low-level functions are meant to be called by an external contract (a router) which makes additional checks.

The only change made to the original contract is an added modifier in mint function which makes safety checks for liquidity tokens not minted to the pair or zero addresses.

No issues were found with the introduced updates.

C23. LBQuoter

Overview

A helper contract for finding best swap path. No changes were made to the original contract.

C24. LBRouter

Overview

A user-interacting contract to swap and manage liquidity stored in LBPairs. No changes were made to the original contract.

C25. LBSwaplineQuoter

Overview

Fork of LBQuoter with implemented overflow fixes.

Issues

C25I8f Unnecessary code

Info Resolved

The original code was written to be compatible with legacy versions. The Swapline project doesn't have any legacy contracts, thus, unused code can be removed.

```
function findBestPathFromAmountIn(
    address[] calldata route,
    uint128 amountIn
) public view returns (Quote memory quote) {
    ...
    for (uint256 i; i < swapLength; i++) {
        if (_factoryV1 != address(0)) {
            ...
        }
        // Fetch swap for V2
        if (_legacyFactoryV2 != address(0)) {
            ...
        }
    }
    ...
 }
```

It should be noted that the unused code only affects deployment costs. Since the `_factoryV1` and `_legacyFactoryV2` variables are declared immutable, there is no gas expenditure on reading them.

Recommendation

Remove unused code.

C25I90 Not implemented safe cast fix in legacy code

Info Resolved

The LBSwapline quoter contract incorporates SafeCast improvements to reduce the risk of integer overflow during mathematical calculations. However, these fixes are not applied to the sections of the contract that interact with legacy contracts.

```
function findBestPathFromAmountIn(
        address[] calldata route,
        uint128 amountIn
    ) public view returns (Quote memory quote) {
        ...
        for (uint256 i; i < swapLength; i++) {
            if (_factoryV1 != address(0)) {
                ...
                        quote.amounts[i + 1] = uint128(
                            JoeLibrary.getAmountOut(quote.amounts[i], reserveIn,
reserveOut)
                        );
                ...
            }
            // Fetch swap for V2
            if (_legacyFactoryV2 != address(0)) {
                ...
                                        quote.fees[i] = uint128((fees * 1e18) /
quote.amounts[i]); // fee percentage in amountIn
                ...
            }
        }
        ...
     }
```

Recommendation

Implement the safecast fixes or remove unused code.

C26. LBToken

Overview

A multi-token contract for batch operations. It's similar to ERC1155 standard but without callbacks and URI function. LBPair inherits from this contract.

The changes made to the original contracts are: removed checks for non-zero or same-contract addresses in the internal `_mint` and `_burn` functions and added check for `_approveForAll()` internal function.

No issues were introduced with the updates.

C27. Encoded

Overview

A helper library used for decoding `bytes32` sample.

No changes were made to the original contract.

C28. LiquidityConfiguration

Overview

A library containing functions to encode and decode the config of a pool and interact with the encoded `bytes32`.

No changes were made to the original contract.

C29. PackedUint128Math

Overview

A library containing functions to encode and decode two `uint128` into a single `bytes32`.

No changes were made to the original contract.

C2a. SafeCast

Overview

A library containing functions to safely cast uint256 to different uint types

No changes were made to the original contract.

C2b. SampleMath

Overview

A library containing functions to encode and decode a sample into a single `bytes32` and interact with the encoded `bytes32`

No changes were made to the original contract.

C2c. TreeMath

Overview

A library contract containing functions to interact with a tree of `TreeUint24` structures. TreeUint24 is a three-level tree used to add, remove, and search for bins.

Issues

C2cl8e Not implemented fix

Low Resolved

Upon reviewing the codebase under audit, it was observed that a significant fix applied to the original repository's TreeMath library is absent. This particular fix is documented in pull request number 104, titled "Tree issue".

Recommendation

We advise reviewing the changes made in pull request #104 in the original repository and integrating the necessary modifications into the code under audit to ensure its security.

C2d. AddressHelper

Overview

Library contract containing functions to check if an address is a contract and catch low level errors.

No changes were made to the original contract.

C2e. BinHelper

Overview

A library containing functions to help interaction with bins

No changes were made to the original contract.

C2f. Clone

Overview

An abstract contract with helper read functions for clone with immutable args.

No changes were made to the original contract.

C30. Constants

Overview

A library containing different constants.

No changes were made to the original contract.

C31. FeeHelper

Overview

A library with helper functions for fee calculation.

No changes were made to the original contract.

C32. ImmutableClone

Overview

A library implementing minimal proxy pattern for gas-efficient contract creation.

No changes were made to the original contract.

C33. OracleHelper

Overview

A library containing functions to manage the oracle.

No changes were made to the original contract.

C34. PairParameterHelper

Overview

A library containing functions to get and set parameters of a pair.

No changes were made to the original contract.

C35. PendingOwnable

Overview

A contract that provides a basic access control mechanism. where there is an owner account that can be granted exclusive access to specific functions, the ownership is transferred in two steps: the current owner sets a pending owner and then that address can claim ownership.

No changes were made to the original contract.

C36. PriceHelper

Overview

A library contract containing funcions for prices calculation. The main purpose is to fetch a price from id and vise versa.

No changes were made to the original contract.

C37. ReentrancyGuard

Overview

A library contract containing functions for preventing reentrancy attacks.

No changes were made to the original contract.

C38. TokenHelper

Overview

A helper library with wrappers around ERC20 token transfers. Handles tokens that do not return a boolean on `transfer()` and `transferFrom()` calls.

No changes were made to the original contract.

6. Conclusion

The code is a fork of traderjoe-xyz/joe-v2 repository which underwent an audit by Paladin Blockchain Security after the commit 7f71d0. The audit confirmed that no discrepancies or vulnerabilities were added compared to the original code.

Appendix A. Issues' severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Informational.** Issues that do not impact the contract operation. Usually, informational severity issues are related to code best practices, e.g. style guide.

Appendix B. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

 contact@hashex.org

 @hashex_manager

 blog.hashex.org

 linkedin

 github

 twitter

#HashEx
BLOCKCHAIN SECURITY