EchoDEX
Audit Report

Fri Aug 11 2023

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

ScaleBit

EchoDEX Audit Report

1 Executive Summary

1.1 Project Information

Description	A decentralized exchange platform built on the Linea Consensys network
Type	Dex
Auditors	ScaleBit
Timeline	Mon Jul 17 2023 - Fri Aug 11 2023
Languages	Solidity
Platform	Linea
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/echodex-io/echodex-contracts
Commits	84c26a1b6dfbfdb7d3b4f733adcef47983fd6d46

1.2 Files in Scope

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

ID	File	SHA-1 Hash
ELI	contracts/libraries/EchodexLibrary.sol	41b978cd047415590f2d171e7eb5b03b2d62c535
MAT	contracts/libraries/Math.sol	1f00b49134faebcd3446d7e5e615770b76dc4435
WETH	contracts/libraries/WETH.sol	fe233bb7a612205842121c7b42831fdb342c6191
SMA	contracts/libraries/SafeMath.sol	384533451967738f6e2f5ff50a88fe61c1de2a56
THE	contracts/libraries/TransferHelper.sol	8bcab3db5a0fae439475640f745dcc2e7b08007d
UQ1X1	contracts/libraries/UQ112x112.sol	1893e45b63f692dd2d1ca610b72e832a0c511d68
EFA	contracts/EchodexFactory.sol	c378c8452a601db52e73b21395e3d2a7db2b02af
MERC2	contracts/utils/MockERC20.sol	304f1152f4b613a4805f7fd6f8c77692088301ce
ERC2	contracts/utils/ERC20.sol	e12414fd73963b3e3c06e1ae2e3d03ff8b51e8af
CON	contracts/utils/Context.sol	a34cc2179b2da819d60afa9d711d0094d5a72799
IERC2	contracts/utils/IERC20.sol	2d6eb8a102a8a92dcfef4d19c977a062e891c7cf
ERO	contracts/EchodexRouter.sol	61cd571a853f320a16ce78e5b180d2c355598dba

EFA	contracts/EchodexFarm.sol	40575256745b74df2855260e319f542f2f9ceb22
ERF	contracts/EchodexRouterFee.sol	350267e4f5ff63fa3aae4c41668e6fba3f7e912d
EERC2	contracts/EchodexERC20.sol	4899e57b09502f3d4a5288f2809accef42aecd79
EPA	contracts/EchodexPair.sol	0e12cd0fd4e9f60ef75dbb45f04fe70eb6901d06

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	4	1	3
Informational	1	1	0
Minor	0	0	0
Medium	3	0	3
Major	0	0	0
Critical	0	0	0

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 EchoDEX to identify any potential issues and vulnerabilities in the source code of the EchoDEX 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 4 issues of varying severity, listed below.

ID	Title	Severity	Status
ERC-1	Approve Function Can Be Front-run	Medium	Acknowledged
ELI-1	Precision Loss Caused By Multiplication Before Division	Medium	Acknowledged
IEP-1	Use SafeTransfer/SafeTransferFrom Consistently Instead of Transfer/TransferFrom	Medium	Acknowledged
ELI-2	Redundant Code	Informational	Fixed

3 Participant Process

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

Owner

- Owner can set whether a mining pool requires blue-check by calling the `setBlueCheck()` function.
- Owner can stop the operation of a specific mining pool by calling the `stopPool()` function.
- Owner has the ability to change the tokenFee, tokenReward, and receiveFeeAddress addresses by calling the respective `setTokenFee()`, `setTokenReward()`, and `setReceiveFeeAddress()` functions.
- Owner can set the fee path for a specific token output by calling the `setFeePath()` function, allowing the contract to calculate fees for different token outputs based on the specified path.
- Owner can set the reward percentage for specific trading pairs using the `setRewardPercent()` function, determining the percentage of rewards distributed to LP providers in each pair.
- Owner can set and manage the fees.

User

- User can stake their liquidity tokens into a specific mining pool using the `stake()` function to participate in the mining activity.
- User can unstake their staked liquidity tokens from the mining pool by calling the `unstake()` function.
- User can claim their earned reward tokens by calling the `harvest()` function.
- User can withdraw excess reward tokens from the mining pool using the `withdrawExcess()` function.
- User can call the `allPairsLength()` function to get the total number of pairs created on the Echodex exchange.
- User can call the `createPair()` function to create a new trading pair by providing the addresses of two tokens. Each pair will have a unique pair address.
- User can call the `calcFeeOrReward()` function to calculate the fee or reward amount for a given token output, considering the specified percentage and fee path.
- User can provide liquidity to the pool by calling the `mint()` function, depositing an equal value of both tokens in the pair.

- User can withdraw their liquidity from the pool by calling the `burn()` function and providing an equivalent amount of LP tokens.
- User can perform token swaps between the two tokens in the pair by calling the `swap()` or `swapPayWithTokenFee()` functions.
- User can add liquidity for two tokens to the trading pair by calling `addLiquidity()` function.
- User can add liquidity for ETH and another token to the trading pair using `addLiquidityETH()` function.
- User can remove liquidity from the trading pair and get back the respective tokens by calling `removeLiquidity()` function.
- User can remove ETH and the corresponding token liquidity from the trading pair using `removeLiquidityETH()` function.
- User can swap tokens with a specified exact output amount using `swapExactTokensForTokens()` function.
- User can swap tokens to get a specified exact output amount using `swapTokensForExactTokens()` function.
- User can swap tokens with a specified exact output amount using ETH by calling `swapExactETHForTokens()` function.
- User can swap tokens to get a specified exact ETH output amount using `swapTokensForExactETH()` function.
- User can swap tokens with a specified exact output amount to get ETH by calling `swapExactTokensForETH()` function.
- User can swap ETH to get a specified exact token output amount by calling `swapETHForExactTokens()` function.

4 Findings

ERC-1 Approve Function Can Be Front-run

Severity: Medium

Status: Acknowledged

Code Location:

contracts/utils/ERC20.sol#L70

Descriptions:

The `approve()` function has a known race condition that can lead to token theft. If a user calls the approve function a second time on a spender that was already allowed, the spender can front-run the transaction and call `transferFrom()` to transfer the previous value and still receive the authorization to transfer the new value.

Suggestion:

Consider implementing functionality that allows a user to increase and decrease their allowance similar to Lido's implementation. This will help prevent users losing funds from front-running attacks.

```
/**
    * @notice Atomically increases the allowance granted to `_spender` by the caller by `_addedValue`.
    *
    * This is an alternative to `approve` that can be used as a mitigation for
    * problems described in:
    * https://github.com/OpenZeppelin/openzeppelin-
contracts/blob/master/contracts/token/ERC20/IERC20.sol#L42
    * Emits an `Approval` event indicating the updated allowance.
    *
    * Requirements:
    *
    * - `_spender` cannot be the the zero address.
    * - the contract must not be paused.
    */
function increaseAllowance(address _spender, uint256 _addedValue) public returns (bool) {
  _approve(msg.sender, _spender, allowances[msg.sender][_spender].add(_addedValue));
  return true;
}

/**
```

```
    * @notice Atomically decreases the allowance granted to `_spender` by the caller by
`_subtractedValue`.
    *
    * This is an alternative to `approve` that can be used as a mitigation for
    * problems described in:
    * https://github.com/OpenZeppelin/openzeppelin-
contracts/blob/master/contracts/token/ERC20/IERC20.sol#L42
    * Emits an `Approval` event indicating the updated allowance.
    *
    * Requirements:
    *
    * - `_spender` cannot be the zero address.
    * - `_spender` must have allowance for the caller of at least `_subtractedValue`.
    * - the contract must not be paused.
    */
function decreaseAllowance(address _spender, uint256 _subtractedValue) public returns (bool) {
  uint256 currentAllowance = allowances[msg.sender][_spender];
  require(currentAllowance >= _subtractedValue, "DECREASED_ALLOWANCE_BELOW_ZERO");
  _approve(msg.sender, _spender, currentAllowance.sub(_subtractedValue));
  return true;
}
```

ELI-1 Precision Loss Caused By Multiplication Before Division

Severity: Medium

Status: Acknowledged

Code Location:

contracts/libraries/EchodexLibrary.sol#L74-87

Descriptions:

Due to the calculation sequence of division followed by multiplication, precision loss may occur. This is because division operations can truncate the result, leading to the loss of precision in the decimal part. Subsequently, the multiplication operation amplifies the precision loss on the truncated result. Such processing may result in the final `amountOut` value not being the expected precise value.

```
uint256 numerator = amountIn.mul(reserveOut);
uint256 denominator = reserveIn.add(amountIn);
amountOut = numerator / denominator;
amountOut = amountOut.mul(997) / 1000;
```

Suggestion:

To minimize precision loss, it is advisable to consider changing the calculation sequence by performing multiplication before division. This ensures that the precision of the result is as high as possible before the division operation, thereby avoiding financial losses.

IEP-1 Use SafeTransfer/SafeTransferFrom Consistently Instead of Transfer/TransferFrom

Severity: Medium

Status: Acknowledged

Code Location:

contracts/interfaces/IEchodexPair.sol#L282

Descriptions:

Some tokens do not revert on failure, but instead return false (e.g. ZRX). https://github.com/d-xo/weird-erc20/#no-revert-on-failure Tranfser/Transferfrom is directly used to send tokens in many places in the contract and the return value is not checked. If the token send fails, it will cause a lot of serious problems.

```
IERC20(tokenFee).transferFrom(msg.sender, address(this), amount);
```

Suggestion:

I am not certain about the token type used here. Even though the current token does not pose any issues, to account for scalability and potential uncertainties in the future, we recommend using SafeTransfer/SafeTransferFrom consistently instead of Transfer/TransferFrom.

ELI-2 Redundant Code

Severity: Informational

Status: Fixed

Code Location:

contracts/libraries/EchodexLibrary.sol#L44

Descriptions:

This is an unnecessary line of code.

```
pairFor(factory, tokenA, tokenB);
```

Suggestion:

Recommend removing redundant code.

Resolution:

The unnecessary code has been removed.

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.