VITAL BLOCK

Security Assessment

ANCORA PROTOCOL

Audit Report Verified On October 3rd, 2023

@Vital-Block

@VB_Audit

info@vitalblock.org

www.vitalblock.org

PREPARED FOR:
ANCORA

INTRODUCTION

Auditing Company	**VITAL BLOCK SECURITY**
Client Project	**ANCORA PROTOCOL**
Methodology	**Automated Analysis, Manual Code Review**
Verified	**No**
Compiler version	**v0.8.18+commit.87f61d96**
Contract Address	**Router:** 0xa043BfFcaA9Ebaa6708FcbFa4909B100Af47Fd15 **Factory:** 0xE7aC188E018f954A83c157ac686De7F66e819a51
Network	**LINEA BLOCKCHAIN**
Optimization	**200 RUNS**
Contract Type	**ERC20**
Website	**https://ancora.finance/**
Telegram	**https://t.me/Ancora_Finance**
Twitter	**https://twitter.com/AncoraFinance**
Discord	**https://discord.com/invite/7QMjDAgqNF**
Prelim Report Date	**October 2ND, 2023**
Final Report Date	**October 3rd, 2023**

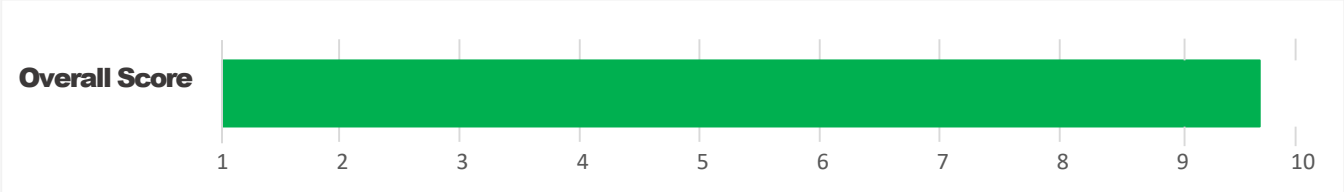
Verify the authenticity of this report on our GitHub Repo: https://www.github.com/vital-block

EXECUTIVE SUMMARY

Vital Block has performed the automated and manual analysis of Ancora Finance Sol code. The code was reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical !	Major "	Medium #	Minor $	Unknown %
Open	0	0	0	2	0
Acknowledged	0	0	1	2	0
Resolved	0	0	0	2	0
Noteworty OnlyOwner Privileges	Set Taxes and Ratios, Airdrop, Set Protection Settings, Set Reward Properties, Set Reflector Settings, Set Swap Settings, Set Pair and Router				

ANCORA FINANCE Smart contract has achieved the following score: 97



- **Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.**
- **Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.**

SCOPE OF WORK

Vital Block was consulted by ANCORA FINANCE to conduct the smart contract audit of its .Sol source code. The audit scope of work is strictly limited to mentioned SOL file only:

- **Router.Sol**
- **Factory.Sol**

ℹ️ **External contracts and/or interfaces dependencies are not checked due to being out of scope.**

Verify audited contract's contract address and deployed link below:

Public Contract.	
Router: 0xa043BfFcaA9Ebaa6708FcbFa4909B100Af47Fd15	
Factory: 0xE7aC188E018f954A83c157ac686De7F66e819a51	
Contract Name	**ANCORA FINANCE**
Ticker	**ACR**
Blockchain	**Linea Blockchain**

AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of Vital Block auditing process and methodology:

CONNECT

- **The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.**

AUDIT

- **Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:**
  - **Remix IDE Developer Tool**
  - **Open Zeppelin Code Analyzer**
  - **SWC Vulnerabilities Registry**
  - **DEX Dependencies, e.g., Pancakeswap, Uniswap**
- **Simulations are performed to identify centralized exploits causing contract and/or trade locks.**
- **A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:**

Centralized Exploits	○ **Token Supply Manipulation** ○ **Access Control and Authorization** ○ **Assets Manipulation** ○ **Ownership Control** ○ **Liquidity Access** ○ **Stop and Pause Trading** ○ **Ownable Library Verification**

Common Contract Vulnerabilities	○ Integer Overflow ○ Lack of Arbitrary limits ○ Incorrect Inheritance Order ○ Typographical Errors ○ Requirement Violation ○ Gas Optimization ○ Coding Style Violations ○ Re-entrancy ○ Third-Party Dependencies ○ Potential Sandwich Attacks ○ Irrelevant Codes ○ Divide before multiply ○ Conformance to Solidity Naming Guides ○ Compiler Specific Warnings ○ Language Specific Warnings

REPORT

- The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.
- The client's development team reviews the report and makes amendments to the codes.
- The auditing team provides the final comprehensive report with open and unresolved issues.

PUBLISH

- The client may use the audit report internally or disclose it publicly.

ℹ️ It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.

RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

Risk Type	Definition
Critical "	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Major "	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
Medium #	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Minor $	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
Unknown %	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

CENTRALIZED PRIVILEGES

Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.

There are some well-intended reasons have privileged roles, such as:

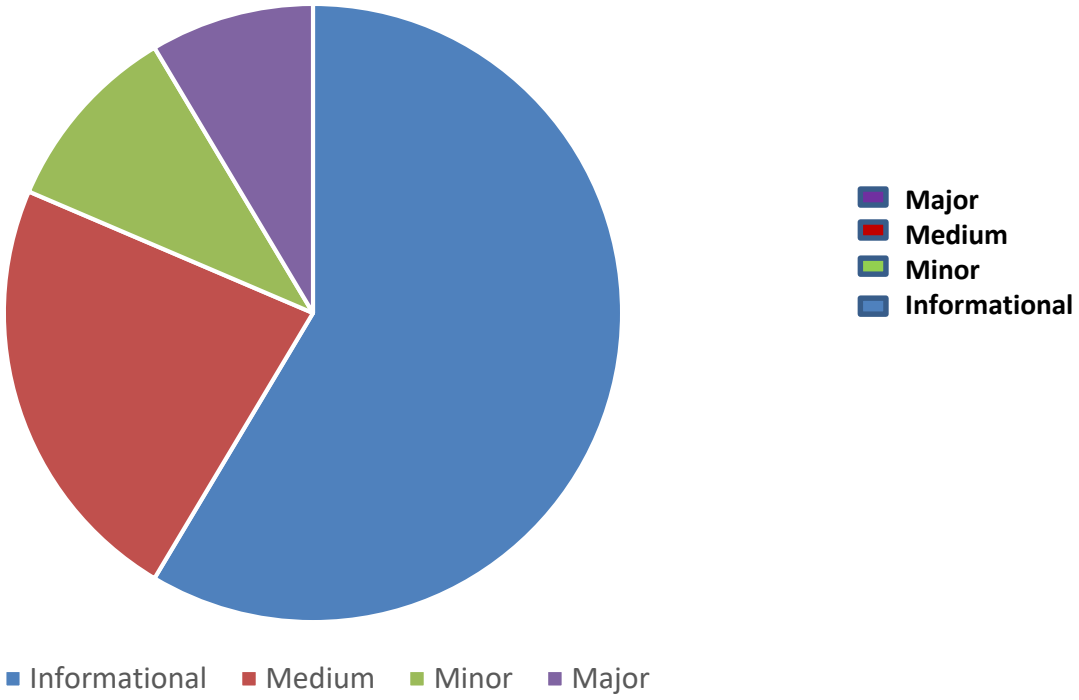
- **Privileged roles can be granted the power to** pause() **the contract in case of an external attack.**
- **Privileged roles can use functions like**, include(), **and** exclude() **to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.**

Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.

- **The client can lower centralization-related risks by implementing below mentioned practices:**
- **Privileged role's private key must be carefully secured to avoid any potential hack.**
- **Privileged role should be shared by multi-signature (multi-sig) wallets.**
- **Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.**
- **Renouncing the contract ownership, and privileged roles.**
- **Remove functions with elevated centralization risk.**

Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.

Finding Summary



Status Icon Definitions

✅	Resolved	🚧	In Progress	ℹ️	Ignored (pro)
❌	Not Resolved	❓	Incorrect	🚫	Ignored (con)

AUTOMATED ANALYSIS

Symbol	Definition
🛑	**Function modifies state**
💵	**Function is payable**
🔒	**Function is internal**
🔐	**Function is private**
❗	**Function is important**

| **ANCORA** | Interface | |||

| └ | totalSupply | External 🔒 | ❗ |NO🔒 |

| └ | decimals | External 🔒 | ❗ |NO🔒 |

| └ | symbol | External 🔒 | ❗ |NO🔒 |

| └ | name | External 🔒 | ❗ |NO🔒 |

| └ | getOwner | External 🔒 | |NO🔒 |

| └ | balanceOf | External 🔒 | ❗ |NO🔒 |

| └ | transfer | External 🔒 | " ❗ 🛑 |NO🔒 |

| └ | allowance | External 🔒 | ❗ |NO🔒 |

| └ | approve | External 🔒 | " ❗ 🛑 |NO🔒 |

| └ | transferFrom | External 🔒 | " |NO🔒 |

||||||

| **IFactoryV2** | Interface | |||

| └ | getPair | External 🔒 | |NO🔒 |

| └ | createPair | External 🔒 | " |NO🔒 |

||||||

| **IV2Pair** | Interface | |||

| └ | factory | External 🔒 | |NO🔒 |

| └ | getReserves | External 🔒 | |NO🔒 |

| └ | sync | External 🔒 | " |NO🔒 |

|||||||

| **IRouter01** | Interface | |||

| └ | factory | External 🔑 | |NO🔑 |

| └ | ETH | External 🔑 | |NO🔑 |

| └ | addLiquidityETH | External 🔑 | # |NO🔑 |

| └ | addLiquidity | External 🔑 | " |NO🔑 |

| └ | swapExactETHForTokens | External 🔑 | # |NO🔑 |

| └ | getAmountsOut | External 🔑 | |NO🔑 |

| └ | getAmountsIn | External 🔑 | |NO🔑 |

|||||||

| **IRouter02** | Interface | IRouter01 |||

| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External 🔑 | " |NO🔑 |

| └ | swapExactETHForTokensSupportingFeeOnTransferTokens | External 🔑 | # |NO🔑 |

| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External 🔑 | " ❗ 🛑 |NO🔑 |

| └ | swapExactTokensForTokens | External 🔑 | " |NO🔑 |

|||||||

| **Protections** | Interface | |||

| └ | checkUser | External 🔑 | " ❗ 🛑 |NO🔑 |

| └ | setLaunch | External 🔑 | " |NO🔑 |

| └ | setLpPair | External 🔑 | " |NO🔑 |

| └ | ACR | External 🔑 | " |NO🔑 |

| └ | removeSniper | External 🔑 | " |NO🔑 |

|||||||

| **Cashier** | Interface | |||

| └ | setRewardsProperties | External 🔑 | " |NO🔑 |

| └ | tally | External 🔑 | " |NO🔑 |

| └ | load | External 🔑 | # |NO🔑 |

| └ | cashout | External 🔑 | " |NO🔑 |

| └ | giveMeWelfarePlease | External 🔑 | " |NO🔑 |

| └ | getTotalDistributed | External 🔑 | |NO🔑 |

| └ | getUserInfo | External 🔑 | |NO🔑 |

| └ | getUserRealizedRewards | External 🔑 | |NO🔑 |

| └ | getPendingRewards | External 🔒 | | |NO🔒 |

| └ | initialize | External 🔒 | " |NO🔒 |

| └ | getCurrentReward | External 🔒 | | |NO🔒 |

||||||

| **SOL** | Implementation | SafeMath |||

| └ | <Constructor> | Public 🔒 | # |NO🔒 |

| └ | transferOwner | External 🔒 | " | onlyOwner |

| └ | renounceOwnership | External 🔒 | " | NO❗|

| └ | setOperator | Public 🔒 | " |NO🔒 |

| └ | renounceOriginalDeployer | External 🔒 | " |NO🔒 |

| └ | <Receive Ether> | External 🔒 | # |NO🔒 |

| └ | totalSupply | External 🔒 | |NO🔒 |

| └ | decimals | External 🔒 | |NO🔒 |

| └ | symbol | External 🔒 | |NO🔒 |

| └ | name | External 🔒 | |NO🔒 |

| └ | getOwner | External 🔒 | ❗ |NO🔒 |

| └ | balanceOf | Public 🔒 | ❗ |NO🔒 |

| └ | allowance | External 🔒 | ❗ |NO🔒 |

| └ | approve | External 🔒 | " ❗ 🛑 |NO🔒 |

| └ | _approve | Internal $ | " | |

| └ | approveContractContingency | Public 🔒 | " ❗ 🛑 | onlyOwner |

| └ | transfer | External 🔒 | " |NO🔒 |

| └ | transferFrom | External 🔒 | " |NO🔒 |

| └ | setNewRouter | External 🔒 | " | onlyOwner |

| └ | setLpPair | External 🔒 | " | onlyOwner |

| └ | setInitializers | External 🔒 | " | onlyOwner |

| └ | isExcludedFromFees | External 🔒 | |NO🔒 |

| └ | isExcludedFromDividends | External 🔒 | |NO🔒 |

| └ | isExcludedFromProtection | External 🔒 | |NO🔒 |

| └ | setDividendExcluded | Public 🔒 | " | onlyOwner |

| └ | setExcludedFromFees | Public 🔒 | " | onlyOwner |

AUDIT SCOPE | ANCORA FINANCE

ID	Repo	Comment	File	SHM321 Checksum
ABY	contracts/ancorafinance/Ancora-v4-core	cC512486	Factory.Sol	6788099YIRHVSK853PKFMGHEF44309200KDHFCBUGIJN
ABI	contracts/ancorafinance/Ancora-v4-core	cC512486	Pool.sol	347520JHDB7549H22H3BVDIOETYUHF009JBIKBDI33BJ4
ABW	contracts/ancorafinance/Ancora-v4-core	cC512486	Pool.sol	1988Y73HUGFDINN353840NFMTEJER73649RGFIMDIDH
ABG	contracts/ancorafinance/Ancora-v4-core	cC512486	BitMath.sol	4438648TEOHBF6378309EHROECNEPOEJDNETE8EYEU3
ABL	contracts/ancorafinance/Ancora-v4-core	cC512486	Factory.sol	66390028765RVNKDBYFTGW553T2KOEHIUUJJIJE
ABA	contracts/ancorafinance/Ancora-v4-core	cC512486	Router.sol	09825539BDYG543DVNKOMIKEBYR JUFHHFHJFIE333222
ABJ	contracts/ancorafinance/Ancora-v4-core	cC512486	BitMath.sol	8654RJVT3DWI865YK26437903JJDGGDHGWY6E
ABE	contracts/ancorafinance/Ancora-v4-core	cC512486	Position.sol	7763888636TGYGFFTFHBETT66TFTCTVYBHBYT
ABP	contracts/ancorafinance/Ancora-smart-order-router/	cC512486	Zapper.sol	88530486494YRHFTEICBGEIEGWTWYWUHEJEHEIE33U3
ABM	contracts/ancorafinance/Ancora-smart-order-router/	cC512486	Position.sol	120987 3KHJLKJNFJHGE98763990029774BCUHHDUU239
ABV	contracts/ancorafinance/Ancora-smart-order-router/	cC512486	SafeCast.sol	23456UGFYUHE98756EFHJHE7654ESDFGHGERTYUJ3897
ABQ	contracts/ancorafinance/Ancora-smart-order-router/	cC512486	Presale.sol	37889UHBIONEO7TYRDFGVBN5678939IJWSFVDYUHDCI
ABS	contracts/ancorafinance/Ancora-smart-order-router/	cC512486	SafeCast.sol	678903098TFHJKFCPOIUGFGHJKE9865ERGBEIVBHE8767
ABR	contracts/ancorafinance/Ancora-smart-order-router/	cC512480	Router.Sol	98765SDFGBNFCOI56789UIYHGGHEJDIUYTRDCVBN3459

Vulnerability Run check

risk detection

Contract source code verified

This token contract is open source, see the contract code for details. Token contracts that do not provide source code are likely to have malicious functions to defraud users of assets.

No bonus issue

Additional issuance functions are transparent or non-existent. Hidden minting may increase the number of tokens in circulation and affect the price of tokens.

Owner cannot change balance

The contract owner does not have the right to modify the token balance of other addresses.

no agency

There is no proxy in the contract. A proxy contract means that the contract owner can modify the functionality of the token and possibly affect the price.

Contract permissions cannot be regained (false abandonment)

If this function exists, it is possible for the project owner to regain ownership even if they abandon it.

Pixiu risk

This doesn't seem to be Pixiu

We did not find any code preventing the token sale.

no anti whale

There is no limit to the number of token transactions. The number of fraudulent token transactions may be limited (Pixiu risk).

no whitelist feature

Discover whitelist functions

No whitelist function

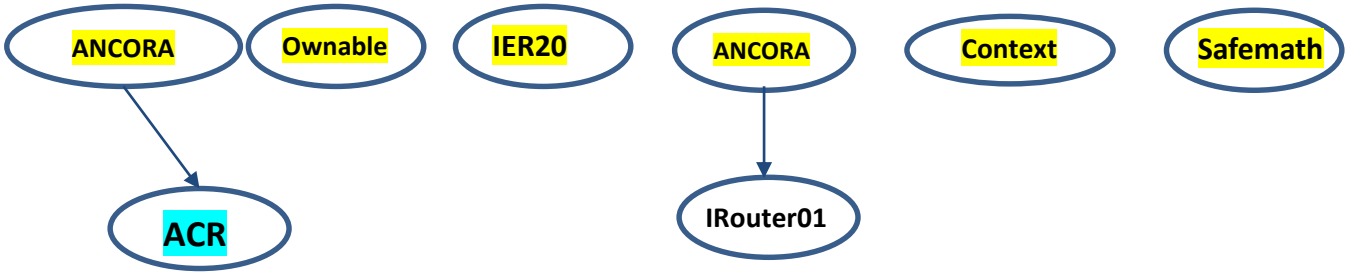
Whitelist function found

The token contract does not have a transaction cooling function. If there is a transaction cooling function, users will not be able to sell tokens within a certain period of time or generate blocks after purchase.

no blacklist function

Does not include whitelist functionality.

INHERITANCE GRAPH



Identifier	Definition	Severity
CEN-12	Centralization privileges of ANCORA FINANCE	Medium #

Vulnerability 0 : No important security issue detected.
Threat level: Low

Home | Factory.sol | Router.sol

```
contract LineaSwapRouter is ILineaSwapRouter {
    address public immutable factory;
    address public immutable WETH;

    modifier ensure(uint deadline) {
        require(deadline >= block.timestamp, 'LineaSwapRouter: EXPIRED');
        _;
    }
    struct User {
        address ref;
        address[] listAddress;
    }
    mapping(address => User) public dataRef;

    constructor(address _factory, address _WETH) {
        factory = _factory;
        WETH = _WETH;
    }

    receive() external payable {
        assert(msg.sender == WETH); // only accept ETH via fallback from the WETH contract
    }
```

STV-03 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Status Mathematical Operations	**Minor**	Factory.Sol/7--12	**INFORMATIONAL**

Description

Mapping State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
contract LineaSwapFactory is ILineaSwapFactory {
  address public feeTo;
  address public feeToSetter;

  mapping(address => mapping(address => address)) public getPair;
  address[] public allPairs;
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the **Mapping** value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

FZT-03 POSSIBLE OVERFLOW

Category	Severity	Location	Status
Inconsistency	**Informational**	Router.Sol/10--21	Acknowledged

Description

In **updateForaddress** , the following equation is used inside an unchecked block

```
   contract LineaSwapRouter is ILineaSwapRouter {
address public immutable factory;
address public immutable WETH;

modifier ensure(uint deadline) {
    require(deadline >= block.timestamp, 'LineaSwapRouter: EXPIRED');
    _;
}
struct User {
    address ref;
    address[] listAddress;
}
```

The function **address** () does not have the override specifier. It should be noted that since price0 > a function that overrides only a single interface function does not require the override specifier (see doc). However, all other instances of this in the code base contain the override specifier.

Recommendation

We recommend either checking for overflow in this case, or ensuring that the PairsIn is close enough it will never cause an overflow.

General Detectors

Incorrect Solidity Version

This contract uses an unconventional or very old version of Solidity.

Attention Required

Public Functions Should be Declared External

Some functions in this contract should be declared as external in order to save gas.

Attention Required

State Variables Should be Declared Constant

Some state variables in this contract should be declared as constant

Attention Required

- No vulnerable withdrawal functions found
- No reentrancy risk found
- No locks detected
- Verified source code found
- No mintable risks found
- Users can always transfer their tokens
- Contract cannot be upgraded
- Wallets cannot be blacklisted from transfering the token
- No transfer fees found
- Token can be sold through regular AMMs
- No transfer limits found
- No ERC20 approval vulnerability found
- Contract owner cannot abuse ERC20 approvals
- No ERC20 interface errors found
- No blocking loops found
- No centralized balance controls found
- No transfer cooldown times found
- No approval restrictions found
- No external calls detected
- No dumping risks found
- No compiler version inconsistencies found
- No unchecked call responses found
- No vulnerable self-destruct functions found
- No assertion vulnerabilities found
- No old solidity code found
- No external delegated calls found
- No external call dependency found
- No vulnerable authentication calls found
- No invalid character typos found
- No RTL characters found
- No dead code found
- No risky data allocation found
- No uninitialized state variables found
- No uninitialized storage variables found
- No vulnerable initialization functions found
- No risky data handling found
- No number accuracy bug found
- No out-of-range number vulnerability found

Vulnerability Scan

REENTRANCY

No reentrancy risk found

Severity	Major
Confidence Parameter	Certain

Vulnerability Description

LIQUIDITY: Any token can Remove its Liquidity by a private wallet or contract. (This is Essentially normal for most contracts)

Scanning Line:

```
  function removeLiquidity(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline,
        address ref
    ) public override ensure(deadline) returns (uint amountA, uint amountB) {
        address pair = LineaSwapLibrary.pairFor(factory, tokenA, tokenB);
        ILineaSwapPair(pair).transferFrom(msg.sender, pair, liquidity); // send
liquidity to pair
        (uint amount0, uint amount1) = ILineaSwapPair(pair).burn(to);
        (address token0, ) = LineaSwapLibrary.sortTokens(tokenA, tokenB);
        (amountA, amountB) = tokenA == token0 ? (amount0, amount1) : (amount1,
amount0);
        require(amountA >= amountAMin, 'LineaSwapRouter: INSUFFICIENT_A_AMOUNT');
        require(amountB >= amountBMin, 'LineaSwapRouter: INSUFFICIENT_B_AMOUNT');
        // handle ref address
        if (ref != address(0)) handleRef(msg.sender, ref);
    }
```

Repository:

https://github.com/ancorafinance/Ancora-v4-core

All Audited Files

Router.sol
Factory.sol

Contract Creator

https://lineascan.build/address/0xdff276be4ac2bbeb8a6a49225a33886c3787be3f

Creator Tnx Hash

0xc7cb5257e30c888bbab9f6a67b25a7201bbe08143cc210dea9fef4b45e8d50b1

Contracts:

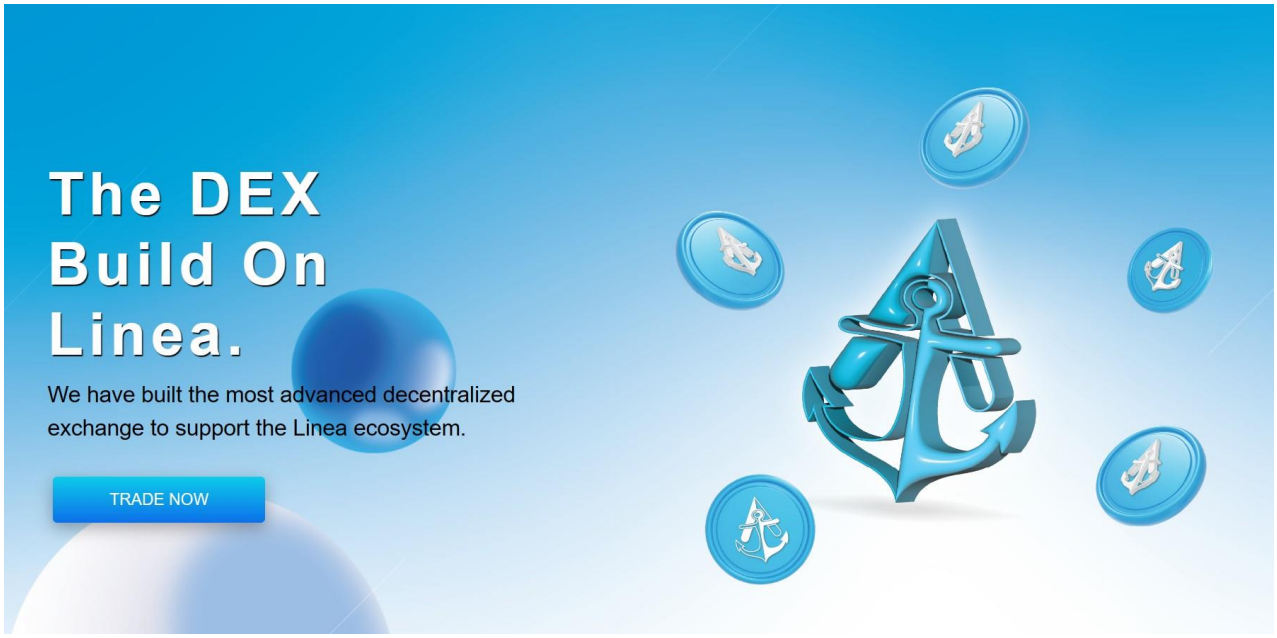
Contract:
Router: 0xa043BfFcaA9Ebaa6708FcbFa4909B100Af47Fd15
Factory: 0xE7aC188E018f954A83c157ac686De7F66e819a51

MANUAL REVIEW

ANCORA FINANCE: is a community-driven organization built to solve what might be called the "liquidity problem." One could define this problem as the inability of disparate forms of liquidity to connect with markets in a decentralized way, and vice versa.
While other solutions provide incrementally progressive advances toward solving the problem of liquidity, Ancora Finance progress is intended to create a broader range of network effects. Rather than limiting itself to a single solution, Ancora Finance intertwines many decentralized markets and instruments.

Project: ANCORA FINANCE
Ticker: ACR
Chain/Standard: Linea Network

Outstanding Features of ANCORA FINANCE Launching On Linea Network



ISSUES CHECKING STATUS

	Issue Description	Checking Status
1.	Compiler errors.	PASSED
2.	Race Conditions and reentrancy. Cross-Function Race Conditions.	PASSED
3.	Possible Delay In Data Delivery.	PASSED
4.	Oracle calls.	PASSED
5.	Front Running.	PASSED
6.	Sol. Dependency.	PASSED
7.	Integer Overflow And Underflow.	PASSED
8.	DoS with Revert.	PASSED
9.	Dos With Block Gas Limit.	PASSED
10.	Methods execution permissions.	PASSED
11.	Economy Model of the contract.	PASSED
12.	The Impact Of Exchange Rate On the solidity Logic.	PASSED
13.	Private use data leaks.	PASSED
14.	Malicious Event log.	PASSED
15.	Scoping and Declarations.	PASSED
16.	Uninitialized storage pointers.	PASSED
17.	Arithmetic accuracy.	PASSED
18.	Design Logic.	PASSED
19.	Cross-Function race Conditions	PASSED
20.	Save Upon solidity contract Implementation and Usage.	PASSED
21.	Fallback Function Security	PASSED

AUDIT RESULT

PASSED

Identifier	Definition	Severity
TEN-02	Transfers User's Tokens	Minor

```
) private returns (uint amountA, uint amountB, address pair) {
    // create the pair if it doesn't exist yet
    if (ILineaSwapFactory(factory).getPair(tokenA, tokenB) == address(0)) {
        ILineaSwapFactory(factory).createPair(tokenA, tokenB);
    }
    pair = LineaSwapLibrary.pairFor(factory, tokenA, tokenB);

    (uint reserveA, uint reserveB) = LineaSwapLibrary.getReserves(factory, tokenA, tokenB);
    if (reserveA == 0 && reserveB == 0) {
        (amountA, amountB) = (amountADesired, amountBDesired);
    } else {
        uint amountBOptimal = LineaSwapLibrary.quote(amountADesired, reserveA, reserveB);
        if (amountBOptimal <= amountBDesired) {
            require(amountBOptimal >= amountBMin, 'LineaSwapRouter: INSUFFICIENT_B_AMOUNT');
            (amountA, amountB) = (amountADesired, amountBOptimal);
        } else {
            uint amountAOptimal = LineaSwapLibrary.quote(amountBDesired, reserveB, reserveA);
            assert(amountAOptimal <= amountADesired);
            require(amountAOptimal >= amountAMin, 'LineaSwapRouter: INSUFFICIENT_A_AMOUNT');
            (amountA, amountB) = (amountAOptimal, amountBDesired);
        }
    }
```

Alleviation:

Any user has the authority to transfer the balance of a user's address if the user has granted allowance. The contract does not subtract the allowance in the mstore(add) method, as a result, the transfer can be repeated until the user's balance go to zero.

RECOMMENDATION

The team is advised to modify the allowance in the mstore(add) method

OPTIMIZATIONS | ANCORA FINANCE

ID	Title	Category	Status
STV	Logarithm Refinement Optimization	Gas Optimization	Acknowledged
SOP	Checks Can Be Performed Earlier	Gas Optimization	Acknowledged
SDP	Unnecessary Use Of SafeMath	Gas Optimization	Acknowledged
SWY	Struct Optimization	Gas Optimization	Acknowledged
SGT	Unused State Variable	Gas Optimization	Acknowledged

RECOMMENDATION

Deployer and/or contract owner private keys are secured carefully.

Please refer to PAGE-09 CENTRALIZED PRIVILEGES for a detailed understanding.

ALLEVIATION

ANCORA FINANCE project team understands the centralization risk. Some functions are provided privileged access to ensure a good runtime behaviour in the project

Identifier	Definition	Severity
TOB-12	Third Party Dependencies	Minor

A smart contract is interacting with third-party protocols e.g., Uniswap, Pancakeswap router, cashier contract,

And protections contract. The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and exploited. Moreover, upgrades in third parties can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

RECOMMENDATION

Inspect and validate third party dependencies regularly, and mitigate severe impacts whenever necessary.

CERTIFICATE BY VITAL BLOCK SECURITY

DISCLAIMERS

Vital Block Security provides the easy-to-understand audit of Solidity, Move, and Raw source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model, or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way

to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

TECHNICAL DISCLAIMER

ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, VITAL BLOCK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, VITAL BLOCK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, VITAL BLOCK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT'S OR ANY OTHER INDIVIDUAL'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

TIMELINESS OF CONTENT

The content contained in this audit report is subject to change without any prior notice. Vital Block does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.

LINKS TO OTHER WEBSITES

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than Vital Block. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites and social accounts owners. You agree that Vital block Security is not responsible for the content or operation of such websites and social accounts and that Vital Block shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

ABOUT VITAL BLOCK

Vital Block provides intelligent blockchain Security Solutions. We provide solidity and Raw Code Review, testing, and auditing services. We have Partnered with 15+ Crypto Launchpads, audited 50+ smart contracts, and analyzed 200,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Aptos, Oasis, etc.

Vital Block is Dedicated to Making Defi & Web3 A Safer Place. We are Powered by Security engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 5 core members, and 4+ casual contributors.

Website: https://www.Vitalblock.org

Email: info@vitalblock.org

GitHub: https://github.com/vital-block

Telegram (Engineering): https://t.me/vital_block

Telegram (Onboarding): https://t.me/vitalblock_cmo

vital-block

info@vitalblock.org

www.Vitalblock.org

Vital Block Dedicated to securing Public and Private Blockchain Ecosystem