VITAL BLOCK

# Security Assessment

# Loozr

**Verified On Oct 24th, 2024**


@Vital-Block

@VB_Audit

info@vitalblock.org

www.vitalblock.org


PREPARED FOR:
Loozr

# TABLE OF CONTENTS

# INTRODUCTION

| | |
|---|---|
| **Auditing Firm** | VITAL BLOCK SECURITY |
| **Client Firm** | Loozr |
| **Methodology** | Automated Analysis, Manual Code Review |
| **Language** | Solidity |
| **Contract** | TOKEN.MO |
| **Network** | NEAR PROTOCOL |
| **Centralization** | Active ownership |
| **Website** | https://loozr.io/ |
| **Docs** | https://docs.loozr.io/ |
| **Twitter** | https://twitter.com/officialloozr |
| **GitHub** | https://github.com/Loozr-Protocol |
| **Prelim Report Date** | Oct 17th , 2024 |
| **Final Report Date** | July 24th 2024 |

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

## Document Properties

| | |
|---|---|
| **Client** | LOOZR |
| **Title** | Smart Contract Audit Report |
| **Target** | LOOZR |
| **Audit Version** | 1.0 |
| **Author** | Akhmetshin Marat |
| **Auditors** | Akhmetshin Marat, James BK, Benny Matin |
| **Reviewed by** | Dima Meru |
| **Approved by** | Prince Mitchell |
| **Classification** | Public |

## Version Info

| Version | Date | Author(s) | Description |
|---|---|---|---|
| 1.0 | October 17th , 2024 | James BK | Final Released |
| 1.0-AP | October 24th , 2024 | Benny Matin | Release Candidate |

## Contact

For more information about this document and its contents, please contact Vital Block Security Inc.



| | |
|---|---|
| **Name** | Akhmetshin Marat |
| **Phone** | +44 7944 248057 |
| **Email** | info@vitalblock.org |

In the following, we show the specific pull request and the commit hash value used in this audit.

- https://github.com/Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend (LOR-79661)
- https://github.com/Loozr-Protocol/lzr-dfinity-point-coin/blob/main/src/lzr_point_coin_backend/main.mo (LOZU544210)

## About Vital Block Security

Vital Block Security provides professional, thorough, fast, and easy-to-understand smart contract security audit. We do in-depth and penetrative static, manual, automated, and intelligent analysis of the smart contract. Some of our automated scans include tools like ConsenSys MythX, Mythril, Slither, Surya. We can audit custom smart contracts, DApps, Rust, NFTs, etc (including the service of smart contract auditing). We are reachable at Telegram (https://t.me/vital_block ), Twitter (http://twitter.com/Vb_Audit ), or Email (info@vitalblock.org).

Table 1.2: Vulnerability Severity Classification

| Impact \ Likelihood | *High* | *Medium* | *Low* |
|---|---|---|---|
| *High* | Critical | High | Medium |
| *Medium* | High | Medium | Low |
| *Low* | Medium | Low | Low |

## Methodology (1)

To standardize the evaluation, we define the following terminology based on the OWASP Risk Rating Methodology [4]:

- Likelihood represents how likely a particular vulnerability is to be uncovered and exploited in the wild;
- Impact measures the technical loss and business damage of a successful attack;
- Severity demonstrates the overall criticality of the risk.

# SCOPE OF WORK

**Vital Block was consulted by LOOZR to conduct the smart contract audit of its. Move source code. The audit scope of work is strictly limited to mentioned .mo file only:**

TOKEN.MO

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 Code Scanned / Audited | |
|---|---|
| Token.mo | |
| Main.mo | |
| Account.mo | |
| Transfer.mo | |
| Types.mo | |
| Utilis.mo | |
| Libs.smo | |
| **Project Name:** | Loozr |

# Table 1.0 The Full Audit Checklist

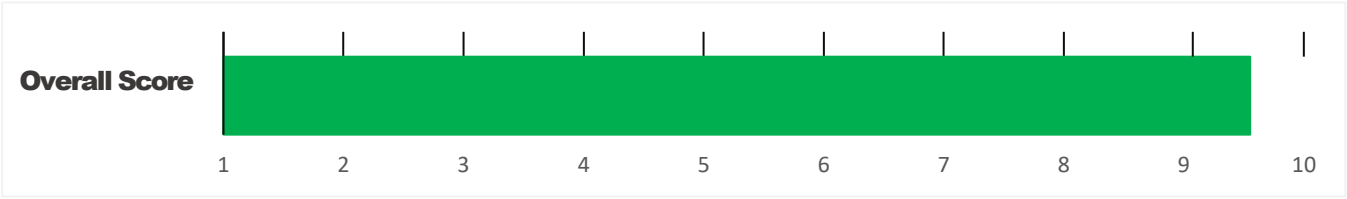
| Category | Checklist Items |
|---|---|
| **Basic Coding Bugs** | Constructor Mismatch |
| | Ownership Takeover |
| | Redundant Fallback Function |
| | Overflows & Underflows |
| | Reentrancy |
| | Money-Giving Bug |
| | Blackhole |
| | Unauthorized Self-Destruct |
| | Revert DoS |
| | Unchecked External Call |
| | Gasless Send |
| | Send Instead Of Transfer |
| | Costly Loop |
| | (Unsafe) Use Of Untrusted Libraries |
| | (Unsafe) Use Of Predictable Variables |
| | Transaction Ordering Dependence |
| | Deprecated Uses |
| **Semantic Consistency Checks** | Semantic Consistency Checks |
| **Advanced DeFi Scrutiny** | Business Logics Review |
| | Functionality Checks |
| | Authentication Management |
| | Access Control & Authorization |
| | Oracle Security |
| | Digital Asset Escrow |
| | Kill-Switch Mechanism |
| | Operation Trails & Event Generation |
| | ERC20 Idiosyncrasies Handling |
| | Frontend-Contract Integration |
| | Deployment Consistency |
| | Holistic Risk Management |
| **Additional Recommendations** | Avoiding Use of Variadic Byte Array |
| | Using Fixed Compiler Version |
| | Making Visibility Level Explicit |
| | Making Type Inference Explicit |
| | Adhering To Function Declaration Strictly |
| | Following Other Best Practices |

# EXECUTIVE SUMMARY

**Vital Block Security has performed the automated and manual analysis of the LOOZR .Mo 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 | 2 | 3 | 1 |
| Resolved | 0 | 0 | 0 | 0 | 0 |
| | | | | | |
| Noteworthy OnlyOwner Privileges | Set Taxes and Ratios, Airdrop, Set Protection Settings, Set Reward Properties, Set Reflector Settings, Set Swap Settings, Set Pair and Router | | | | |

**FENTURE FINANCE Smart contract has achieved the following score: %94.5**




ℹ **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.**

ℹ **Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.**

# 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<br>○ Access Control and Authorization<br>○ Assets Manipulation<br>○ Ownership Control<br>○ Liquidity Access<br>○ Stop and Pause Trading<br>○ Ownable Library Verification |

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

ℹ️ **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.**

## Key Findings

Overall, these contracts are well-designed and engineered, though the implementation can be improved by resolving the identified issues (shown in Table 2.1), 0 medium-severity vulnerabilities, 3 low-severity vulnerabilities, and 1 informational recommen- dations.

**Table 2.1: Key LOOZR Audit Findings**

| ID | Severity | Title | Category | Status |
|---|---|---|---|---|
| LZR-01 | low | In UncheckedForTransfer , the following equation is used inside an unchecked block | Business Logic | Acknowledged |
| LZR-02 | Informational | In updateForMinter , the following equation is used inside an unchecked block | Mathematical Operations | Acknowledged |

Beside the identified issues, we emphasize that for any user-facing applications and services, it is always important to develop necessary risk-control mechanisms and make contingency plans, which may need to be exercised before the mainnet deployment. The risk-control mechanisms should kick in at the very moment when the contracts are being deployed on mainnet. Please refer to page 10 for details.

# AUDIT SCOPE | LOOZR

| ID | Repo | Comment | File | SHM211 Checksum |
|---|---|---|---|---|
| LZR | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | Cc51d21 | Token.mo | 3426670yube0fd50f8632d8433cccc9db6f4b39f9e566d1fa78de54b84bddr54 |
| LZR | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D32 | Token.mo | 8oippkjjjk96be0fd50f8632d8433cccc9db6f4b39f9e566d1yhhg8765fffckiuybb |
| LZR | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D42 | Token.mo | 4311280uj908766362fvyga98jdkl88648yhfbqt37409owehbgwhuyyyg223738 |
| LZR | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D44 | Token.mo | 75uuyriy399787390uhbiiuhghhdg7guu30oi7799u9359ydfgdgygeigi3ioueyy78 |
| TTR | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D46 | Account.mo | 450efgywqutfeuh87872t15378837983639293763hhegetgjfwjk89336668862 |
| TOP | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D48 | Account.mo | 16363ttebnve88329973mvvdsggct478153ytgdfdxy792635fgdjgi1900990908 |
| TDP | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D49 | Account.mo | 32156990327hudbinnjntr6729dchjld0993ytyy3vq63235727879889073 |
| TWY | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D53 | Transfer.mo | bff08692343d1cc36eaf196046d7a528d153abd55ba20e82f1d57c22fcd92675 |
| TKB | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D62 | Transfer.mo | 8448b3af42497f5f74e53424ee3e6c551f51356945108d22a893d608a7990542 |
| TXY | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D63 | Transfer.mo | 5c86aa1dd3889db5fcd17a80214b226fc784f268ab9db82df97c1d2459467831 |
| TCB | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D63 | Types.mo | b8244da33db171e5533d77bef4a35703df1de2cebea5f35cb38ce6a26c778cf1 |
| TWO | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D67 | Types.mo | 3d408b8f2cc56f9699a402b5151de90671de089c3007afc9e4fc867c04152e7c |
| TGT | Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend/ICRC1 | cC51D68 | Utils.mo | 9d751621c3501102e4b50005ca3314ec6e04e6ff8bbb30852d1c7edfff3f8cef |

# AUTOMATED ANALYSIS

| Symbol | Definition |
|---|---|
| | **Function modifies state** |
| | **Function is payable** |
| | **Function is internal** |
| | **Function is private** |
| | **Function is important** |

| **LOOZR ** | 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❗ |
| └ | NEAR | External ❗ | NO❗ |
| └ | addLiquidityNEAR | External ❗ | # NO❗ |
| └ | addLiquidity | External ❗ | " NO❗ |
| └ | swapExacNEARForTokens | External ❗ | # NO❗ |
| └ | getAmountsOut | External ❗ | NO❗ |
| └ | getAmountsIn | External ❗ | NO❗ |
| **IRouter02** | Interface | IRouter01 | |
| └ | swapExactTokensForNEARSupportingFeeOnTransferTokens | External ❗ | " NO❗ |
| └ | swapExactNEARForTokensSupportingFeeOnTransferTokens | External ❗ | # NO❗ |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗ | " ! 🛑 NO❗ |
| └ | swapExactTokensForTokens | External ❗ | " NO❗ |
| **Protections** | Interface | | |
| └ | checkUser | External ❗ | " ! 🛑 NO❗ |
| └ | setLaunch | External ❗ | " NO❗ |
| └ | setLpPair | External ❗ | " NO❗ |
| └ | NEAR | 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❗ |

||||||

| **NEAR** | Implementation | ICRC1 |||

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

| └ | transferOwner | External ❗ | " | | onlyOwner |

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

| └ | setOperator | Public ❗ | " | |NO❗ |

| └ | renounceOriginalDeployer | External ❗ | " | |NO❗ |

| └ | <Receive ETH> | 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 |

## Lzr-01 Key Findings

| Category | Severity ● | Location | Status |
|---|---|---|---|
| **Business Logic** | **Low** | Contract/Transfer.mo 37-42 | Informational |

## Description

In **UncheckedForTransfer ,** the following equation is used inside an unchecked() block

```
public func is_too_old(token : T.TokenData, created_at_time : Nat64) : Bool {
        let { permitted_drift; transaction_window } = token;

        let lower_bound = Time.now() - transaction_window - permitted_drift;
        Nat64.toNat(created_at_time) < lower_bound;
    };
```

**Note** that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the **Loozr** contract.

## Recommendation

Incorporate the following verification within process approve account to confirm that the contract account's associated transfer aligns with the mint for which the confidential transfer approval is sought.

## Lzr-02 POSSIBLE OVERFLOW

| Category | Severity | Location | Status |
|---|---|---|---|
| **Mathematical Operations** | **Informational** | Contract/code/**Main.mo/** | Acknowledged |

## Description

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

```
stable let token = ICRC1.init({
        token_args with minting_account = Option.get(
            token_args.minting_account,
            minting_account,
        );
    });
```

**Minter** can not issue more **Loozr** tokens indefinitely.
Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the **Loozr** contract.

## Recommendation

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

## OPTIMIZATIONS | Loozr

| ID | Title | Category | Status |
|---|---|---|---|
| FTV | Logarithm Refinement Optimization | Gas Optimization | Acknowledged |
| FOP | Checks Can Be Performed Earlier | Gas Optimization | Acknowledged |
| FDP | Unnecessary Use Of SafeMath | Gas Optimization | Acknowledged |
| FWY | Struct Optimization | Gas Optimization | Acknowledged |
| FGT | Unused State Variable | Gas Optimization | Acknowledged |

## Vulnerability Scan

**REENTRANCY**

No reentrancy risk found

| | |
|---|---|
| Severity | Minor |
| Confidence Parameter | Certain |

## Vulnerability Description

**Not Mintable**: A large amount of this token can not be minted by a private wallet or contract.

## Scanning Line:

```
let token_args : ICRC1.TokenInitArgs = {
        advanced_settings = null;
        decimals = 18;
        fee = 1_000;
        initial_balances = [(
            {
                owner = _owner;
                subaccount = null;
            },
            0,
        )];
        min_burn_amount = 0;
        minting_account = null;
        name = name;
        symbol = symbol;
    };

    stable let token = ICRC1.init({
        token_args with minting_account = Option.get(
            token_args.minting_account,
            minting_account,
        );
    });

    /// Functions for the ICRC1 token standard
    public shared query func icrc1_name() : async Text {
        ICRC1.name(token);
    };
```

| Identifier | Definition | Severity |
| --- | --- | --- |
| CEN-02 | Initial asset distribution | Minor |

```
public shared ({ caller }) func mint(account : Principal, amount :
ICRC1.Balance) : async ICRC1.TransferResult {
        let to_account : ICRC1.Account = {
            owner = account;
            subaccount = null;
        };
        let args = {
            to = to_account;
            amount = amount;
            memo = null;
            created_at_time = null;
        };
```

## Description:

Floating point calculations can vary across different architectures.

## Alleviation:

This exhibit was acknowledged and ultimately discarded by the Loozr team due to low severity. We consider the exhibit fully attended to as it doesn't impose any meaningful security concerns.

**RECOMMENDATION**

**Project stakeholders should be consulted during the initial asset distribution process.**

## Repository:

https://github.com/Loozr-Protocol/lzr-dfinity-point-coin/tree/main/src/lzr_point_coin_backend

## Scanned Code Files

Token.mo

Main.mo

Account.mo

Transfer.mo

Types.mo

Utilis.mo

Libs.smo

## Contract Deployed

Contract:
Token.Mo

## Vulnerability Run check

### Risk Analysis

**Contract source code verified**

This token contract is open source. You can check the contract code for details. Unsourced token contracts are likely to have malicious functions to defraud their users of their assets.

**No Proxy**

There is no proxy in the contract. The proxy contract means contract owner can modify the function of the token and possibly effect the price.

**No mint function**

Mint function is transparent or non-existent. Hidden mint functions may increase the amount of tokens in circulation and effect the price of the token.

**No function to retrieve ownership**

If this function exists, it is possible for the project owner to regain ownership even after relinquishing it.

**Owner cant change balance**

The contract owner does not have the authority to modify the balance of tokens at other addresses.

### Honeypot Risk

**This does not appear to be a honeypot**

We are not aware of any code that prevents the sale of tokens.

**No trading cooldown**

The token contract has no trading cooldown function.If there is a trading cooldown function, the user will not be able to sell the token within a certain time or block after buying.

**No Anti Whale**

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

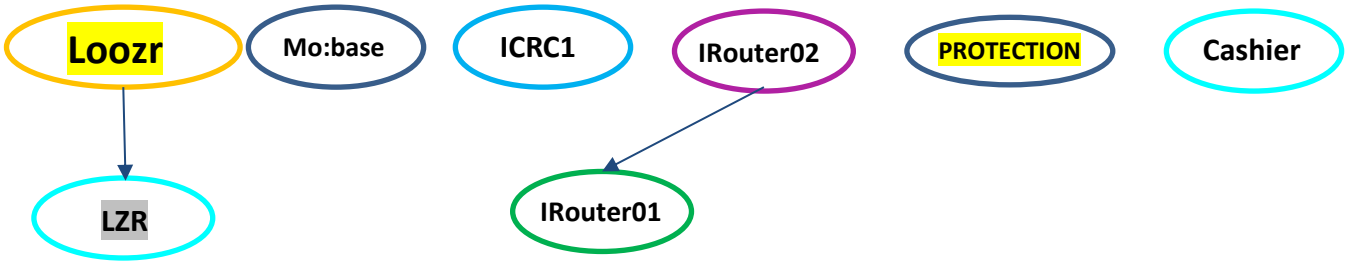
**No blacklist function**

No blacklist function is included.

**No whitelist function**

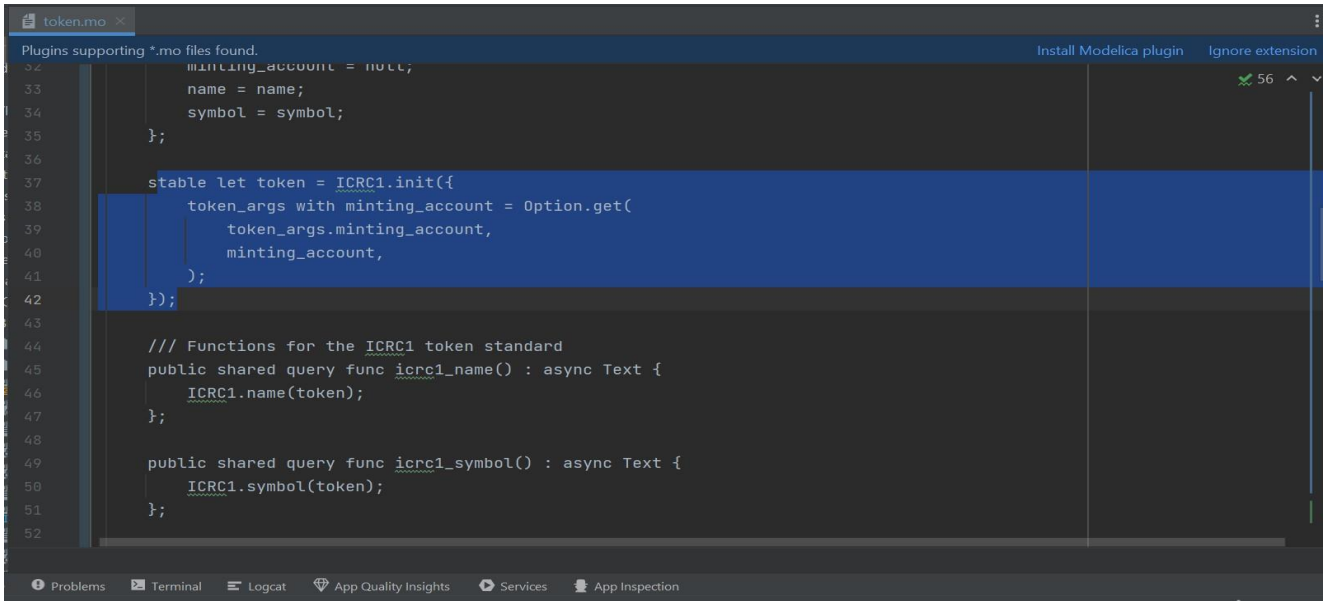
Whitelist function found

# INHERITANCE GRAPH




| Identifier | Definition | TIVEL FINANCE |
|---|---|---|
| CEN-12 | Centralization privileges of LOOZR | Medium # 🟡 |

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



# MANUAL REVIEW

**Loozr introduces an exciting music game on Telegram, where participants mine community GENRE tokens and partake in multiple EPOCH rewards, paving the way for the platform's Music-SocialFi and crowd-investment launch in Q1 2025.**

This groundbreaking initiative combines gamification, music culture, community-centric approach, and decentralized finance (DeFi), offering a dynamic platform where music culture thrives, and fans and artists collaborate in an interactive ecosystem..

**TOKEN NAME:** LOOZR
**Ticker**: LZR

**Network:** NEAR PROTOCOL

**Token Type:** Utility

**Total Supply: .** 160,000,000

**The LOOZR Platform Is Launching On NEAR Protocol**

# 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. | Mo 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 Move 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 Move contract Implementation and Usage. | PASSED |
| 21. | Fallback Function Security | PASSED |

## AUDIT RESULT

**PASSED**

SMART CONTRACT AUDIT OF LOOZR

| Identifier | Definition | Severity |
| --- | --- | --- |
| CEN-02 | Initial asset distribution | Minor |

All of the initially minted assets are sent to the contract deployer when deploying the contract. This can be an issue as the deployer and/or contract owner can distribute tokens without consulting the community.

```
let token_args : ICRC1.TokenInitArgs = {
        advanced_settings = null;
        decimals = 18;
        fee = 1_000;
        initial_balances = [(
            {
                owner = _owner;
                subaccount = null;
            },
```

## RECOMMENDATION

Project stakeholders should be consulted during the initial asset distribution process.

## RECOMMENDATION

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

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

## ALLEVIATION

The LOOZR project team understands the centralization risk. Some functions are provided privileged access to ensure a good runtime behavior in the project

| Identifier | Definition | Severity |
| --- | --- | --- |
| COD-10 | Third Party Dependencies | Minor |

Smart contract is interacting with third party protocols e.g., Pancakeswap router, cashier contract, 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.

# DISCLAIMERS

**Vital Block 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://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