Coinsult

Request your audit at coinsult.net

Advanced Manual

Smart Contract Audit

October 23, 2023

Audit requested by

Landrocker Contracts

Audit Summary

Project Name	Landrocker
Website	https://landrocker.io/
Blockchain	Polygon (Mumbai)
Smart Contract Language	Solidity
Audit Method	Static Analysis, Manual Review
Start date of audit	October 23, 2023

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

The audit of the Solidity codebase presented significant challenges due to its disorganized nature and lack of accompanying documentation. The convoluted structure of the code made it increasingly difficult to discern the intended functionality of individual components. Furthermore, the absence of any guiding documentation surrounding the functions made it even more arduous to review and assess the overall quality of the code. Despite these obstacles, the auditing process was diligently carried out to ensure the highest standards of security and stability.

Audit Scope

Source Code

Coinsult was comissioned to do a manual code review by LRT based on the following code:

Access Restriction: 0xB21bb1Ab0012236E3CF889FCcE00a4F3d9aF55c4

LRTDistributor: 0xdF7fE8faec166044819E4E70B983CF0E135016f5

LRT Vesting: 0x6f4592838fcE61ed1eFCf54E3545530a7b0a822C

LRT Pre sale : 0xDCB498eEB98Ad163771416e0DB179e1350475344

Note that we only audited the code available to us on this URL at the time of the audit. If the URL is not from any block explorer (main net), it may be subject to change. Always check the contract address on this audit report and compare it to the token you are doing research for.

Audit Method

Coinsult's manual smart contract audit is an extensive methodical examination and analysis of the smart contract's code that is used to interact with the blockchain. This process is conducted to discover errors, issues and security vulnerabilities in the code in order to suggest improvements and ways to fix them.

Automated Vulnerability Check

Coinsult uses software that checks for common vulnerability issues within smart contracts. We use automated tools that scan the contract for security vulnerabilities such as integer-overflow, integer-underflow, out-of-gas-situations, unchecked transfers, etc.

Manual Code Review

Coinsult's manual code review involves a human looking at source code, line by line, to find vulnerabilities. Manual code review helps to clarify the context of coding decisions. Automated tools are faster but they cannot take the developer's intentions and general business logic into consideration.

Used Tools

- ✓ Slither: Solidity static analysis framework
- ✓ Remix: IDE Developer Tool
- ✓ CWE: Common Weakness Enumeration
- ✓ SWC: Smart Contract Weakness Classification and Test Cases
- ✓ DEX: Testnet Blockchains

Audit Findings

LRT Vesting:

Centralization – Admin of the contract is able to create a vesting plan with arbitrary amounts and unlock time for an arbitrary address. A malicious owner might use this feature to claim excessive amount of tokens from distributor contract.

Input validation – at createVestingPlan, startDate is not validated to be bigger than current time stamp

Logical – createVesting requires _planID to be less than or equal to current plan Id. Current plan ID is not initialized

Logical – setDebt is not skipping revoked vestings. Gas optimization – define lrtDistributor and accessRestriction as immutable

LRT Distributor:

Input validation - Ensure that "_to" is not zero address

Input validation – Ensure that "_to" is not distributor address itself, otherwise tokens will be lost

Code quality – return "true" from "swap" function (returning value of _distribute is ignored)

Gas optimization – define token as immutable

Gas optimization – define accessRestriction as immutable

Events – use "indexed" keyword for below events: • event TokenDistributed(bytes32 poolName, uint256 _amount, address _to); • event TokenSwapped(address _to, uint256 _amount); • event TransferredLiquidity(• bytes32 _fromPoolName, • bytes32 _destPoolName, • uint256 _amount);

Access Control:

All contracts are using a single contract as the the AccessControl contract, owner of this contract is an EOA wallet with the privilege to grant roles to any arbitrary address or revoke roles from any address. A malicious owner might use this privilege to gain access to critical functions of the system such as distributor contract, and be able to claim excessive amount of tokens from there

Disclaimer

The audit of the Solidity codebase presented significant challenges due to its disorganized nature and lack of accompanying documentation. The convoluted structure of the code made it increasingly difficult to discern the intended functionality of individual components. Furthermore, the absence of any guiding documentation surrounding the functions made it even more arduous to review and assess the overall quality of the code. Despite these obstacles, the auditing process was diligently carried out to ensure the highest standards of security and stability.