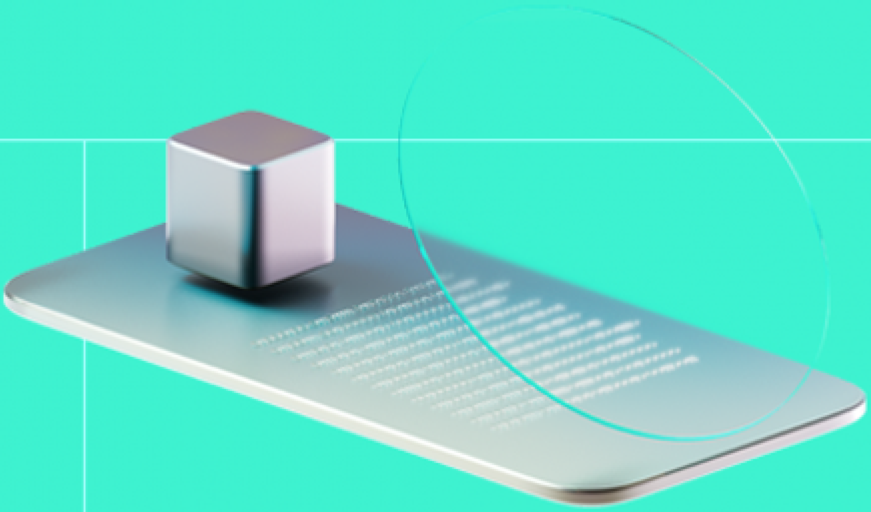
Smart Contract Code Review And Security Analysis Report



Customer: Alcazar

Date: 01/05/2023

We express our gratitude to the Alcazar team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

LEO is a lossless ERC20 token.

Platform: BNB Chain

Language: Solidity

Tags: ERC20

Timeline: 20/04/2023 - 01/05/2023

Methodology: https://hackenio.cc/sc_methodology

Review

Scope

Repository	https://bscscan.com/token/0x56b331c7e3d68306f26e07492125f0faa9d95343
Commit	0x56b331c7e3d68306f26e07492125f0faa9d95343

Audit Summary

10/10	10/10	n/a	3/10
Security score	Code quality score	Test coverage	Documentation quality score

Total 9.3/10

The system users should acknowledge all the risks summed up in the risks section of the report

1	0	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	0
Low	1

Vulnerability	Status
F-2023-1434 - Floating Pragma	Pending Fix

Document

Name	Smart Contract Code Review and Security Analysis Report for Alcazar
Audited By	Hacken
Changelog	01/05/2023 – Initial Review

Table of Contents

System Overview

LEO is a lossless ERC20 token with the following components:

- Context: This abstract contract provides utility functions to access the transaction context, such as the sender of the transaction and the transaction data. It is inherited by the LERC20 contract to make use of these functions.
- Token: LERC20 is an ERC-20 token that supports lossless features. It mints the initial supply during deployment and does not allow further minting. It has the following attributes:
  -Name: Leo.
  -Symbol: LEO.
  -Decimals: 18.
  -Total supply: 1.000.000.000.
  -Recovery Admin: 0xb2a8ea4b7b385746ca4ea9b9a10a0559721680d5.
  -Admin: 0x2ff9d7be466f674c8640466a55ffdd02b3a00864.
  -LossLess: 0xdbb5125ceeaf7233768c84a5df570aeecf0b4634.
  -Timelock Period: 86400 (1 day).
- Lossless: The token interacts with a lossless controller (*ILosslessController*) to help prevent fraudulent transactions. The lossless feature can be turned on or off by the recovery admin.

Privileged roles

- Admin: The admin can be changed by the recovery admin and has the authority to change certain aspects of the lossless controller.
- Recovery Admin: The recovery admin has several privileges, including transferring recovery admin ownership, accepting recovery admin ownership, setting the lossless admin, and managing the state of the lossless feature (turning it on or off).

Executive Summary

This report presents an in-depth analysis and scoring of the customer's smart contract project. Detailed scoring criteria can be referenced in the scoring methodology.

Documentation quality

The total Documentation quality score is **3** out of **10**.

- Functional requirements are not provided.
- Technical description is partially provided.
- Only the Lossless token generator is documented.
- NatSpec is not provided.
- The development environment is not described.

Code quality

The total Code quality score is **10** out of **10**.

- The code follows best practices.
- The token generator from Lossless was used to deploy the code.

Test coverage

The project has less than 250 lines of code. Tests are not required.

Security score

Upon auditing, the code was found to contain **0** critical, **0** high, **0** medium, and **1** low severity issues. Out of these, **0** issues have been addressed and resolved, leading to a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's smart contract yields an overall score of **9.3**. This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

- The *LssController* contract is not within the scope of the audit but has a significant impact on the audited contracts:
  -it can transfer tokens from the user to itself.
  -it has the ability to reject token transfers.
  -it has the ability to reject changing allowances.
- The *LosslessController* address defined in the contract points to the *LosslessControllerV3.sol*, the latest version of the *LosslessController* is V4, the contract uses an older version of the controller.

Findings

Vulnerability Details

F-2023-1434 - Floating Pragma - Low

Description:	The project uses floating pragmas ^0.8.0. This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with. For example, they might be deployed using an outdated pragma version which may include bugs that affect the system negatively.
Status:	Pending Fix

Classification

Impact:	2/5
Likelihood:	1/5
Severity:	Low

Recommendations

Remediation:	Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment. Consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details

Whitepaper	Not provided
Requirements	Not provided
Technical Requirements	Partially provided

Deployed contract

https://bscscan.com/token/0x56b331c7e3d68306f26e07492125f0faa9d95343