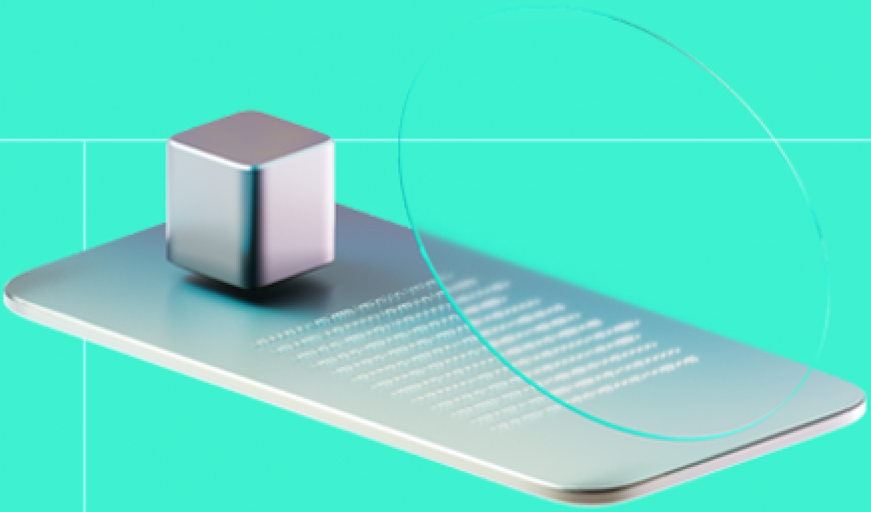
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Codyfight

Date: 24/04/2024

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

Codyfight is a competitive turn-based strategy RPG packed with chess-like depth and ever-evolving battles against AI-controlled NPCs.

Platform: Arbitrum One

Language: Solidity

Tags: LERC20, LERC20Burnable

Timeline: 22/04/2024 - 24/04/2024

Methodology: https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/codyfight/token-contract
Commit	af6ea26

Audit Summary

10/10	10/10	100%	10/10
Security Score	Code quality score	Test coverage	Documentation quality score

Total 10/10

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

0	0	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	0
Low	0

Document

Name	Smart Contract Code Review and Security Analysis Report for Codyfight
Audited By	Eren Gonen
Approved By	Ataberk Yavuzer
Website	https://codyfight.com/
Changelog	22/04/2024 - Preliminary Report
	24/04/2024 - Final Report

Table of Contents

System Overview

The CodyfightToken (CTOK) operates on the Arbitrum blockchain, enhancing Codyfight's gaming economy through asset exchanges and player rewards. It is a LERC-20 Burnable token equipped with lossless functionality to freeze fraudulent transactions and features such as token minting and burning, aimed at reducing transaction costs and incentivizing community participation. The protocol contracts:

CodyfightToken — The official token of Codyfight fully inherits the LERC20Burnable contract, which implements the LERC20 standard with burn and lossless features. It mints the total supply to the owner in the constructor, and additional minting is not allowed.

It has the following attributes:

- **Name:** Codyfight Token
- **Symbol:** CTOK
- **Decimals:** 18
- **Total supply:** 127,00,00,01

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 **10** out of **10**.

- Functional requirements are provided
- Technical description is provided.

Code quality

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

- Best practices are followed.
- The development environment is configured.
- Natspec is sufficient.

Test coverage

Code coverage of the project is **100%** (branch coverage).

- Everything covered with tests.

Security score

Upon auditing, the code was found to contain **0** critical, **0** high, **0** medium, and **0** low severity issues, 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 **10**. This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

- The **CodyfightToken** contract heavily relies on the **LERC20Burnable** contract previously audited by Hacken, is **out of the scope** of this audit.

Findings

Vulnerability Details

Observation Details

F-2024-1481 - Missing Zero Address Validation - Info

Description:

In Solidity, the Ethereum address `0x0000000000000000000000000000000000000000` is known as the "**zero address**". This address has significance because it is the default value for uninitialized address variables and is often used to represent an invalid or non-existent address.

The "**Missing zero address control**" issue arises when a Solidity smart contract does not properly check or prevent interactions with the zero address, leading to unintended behavior.

For instance, a contract might allow tokens to be sent to the zero address without any checks, which essentially burns those tokens as they become irretrievable. While sometimes this is intentional, without proper control or checks, accidental transfers could occur.

Missing check were observed in the following contract:

- `./CodyfightToken.sol: constructor()`

Assets:

- CodyfightToken.sol [https://github.com/codyfight/token-contract]

Status: Fixed

Recommendations

Remediation:

Implement zero address validation for the given parameters. This can be achieved by adding require statements that ensure address parameters are not the zero address.

Remediation(e5134f5): The team implemented zero address validation for the `lossless_` input parameter.

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

Repository	https://github.com/codyfight/token-contract
Commit	af6ea2680266958b23ea300b41e16ac38d42bf3b
Whitepaper	https://codyfight.gitbook.io/white-paper
Requirements	https://github.com/codyfight/token-contract/blob/main/README.md
Technical Requirements	https://github.com/codyfight/token-contract/blob/main/README.md

Contracts in Scope

./contracts/CodyfightToken.sol