



Request an audit

Hacken → Audits → Zkrace → [SCA] ZkRace / ERC20 / Mar2024



Audit name:  
[SCA] zkRace / ERC20 / Mar2024

Date:  
Apr 14, 2024

Table of Content

- Introduction
  - Audit Summary
  - Document Information
  - System Overview
  - Executive Summary
  - Risks
  - Findings
  - Appendix 1. Severity Definitions
  - Appendix 2. Scope
  - Disclaimer

Want a comprehensive audit report like this?

Start your audit now →

# Introduction

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

zkRace is deploying a new ERC20 token without burning/minting functionality.

# Audit Summary



Security Score

8/10

Test Coverage

100%

Code Quality Score

10/10

Documentation Quality Score

8/10

5

Total Findings

0

Resolved

5

Accepted

2 Medium



3 Observation



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

## Document Information

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

### Document

**Name**

Smart Contract Code Review and Security Analysis Report for zkRace

**Audited By**

Luis Arroyo

**Approved By**

Grzegorz Trawinski

**Website**

<https://zkpace.com/>

**Changelog**

28/03/2024 – Preliminary Report

02/04/2024 – Remediation Update

## System Overview

W3Forge ERC20 Token is a ERC20 token with the following contracts:

Token — simple ERC-20 token that mints **all** initial supply to a deployer. Additional minting is not allowed.

It has the following attributes:

- Name: zkRace
- Symbol: ZERC
- Decimals: 18
- Total supply: 1200000000e18 tokens

## Privileged roles

- The contract is not address privileged or has owner access.

# Executive Summary

## Documentation quality

The total Documentation Quality score is **8** out of **10**.

- No whitepaper.
- No futures description.
- No Tokenomics.

## Code quality

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

## Test coverage

Code coverage of the project is **100%** (branch coverage), with a mutation score of **100%**.

## Security score

Upon auditing, the code was found to contain **0** critical, **0** high, **2** medium, and **0** low severity issues, leading to a Security score of **8** 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 **8.4**. This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

# Risks



 Existing issues that could cause distrust in the community due to over-centralization concerns.

# Findings

F-2024-1814 Lack of token distribution leads to centralization problems		<div>Q</div>
STATUS	Accepted	
SEVERITY	Medium	

F-2024-1810 Total supply is not hardcoded		<div>Q</div>
STATUS	Accepted	
SEVERITY	Medium	

F-2024-1822 Floating Pragma		<div>Q</div>
STATUS	Accepted	
SEVERITY	Observation	

F-2024-1819 Update import usages to add modularity		<div>Q</div>
STATUS	Accepted	
SEVERITY	Observation	

F-2024-1818 Constructor can be marked as payable		<div>Q</div>
STATUS	Accepted	
SEVERITY	Observation	

Identify vulnerabilities in your smart contracts.

Request an audit →

## 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**

Critical

**Description**

Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.

---

**Severity**

High

**Description**

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.

---

**Severity**

Medium

**Description**

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.

---

**Severity**

Low

**Description**

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/W3Forge/erc20>

---

**Commit**

2bbdbbc7eeb1a7a398a84c257a90d6185db4b204

---

**Whitepaper**