



**HACKEN**

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer:** OBRok Token

**Date:** April 19<sup>th</sup>, 2022

This document 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 containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed – upon a decision of the Customer.

## Document

<b>Name</b>	Smart Contract Code Review and Security Analysis Report for OBROK Token.
<b>Approved By</b>	Evgeniy Bezuglyi   SC Department Head at Hacken OU
<b>Type of Contracts</b>	ERC20 token;
<b>Platform</b>	EVM
<b>Language</b>	Solidity
<b>Methods</b>	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
<b>Timeline</b>	15.04.2022 - 19.04.2022
<b>Changelog</b>	19.04.2022 - Initial Review



## Table of contents

Introduction	4
Scope	4
Executive Summary	5
Severity Definitions	7
Findings	8
Disclaimers	10

## Introduction

Hacken OÜ (Consultant) was contracted by OBRok Token (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

## Scope

The scope of the project is smart contracts in the repository:

**Repository:**

<https://bscscan.com/address/0x205afd08cefe438377a0abc5a20cb4462e1a8c5c#code>

**Documentation:** Yes (<https://obroktoken.com/white-paper/>)

**JS tests:** No

**Contracts:**

CoinToken.sol

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	<ul style="list-style-type: none"><li>▪ Reentrancy</li><li>▪ Ownership Takeover</li><li>▪ Timestamp Dependence</li><li>▪ Gas Limit and Loops</li><li>▪ Transaction-Ordering Dependence</li><li>▪ Style guide violation</li><li>▪ EIP standards violation</li><li>▪ Unchecked external call</li><li>▪ Unchecked math</li><li>▪ Unsafe type inference</li><li>▪ Implicit visibility level</li><li>▪ Deployment Consistency</li><li>▪ Repository Consistency</li></ul>
Functional review	<ul style="list-style-type: none"><li>▪ Business Logics Review</li><li>▪ Functionality Checks</li><li>▪ Access Control &amp; Authorization</li><li>▪ Escrow manipulation</li><li>▪ Token Supply manipulation</li><li>▪ Assets integrity</li><li>▪ User Balances manipulation</li><li>▪ Data Consistency</li><li>▪ Kill-Switch Mechanism</li></ul>

## Executive Summary

The score measurements details can be found in the corresponding section of the [methodology](#).

### Documentation quality

The Customer provided a whitepaper with tokenomics and no technical requirements. The total Documentation Quality score is **5** out of **10**.

### Code quality

The total CodeQuality score is **4** out of **10**. Different code formatting. No unit tests were provided.

### Architecture quality

The architecture quality score is **9** out of **10**. All the logic is implemented in one file.

### Security score

As a result of the audit, security engineers found **2** high, and **2** low severity issues. The security score is **0** out of **10**. All found issues are displayed in the “Issues overview” section.

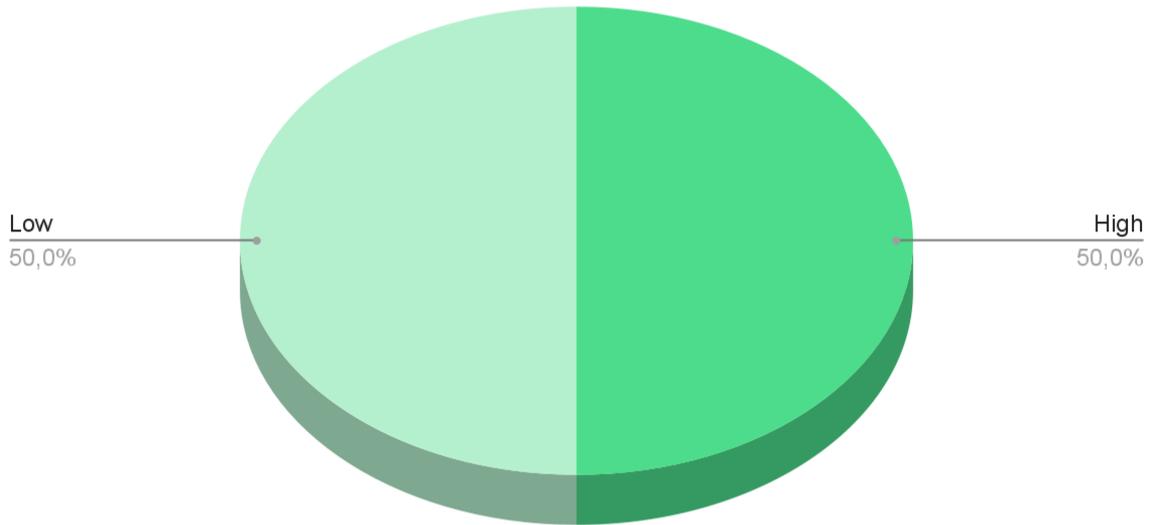
### Summary

According to the assessment, the Customer's smart contract has the following score: **1.8**



You are here

*Graph 1. The distribution of vulnerabilities after the audit.*



## Severity Definitions

Risk Level	Description
<b>Critical</b>	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
<b>High</b>	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
<b>Medium</b>	Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations.
<b>Low</b>	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution

## Findings

### Critical

No critical severity issues were found.

### High

#### 1. Token minting.

According to the tokenomics maximum total supply is 10,000,000,000,000, but the functionality allows the owner to mint more.

**Contracts:** CoinToken.sol

**Function:** mint

**Recommendation:** remove the ability to mint more than stated in tokenomics.

**Status:** New

#### 2. Pausing of all the token transfers.

The functionality allows the owner to pause all the token transfers anytime. Pausing functionality should be limited by clear contract rules. The documentation does not mention the functionality of transfers stopping.

**Contracts:** CoinToken.sol

**Function:** pause

**Recommendation:** change pausing functionality.

**Status:** New

### Medium

No medium severity issues were found.

### Low

#### 1. Functions that can be declared as external.

The contract has public functions that are not used internally and specified as public which uses more gas than external.

**Contracts:** CoinToken.sol

**Function:** mint, updateFee, burn

**Recommendation:** replace visibility to external.

**Status:** New

#### 2. Unlocked pragma and using not latest solidity version



Contracts with unlocked pragmas may be deployed by the latest compiler, which may have higher risks of undiscovered bugs. The latest version of solidity is 0.8.13, it is recommended to use it.

**Function:** -

**Contracts:** CoinToken.sol

**Recommendation:** lock pragmas to a specific compiler version, and use the latest solidity version.

**Status:** New



## Disclaimers

### Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date 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 audit makes no statements or warranties on the security of the code. It also cannot be considered a 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.

### 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 audit cannot guarantee the explicit security of the audited smart contracts.