HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Oracula

**Date**: February 11th, 2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io



## Document

| | |
|---|---|
| **Name** | Smart Contract Code Review and Security Analysis Report for Oracula. |
| **Approved by** | Andrew Matiukhin \| CTO Hacken OU |
| **Type** | ERC20 token; Transfer controller |
| **Platform** | EVM |
| **Language** | Solidity |
| **Methods** | Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| **Repository** | No |
| **Deployed contract** | Contract Address 0x85f3ec4EC49aB6a5901278176235957ef521970d \| BscScan |
| **Technical Documentation** | No |
| **JS tests** | No |
| **Website** | https://oracula.io/pitchdeck.pdf |
| **Timeline** | 03 FEB 2022 - 05 FEB 2022 |
| **Changelog** | 03 FEB 2022 - Initial Audit |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Introduction

Hacken OÜ (Consultant) was contracted by Oracula (Client) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between Feb 03rd, 2022 - Feb 05th, 2022.

# Scope

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

**Repository:**
Contract Address 0x85f3ec4EC49aB6a5901278176235957ef521970d | BscScan
**Commit:**
-
**Technical Documentation:** No
**JS tests:** No
**Contracts:**
oracula.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 | ▪ Reentrancy<br>▪ Ownership Takeover<br>▪ Timestamp Dependence<br>▪ Gas Limit and Loops<br>▪ DoS with (Unexpected) Throw<br>▪ DoS with Block Gas Limit<br>▪ Transaction-Ordering Dependence<br>▪ Style guide violation<br>▪ Costly Loop<br>▪ ERC20 API violation<br>▪ Unchecked external call<br>▪ Unchecked math<br>▪ Unsafe type inference<br>▪ Implicit visibility level<br>▪ Deployment Consistency<br>▪ Repository Consistency<br>▪ Data Consistency |

| | |
|---|---|
| Functional review | ▪ Business Logics Review<br>▪ Functionality Checks<br>▪ Access Control & Authorization<br>▪ Escrow manipulation<br>▪ Token Supply manipulation<br>▪ Assets integrity<br>▪ User Balances manipulation<br>▪ Data Consistency manipulation<br>▪ Kill-Switch Mechanism<br>▪ Operation Trails & Event Generation |

## Executive Summary

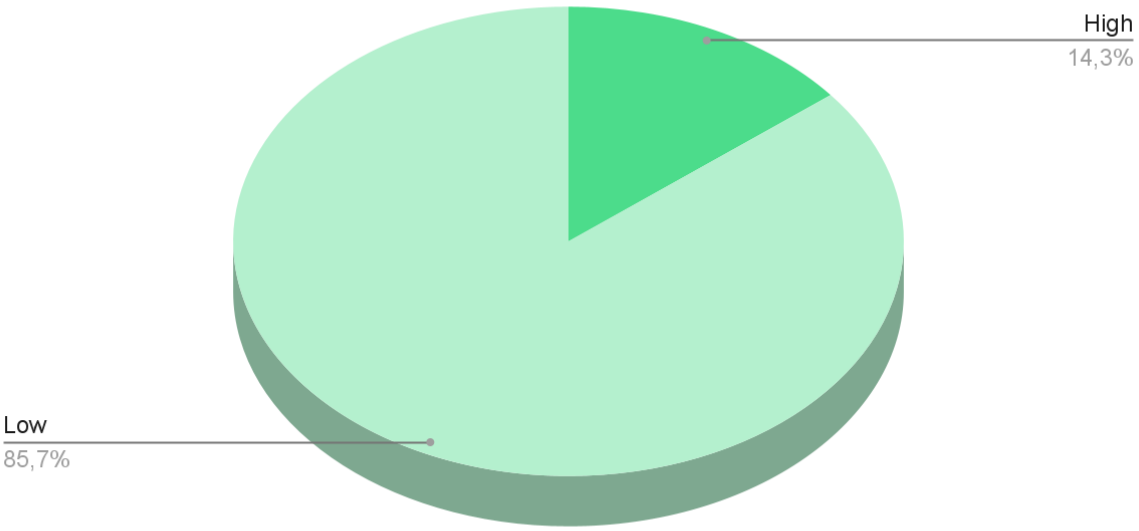
According to the assessment, the Customer's smart contracts are well-secured.


Our team performed an analysis of code functionality, manual audit, and automated checks with Mythx and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **1** high**, 6** low severity issues.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Severity Definitions

| Risk Level | Description |
|---|---|
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to asset loss or data manipulations. |
| High | 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 |
| Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to asset loss or data manipulations. |
| Low | Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Audit overview

## Critical

No critical issues detected.

## High

1. Transfer Receives Fees.

In the _transfer() function fees are deducted from the amount. This causes the receiver to get less than desired amount every time. Also it is impossible to predict fees.

**Contracts**:Oracula.sol

**Functions**: _transfer(address _from, address _to, uint256 _amount)

**Recommendation**: Transfer and transferFrom should not take a fee.

## Medium

No medium issues detected.

## Low

2. Variable Shadowing.

The owner variable used in the *allowance* function shadows the Ownable contract's owner function.

**Contracts**:Oracula.sol

**Functions**: owner(), approve(address owner, address spender, uint256 amount), allowance(address owner, address spender)

3. Boolean Equality.

Boolean constants can be used directly and do not need to be compared to true or false

**Contracts**:Oracula.sol

**Functions**: _setExcanhePairs(address pair, bool value)

**Recommendation**: Avoid using boolean equalities.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

**Status**:

4. State Variable Visibility not Set (SWC-108).

The visibility for state variables, burnFee, devFee, rewardFee, maxBurnFee, maxDevFee, maxRewardFee, maximumFee, denominator, burnAddress, devAddress, rewardAddress are not set.

**Contracts**:Oracula.sol

**Functions**: -

**Recommendation**: Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

5. State Variables that could be Declared as *constant*

State variables that don't change their (_decimals, _name, _symbol, _totalSupply, denominator) value should be declared constant to save gas.

**Contracts**:Oracula.sol

**Functions**: -

**Recommendation**:Declare above mentioned variables as constants.

**Status**:

6. Functions that can be Declared as *external*

In order to save Gas, public functions that are never called in the contract should be declared as *external*.

**Contracts**: Oracula.sol

**Functions:** renounceOwnership(), transferOwnership(address newOwner), increaseAllowance(address spender, uint256 addedValue), decreaseAllowance(address spender, uint256 subtractedValue), approve(address owner, address spender, uint256 amount), allowance(address owner, address spender), transferFrom(address sender, address recipient, uint256 amount)

**Recommendation:** Aforementioned should be declared as *external*.

**Status:**

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **1** high, **6** low severity issues.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Disclaimers

## Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to 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 as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. 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 can't guarantee the explicit security of the audited smart contracts.