HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: BotPlanet
Date: April 06th, 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 BotPlanet.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type of Contracts	DEX Core
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://www.botpla.net/
Timeline	16.02.2022 - 06.04.2022
Changelog	23.03.2022 - Initial Review 06.04.2022 - Revise

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 BotPlanet (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://github.com/BOTDeFi/botdex-contracts-core
Commit:
215e8490d064f5d8e674e049fdde73fe59ce3559
Technical Documentation: Yes
(https://www.botpla.net/wp-content/uploads/2022/02/White-Paper.pdf;
https://docs.google.com/document/d/1WMOkMi4xAX7MburhZyQ8wy6Axmyf9xXtpnvL15yuNgs/edit)
JS tests: Yes (link)/No?
Contracts:
interfaces/IBotdexCallee.sol
interfaces/IBotdexERC20.sol
interfaces/IBotdexFactory.sol
interfaces/IBotdexPair.sol
interfaces/IERC20.sol
libraries/Math.sol
libraries/UQ112x112.sol
test/ERC20.sol
BotdexERC20.sol
BotdexPair.sol
BotdexFactory.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 ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ EIP standards violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency

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

Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency ▪ Kill-Switch Mechanism

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

Executive Summary

Score measurements details can be found in the corresponding section of the methodology.

Documentation quality

The customer provided a whitepaper as functional requirements and a short description of functions as technical requirements. Counting that the implementation is a 1-to-1 clone of the UniswapV2-core which is fully documented, the total Documentation Quality score is **8** out of **10**.

Code quality

Total CodeQuality score is **8** out of **10**. The code is a copy of the UniswapV2-core codebase. Some parts are improved some are increased gas usage.

Architecture quality

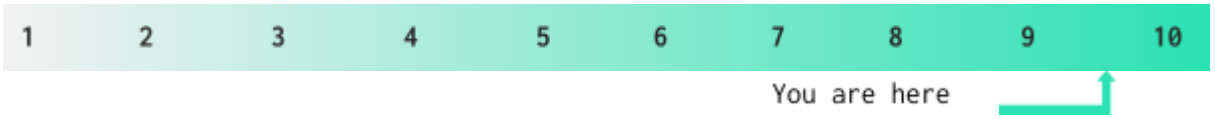
The architecture quality score is **9** out of **10**. The architecture is standard for DEX core. Improved for the factory to contain the “INIT_CODE_PAIR_HASH” constant.

Security score

As a result of the audit, security engineers found **no security** issues. The security score is **10** out of **10**. All found issues are displayed in the “Issues overview” section of the report.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.5**



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 assets 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 assets 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

Findings

Critical

No critical severity issues were found.

High

No high severity issues were found.

Medium

1. **Tests failed**

   Two tests of 32 are failed. Both tests are for gas usage: `createPair:gas` and `swap:gas`.

   Scope: testing

   Recommendation: please make sure all tests are passing.

   Status: Fixed (Revised Commit: 215e849)

Low

1. **Floating solidity version**

   It is recommended to specify the exact solidity version in the contracts.

   Contracts: all

   Recommendation: please specify exact solidity version (ex. pragma solidity 0.8.4 instead of pragma solidity >=0.8.0).

   Status: Fixed (Revised Commit: 215e849)

2. **SPDX license identifier not provided in a source file.**

   Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.

   Contracts: all

   Recommendation: add SPDX-license identifiers.

   Status: Fixed (Revised Commit: 215e849)

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 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.