HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: BotPlanet
Date: April 07th, 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; Farms; Staking; MasterChef
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://website.com
Timeline	16.02.2022 - 07.04.2022
Changelog	23.03.2022 - Initial Review 07.04.2022 - 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-farm
Commit:
e22a94489331fafe13f6ff1170aeb7c579190192
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 (test)
Contracts:
pancake-swap/interfaces/IPancakeFactory.sol
pancake-swap/libraries/FullMath.sol
pancake-swap/libraries/PancakeOracleLibrary.sol
pancake-swap/libraries/Babylonian.sol
pancake-swap/WETH.sol
BotdexStaking.sol
test/BOTToken.sol
pancake-swap/libraries/FixedPoint.sol
libs/Multicall2.sol
libs/Multicall.sol
pancake-swap/interfaces/IWETH.sol
pancake-swap/libraries/UQ112x112.sol
pancake-swap/PancakeRouter01.sol
test/BOTTokenTest.sol
pancake-swap/interfaces/IPancakeRouter02.sol
libs/rocketswap-lib/access/Ownable.sol
pancake-swap/PancakeFactory.sol
libs/rocketswap-lib/access/Context.sol
libs/token/IBEP20.sol
pancake-swap/PancakeRouter.sol
libs/WBNB.sol
libs/rocketswap-lib/utils/Address.sol
pancake-swap/interfaces/IPancakeCallee.sol
libs/MockBEP20.sol
pancake-swap/PancakePair.sol
libs/token/SafeBEP20.sol
pancake-swap/RocketToken.sol
pancake-swap/interfaces/IPancakeERC20.sol
pancake-swap/libraries/BitMath.sol
pancake-swap/libraries/TransferHelper.sol
pancake-swap/PancakeERC20.sol
libs/rocketswap-lib/math/SafeMath.sol
RocketPropellantToken.sol

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

```
pancake-swap/libraries/PancakeLibrary.sol
libs/token/BEP20.sol
libs/Migrations.sol
MasterBotdex.sol
pancake-swap/interfaces/IPancakePair.sol
interfaces/IMasterBotdex.sol
pancake-swap/libraries/Math.sol
pancake-swap/interfaces/IPancakeRouter01.sol
interfaces/PancakeVoteProxy.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
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 partial clone of the PancakeSwap farms which is fully documented, the total Documentation Quality score is **8** out of **10**.

Code quality

Total CodeQuality score is **8** out of **10**. Parts of the code are a copy of the PancakeSwap farms' codebase. Some parts are improved some are increased gas usage.

Architecture quality

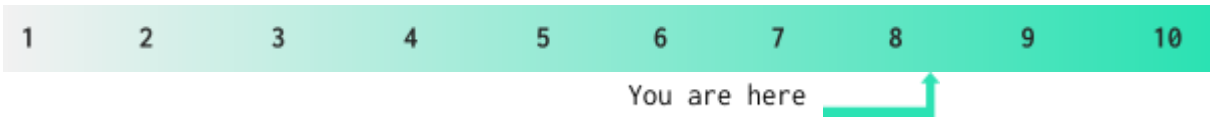
The architecture quality score is **10** out of **10**. Clean and functional.

Security score

As a result of the audit, security engineers found **1** medium severity issue. The security score is **8** 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: **8.2**



Notices

1. The owner can withdraw all LP tokens from the MasterBotdex contract by setting the migrator contract.

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

1. **Possible rewards lost or receive more**

   Changing **allocPoint** in the MasterBotdex.set method while **_withUpdate** flag set to **false** may lead to rewards lost or receiving rewards more than deserved.

   Contract: MasterBotdex.sol

   Function: set

   Recommendation: Please call updatePool(_pid) in the case if **_withUpdate** flag is **false** and you don't want to update all pools.

   Status: Fixed (Revised Commit: e22a944)

Medium

1. **Privileged ownership**

   The owner of the MasterBotdex contract has permission to updateMultiplier, add new pools, change the pool's allocation points and set migrator contract (which will move all LPs from the pool to itself) without community consensus.

   Contract: MasterBotdex.sol

   Recommendation: Please consider using one of the following methodologies:

   - Transfer ownership to Time-lock contract with reasonable latency (ie. 24h) so the community may react to changes;
   - Transfer ownership to a multi-signature wallet, to prevent a single point of failure;
   - Transfer ownership to DAO so the community could decide whether the privileged operations should be executed by voting.

   Status: Not Fixed (Revised Commit: e22a944)

Low

1. **Unnecessary operations**

   When allocPoint is not changed for the pool, there is still an assignment for a new value, which just consumes gas doing nothing.

   Contract: MasterBotdex.sol

   Function: set

   Recommendation: Please move "poolInfo[_pid].allocPoint = _allocPoint" assignment inside the *if* block.

Status: Fixed (Revised Commit: e22a944)

2. **Missing Emit Events**

   Functions that change critical values should emit events for better off-chain tracking.

   Contract: MasterBotdex.sol

   Function: setMigrator, updateMultiplier

   Recommendation: Consider adding events when changing critical values, and emit them in the function.

   Status: Fixed (Revised Commit: e22a944)

3. **Using solidity time units**

   Solidity provides time unit suffixes to use to describe a time in seconds.

   Contract: BotdexStaking.sol

   Constant: YEAR

   Recommendation: consider using time unit suffixes
   (ie. *YEAR = 365* ***days***).

   Status: Fixed (Revised Commit: e22a944)

4. **Unused structure field**

   The structure has a field `stakeAmount` which is neither set nor accessed.

   Contract: MasterBotdex.sol

   Structure: UserInfo

   Recommendation: remove unused field.

   Status: Fixed (Revised Commit: e22a944)

5. **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 0.8.7 instead of pragma solidity ^0.8.7).

   Status: Fixed (Revised Commit: e22a944)

6. **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: e22a944)

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.