HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Bolide
Date: July 7th, 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 Bolide.
Approved By	Andrew Matiukhin | CTO Hacken OU
Type of Contracts	ERC20 token; Farming; TokenSale; Strategy; Vesting
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://bolide.fi/
Timeline	21.03.2022 - 07.07.2022
Changelog	30.03.2022 - Initial Review 18.04.2022 - Revise 07.06.2022 - Revise 07.07.2022 - 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 Bolide (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/bolide-fi/contracts
Commit:
57f192ff4c4f1c8e8fbb99c60757f0327d76716c
Documentation: Yes
JS tests: Yes
Contracts:
farming/contracts/libs/PancakeVoteProxy.sol
farming/contracts/libs/Migrations.sol
farming/contracts/MasterChef.sol
farming/contracts/Timelock.sol
farming/contracts/libs/MockBEP20.sol

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

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

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

Documentation quality

The Customer provided some functional requirements and no technical requirements. The contracts are forks of well-known ones. The total Documentation Quality score is **10** out of **10**.

Code quality

The total CodeQuality score is **9** out of **10**. No NatSpecs.

Architecture quality

The architecture quality score is **10** out of **10**.

Security score

As a result of the audit, security engineers found **1** low severity issue. The security score is **10** 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: **9.9**

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 cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot 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 receiving more**

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

   Contract: MasterChef.sol

   Function: set

   Recommendation: Call updatePool(_pid) in the case if **_withUpdate** flag is **false** and you do not want to update all pools.

   Status: Fixed. (Revised Commit: 9ca0cf0)

Medium

1. **Privileged ownership**

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

   Contract: MasterChef.sol

   Recommendation: Consider using one of the following methodologies:

   - Transfer ownership to a Time-lock contract with reasonable latency (i.e. 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: Fixed; Moved ownership to a Timelock (Revised Commit: 9ca0cf0)

Low

1. **Excess writing operation**

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

   Contract: MasterChef.sol

   Function: set

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

Status: Fixed (Revised Commit: 9378f79)

2. **Missing Emit Events**

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

   Contract: MasterChef.sol

   Function: setMigrator, updateMultiplier, setBlidPerBlock

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

   Status: Fixed (Revised Commit: 9378f79)

3. **Floating solidity version**

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

   Contracts: all

   Recommendation: Specify the exact solidity version (ex. pragma solidity 0.8.10 instead of pragma solidity ^0.8.0).

   Status: Fixed (Revised Commit: 9378f79)

4. **Balance updated after transfer**

   It is recommended to update the balance state before doing any token transfer.

   Contract: MasterChef.sol

   Functions: emergencyWithdraw, migrate

   Recommendation: Update the balance and do transfer after that.

   Status: Reported (Revised Commit: 9378f79)

5. **A public function that could be declared external**

   Public functions that are never called by the contract should be declared **external**.

   Contracts: MasterChef.sol

   Functions: updateMultiplier, add, set, setBlidPerBlock, setMigrator, setExpenseAddress, migrate, deposit, withdraw, enterStaking, leaveStaking

   Recommendation: Use the **external** attribute for functions never called from the contract.

   Status: Fixed (Revised Commit: 9378f79)

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.