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:
strategies/low_risk/contracts/libs/Aggregator.sol
strategies/low_risk/contracts/libs/ERC20ForTestStorage.sol
strategies/low_risk/contracts/libs/Migrations.sol
strategies/low_risk/contracts/Logic.sol
strategies/low_risk/contracts/Storage.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 functional requirements and technical requirements. The total Documentation Quality score is **10** out of **10**.

Code quality

The total CodeQuality score is **7** out of **10**. Code duplications. Not following solidity code style guidelines. Gas over-usage.

Architecture quality

The architecture quality score is **8** out of **10**. Logic is split into modules. Contracts are self-descriptive. No thinking about gas efficiency. Room for improvements in code structuring.

Security score

As a result of the audit, security engineers found **no issues**. 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: **9.5**

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 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

No high severity issues were found.

Medium

1. **Test failed**

   One of the two tests is failing. That could be either an issue in the test or an error in the contract logic implementation.

   Scope: strategies

   Recommendation: Ensure that the tests are successful and cover all the code branches.

   Status: 6 of 73 tests are failing (Revised Commit: 9378f79)

Low

1. **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)

2. **Excessive state access**

   It is not recommended to read the state at each code line. It would be much more gas effective to read the state value into the local memory variable and use it for reading.

   Contract: StorageV0.sol

   Recommendation: Read the state variable to a local memory instead of multiple reading.

   Status: Fixed (Revised Commit: 9ca0cf0)

3. **Not emitting events**

   StorageV0 and Logic are not emitting events on state changes. There should be events to allow the community to track the current state off-chain.

   Contract: StorageV0.sol, Logic.sol

   Functions: setBLID, addToken, setLogic, setStorage, setAdmin

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

Status: Fixed (Revised Commit: 9ca0cf0)

4. **Implicit variables visibility**

   State variables that do not have specified visibility are declared **internal** implicitly. That could not be obvious.

   Contract: StorageV0.sol

   Variables: earnBLID, countEarns, countTokens, tokens, tokenBalance, oracles, tokensAdd, deposits, tokenDeposited, tokenTime, reserveBLID, logicContract, BLID

   Recommendation: Always declare visibility explicitly.

   Status: Fixed (Revised Commit: 9378f79)

5. **Reading state variable's `length` in the loop**

   Reading `length` attribute in the loop may cost excess gas fees.

   Contract: Logic.sol

   Function: returnToken

   Recommendation: Save `length` attribute value into a local memory variable.

   Status: Fixed (Revised Commit: 9378f79)

6. **Reading state variable in the loop**

   Reading `countTokens` state variable in the loop would cost excess gas fees.

   Contract: StorageV0.sol

   Function: addEarn, _upBalance, _upBalanceByItarate, balanceEarnBLID, balanceOf, getTotalDeposit

   Recommendation: Save `countTokens` value into a local memory variable.

   Status: Fixed (Revised Commit: 9ca0cf0)

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

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

   Contracts: Logic.sol, StorageV0.sol

   Functions: Logic.getReservesCount, Logic.getReserve, StorageV0.initialize, StorageV0._upBalance, StorageV0._upBalanceByItarate, StorageV0.balanceOf, StorageV0.getBLIDReserve, StorageV0.getTotalDeposit, StorageV0.getTokenBalance, StorageV0.getTokenDeposit, StorageV0._isUsedToken, StorageV0.getCountEarns

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

Status: Fixed (Revised Commit: 9ca0cf0)

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.