HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: TeraBlock
Date: April 20th, 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 TeraBlock.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type of Contracts	Staking
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://terablock.com
Timeline	04.04.2022 - 20.04.2022
Changelog	06.04.2022 - Initial Review 18.04.2022 - Remediation Check 20.04.2022 - Remediation Check 2

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 TeraBlock (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/TeraBlock/tb-stake-v1-contracts - Initial Audit
https://github.com/TeraBlock/tb-stake-v1-contracts/tree/eea11145ecff797b05e20283905b422b82aa15a5 - Remediation Check
Commit:
07427df50fc509de887a009489872a6202180a1e - Initial Audit
eea11145ecff797b05e20283905b422b82aa15a5 - Remediation Check
Documentation: No - https://docs.google.com/document/d/1_vhOTHjb06P1wbFtTY_yDFLRO-YhAdAk-qRq4_V-aSM/edit
JS tests: Yes - https://github.com/TeraBlock/tb-stake-v1-contracts/tree/07427df50fc509de887a009489872a62
Contracts:
Pool.sol
PoolFactory.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

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 **10** out of **10**. Code follows official language style guides. Unit tests were provided.

Architecture quality

The architecture quality score is **10** out of **10**. Smart contracts of the project follow the best practices, and the project has a clear architecture.

Security score

As a result of the audit, security engineers found **1** critical, **3** high, **4** medium, and **7** low severity issues. The security score is **0** out of **10**.

As a result of the second review, security engineers found **1** low severity issue. All previously found issues were fixed. The security score is **10** out of **10**.

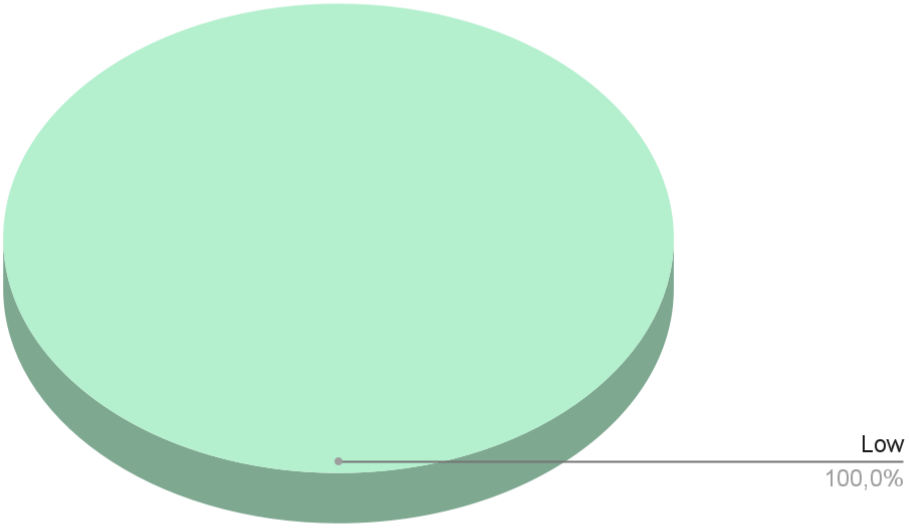
All found issues are displayed in the “Issues overview” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **10.0**

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 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 issues were found.

High

1. **The owner can Withdraw Both Reward and Staking Tokens.**

   The owner can withdraw both reward and staking tokens from the pools. This withdrawal can be done at any time without informing the users, which can lead to sudden balance changes in the pools. From its name (recoverTokens()), it is thought that this function was created to recover tokens stuck in the pool. However, a definite conclusion could not be reached due to the lack of documentation.

   This can lead to sudden token depletion in the pool.

   Contracts: Pool.sol, PoolFactory.sol,

   Function: recoverTokens()

   Recommendation: Remove this functionality or inform users in the documentation.

   Status: Fixed

2. **Missing Staking Token Balance Check.**

   Staking tokens are sent to the user in the form of reward + amount. However, before this process, only the amount that the user has deposited in the relevant stake is checked. It is not checked whether there are enough tokens in the system.

   This can lead to reverts in the withdrawal process.

   Contracts: Pool.sol

   Function: withdraw()

   Recommendation: Implement control mechanisms

   Status: Mitigated (The customer approved that this behaviour is expected)

Medium

1. **Missing Allowance Check.**

   The safeTransferFrom() function, of ERC20, is being called in other functions, but they never check if there is enough allowance before calling it.

   This can lead to reverts in the calling functions.

   Contracts: Pool.sol, PoolFactory.sol

   Function: deposit(), depositRewards()

Recommendation: Add control mechanisms for allowances. Adjust the allowance before calling the safeTransferFrom() function.

Status: Fixed

2. **Possible Out-of-Gas Exception.**

   Iterating over stakeIds and amounts may lead to enormous Gas consumption due to the arrays' size.

   This could lead to a potential Out-of-Gas exception.

   Contracts: PoolFactory.sol

   Function: withdrawBatch()

   Recommendation: Implement array size limitations.

   Status: Mitigated

3. **Redundant Subtraction.**

   1 is subtracted from stakeId. This operation could be performed off-chain, and a valid id can be passed.

   Contracts: Pool.sol

   Function: withdraw(), getRewardsEarned(), isEligibleForRewards()

   Recommendation: Remove redundant subtraction.

   Status: Fixed

Low

1. **Checks-Effects-Interactions Pattern Violation**

   The state variables of a pool are updated after depositing or withdrawing tokens from the farm contract.

   Contracts: PoolFactory.sol

   Function: depositRewards()

   Recommendation: Make token transfers after user balances are decreased or zeroed.

   Status: Fixed

2. **Missing Zero Address Validation**

   Address parameters are being used without checking against the possibility of 0x0.

   This can lead to unwanted external calls to 0x0.

   Contracts: AccessProtected.sol, Pool.sol,

   Function: setAdmin(), getStakes(), getTotalStaked(), getRewardsEarned()

   Recommendation: Implement zero address checks.

Status: Fixed

3. **Outdated Encoder Use**

   The statement “pragma experimental ABIEncoderV2” is outdated.

   Contracts: Pool.sol

   Function: -

   Recommendation: Use “pragma abicoder v2” instead.

   Status: Fixed

4. **Use of Hardcoded Values**

   Hardcoded values are used in computations.

   Contracts: Pool.sol

   Function: getPeriod()

   Recommendation: Move hardcoded values to constants.

   Status: Reported

5. **Functions That Can be Declared as *external***

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

   Contracts: AccessProtected.sol, Pool.sol, PoolFactory.sol

   Function: isAdmin(), mint(), burn(), pause(), unpause(), recoverTokens(), deposit(), withdrawBatch(), setRewardsDeposited(), depositRewards(), setStakingToken(), setTGBToken(), recoverTokens(), getStakes(), getTotalRewardsEarned(), getTotalStaked, poolCount()

   Recommendation: Move hardcoded values to constants.

   Status: Fixed

6. **Floating Pragma**

   The project uses floating pragma ^0.6.12.

   Contracts: AccessProtected.sol, Pool.sol, PoolFactory.sol

   Function: -

   Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

   Status: Fixed

7. **Outdated Solidity Version**

   Using an old version prevents access to new Solidity security checks.

   Contracts: AccessProtected.sol, Pool.sol, PoolFactory.sol

Function: -

Recommendation: Consider using one of these versions: *0.8.6*, *0.8.9*, or *0.8.11*.

Status: Fixed

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.