HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Pera
Date: April 05th, 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 Pera.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type of Contracts	ERC20 token; Staking; Yield Farming
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://www.pera.finance
Timeline	16.03.2022 - 05.04.2021
Changelog	23.03.2022 - Initial Review 05.04.2022 - Remediations Check

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

Table of contents

Introduction 4

Scope 4

Executive Summary 5

Severity Definitions 7

Findings 8

Disclaimers 12

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

Introduction

Hacken OÜ (Consultant) was contracted by Pera (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:
Initial Review
https://github.com/perafinance/peraplatform/commit/2e02b430c7641121e8053cf6718087115a5cf9d5 - Pera Farming
https://github.com/perafinance/pera-staking/commit/952230ed67a040c9b0f37b20511385fbf61195f5 - Pera Staking
Remediation Check
https://github.com/perafinance/peraplatform/
https://github.com/perafinance/pera-staking/
Commit:
Initial Review
952230ed67a040c9b0f37b20511385fbf61195f5 - Pera Staking
2e02b430c7641121e8053cf6718087115a5cf9d5 - Pera Farming
Remediation Check
e1a34b8e51ec9564ea934c659642b099e8732808 - Pera Staking
7969ae040763fb3e882ac731882676d09d024036 - Pera Farming
Technical Documentation: Yes - https://www.pera.finance/info/PeraFinanceNewDeck.pdf
JS tests: Yes
Contracts:
PeraStaking.sol
TradeFarming.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

Executive Summary

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

Documentation quality

The Customer provided superficial functional requirements and no technical requirements. Total Documentation Quality score is **3** out of **10**.

Code quality

The total CodeQuality score is **5** out of **10**. Code follows best practices. Unit tests were provided. Test coverage is very low and does not check for multiple user-related cases.

Architecture quality

The architecture quality score is **10** out of **10**. Both projects follow recommended design patterns and best practices. Projects have clean and clear structures.

Security score

As a result of the audit, security engineers found **5** high, **4** medium, and **5** low severity issues. The security score is **0** out of **10**. All found issues are displayed in the “Issues overview” section.

As a result of the remediations review, Customer’s smart contracts contain **2** medium and **3** low severity issues. The security score is **10** out of **10**

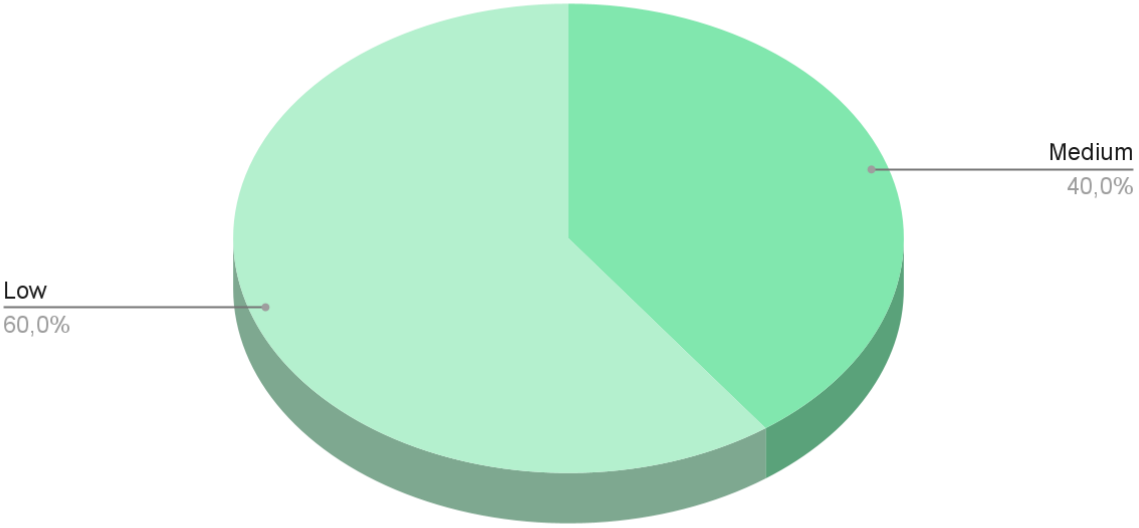
Summary

According to the assessment, the Customer's smart contract has the following score: **8.8**

1.

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. **Undocumented owner functionality**

   The owner can withdraw all tokens from the contract. This behavior is not mentioned in the provided documentation.

   Contracts: PeraStaking.sol

   Function: withdrawTokens()

   Recommendation: remove the function or provide the description in the documentation.

   Status: Fixed

2. **Potential DoS Risk**

   Iteration over the activeRewards array can consume much Gas as the size of the array grows.

   This could lead to potential Denial of Service.

   Contracts: PeraStaking. sol

   Function: updateReward()

   Recommendation: the contract design should be changed to avoid data updates for all its users during one single call.

   Status: Mitigated

3. **Potential Reward Token Balance Shortage**

   Rewards in all available tokens are transferred during one call. If there is at least one token whose balance is insufficient, the whole call will fail.

   Contracts: PeraStaking. sol

   Function: claimAllRewards()

   Recommendation: allow to claim rewards one by one or by batches provided by a caller.

   Status: Mitigated

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

4. **Missing Reward Balance Update**

   claimAllRewards function does not decrease totalRewardBalance after reward transfer to a user.

   This can lead to an incorrect reward token balance. If the staking and reward tokens are the same, this can lead to the usage of deposited funds as reward funds.

   Contracts: TradeFarming. sol

   Function: claimAllRewards()

   Recommendation: Decrease totalRewardBalance during rewards claiming.

   Status: Mitigated

5. **Reward and Staking Token Balances Should be Separate**

   The contract should separate reward and staking token balances.

   In the case of these tokens being the same, this could lead to the use of staking tokens in the name of reward tokens.

   Contracts: PeraStaking. sol

   Function: -

   Recommendation: Review and check this logic.

   Status: Fixed

Medium

1. **Missing Allowance Check**

   The safeTransferFrom function is being called in other functions, but they never check if there is enough allowance prior to calling it.

   This can lead to reverts in the calling functions.

   Contracts: PeraStaking. sol, TradeFarming.sol

   Function: depositRewards(), swapExactTokensForETH(), swapTokensForExactETH(), swapETHForExactTokens(), swapExactETHForTokens()

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

   Status: Mitigated

2. **Revert due to transfer Function Gas Limitation**

   The swapETHForExactTokens functions cannot be called from another contract with a fallback function. This is because the transfer function has a hardcoded Gas upper limit, used in the refunding logic.

   This can lead to limitations in the system.

   Contracts:TradeFarming.sol

   Function: swapETHForExactTokens()

   Recommendation: Use the call function, which allows the caller to send all the Gas.

   Status: Reported

3. **Unused Return**

   There are calls to EnumerableSet.UintSet's functions but return values of these calls are ignored.

   This can lead to unexpected behaviors in the function execution.

   Contracts: TradeFarming. sol

   Function: swapExactETHForTokens(), swapETHForExactTokens(), swapExactTokensForETH(), swapTokensForExactETH()

   Recommendation: Implement control mechanisms.

   Status: Reported

Low

1. **Missing Zero Address Validation**

   The constructor and depositRewardTokens take address parameters but do not check if they are zero address.

   This can lead to unwanted external calls to 0x0.

   Contracts: TradeFarming.sol

   Function: constructor() and depositRewardTokens()

   Recommendation: Implement zero address checks.

   Status: Fixed

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

2. **Unused State Variable**

   Field `MAX_UINT` is never used.

   Contracts: TradeFarming.sol

   Function: -

   Recommendation: Remove unused variable.

   Status: Fixed

3. **Use of Hardcoded Values**

   The constructor and mulDiv functions use hardcoded values in their computations.

   In PeraStaking.sol contract, withdraw, calcWeight, and _decrease functions use hardcoded values in their computations.

   Contracts: TradeFarming.sol, PeraStaking.sol

   Function: withdraw(), calcWeight(), _decrease(), constructor(), mulDiv()

   Recommendation: Move hardcoded values to constants.

   Status: Reported

4. **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: PeraStaking.sol

   Function: calcWeight()

   Recommendation: Aforementioned function should be declared as *external*.

   Status: Reported

5. **Floating Pragma**

   The PeraStaking.sol, and TradeFarming.sol contracts use floating pragma ^0.8.11

   Contracts: PeraStaking.sol, TradeFarming.sol

   Function: -

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

   Status: Reported

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.