HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: PeakDeFi

Date: April 14th, 2021

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

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities fixed - upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for PeakDeFi.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token, Staking, Governance, Defi
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/PeakDeFi/peakdefi-contracts/tree/feature/protected_staking
Commit	c14bf25c5cb047a76343ea01f3268d1323b0e0e6
Deployed contract	
Timeline	12 APR 2021 – 14 APR 2021
Changelog	14 APR 2021 – INITIAL AUDIT 26 APR 2021 – SECONDARY AUDIT

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 PeakDeFi (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted between April 12th, 2021 – April 14th, 2021.

The secondary review conducted on April 26th, 2021.

Scope

The scope of the project is smart contracts in the repository:

Contract deployment address:
Repository
File:
ProtectionStaking.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 ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data 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 ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.

Insecure	Poor secured	Secured	Well-secured

You are here 

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section, and all found issues can be found in the Audit overview section.

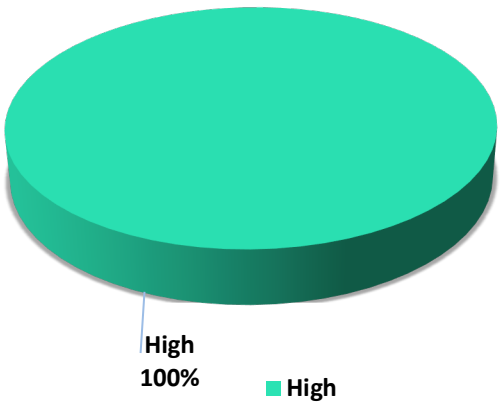
Security engineers found **1** high issue during the audit.

After the **second** review no vulnerabilities were found.

Notice: the audit scope is limited and not include all files in the repository. Though, reviewed contracts are secure, we may not guarantee secureness of contracts that are not in the scope.

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

Graph 1. The distribution of vulnerabilities after the first review.



After the second review no vulnerabilities were found.

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
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

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

AS-IS overview

ProtectionStaking.sol

Description

Imports

ProtectionStaking has following imports:

- SafeMath.sol – from the OpenZeppelin.
- Ownable.sol – from the OpenZeppelin.
- import "../../lib/ReentrancyGuard.sol";
- import "../../Utils.sol";
- import "../../PeakDeFiFund.sol";
- import "../../PeakDeFiStorage.sol";
- import "../../interfaces/IMiniMeToken.sol";
- import "../IUniswapOracle.sol";

Inheritance

ProtectionStaking is

- Ownable
- ReentrancyGuard.

Usages

ProtectionStaking contract has following usages:

- SafeMath for uint.
- SafeERC20 for PeakToken;
- SafeERC20 for IERC20;

Structs

ProtectionStaking contract has no data structures.

Enums

ProtectionStaking contract has no enums.

Events

ProtectionStaking contract has following events:

- event ClaimCompensation(address investor, uint256 amount, uint256 timestamp);
- event RequestProtection(address investor, uint256 amount, uint256 timestamp);
- event Withdraw(address investor, uint256 amount, uint256 timestamp);
- event ProtectShares(address investor, uint256 amount, uint256 timestamp);
- event WithdrawShares(address investor, uint256 amount, uint256 timestamp);
- event AdminWithdrawToken(address token, uint256 amount, uint256 timestamp);
- event ChangePeakMintCap(uint256 newAmmount);

Modifiers

ProtectionStaking has following modifiers:

- during(PeakDeFiStorage.CyclePhase phase)
- ifNoCompensation()

Fields

ProtectionStaking contract has following fields and constants:

- PeakDeFiFund public fund;
- PeakToken public peakToken;
- address public sharesToken;
- IUniswapOracle public uniswapOracle;
- mapping(address => uint256) public peaks;
- mapping(address => uint256) public shares;

- mapping(address => uint256) public startProtectTimestamp;
- mapping(address => uint256) internal _lastClaimTimestamp;
- mapping(address => uint256) public lastClaimAmount;
- uint256 public mintedPeakTokens;
- uint256 public peakMintCap = 3 * 10 ** 16;
- uint256 internal constant PEAK_PRECISION = 10 ** 8;
- uint256 internal constant USDC_PRECISION = 10 ** 6;
- uint256 internal constant PERCENTS_DECIMALS = 10 ** 20;

Functions
ProtectionStaking has following public functions:

- ***constructor***
- *calculateCompensating*
- *claimCompensation*
- *requestProtection*
- *withdraw*
- *protectShares*
- *withdrawShares*
- *setPeakMintCap*
- *adminWithdrawToken*

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

Audit overview

Critical

No critical issues were found.

High - Resolved

- The function *adminWithdrawToken* provide ability to withdraw any funds to the smart contract owner.

Fixed before the second audit.

Medium

No critical issues were found.

Low

No low severity issues were found.

Lowest / Code style / Best Practice

No critical issues were found.

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

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, high-level description of functionality was presented in As-Is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found **1** high issue during the audit.

After the **second** review no vulnerabilities were found.

Notice: the audit scope is limited and not include all files in the repository. Though, reviewed contracts are secure, we may not guarantee secureness of contracts that are not in the scope.

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 in accordance with the best industry practices at the date of this report, in relation to 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 security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. 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 security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.