HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: StrongBlock
Date: November 23rd, 2021

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 StrongBlock.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/StrongBlock/eth2.pool.contract
Commit	ea3eb31067b2cf193e98bfdd310d89807df9eb21
Technical Documentation	NO
JS tests	NO
Website	strongblock.com
Timeline	09 NOVEMBER 2021 - 23 NOVEMBER 2021
Changelog	12 NOVEMBER 2021 - INITIAL AUDIT 19 NOVEMBER 2021 - SECOND REVIEW 23 NOVEMBER 2021 - THIRD REVIEW

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 StrongBlock (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 contract and its code review conducted between November 9th, 2021 - November 12th, 2021.

Second review conducted on November 19th, 2021.

Third review conducted on November 23rd, 2021.

Scope

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

Repository:
https://github.com/StrongBlock/eth2.pool.contract
Commit:
ea3eb31067b2cf193e98bfdd310d89807df9eb21
Technical Documentation: No
JS tests: No
Contracts:
Context.sol
ETHPool.sol
ETHPoolInterface.sol
IERC20.sol
Ownable.sol
PlatformFees.sol
PlatformFeesInterface.sol
ReentrancyGuard.sol
SafeMath.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 ▪ 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
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

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

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. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **1** high and **3** low severity issues.

After the second review security engineers found that all previously found issues were fixed but **1** high severity issue was added to the code.

After the third review security engineers found that **all** issues were addressed.

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

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

Audit overview

Critical

No critical issues were found.

High

1. ETH could be locked.

The transfer function may fail if a recepient is the contract address with fallback function. As a result, funds may be locked.

Contract: ETHPool.sol

Functions: processClaimPayment, processFee

Recommendation: stop using transfer() or send() and switch to using call() instead.

Status: Fixed

2. Incorrect ETH recipient.

processPayment function always sends ETH to the getFeeWallet() and never to the “recepient”

Contract: ETHPool.sol

Functions: processPayment

Recommendation: make sure you’re sending ETH to the correct address.

Status: Fixed

Medium

No medium severity issues were found.

Low

1. Boolean equality.

Boolean constants can be used directly and do not need to be compared to **true** or **false**.

Contract: ETHPool.sol

Functions: unStake, claimReward

Recommendation: Remove the equality to the boolean constant.

Status: Fixed

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

2. Repeated function call.

calculateReward function called twice with the same arguments which burns excess gas.

Contract: ETHPool.sol

Functions: claimReward

Recommendation: Please save the result of the first call to the local variable.

Status: Fixed

3. A public function that could be declared external.

public functions that are never called by the contract should be declared **external** to save gas.

Contracts: ETHPool.sol, Ownable.sol, PlatformFees.sol

Functions: ETHPool.stake, ETHPool.unStake, ETHPool.claimReward, ETHPool.changeEpochTime, ETHPool.addNewEpoch, ETHPool.upcomingWeightOf, ETHPool.getStakeInfo, ETHPool.getUserIds, ETHPool.getUserIdIndex, Ownable.renounceOwnership, Ownable.transferOwnership, PlatformFees.setStakeFeeNumenator, PlatformFees.setStakeFeeDenominator, PlatformFees.setUnstakeFeeNumenator, PlatformFees.setUnstakeFeeDenominator, PlatformFees.setClaimFeeNumenator, PlatformFees.setclaimFeeDenominator, PlatformFees.setMinStakeAmount, PlatformFees.setStakeLimit, PlatformFees.setFeeWallet

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

Status: Fixed

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.

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

As a result of the audit, security engineers found **1** high and **3** low severity issues.

After the second review security engineers found that all previously found issues were fixed but **1** high severity issue was added to the code.

After the third review security engineers found that **all** issues were addressed.

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 the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free 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 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 can't guarantee the explicit security of the audited smart contracts.