HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Thorstarter
Date: October 25th, 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 Thorstarter.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Allocations; Sales; Staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	1. https://raw.githubusercontent.com/Thorstarter/thorstarter-contracts/d0e96db4b5ab5fd1ec64354ccd2d37636da78467/contracts/Tiers.sol 2. https://raw.githubusercontent.com/Thorstarter/thorstarter-contracts/fc8ab428222c668bd6ba8580d911565729fbdb72/contracts/SaleFcfs.sol 3. https://raw.githubusercontent.com/Thorstarter/thorstarter-contracts/73699f9b55f2dffa6e606958f266cecb03f540ae/contracts/StakingSingle.sol
Commit	1. d0e96db4b5ab5fd1ec64354ccd2d37636da78467 2. fc8ab428222c668bd6ba8580d911565729fbdb72 3. 73699f9b55f2dffa6e606958f266cecb03f540ae
Technical Documentation	YES
JS tests	YES
Website	thorstarter.org
Timeline	19 OCTOBER 2021 - 25 OCTOBER 2021
Changelog	25 OCTOBER 2021 - Initial 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 Thorstarter (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 October 19th, 2021 - October 25th, 2021.

Scope

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

Repository 1:
https://raw.githubusercontent.com/Thorstarter/thorstarter-contracts/d0e96db4b5ab5fd1ec64354ccd2d37636da78467/contracts/Tiers.sol
Commit 1:
d0e96db4b5ab5fd1ec64354ccd2d37636da78467

Repository 2:
https://raw.githubusercontent.com/Thorstarter/thorstarter-contracts/fc8ab428222c668bd6ba8580d911565729fbdb72/contracts/SaleFcfs.sol
Commit 2:
fc8ab428222c668bd6ba8580d911565729fbdb72

Repository 3:
https://raw.githubusercontent.com/Thorstarter/thorstarter-contracts/73699f9b55f2dffa6e606958f266cecb03f540ae/contracts/StakingSingle.sol
Commit 3:
73699f9b55f2dffa6e606958f266cecb03f540ae
Technical Documentation: Yes (few words description)
JS tests: Yes (included in “/tests/” directory)
Contracts:
SaleFcfs.sol
StakingSingle.sol
Tiers.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

Executive Summary

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

Insecure	Poor secured	Secured	Well-secured

You are here ↑

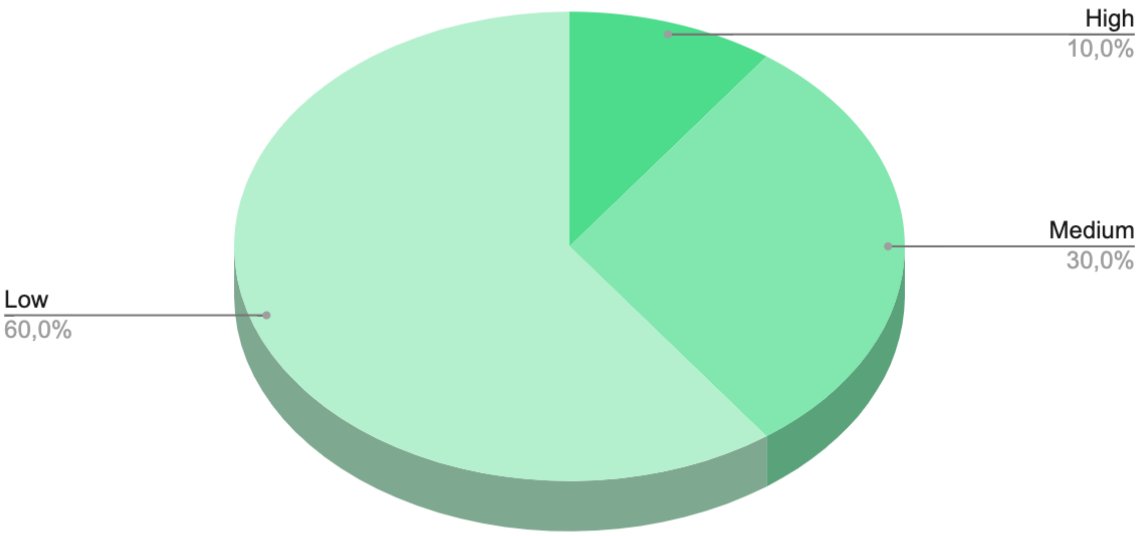
Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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, **3** medium and **6** low severity issues.

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 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

Possibility of missing rewards or receiving more

While StakingSingle contract has a function to change the rewardPerBlock value and this value is used to calculate rewards, there is a possibility when users that didn't update their pending rewards would receive more or fewer tokens than they should, depending on the comparison of the values.

It is also possible that setting rewardPerBlock to zero will not only cancel all future rewards, but also will zero all previously not updated rewards, since only.

Contracts: SaleFcfs.sol

Function: setRewardsPerBlock

Recommendation: There are two possible ways to resolve an issue:

1. try to call "update()" for each stakeholder before changing the "rewardsPerBlock" value. This would cost gas increasingly more for each new stakeholder, but that's the payment for changing rewards;
2. create a list of rewardsPerBlock changes with block numbers when it was changed and use that while calculating the rewards.

Medium

1. No event on changing critical values

Contract SaleFcfs changes the following values without emitting any events: tiers, tiersAllocation, tiersLevels, tiersMultipliers, raisingAmount, paused. Events are used to track such changes off-chain.

Contracts: SaleFcfs.sol

Function: configureTiers, setRaisingAmount, togglePaused

Recommendation: Please emit events on values changes.

2. No event on changing critical values

Contract StakingSingle changes the following values without emitting any events: rewardPerBlock. Events are used to track such changes off-chain.

Contracts: StakingSingle.sol

Function: setRewardsPerBlock

Recommendation: Please emit events on values changes.

3. Accessing state variable in the loop

Reading the state variable values (tokens.length(), nfts.length()) in the loop is very expensive in the gas manner. It is much more sufficient to save the value into the local variable and then use it in the code.

Contracts: Tiers.sol

Functions: userInfoAmounts, userInfoTotal, totalAmount

Recommendation: Please store values to the local variables.

Low

1. Constructor visibility.

Starting Solidity version 0.7.0 constructor visibility is ignored.

Contracts: Tiers.sol, StakingSingle.sol

Function: constructor

Recommendation: Please remove constructor visibility.

2. Too many digits

Literals with many digits are difficult to read and review.

Contracts: SaleFcfs.sol

Function: constructor

Recommendation: Please use either scientific notation or ether units suffix or both (ie. 1e10 instead of 10000000000)

3. Variable could be declared as immutable.

State variable which never changes its value and initialized in the constructor should be declared immutable to save gas.

Contracts: SaleFcfs.sol

Variables: paymentToken, offeringToken, startTime, endTime, offeringAmount, perUserCap, staking

Recommendation: Please use the **immutable** keyword to make such variables immutable and save gas.

4. Unused return value

Function “updateToken” calls EnumerableSet.add function without checking the boolean result of the operation.

Contracts: Tiers.sol

Function: updateToken

Recommendation: Please check the return value.

5. Unused return value

Function “updateNft” calls EnumerableSet.add function without checking the boolean result of the operation.

Contracts: Tiers.sol

Function: updateNft

Recommendation: Please check the return value.

6. 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: Tiers.sol, SaleFcfs.sol, StakingSingle.sol

Functions:
- Tiers.userInfoAmounts, Tiers.onTokenTransfer, Tiers.migrateRewards
- SaleFcfs.configureTiers, SaleFcfs.setRaisingAmount, SaleFcfs.togglePaused, SaleFcfs.finalize, SaleFcfs.deposit, SaleFcfs.onTokenTransfer, SaleFcfs.harvest, SaleFcfs.withdrawToken
- StakingSingle.setRewardsPerBlock, StakingSingle.deposit, StakingSingle.withdrawAndHarvest, StakingSingle.emergencyWithdraw

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

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, **3** medium and **6** low severity issues.

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.