HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Thorstarter
**Date**: August 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 (a part audit). |
| **Approved by** | Andrew Matiukhin \| CTO Hacken OU |
| **Type** | ERC20 token; Transfer controller |
| **Platform** | Ethereum / Solidity |
| **Methods** | Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| **Zip archive** | thorstarter-contracts-ido-eb1769901092068194e8cebf2965e5e13a0ad200.zip |
| **Files** | Sale.sol<br>SaleFloating.sol |
| **Technical Documentation** | NO |
| **JS tests** | YES |
| **Timeline** | 20 AUG 2021 - 25 AUG 2021 |
| **Changelog** | 25 AUG 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 Customer's smart contract and its code review conducted between Aug 20th, 2021 - Aug 25th, 2021.

# Scope

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

**Repository:**
https://github.com/Thorstarter/thorstarter-contracts
**Zip archive:**
thorstarter-contracts-ido-eb1769901092068194e8cebf2965e5e13a0ad200.zip
**Technical Documentation:** No
**JS tests:** Yes
**Contracts:**
Sale.sol
SaleFloating.sol

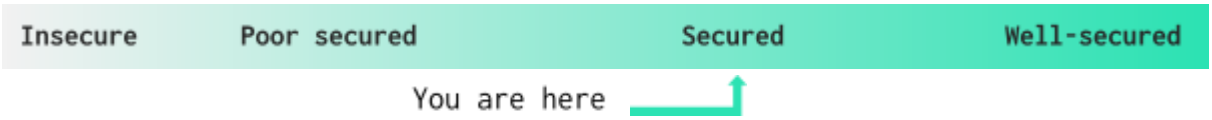
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

| Category | Check Item |
|---|---|
| Code review | ▪ Reentrancy<br>▪ Ownership Takeover<br>▪ Timestamp Dependence<br>▪ Gas Limit and Loops<br>▪ DoS with (Unexpected) Throw<br>▪ DoS with Block Gas Limit<br>▪ Transaction-Ordering Dependence<br>▪ Style guide violation<br>▪ Costly Loop<br>▪ ERC20 API violation<br>▪ Unchecked external call<br>▪ Unchecked math<br>▪ Unsafe type inference<br>▪ Implicit visibility level<br>▪ Deployment Consistency<br>▪ Repository Consistency<br>▪ Data Consistency |

| | |
|---|---|
| Functional review | ▪ Business Logics Review<br>▪ Functionality Checks<br>▪ Access Control & Authorization<br>▪ Escrow manipulation<br>▪ Token Supply manipulation<br>▪ Assets integrity<br>▪ User Balances manipulation<br>▪ Data Consistency manipulation<br>▪ Kill-Switch Mechanism<br>▪ Operation Trails & Event Generation |

## Executive Summary

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




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.

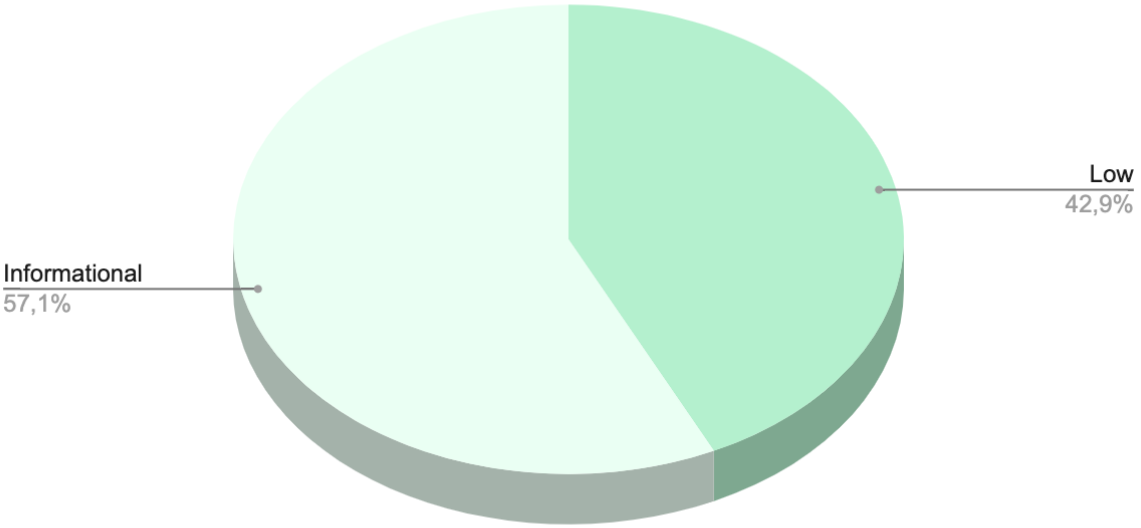
As a result of the audit, security engineers found **3** low and **4** informational severity issues.

**Notice:**

There are the same issues as in SaleFloating.sol in the Sale.sol file.

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

No high severity issues were found.

## Medium

No medium severity issues were found.

## Low

1. There are several redundant if conditions, which should be removed to decrease gas usage:
   a. if (_paymentAmount > 0) - redundant check
   b. if (_offeringAmount > 0) - redundant check

   **Contracts**:SaleFloating.sol, Sale.sol

   **Function**: finalWithdraw()

   **Recommendation**: remove redundant operations.

2. Increasing user deposit is forbidden. A client is able to create the deposit and buy tokens only once, without ability to call deposit function and buy more tokens.
   a. require(userInfo[msg.sender].amount == 0, 'already participated');

   **Contracts**:SaleFloating.sol

   **Function**: deposit()

   **Recommendation**: Allow client to increase user deposit

3. perUserCap condition check should be called earlier in this function. It will decrease gas usage in some situations.

   **Contracts**:Sale.sol

   **Function**: deposit()

   **Recommendation**: Allow client to increase user deposit

## Lowest

1. It is better to reuse already created and checked libs. We recommend you to use @openzeppelin Ownable instead of creating your own modifiers.

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

**Contracts**:SaleFloating.sol, Sale.sol

**Function**: modifier onlyOwner

**Recommendation**: use the library.

2. There should be ‘less or equal’ and ‘greater or equal’ operators, not strict less and greater.
   a. require(block.number > startBlock && block.number < endBlock, 'sale not active');

**Contracts**:SaleFloating.sol, Sale.sol

**Function**: deposit

**Recommendation**: Change operator.

3. It is a good practice to prepare the basic technical documentations of the contract. Goals of the functions like balanceOfAt() is unobvious

**Contracts**:SaleFloating.sol, Sale.sol

**Recommendation**: Create basic technical documentation.

4. Each next token, which would be bought by the client, will cost more then previous. But it produces unequal conditions for buyers depending on order size. In this case, it will be better to use Volume Weighted Average price as a price rise delta.
   a. uint price = startPrice + ((totalAmount * priceVelocity) / 1e18);

**Contracts**:SaleFloating.sol, Sale.sol

**Function**: deposit()

**Recommendation**: change the price changing algorithm.

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.

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

As a result of the audit, security engineers found **3** low and **4** informational severity issues.

**Notice:**

There are the same issues as in SaleFloating.sol in the Sale.sol file.

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, 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 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.