HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Thorstarter
Date: August 20th, 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	Custom ERC20-like token with DAO, locking and no transfer methods
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/Thorstarter/thorstarter-contracts-ido/blob/master/contracts/Voters.sol
Commit	eb1769901092068194e8cebf2965e5e13a0ad200
Technical Documentation	NO
JS tests	NO
Timeline	18 AUGUST 2021 - 20 AUGUST 2021
Changelog	20 AUGUST 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 security assessment findings of the Customer's smart contract and its code review conducted between August 18th, 2021 - August 20th, 2021.

Scope

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

Repository:
https://github.com/Thorstarter/thorstarter-contracts-ido/blob/master/contracts/Voters.sol
Commit:
eb1769901092068194e8cebf2965e5e13a0ad200
Technical Documentation: No
JS tests: No
Contracts:
Voters.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 ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are secured but have slightly gas over usage.

Insecure	Poor secured	Secured	Well-secured

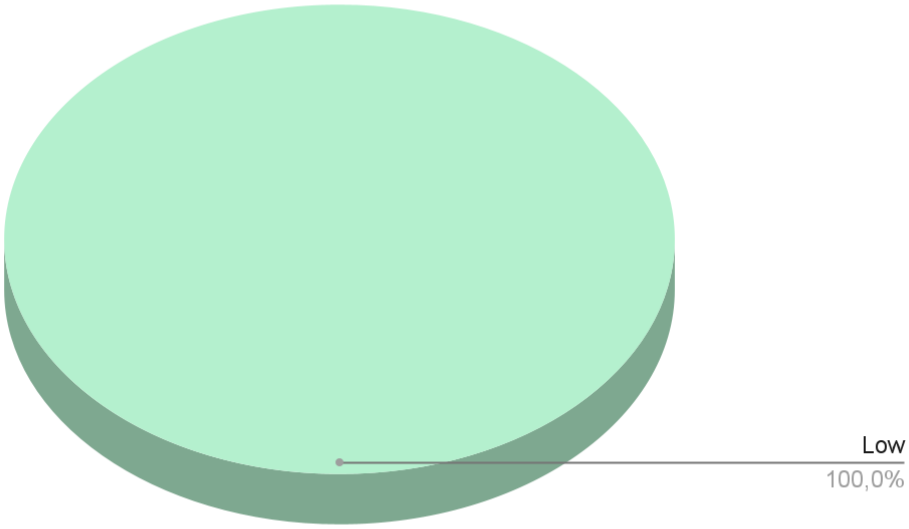
You are here

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 **3** 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

No high severity issues were found.

Medium

No medium severity issues were found.

Low

1. State variables that could be declared constant

   Constant state variables should be declared constant to save gas.

   Recommendation: Add the constant attributes to state variables that never change.

Lines: #37-38

```
string public name = "Thorstarter Voting Token";
string public symbol = "vXRUNE";
```

2. Public function that could be declared external

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

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

Lines: #63

```
function userInfo(address user) public view returns (uint, uint, uint,
uint, uint, uint, address) {
```

Lines: #81

```
function balanceOfAt(address user, uint snapshotId) override public view
returns (uint) {
```

Lines: #90

```
function votesAt(address user, uint snapshotId) override public view
returns (uint) {
```

Lines: #95

```
function totalSupplyAt(uint snapshotId) override public view returns
(uint) {
```

Lines: #262

```
function updateTclp(address[] calldata users, uint[] calldata amounts,
```

```
uint[] calldata values) public {
```

Lines: #315

```
function historicalTcLpsList(uint page, uint pageSize) public view
returns (address[] memory) {
```

Lines: #327

```
function donate(uint amount) override public {
```

3. This declaration shadows an existing declaration

   The local variable in multiple functions, as well as an argument in one function are shadows the function name. It is hard to distinguish between which one is used in the code.

   Recommendation: Please rename local variables so those have the name different from the function name.

Lines: #63

```
function userInfo(address user) public view returns (uint, uint, uint,
uint, uint, uint, address) {
```

Lines: #64

```
UserInfo storage userInfo = _userInfos[user];
```

Lines: #77

```
UserInfo storage userInfo = _userInfos[user];
```

Lines: #162

```
UserInfo storage userInfo = _userInfos[msg.sender];
```

Lines: #176

```
UserInfo storage userInfo = _userInfo(msg.sender);
```

Lines: #191

```
UserInfo storage userInfo = _userInfo(msg.sender);
```

Lines: #209

```
UserInfo storage userInfo = _userInfo(msg.sender);
```

Lines: #242

```
UserInfo storage userInfo = _userInfo(msg.sender);
```

Lines: #267

```
UserInfo storage userInfo = _userInfo(user);
```

Lines: #292

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

```
UserInfo storage userInfo = _userInfos[user];
```

Lines: #323

```
function _userInfoTotal(UserInfo storage userInfo) private view returns
(uint) {
```

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 **3** 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.