HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: TosDis
Date: January 19th, 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 TosDis Finance
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token sale contract
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/tosdis/Contracts/blob/main/ITOPool.sol
Commit	
Deployed contract	
Timeline	19 JAN 2021
Changelog	19 JAN 2021 - Initial Audit 19 JAN 2021 - Remediation Check 20 JAN 2021 - Remediation Check

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

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

Table of contents

This document is proprietary and confidential. No part of this document may be disclosed
in any manner to a third party without the prior written consent of Hacken.

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

Introduction

Hacken OÜ (Consultant) was contracted by TosDis Finance (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 on January 19th, 2021.

Remediation Check conducted - Jan 19th, 2021
Remediation Check conducted - Jan 20th, 2021

Scope

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

```
Contract deployment address:
Repository
Commit
Files:
      ITOPool.sol
(b5d8da0cf1f4c310bcc3a3d0853fac0e7b125c52c6d8084df70c61b96ec3d5a6)
```

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

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

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 secure and can be used in production.

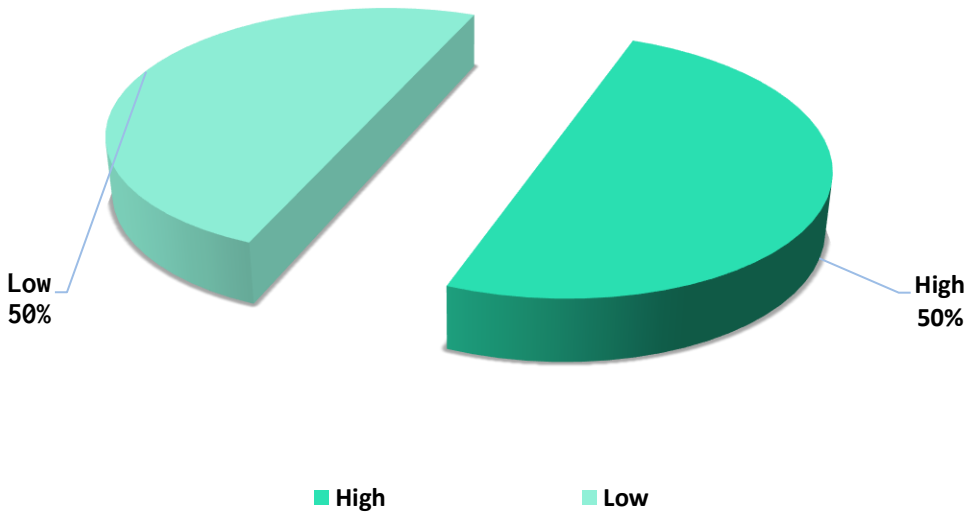
Insecure	Poor secured	Secured	Well-secured

You are

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.

After the first review, Customers` smart contracts contained **1** high and **1** low severity issue.

Graph 1. Distribution of vulnerabilities after the first review.

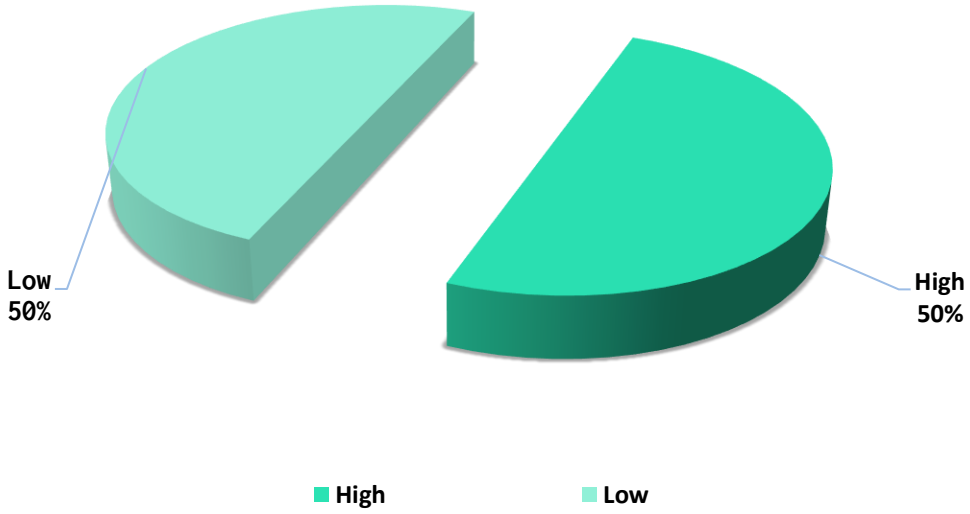


This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

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

After the second review, Customers` smart contracts contains **1** high and **1** low severity issue.

Graph 2. Distribution of vulnerabilities after the second review.



After the third review, Customers` smart contracts do not contain vulnerabilities.

Notice for the contract users: before using the contract ensure that all parameters are set correctly, and the contract has at least `maxDistributedTokenAmount / tokenPrice * 10^decimals` of tokens on its balance.

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

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.

This document is proprietary and confidential. No part of this document may be disclosed
in any manner to a third party without the prior written consent of Hacken.

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

AS-IS overview

ITOPool.sol

Description

ITOPool is a token sale contract.

Imports

ITOPool contract has following imports:

- openzeppelin/contracts/access/Ownable.sol
- openzeppelin/contracts/utils/ReentrancyGuard.sol
- openzeppelin/contracts/math/SafeMath.sol
- openzeppelin/contracts/token/ERC20/SafeERC20.sol
- openzeppelin/contracts/token/ERC20/ERC20.sol

Inheritance

ITOPool contract is Ownable, ReentrancyGuard.

Usages

ITOPool contract has the following custom usages:

- SafeMath for uint256
- SafeERC20 for ERC20

Structs

ITOPool contract has following data structures:

- UserInfo

Enums

ITOPool contract has no custom enums.

Events

ITOPool contract has the following events:

- UpdatedSettings
- TokensDebt
- TokensWithdrawn

Modifiers

ITOPool has no custom modifiers.

This document is proprietary and confidential. No part of this document may be disclosed
in any manner to a third party without the prior written consent of Hacken.

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

Fields

ITOPool contract has following fields and constants:

- uint256 public tokenPrice
- ERC20 public rewardToken
- uint256 public decimals
- uint256 public startTimestamp
- uint256 public finishTimestamp
- uint256 public startClaimTimestamp
- uint256 public minEthPayment
- uint256 public maxEthPayment
- uint256 public maxDistributedTokenAmount
- uint256 public tokensForDistribution
- uint256 public distributedTokens
- mapping(address => UserInfo) public userInfo;

Functions

ITOPool has following public functions:

- ***constructor***
  Description
  Sets default parameters of the contract.
  Visibility
  public
  Input parameters
  - uint256 _tokenPrice
  - ERC20 _rewardToken
  - uint256 _startTimestamp
  - uint256 _finishTimestamp
  - uint256 _startClaimTimestamp
  - uint256 _minEthPayment
  - uint256 _maxEthPayment
  - uint256 _maxDistributedTokenAmount

  Constraints
  - Start timestamp must be less than finish timestamp.
  - Finish timestamp must be more than current block.

  Events emit
  None
  Output
  None
- ***pay***
  Description
  Pay ETH in exchange to tokens. Tokens withdrawal will be available after `startClaimTimestamp` is reached.

This document is proprietary and confidential. No part of this document may be disclosed
in any manner to a third party without the prior written consent of Hacken.

Visibility
payable external
Input parameters
None
Constraints
- msg.value should be between `minEthPayment` and `maxEthPayment`.
- Current timestamp should be between `startTimestamp` and `finishTimestamp`
- Total purchase sum should not exceed the `maxEthPayment`.

Events emit
Emits the `TokensDebt` event.
Output
None

- ***claimFor***
  Description
  Claims tokens on behalf of a `_user`.
  Visibility
  external
  Input parameters
  - address _user

  Constraints
  - Can only be called after the `startClaimTimestamp` is reached.

  Events emit
  Emits the `TokensWithdrawn` event.
  Output
  None
- ***claim***
  Description
  Claims tokens on behalf of a message sender.
  Visibility
  external
  Input parameters
  None
  Constraints
  - Can only be called after the `startClaimTimestamp` is reached.

  Events emit
  Emits the `TokensWithdrawn` event.
  Output
  None
- ***withdrawETH***
  Description
  Withdraw an `amount` of eth.

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

Visibility
external
Input parameters
- uint256 amount

Constraints
- Can only be called by the owner.

Events emit
None
Output
None

- ***withdrawNotSoldTokens***
  Description
  Withdraw all unsold tokens.
  Visibility
  external
  Input parameters
  - uint256 amount

  Constraints
  - Can only be called by the owner.
  - `finishTimestamp` should be reached.

  Events emit
  None
  Output
  None
- ***setFinishTimestamp, setStartClaimTimestamp, setMaxDistributedTokenAmount***
  Description
  Setter functions. Can only be used by the owner.
  All functions were removed.

This document is proprietary and confidential. No part of this document may be disclosed
in any manner to a third party without the prior written consent of Hacken.

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. setStartClaimTimestamp can be used by owners to change the `startClaimTimestamp` value. As a result, customers who already purchased tokens may not receive them when they expect to.

   Fixed before the second review. The function was removed.

2. setFinishTimestamp can be used by owners to change the `finishTimestamp` value. This allows to transfer unsold tokens at any time. As a result, customers who already purchased tokens may not receive them when they expect to.

   Fixed before the second review. The function was removed.

Medium

No medium severity issues were found.

Low

1. Usage of the `ReentrancyGuard` is redundant..

Fixed before the second review.

Lowest / Code style / Best Practice

No informational issues were found.

This document is proprietary and confidential. No part of this document may be disclosed
in any manner to a third party without the prior written consent of Hacken.

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.

After the first review, Customers` smart contracts contained **1** high and **1** low severity issue.

After the second review, Customers` smart contracts contains **1** low severity issue.

After the third review, Customers` smart contracts contains **1** low severity issue.

Notice for the contract users: before using the contract ensure that all parameters are set correctly, and the contract has at least `maxDistributedTokenAmount / tokenPrice * 10^decimals` of tokens on its balance.

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

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.

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.