SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: TrustSwap
Date: March 7th, 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 TrustSwap - Draft
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Vesting
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Deployed contract	https://ropsten.etherscan.io/address/0xf518014ad3a6b0b758f9742f8fb082730ea95805#code
Timeline	04 March 2021 - 07 March 2021
Changelog	07 March 2021 - INITIAL AUDIT 13 March 2021 - SECOND REVIEW

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

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by TrustSwap (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 March 7th, 2021.

Remediation check was done March 15th, 2021.

Scope

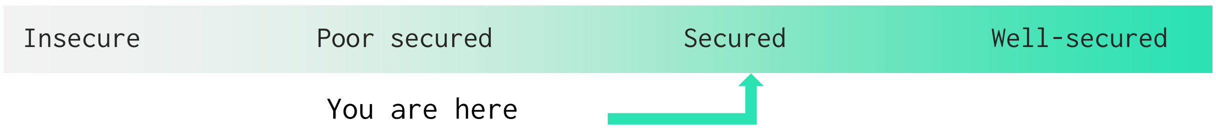
The scope of the project is smart contracts in the repository:
Contract deployment address:
https://ropsten.etherscan.io/address/0x8733caa60eda1597336c0337efde27c1335f7530#code
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known 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 - Asset’s integrity - User Balances manipulation - Kill-Switch Mechanism - Operation Trails & Event Generation

Executive Summary

According to the assessment, after remediation check, the Customer's smart is Secured.

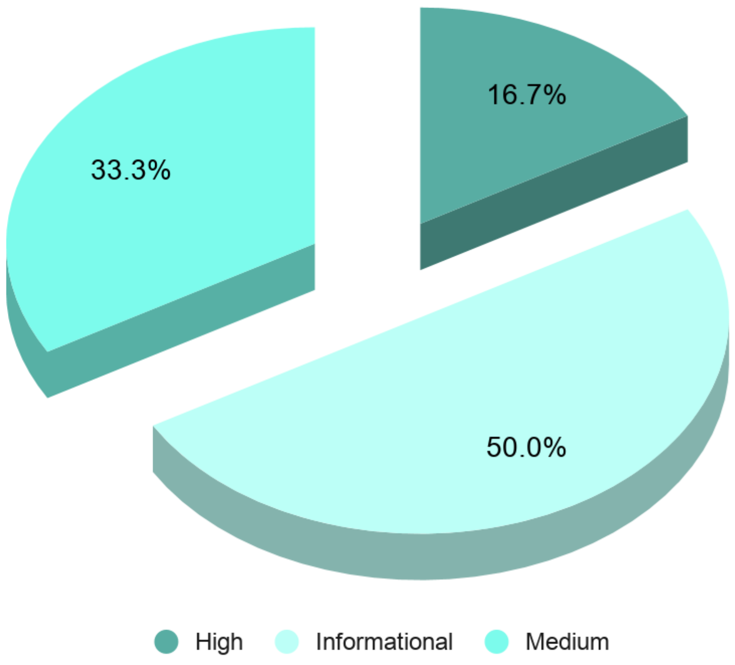


Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section, and all found issues can be found in the Audit overview section.

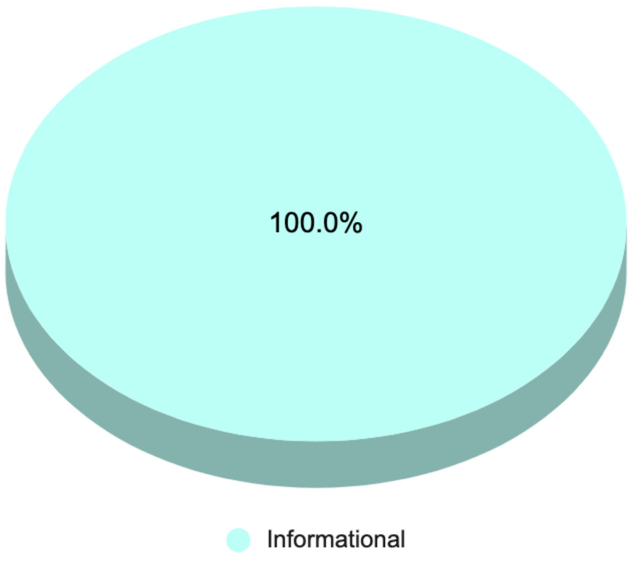
Security engineers found 1 high, 2 medium, 3 informational issues during first review.

After the second review, security engineers found 3 informational issues.

Graph 1. The distribution of vulnerabilities after the first review.



Graph 2. The distribution of vulnerabilities after the second review.



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.

Audit overview

■ ■ ■ ■ Critical

No Critical severity issues were found.

■ ■ ■ High

1. Vulnerability: Nested Loops
Contract: lockToken
Method(s):
    - ➢ transferLocks(uint256 _id, address _receiverAddress)
    - ➢ withdrawTokens(uint256 _id)

Fixed before the second review.

■ ■ Medium

1. Vulnerability: Redundant approve
Contract: lockToken
Method(s):
    - ➢ lockTokens(address _tokenAddress, address _withdrawalAddress, uint256 _amount, uint256 _unlockTime)
    - ➢ createMultipleLocks(address _tokenAddress, address _withdrawalAddress, uint256[] _amounts, uint256[] _unlockTimes)

Fixed before the second review.

2. Vulnerability: Using `.length` in for loop
Contract: lockToken
Method: withdrawTokens(uint256 _id)

Fixed before the second review.

■ Lowest / Code style / Best Practice

1. Lines 95, 100, 119, 129, 167, 170, 177, 200, 207, 237 of the code are above the recommended maximum line length.
2. Contracts (lockToken) and libraries (owned) should be named using the CapWords style.
3. Lines 167-178 and 200-208 consider using a variable for best reading and eliminating continuous access to the state.

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.

Security engineers found 1 high, 2 medium, 3 informational issue during the audit.

After second review security engineers found 3 informational issues.

Category	Check Items	Comments
Code Review	Maximum Code Length	Recommended maximum line length.
	Naming Style	Contracts and libraries should be named using the CapWords style
	Readability	Consider using a variable for best reading and eliminating continuous access to the state

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.