HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: ScaleSwap
Date: May 6th, 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 ScaleSwap.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Multiple purposes contracts
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/scaleswap-io/contracts
Commit	B2449EA1CB5FCB16315877F84DFA36CDEE99EA48
Deployed contract	
Timeline	29 APR 2021 - 06 MAY 2021
Changelog	06 MAY 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 ScaleSwap (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 April 29th, 2021 - May 06th, 2021.

Scope

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

```
Repository:  https://github.com/scaleswap-io/contracts
Commit: B2449EA1CB5FCB16315877F84DFA36CDEE99EA48
Files:
      ├── ScaleSwapToken.sol
      ├── interfaces
      │   ├── IERC20Detailed.sol
      │   ├── IERC20DetailedBurnable.sol
      │   └── TokensaleCommon.sol
      ├── mocks
      │   └── TokenMock.sol
      └── tokensale
          ├── ScaleSwap.sol
          ├── ScaleSwapAccessControl.sol
          └── ScaleSwapFactory.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

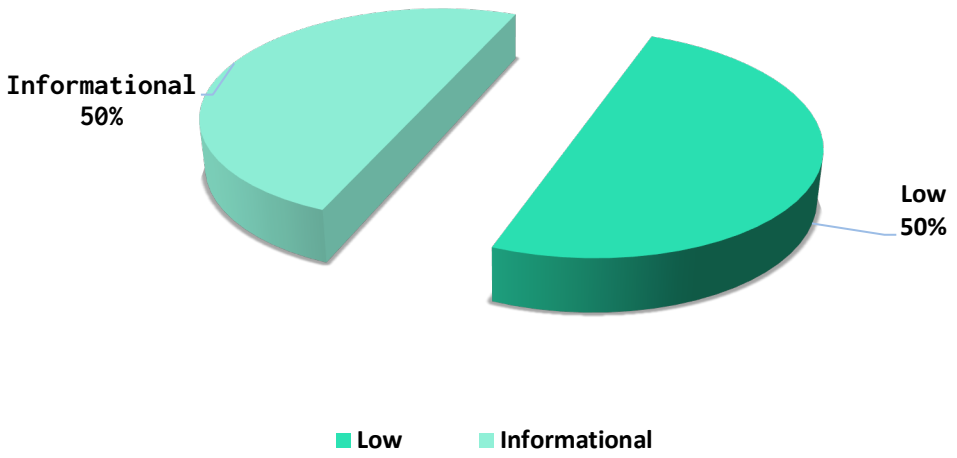
According to the assessment, the Customer's smart contracts are well-secured.

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. A general overview is presented in AS-IS section, and all found issues can be found in the Audit overview section.

Security engineers found **2** low, and **2** informational issues during the audit.

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



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.

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 issues were found.

Medium

No medium issues were found.

Low

1. Administrative keys are highly permissive and should be moved to a multi-signature wallet.

Customer notice: “That is what we do in production”.

2. In the event that a transaction fails due to overflow, this is not explicitly stated to the user as the contract takes advantage of the 0.8.0 and above overflow arithmetic. Where possible, this should be made explicit.

Customer notice: “Ok, we will proceed with such behaviour. It's ok for us as soon as our clients should only use our UI to communicate with contracts and all edge cases are validated before sending a transaction”.

Lowest / Code style / Best Practice

1. Token approval and fee changes should emit an event.
2. Multiple typos exist within interface code, namely those within MetaTxToken.

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 the As-Is overview section of the report.

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

Security engineers found **2** low, and **2** informational issues during the audit.

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.