HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: ScaleSwap
Date: June 22nd, 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 are fixed - upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for ScaleSwap - Initial Audit
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 Token using LiquidityProtection
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Git repository	https://github.com/scaleswap-io/SCA-token-protected
Zip Archive	SCA-token-protected-main.zip (**md5:** 6519c6edfc008a4d9964b470fc4c197a)
Changelog	22 JUNE 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 on June 22nd, 2021.

Scope

The scope of the project is the smart contracts in Zip Archive:

```
Repository:
      https://github.com/scaleswap-io/SCA-token-protected

Zip archive:
      SCA-token-protected-main.zip (md5: 6519c6edfc008a4d9964b470fc4c197a)

File:
      ScaleSwapToken.sol  md5: bd14d3140eab4b9dea372949ed6e7988
```

We have scanned these smart contracts 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

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

Executive Summary

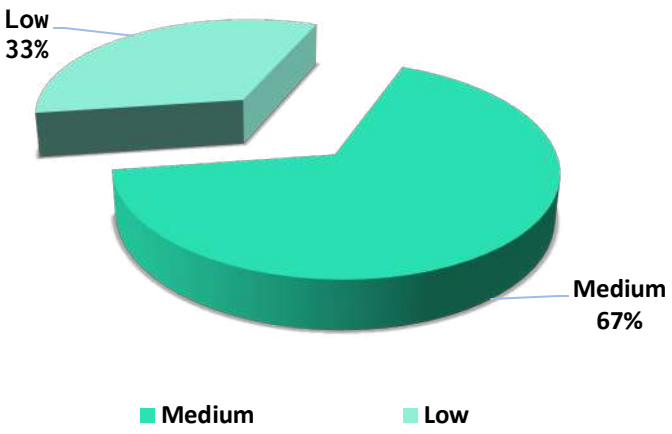
According to the assessment, the Customer's smart contract is secure but consist of non-audible external dependency.

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.

Security engineers found **1** low and **2** informational issues during the first review.

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 severity issues were found.

High

No High severity issues were found.

Medium

No Medium severity issues were found.

Low

Vulnerability: Non-audible External Dependency

All token transfers which would be done before **Monday, June 28, 2021 11:59:59 PM GMT** are subject to be pre-processed by the external smart contract on address **0x9420FE350eEF12778bDA84f61dB0FcB05aaE31C5** which is not exposed its source code thus cannot be audited.

Recommendation: Please consider not using non-audible external dependencies or please as 3rd party to upload verified security audit to their contract address (0x9420FE350eEF12778bDA84f61dB0FcB05aaE31C5)

Lines: #733-735

```
function liquidityProtectionService() internal pure override
returns(address) {
    return 0x9420FE350eEF12778bDA84f61dB0FcB05aaE31C5;
}
```

Lines: #641-643

```
function lps() private pure returns(IPLPS) {
    return IPLPS(liquidityProtectionService());
}
```

Lines: #645-652

```
function LiquidityProtection_beforeTokenTransfer(address _from, address
_to, uint _amount) internal virtual {
    if (protectionChecker()) {
        if (!protected) {
            return;
        }
```

```
lps().LiquidityProtection_beforeTokenTransfer(getLiquidityPool(),
_from, _to, _amount);
    }
}
```

Lines: #745-748

```
function _beforeTokenTransfer(address _from, address _to, uint _amount)
internal override {
    super._beforeTokenTransfer(_from, _to, _amount);
    LiquidityProtection_beforeTokenTransfer(_from, _to, _amount);
}
```

Lines: #752-756

```
    function protectionChecker() internal view override returns(bool) {
        return ProtectionSwitch_timestamp(1624924799); // Switch off
protection on Monday, June 28, 2021 11:59:59 PM GTM.
       // return ProtectionSwitch_block(13000000); // Switch off
protection on block 13000000.
//        return ProtectionSwitch_manual(); // Switch off protection by
calling disableProtection(); from owner. Default.
    }
```

Client comments: this part of the code is an integration for 3rd party anti-signal bot solution(e.g. LiquidityProtection).

Lowest / Code style / Best Practice

1. **Vulnerability**: Too many digits

   Literals with many digits are difficult to read and review

   Recommendation: Please consider using ether suffix and scientific notation

   Lines: #764

   ```
   _mint(owner(), 25000000 * 1e18);
   ```

2. **Vulnerability:** Public function that could be declared external

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

   Lines: #679

   ```
   function isProtected() public view returns(bool) {
   ```

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.

Security engineers found **1** low and **2** informational issues during the first review.

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 the 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.