HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer:** Securitance
**Date:** May 4th, 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 Sekuritance - Initial Audit |
| **Approved by** | Andrew Matiukhin \| CTO Hacken OU |
| **Type** | ERC20 Token |
| **Platform** | Ethereum / Solidity |
| **Methods** | Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| **Deployed mainnet** | https://etherscan.io/address/0x887168120cb89fb06f3e74dc4af20d67df0977f6#code |
| **Timeline** | 04 MAY 2021 - 04 MAY 2021 |
| **Changelog** | 04 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 Sekuritance (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 May 4th, 2021.

# Scope

The scope of the project is the smart contract deployed in the Ethereum mainnet:

https://etherscan.io/address/0x887168120cb89fb06f3e74dc4af20d67df0977f6#code

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<br>▪ Ownership Takeover<br>▪ Timestamp Dependence<br>▪ Gas Limit and Loops<br>▪ DoS with (Unexpected) Throw<br>▪ DoS with Block Gas Limit<br>▪ Transaction-Ordering Dependence<br>▪ Style guide violation<br>▪ Costly Loop<br>▪ ERC20 API violation<br>▪ Unchecked external call<br>▪ Unchecked math<br>▪ Unsafe type inference<br>▪ Implicit visibility level<br>▪ Deployment Consistency<br>▪ Repository Consistency<br>▪ Data Consistency |

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

| | |
|---|---|
| Functional review | ▪ Business Logics Review<br>▪ Functionality Checks<br>▪ Access Control & Authorization<br>▪ Escrow manipulation<br>▪ Token Supply manipulation<br>▪ Asset’s integrity<br>▪ User Balances manipulation<br>▪ Kill-Switch Mechanism<br>▪ 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 secured

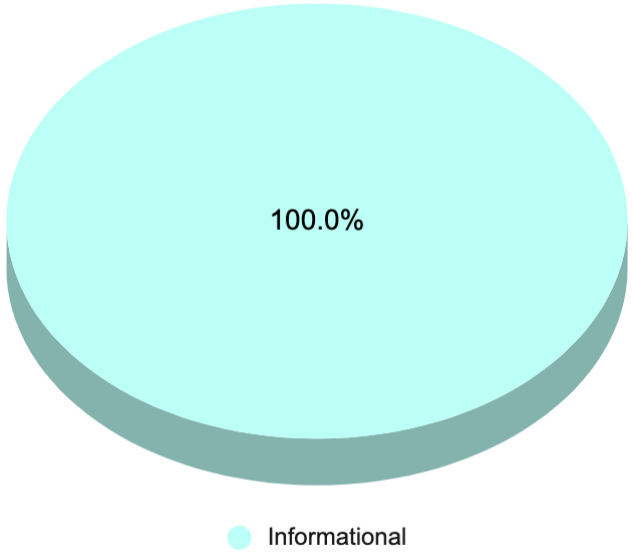
| Insecure | Poor secured | Secured | Well-secured |
|---|---|---|---|

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. All found issues can be found in the Audit overview section.

Security engineers found **4** 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

No Low severity issues were found.

## Lowest / Code style / Best Practice

1. **Vulnerability:** Incorrect versions of Solidity

   Solidity version 0.5.12 is not recommended for deployment. Please consider using any of the following Solidity versions:

   - 0.5.16 - 0.5.17
   - 0.6.11 - 0.6.12
   - 0.7.5 - 0.7.6

   Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

2. **Vulnerability:** State variable that should be immutable

   State variables that got initialized in the constructor and don't change their value should be declared immutable to save gas.

   **Lines**: #443-445

   ```
   string private _name;
   string private _symbol;
   uint8 private _decimals;
   ```

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

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

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

**Lines**: #471

```
function name() public view returns (string memory) {
```

**Lines**: #480

```
function symbol() public view returns (string memory) {
```

**Lines**: #487

```
function decimals() public view returns (uint8) {
```

**Lines**: #494

```
function totalSupply() public view override returns (uint256) {
```

4. Line 455 is above the recommended maximum line length.

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 **4** informational issues during the first review.

| Category | Check Items | Comments |
|---|---|---|
| Code Review | Style guide violation | ➔ Public function that could be declared external<br>➔ State variable should be immutable<br>➔ Maximum line length<br>➔ Incorrect solidity version |

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.