# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Plethori
**Date**: July 19th, 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 Plethori - 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 |
| **Zip archive** | PLE-Token-4784b781b574c7e4f1d09c77b96a0e2ac5f08565.zip (md5: 9154352b5daea84ae316edb18c6c58d4) |
| **Timeline** | 12 July 2021 - 19 July 2021 |
| **Changelog** | 12 July 2021 - Initial Audit<br>19 July 2021 - Second Review |

## Table of contents

## Introduction

Hacken OÜ (Consultant) was contracted by Plethori (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 July 12th, 2021 - July 15th, 2021. The second review conducted on July 19th, 2021.

## Scope

The scope of the project is the smart contracts in zip archive:

Zip archive:
PLE-Token-4784b781b574c7e4f1d09c77b96a0e2ac5f08565.zip
md5: 9154352b5daea84ae316edb18c6c58d4

Scope:
PleToken.sol md5: 9b293a1ba350de8c2b4e2d05a2cde759

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 |

| | |
|---|---|
| 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 |

## Executive Summary

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

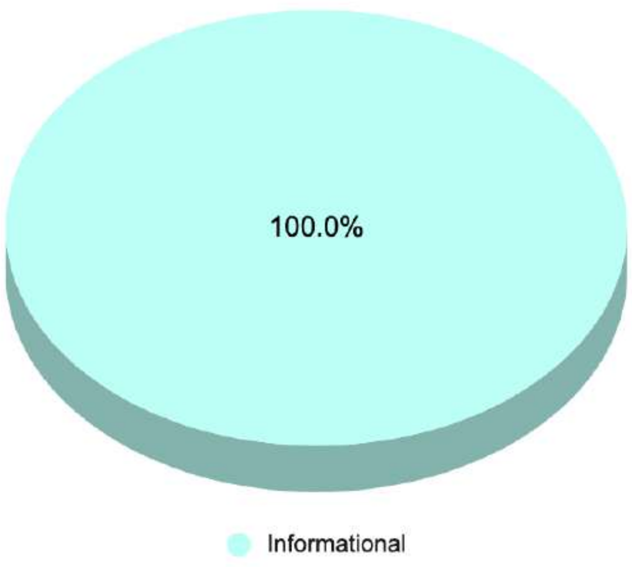
| Insecure | Poor secured | Secured | Well-secured |
|---|---|---|---|
| | | You are here | ↑ |

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

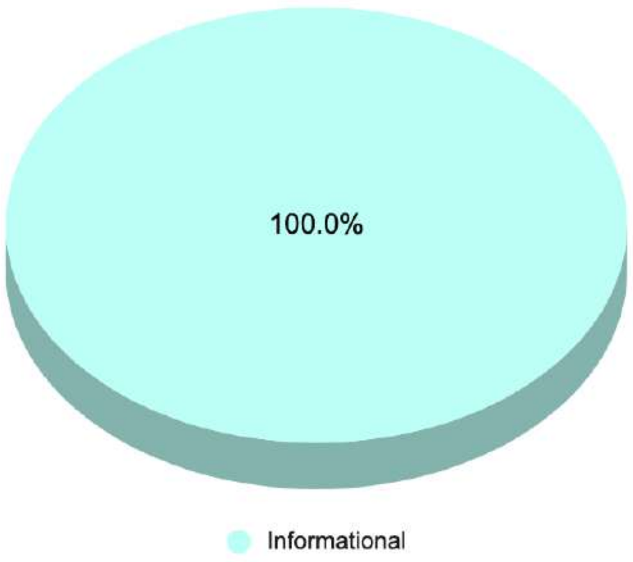
Security engineers found **3** informational issues during the first review.

Security engineers found **1** informational issue during the second review.

*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

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:** Too many digits.

   Literals with many digits are difficult to read and review.

   **Recommendation**: Please consider using ether suffix and the scientific notation. Example: **100e6 ether**

   **Fixed before the second review.**

2. **Vulnerability:** SafeMath in solidity >0.8.

   Starting solidity version 0.8 and later math is already safe from the over and underflow. And it's not needed to check the result anymore.

   **Recommendation:** Please consider not using SafeMath with solidity >= 0.8.

   **Fixed before the second review.**

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.

   **Lines**: #222

```
function transferFrom(address sender, address recipient, uint256
amount) public override returns (bool) {
```

## 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 **3** informational issues during the first review.

Security engineers found **1** informational issue during the second review.

## 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.