

**HACKEN**

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer:** Snail Trail  
**Date:** April 18<sup>th</sup>, 2022

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

<b>Name</b>	Smart Contract Code Review and Security Analysis Report for Snail Trail.
<b>Approved By</b>	Evgeniy Bezuglyi   SC Department Head at Hacken OU
<b>Type of Contracts</b>	ERC721 token; Token sale
<b>Platform</b>	EVM
<b>Language</b>	Solidity
<b>Methods</b>	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
<b>Website</b>	<a href="https://www.snailtrail.art/">https://www.snailtrail.art/</a>
<b>Timeline</b>	30.03.2022 - 18.04.2022
<b>Changelog</b>	02.04.2022 - Initial Review 08.04.2022 - Revising 18.04.2022 - Second Revision



## Table of contents

Introduction	4
Scope	4
Executive Summary	5
Severity Definitions	7
Findings	8
Disclaimers	10

## Introduction

Hacken OÜ (Consultant) was contracted by Snail Trail (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

## Scope

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

**Repository:**

**Commit:**

**Documentation:** Yes (<https://docs.snailtrail.art/>)

**JS tests:** Yes

**Contracts:**

`./contracts/ERC721/SnailTrailNFT.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	<ul style="list-style-type: none"> <li>▪ Reentrancy</li> <li>▪ Ownership Takeover</li> <li>▪ Timestamp Dependence</li> <li>▪ Gas Limit and Loops</li> <li>▪ Transaction-Ordering Dependence</li> <li>▪ Style guide violation</li> <li>▪ EIP standards violation</li> <li>▪ Unchecked external call</li> <li>▪ Unchecked math</li> <li>▪ Unsafe type inference</li> <li>▪ Implicit visibility level</li> <li>▪ Deployment Consistency</li> <li>▪ Repository Consistency</li> </ul>
Functional review	<ul style="list-style-type: none"> <li>▪ Business Logics Review</li> <li>▪ Functionality Checks</li> <li>▪ Access Control &amp; Authorization</li> <li>▪ Escrow manipulation</li> <li>▪ Token Supply manipulation</li> <li>▪ Assets integrity</li> <li>▪ User Balances manipulation</li> <li>▪ Data Consistency</li> <li>▪ Kill-Switch Mechanism</li> </ul>

## Executive Summary

The score measurements details can be found in the corresponding section of the [methodology](#).

### Documentation quality

The Customer provided functional requirements and superficial technical requirements. The total Documentation Quality score is **8** out of **10**.

### Code quality

The total CodeQuality score is **10** out of **10**. The code is clean and clear. Unit tests were provided.

### Architecture quality

The architecture quality score is **8** out of **10**. Storing library code in the repository is against best practices.

### Security score

As a result of the audit, security engineers found **2** high, **1** medium, and **2** low severity issues.

As a result of the revision, security engineers found **no** new issues, **1** high, **1** medium, and **2** low previously found severity issues were fixed, and **1** high severity issue was mitigated to medium severity level.

As a result of the second revision, security engineers found **1** new medium severity issue.

The security score is **10** out of **10**. All found issues are displayed in the "Findings" section.

### Summary

According to the assessment, the Customer's smart contract has the following score: **9.6**

