

HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: SnapEx

Date: April 4th, 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

Name	Smart Contract Code Review and Security Analysis Report for SnapEx.
Approved By	Evgeniy Bezuglyi SC Department Head at Hacken OU
Type of Contracts	ERC-20 token
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://token.snapex.com/
Timeline	01.04.2022-04.04.2022
Changelog	04.04.2022 - Initial Review



Table of contents

Introduction	4
Scope	4
Executive Summary	5
Severity Definitions	7
Findings	8
Disclaimers	9

Introduction

Hacken OÜ (Consultant) was contracted by SnapEx (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 deployed smart contracts:

Deployed contract:

<https://bscscan.com/address/0x57cd5a91c18d21dedd72d3bcf255b60aef290f53#code>

Technical Documentation: Yes

<https://token.snapex.com/wp-content/uploads/2021/11/Whitepaper-November-2021.pdf>

JS tests: No

Contracts:

SnapToken

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

Executive Summary

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

Documentation quality

The Customer provided a whitepaper with functional and technical requirements. The code contains detailed documentation. The total Documentation Quality score is **10** out of **10**.

Code quality

The total CodeQuality score is **5** out of **10**. Code follows official language style guides, but no unit tests were provided.

Architecture quality

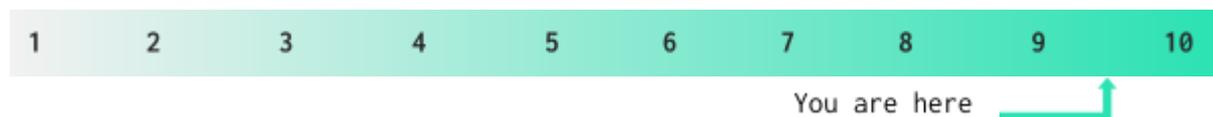
The architecture quality score is **10** out of **10**. Smart contract use best practices.

Security score

As a result of the audit, security engineers found **2** low severity issues. The security score is **10** out of **10**. All found issues are displayed in the “Findings” section.

Summary

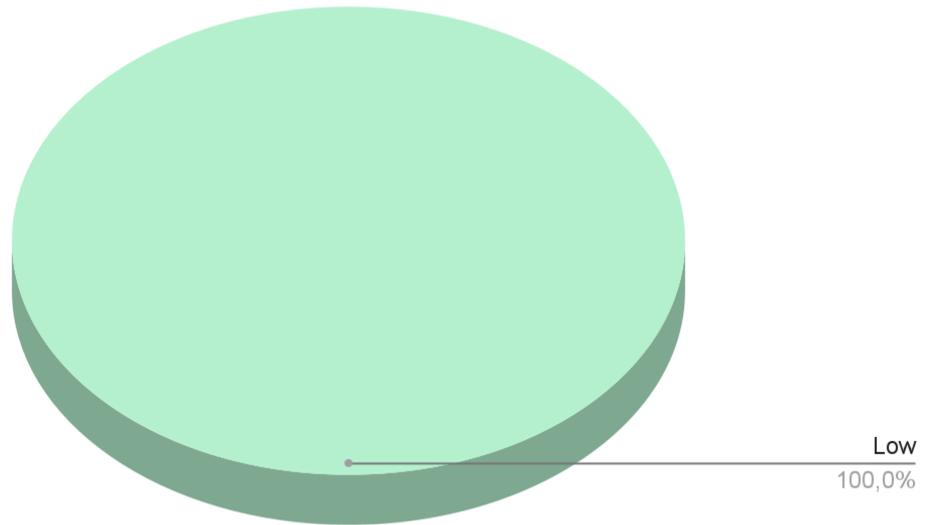
According to the assessment, the Customer's smart contract has the following score: **9.5**



Notices

1. The owner has the ability to stop all token transfers using the pause function. The need for this functionality is described in the [article](#).

Graph 1. The distribution of vulnerabilities after the audit.



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 cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution

Findings

Critical

No critical severity issues were found.

High

No high severity issues were found.

Medium

No medium severity issues were found.

Low

1. Floating pragma

The testToken.sol contract use floating pragma ^0.5.3

Contracts: testToken.sol

Recommendation: Please, consider locking the pragma version whenever possible.

2. Deprecated pragma

The testToken.sol contract use floating pragma ^0.5.3 which is deprecated

Contracts: testToken.sol

Recommendation: Please, use the latest version of the Solidity language.



Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with 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 the security of the code. It also cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements. 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 the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit cannot guarantee the explicit security of the audited smart contracts.