HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: QANplatform
**Date**: April 06th, 2022

## Document

| | |
|---|---|
| **Name** | Smart Contract Code Review and Security Analysis Report for QANplatform. |
| **Approved By** | Evgeniy Bezuglyi \| SC Department Head at Hacken OU |
| **Type of Contracts** | ERC20 token |
| **Platform** | EVM |
| **Language** | Solidity |
| **Methods** | Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| **Website** | https://www.qanplatform.com/ |
| **Timeline** | 06.04.2022 |
| **Changelog** | 06.04.2022 - Initial Review |

## Table of contents

## Introduction

Hacken OÜ (Consultant) was contracted by QANplatform (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:**
    https://github.com/qanplatform/qanx-token
**Commit:**
    ebf0352af6ccb8d991827a0fba3ac5cfe9ca6b70
**Documentation:** Yes
**JS tests:** Yes
**Contracts:**
    QANX.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 | ▪ Reentrancy<br>▪ Ownership Takeover<br>▪ Timestamp Dependence<br>▪ Gas Limit and Loops<br>▪ Transaction-Ordering Dependence<br>▪ Style guide violation<br>▪ EIP standards violation<br>▪ Unchecked external call<br>▪ Unchecked math<br>▪ Unsafe type inference<br>▪ Implicit visibility level<br>▪ Deployment Consistency<br>▪ Repository Consistency |
| Functional review | ▪ Business Logics Review<br>▪ Functionality Checks<br>▪ Access Control & Authorization<br>▪ Escrow manipulation<br>▪ Token Supply manipulation<br>▪ Assets integrity<br>▪ User Balances manipulation<br>▪ Data Consistency<br>▪ Kill-Switch Mechanism |

## Executive Summary

The score measurements details can be found in the corresponding section of the methodology.

### Documentation quality

The Customer provided functional requirements and technical requirements as the detailed comments in the contract. The total Documentation Quality score is **10** out of **10**.

### Code quality

The total CodeQuality score is **10** out of **10**. Unit tests were provided.

### Architecture quality

The architecture quality score is **10** out of **10**. The architecture is clear.

### 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 "Issues overview" section.

### Summary

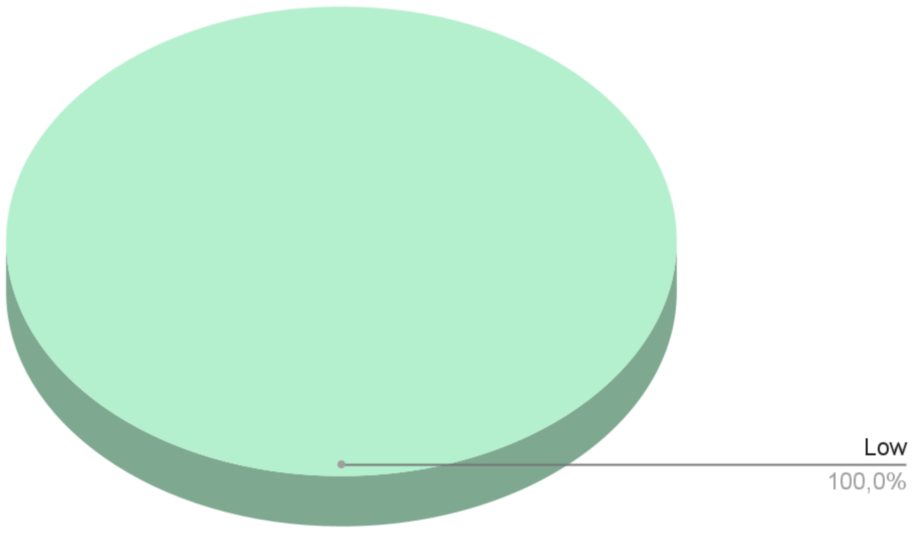
According to the assessment, the Customer's smart contract has the following score: **10**

1
2
3
4
5
6
7
8
9
10
You are here

### Notices

**1.** The repository contains contracts that are out of scope.
**2.** The locking mechanism emits self-transfer events when locked tokens of an address get unlocked, as the locked balance is moved to the unlocked balance of the particular account. This operation is compliant with ERC-20 standards. However, systems monitoring Transfer events should check both the sender and recipient of all Transfer events to keep their internal accounting state correct.

*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. **Unlocked pragma**

   Contracts with unlocked pragmas may be deployed by the latest compiler, which may have higher risks of undiscovered bugs.

   **Contracts**: QANplatform.sol

   **Function**: -

   **Recommendation**: lock pragmas to a specific compiler version.

   **Status**: Acknowledged

   **Clarification:** the contract has been compiled using Solidity compiler v0.8.4+commit.c7e474f2 and is already deployed on the Ethereum and BinanceSmart Chain mainnets at address 0xAAA7A10a8ee237ea61E8AC46C50A8Db8bCC1baaa. There are no known bugs in this specific compiler version according to the official Solidity documentation https://docs.soliditylang.org/en/v0.8.11/bugs.html.

2. **Redundant check**

   Checking if block.timestamp > _locks[account].hardLockUntil is redundant because it is already included in unlockableBalanceOf function which is called before.

   **Contracts**: QANplatform.sol

   **Function**: unlock#549

   **Recommendation**: remove the redundant checks

   **Status**: Acknowledged

   **Clarification:** this issue does not have any negative effect on security in any way, but this redundant check costs extra gas for the unlocker when calling the unlock() method, which could have been spared.

# 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.