HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: SubQuery Pte Ltd
Date: April 06th, 2022

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 SubQuery Pte Ltd.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type of Contracts	ERC-20 token; Staking; Vesting
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://subquery.network/
Timeline	21.03.2022 - 06.04.2022
Changelog	24.03.2022 - Initial Review 06.04.2022 - Revising

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 SubQuery Pte Ltd (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.

The second review was conducted on April 06th, 2022.

Scope

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

Repository:
https://github.com/hknio/contract-audit-batch1-440659869/tree/audit/update-with-audit
Commit:
fc45d60878990a96d45ab805db0580ee00a6b53c
Technical Documentation: Yes
JS tests: Yes
Contracts:
./contracts/AdminUpgradabilityProxy.sol
./contracts/EraManager.sol
./contracts/InflationController.sol
./contracts/ProxyAdmin.sol
./contracts/Settings.sol
./contracts/SQToken.sol
./contracts/Staking.sol
./contracts/VestingContract.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 ▪ 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	▪ 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

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

Documentation quality

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

Code quality

The total Code Quality score is **7** out of **10**. Code follows official language style guides. Unit tests were provided. The code is not properly commented.

Architecture quality

The architecture quality score is **10** out of **10**. The project has clear and clean architecture.

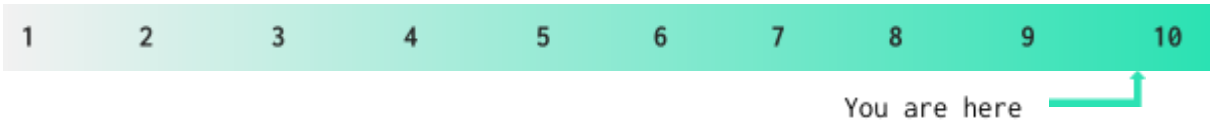
Security score

As a result of the audit, security engineers found **2** high, **1** medium, and **5** low severity issues. The security score is **0** out of **10**. All found issues are displayed in the “Issues overview” section of the report.

After the second review, the code contains **no** issues. The security score is **10** out of **10**.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.7**



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

Notices

1. SQTokens may be unlimitedly minted by the `minter`.
2. The distribution of the tokens emitted by the InflationController contract and staking reward distribution is out of the audit scope.
3. The Staking and InflationController contracts are upgradable.

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

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

Findings

Critical

No critical severity issues were found.

High

1. **The vesting contract may have insufficient funds.**

   The contract has the function `allocateVesting` that allocates tokens in the vesting, but the function logic does not guarantee that the contract has enough tokens.

   This may lead to the inability to claim tokens from the vesting contract.

   Contracts: VestingContract.sol

   Function: allocateVesting

   Recommendation: Add logic to the `allocateVesting` to check if the contract has enough funds.

   Status: Fixed

2. **The owner may withdraw allocated tokens.**

   The contract has the function `withdrawAllByAdmin` that allows the contract owner to withdraw all the tokens from the vesting contract.

   This may lead to the inability to claim tokens from the vesting contract.

   Contracts: VestingContract.sol

   Function: withdrawAllByAdmin

   Recommendation: Rework the function `withdrawAllByAdmin` to allow the owner to withdraw only not allocated tokens.

   Status: Fixed

Medium

1. **New value is not properly checked.**

   The staking contract has the function `setUnbondFeeRateBP` which sets a new value for withdraw fee, but the new value is not checked.

   This may lead to the setting the value which does not allow to withdraw staking funds.

   Contracts: Staking.sol

   Function: setUnbondFeeRateBP

   Recommendation: Add require statement to check if the new fee value is lower than `UNNOMINATE_BURNRATE_MULTIPLIER` constant.

   Status: Fixed

Low

1. **The function has no implementation.**

   The staking contract has the function `_onDelegationChange` without implementation.

   Contracts: VestingContract.sol

   Functions: _onDelegationChange

   Recommendation: Remove the function or add the logic to it.

   Status: Fixed

2. **Unused import statement.**

   The contract has an unused import statement: `Address`, `IVestingContract`.

   Contracts: VestingContract.sol

   Recommendation: Remove unused import statements.

   Status: Fixed

3. **Unused import statement.**

   The contract has an unused import statement: `Address`.

   Contracts: SQToken.sol

   Recommendation: Remove unused import statement.

   Status: Fixed

4. **Unused import statement.**

   The contract has an unused import statement: `Address`.

   Contracts: Settings.sol

   Recommendation: Remove unused import statement.

   Status: Fixed

5. **Unused import statement.**

   The contract has an unused import statement: `IRewardsDistributer`, `IIndexerRegistry`, `ISQToken`.

   Contracts: Staking.sol

   Recommendation: Remove unused import statements.

   Status: Fixed

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

Recommendations

1. It is recommended to add staking reward distribution logic to the audit scope. According to the current staking implementation and audit scope, the reward distribution logic is not clear. Staking contract logic which is related to `StakingAmount` data structure may be redundant.

   Contracts: Staking.sol

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 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.