SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: SDAO
Date: March 16th, 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 SDAO.
Approved by	Andrew Matiukhin | CTO Hacken OU Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type	ERC20 Converter
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/singnet/converter-token-manager
Commit	3cf70b374253fcba7c55e73cbd32551c43c6f731
Technical Documentation	YES
JS tests	YES
Website	https://singularitydao.ai/
Timeline	04 MARCH 2022 - 16 MARCH 2022
Changelog	10 MARCH 2022 - Initial Audit 15 MARCH 2022 – Remediation Checks 16 MARCH 2022 – Remediation Checks

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by SDAO (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/singnet/converter-token-manager
Commit:
3cf70b374253fcba7c55e73cbd32551c43c6f731
Technical Documentation: Yes; SNETP-ERC20ConverterDesign.pdf
JS tests: Yes; in tests directory
Contracts:
TokenConversionManager.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 and technical requirements with diagrams and descriptions.
Total Documentation Quality score is **10** out of **10**.

Code quality

Total CodeQuality score is **7** out of **10**. Too long lines. Some uncompleted DocBlocks. Spaces issues. Variable names.

Architecture quality

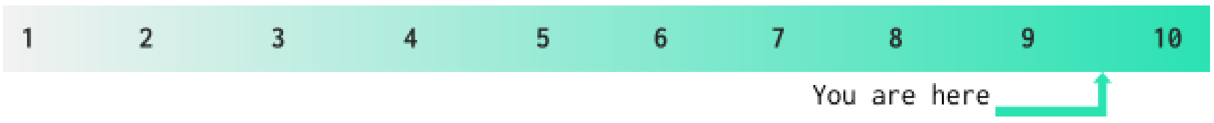
Architecture quality score is **9** out of **10**. All the logic is implemented in one file but there’s no need to extend it. The outdated compiler version is used. Some validations could be put to modifiers.

Security score

As a result of the audit, security engineers found **no** security issues. The security score is **10** out of **10**. All found issues are displayed in the “Issues overview” section of the report.

Summary

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



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

Audit overview

🟥🟥🟥🟥 Critical

No critical issues were found.

🟨🟨🟨 High

No high severity issues were found.

🟩🟩 Medium

No medium severity issues were found.

🟦 Low

A public function that could be declared external

public functions that are never called by the contract should be declared **external**.

Contract: TokenConversionManager.sol

Functions: updateAuthorizer, updateConfigurations

Recommendation: Use the **external** attribute for functions never called from the contract.

Status: Fixed

Recommendations

1. Please follow solidity recommendations for the code formatting (line length, spacings, variables names, etc.)
2. Try using a more modern compiler version, e.g. **0.8.11**. Doesn’t matter which version was used to compile your previous contracts, always try to use the latest stable version for any new contracts.

   Contracts: TokenConversionManager.sol

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 the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free 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 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 can't guarantee the explicit security of the audited smart contracts.