HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: SDAO
Date: December 15th, 2021

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
Type	Dynaset
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/Singularity-DAO/Dynaset
Commit	d1a942088e1c558c867d5fbcb0cc8cb09cbab65e
Technical Documentation	NO
JS tests	YES
Website	singularitydao.ai
Timeline	07 DECEMBER 2021 - 15 DECEMBER 2021
Changelog	10 DECEMBER 2021 - INITIAL AUDIT 15 DECEMBER 2021 - SECOND REVIEW

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 contract and its code review conducted between December 7th, 2021 - December 10th, 2021.

Second review conducted on December 15th, 2021.

Scope

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

Repository:
https://github.com/Singularity-DAO/Dynaset
Commit:
d1a942088e1c558c867d5fbcb0cc8cb09cbab65e
Technical Documentation: No
JS tests: Yes (https://github.com/Singularity-DAO/Dynaset/blob/d1a942088e1c558c867d5fbcb0cc8cb09cbab65e/test/dynaset.js)
Contracts:
BConst.sol
BMath.sol
BNum.sol
DToken.sol
Dynaset.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 ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency

	▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.

Insecure	**Poor secured**	**Secured**	**Well-secured**
		You are here	↑

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

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

After the second review security engineers found **no security issues**.

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

Contracts that lock Ether

Contract with a payable function, but without a withdrawal capacity.

Contract: Dynaset.sol

Functions: swapUniswap, swapOneInch, swapOneInchUniV3

Recommendation: Remove the payable attribute or add a withdraw function.

Status: Fixed

🟦 Low

1. Uninitialized state variable

The contract has an uninitialized state variable “_admin” which is used in the modifier “_admin_”. But since this modifier is never used in the contract, those both could be removed or commented in the code.

Contract: Dynaset.sol

Recommendation: Please remove both the variable and the modifier.

Status: Fixed

2. A public function that could be declared external

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

Contract: Dynaset.sol

Functions: getCurrentDesiredTokens, getDenormalizedWeight, getTotalDenormalizedWeight, getBalance

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

Status: Fixed

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

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

After the second review security engineers found **no security issues**.

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.