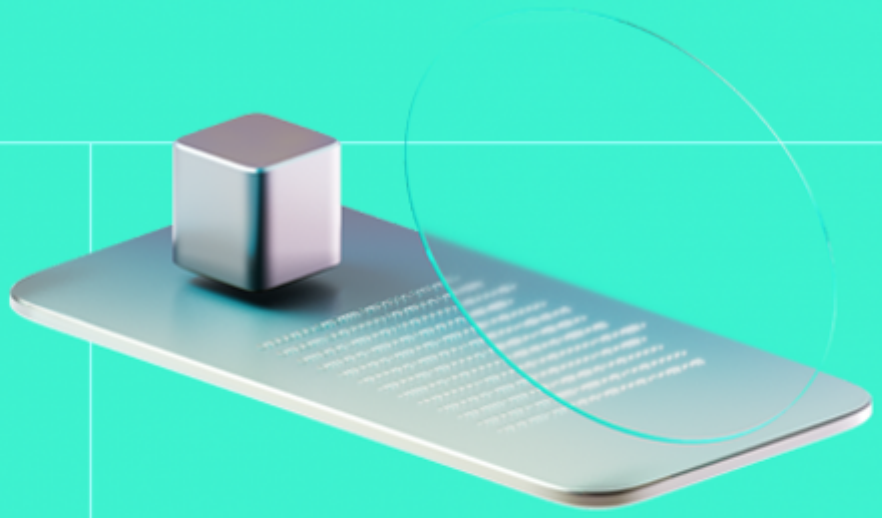
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Sensay

Date: 27/03/2024

We express our gratitude to the Sensay team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

Sensay is dedicated to creating a seamless integration between the digital and physical worlds, utilizing cutting-edge technology to develop personalized digital replicas.

Platform: EVM

Language: Solidity

Tags: Fungible Token

Timeline: 25/03/2024 - 26/03/2024

Methodology: https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/sensay-io/smart-contracts
Commit	01c95d6

Audit Summary

10/10	10/10	0%	10/10
Security Score	Code quality score	Test coverage	Documentation quality score

Total 10/10

The system users should acknowledge all the risks summed up in the risks section of the report

0	0	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	0
Low	0

This report 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 can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for Sensay
Audited By	Ivan Bondar
Approved By	Grzegorz Trawinski
Website	https://www.snsy.ai/
Changelog	27/03/2024 - Final Report

Table of Contents

System Overview

The Sensay token (SNSY) is a cutting-edge digital asset designed for operation on the Ethereum blockchain. It incorporates the LayerZero Omnichain Fungibility (OFT) protocol, which allows for efficient and seamless cross-chain token transfers. The Sensay token is crafted to address the limitations of traditional blockchain bridges, offering a more flexible and interoperable solution.

Token Specifications:

- Token Name: Sensay
- Symbol: SNSY
- Decimal Precision: 18
- Total Supply: 10 Billion (10,000,000,000) SNSY Tokens

Privileged roles

- Owner :
  - Delegate Management (setDelegate Function)
    - Assigns a delegate who can implement custom configurations on behalf of the contract owner.
    - The delegate gains the ability to manage critical tasks, including setting configurations and MessageLibs, and handling payload-related operations for the OFT.
    - Enables flexible management of the token's cross-chain functionalities.
  - Cross-Chain Communication Setup (setPeer Function)
    - Opens the messaging channel and connects the OFT deployment to different blockchain networks.
    - Inputs:
      - _eid: The endpoint ID for the destination chain where the other OFT contract resides.
      - _peer: The address of the destination OFT contract in bytes32 format.
    - Essential for enabling the OFT to start receiving messages from other chains, a critical step in establishing cross-chain functionality.
- Enforcement of Execution Options (setEnforcedOptions Function)
  - Specifies mandatory execution options to ensure the application behaves as expected during user interactions.
  - Input:
    - EnforcedOptionParam[], a struct defining execution options per message type and destination chain.
  - Provides control over how the token operates across chains.
- Message Inspection Setup (setMsgInspector Function)
  - Sets the address of the message inspector for the OFT, which is an optional role.
  - The message inspector can examine 'message' and 'options' if enabled.
  - Options:
    - Set to address(0) to disable the message inspector.
    - Set to a specific contract address to enable and define its behavior.
  - Offers an additional layer of inspection for cross-chain messages.
- Transfer of Ownership (transferOwnership Function)
  - This function enables the current owner of the contract to transfer ownership to a new address. It's a key aspect of the contract's governance.

- Transferring ownership is a significant action as it involves handing over control of the contract's critical administrative functions. This includes the ability to set delegates, configure cross-chain communication, enforce execution options, and set up a message inspector.

Executive Summary

This report presents an in-depth analysis and scoring of the customer's smart contract project. Detailed scoring criteria can be referenced in the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are detailed.
  - Project overview is detailed
  - All roles in the system are described.
  - Use cases are described and detailed.
  - For each contract, all futures are described.
  - All interactions are described.
- Technical description is limited.
  - Run instructions are provided.
  - Technical specification is provided.
  - The NatSpec documentation is sufficient.

Code quality

The total Code Quality score is **10** out of **10**.

Test coverage

Code coverage of the project is **0%** (branch coverage).

- Tests are not provided.

Security score

Upon auditing, the code was found to contain **0** critical, **0** high, **0** medium, and **0** low severity issues, leading to a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's smart contract yields an overall score of **10.0**. This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

- Centralization of Initial Supply
  - The contract mints the entire initial supply of 10 billion tokens to the deployer's address. This concentration of tokens in a single address can pose significant risks in terms of centralization and potential manipulation.
- Lack of On-Chain Enforcement of Tokenomics
  - The provided tokenomics outline a detailed allocation plan (e.g., Public Sale, Platform Development, Team and Advisors, etc.). However, the current smart contract implementation does not enforce these allocations on-chain.
  - Without on-chain enforcement, there's no guarantee that the tokens will be distributed according to the stated tokenomics.
- Vesting and Lock-up Periods
  - Tokenomics mention specific vesting and lock-up periods for different stakeholders (e.g., Team and Advisors, Future Team). However, these restrictions are not coded into the contract.
  - The lack of on-chain mechanisms to enforce vesting schedules and lock-up periods means that large amounts of tokens could potentially be moved or sold earlier than intended, impacting the token's market stability.
- Reliance on External Protocols
  - The token relies on Layer Zero's OFT for cross-chain functionality. Any vulnerabilities or issues in the Layer Zero protocol could directly impact the Sensay token.
  - Cross-chain transfers also add complexity, which can introduce additional points of failure or security considerations.

Findings

Vulnerability Details

Observation Details

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing 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 report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and 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.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

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 Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details

Repository	https://github.com/sensay-io/smart-contracts
Commit	01c95d6b34dbf13bca3f33127708558dc61cae09
Deployed Address	https://etherscan.io/address/0×82a605D6D9114F4Ad6D5Ee461027477EeED31E34
Whitepaper	https://docsend.com/view/5zdkjv3nbfu4r5dx
Requirements	https://github.com/sensay-io/smart-contracts/blob/main/README.md
Technical Requirements	https://github.com/sensay-io/smart-contracts/blob/main/README.md

Contracts in Scope

sensay.sol