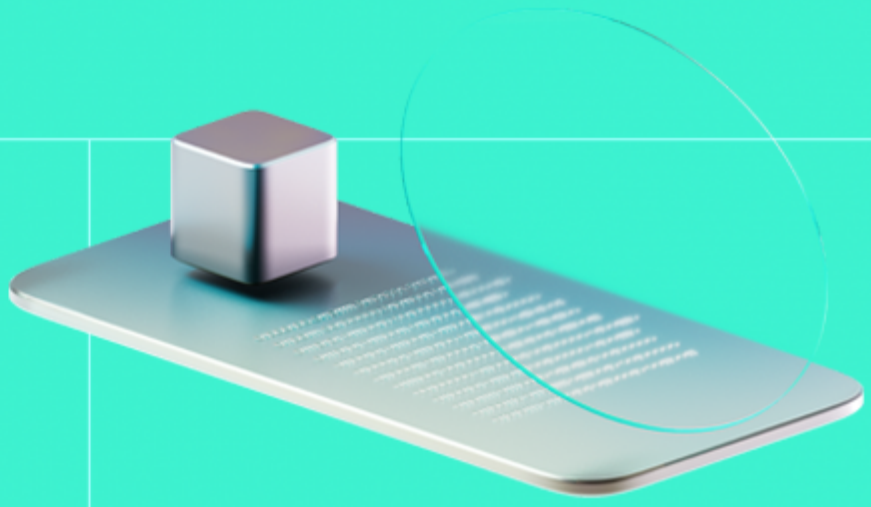
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Zesh AI Layer

Date: 17/12/2024

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

Zesh AI Layer (ZAI) currency is a SUI coin that facilitates seamless initialization and minting of tokens to the deployer address.

Document

Name	Smart Contract Code Review and Security Analysis Report for Zesh AI Layer
Audited By	Jakub Heba
Approved By	Przemyslaw Swiatowiec
Website	https://zesh.ai
Changelog	16/12/2024 - Preliminary Report, 17/12/2024 - Secondary Report
Platform	Sui
Language	MOVE
Tags	Sui, Coin
Methodology	https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/ZeshDev/zesh-coin
Commit	6f34f9f34e4f5e579ca4d3bb1ee62020218feef8

Audit Summary

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

1	1	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by Severity

Severity	Count
Critical	0
High	0
Medium	1
Low	0

Vulnerability	Severity
F-2024-7705 - Wrong value hardcoded in TOTAL_SUPPLY leads to too few minted tokens	Medium

Documentation quality

- Functional requirements are listed in README.md file.
- Technical description is partially provided.

Code quality

- The code duplicates commonly known Coin initialization standard.

Test coverage

Code coverage of the project is **0%** (branch coverage), with a mutation score of **0%**.

- All tests are commented-out.

Table of Contents

System Overview

ZAI — simple Sui coin that mints all initial supply to a deployer. Additional minting is not allowed.

It has the following attributes:

- Name: Zesh AI Layer
- Symbol: ZAUI
- Decimals: 6
- Total supply: 500m tokens.

Privileged roles

- The deployer of ZAI is minting 500m supply of tokens to himself during initialization.
- No other privileges are assigned to that address.

Potential Risks

- Total token supply is minted to the deployer account without allocation logic, which could lead to concerns over transparency and control in token distribution.

Findings

Vulnerability Details

F-2024-7705 - Wrong value hardcoded in TOTAL_SUPPLY leads to too few minted tokens - Medium

Description:

During the initialization of the ZAI coin, the total supply is defined as a hardcoded constant `TOTAL_SUPPLY`, which according to the commentary and technical documentation should be `1 billion` tokens.

Then, these coins are minted by the `mint()` function and sent to the deployer for further distribution.

However, it was found that this coin has a decimal of `6`. Therefore, the number of coins needed to reach the displayed value of `1 billion` must be increased by six zeros so that the calculations are correct.

Currently, `TOTAL_SUPPLY` has the following value:

```
// Total supply: 1 billion tokens
const TOTAL_SUPPLY: u64 = 1_000_000_000;
```

This is incorrect, because after "cutting off" six decimal places, the actual number of displayed coins will be `1_000`.

Assets:

- zesh_coin.move [https://github.com/ZeshDev/zesh-coin]

Status: Fixed

Classification

Impact: 2/5

Likelihood: 5/5

Exploitability: Independent

Complexity: Simple

Severity: Medium

Recommendations

Remediation: We recommend adjusting the value in `TOTAL_SUPPLY` to `1_000_000_000_000_000`, so that after taking into account decimals, there are actually `1 billion` tokens available.

Resolution: Issue was fixed in `005c8a6b87239b8611dac9e869a0fd5e799ff8da` commit by changing the `TOTAL_SUPPLY` constans to `1_000_000_000_000_000`.

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. Definitions

Severities

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.

Potential Risks

The "Potential Risks" section identifies issues that are not direct security vulnerabilities but could still affect the project's performance, reliability, or user trust. These risks arise from design choices, architectural decisions, or operational practices that, while not immediately exploitable, may lead to problems under certain conditions. Additionally, potential risks can impact the quality of the audit itself, as they may involve external factors or components beyond the scope of the audit, leading to incomplete assessments or oversight of key areas. This section aims to provide a broader perspective on factors that could affect the project's long-term security, functionality, and the comprehensiveness of the audit findings.

Appendix 2. Scope

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

Scope Details	
Repository	https://github.com/ZeshDev/zesh-coin
Commit	6f34f9f34e4f5e579ca4d3bb1ee62020218feef8
Whitepaper	-
Requirements	https://github.com/ZeshDev/zesh-coin/README.md
Technical Requirements	https://github.com/ZeshDev/zesh-coin/README.md

Asset	Type
zesh_coin.move [https://github.com/ZeshDev/zesh-coin]	Smart Contract

Appendix 3. Additional Valuables

Additional Recommendations

The smart contracts in the scope of this audit could benefit from the introduction of automatic emergency actions for critical activities, such as unauthorized operations like ownership changes or proxy upgrades, as well as unexpected fund manipulations, including large withdrawals or minting events. Adding such mechanisms would enable the protocol to react automatically to unusual activity, ensuring that the contract remains secure and functions as intended.

To improve functionality, these emergency actions could be designed to trigger under specific conditions, such as:

- Detecting changes to ownership or critical permissions.
- Monitoring large or unexpected transactions and minting events.
- Pausing operations when irregularities are identified.

These enhancements would provide an added layer of security, making the contract more robust and better equipped to handle unexpected situations while maintaining smooth operations.