SALUS SECURITY

JAN 2026

CODE SECURITY ASSESSMENT

CODEX

Overview

Project Summary

- Name: CODEX
- Platform: BSC Chain
- Address: 0x4AF228a5fba8fb4fF64b5b9563Fd114595Fa6686
- Language: Solidity
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Codex
Version	v2
Type	Solidity
Dates	Jan 04 2026
Logs	Jan 04 2026; Jan 04 2026

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	0
Total informational issues	2
Total	2

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Lack of Security Contact Information for Responsible Disclosure	Informational	Configuration	Acknowledged
2	Interface Should Be Placed in a Separate File	Informational	Code Quality	Acknowledged

2.2 Informational Findings

1. Lack of Security Contact Information for Responsible Disclosure

Severity: Informational	Category: Configuration
Target: - CodexToken.sol	

Description

The contract CodexToken does not specify a security contact point. Including a designated security contact (such as an email address or ENS name) in the contract's NatSpec header facilitates responsible vulnerability disclosure. This makes it easier for external researchers to quickly reach the appropriate team in the event a vulnerability is identified, helping minimize the time window between discovery and mitigation. The Ethereum community has begun standardizing this practice using the `@custom:security-contact` tag, adopted by tools such as OpenZeppelin Wizard and ethereum-lists.

Recommendation

It is recommended to use a locked Solidity version throughout the project. It is also recommended to use the most stable and up-to-date version.

Status

This issue has been acknowledged by the team.

2. Interface Should Be Placed in a Separate File

Severity: Informational	Category: Code Quality
Target: - CodexToken.sol	

Description

The `IERC20/IERC20Metadata` interfaces are currently declared together with the CodexToken contract in CodexToken.sol file. Combining an interface and a contract in the same file increases code complexity and reduces readability, making it harder for developers and auditors to navigate, understand, and maintain the codebase.

Recommendation

It is recommended to separate the `IERC20/IERC20Metadata` interfaces into its own dedicated files. Doing so improves the clarity, organization, and maintainability of the codebase, while aligning the project with standard Solidity best practices.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the contracts on 0x4AF228a5fba8fb4fF64b5b9563Fd114595Fa6686.