SALUS SECURITY

DEC 2024

CODE SECURITY ASSESSMENT

AVALON

Overview

Project Summary

- Name: Avalon - Governance Token
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/avalonfinancexyz/avalon-governance-token
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Avalon - Governance Token
Version	v1
Type	Solidity
Dates	Dec 11 2024
Logs	Dec 11 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	1
Total informational issues	2
Total	4

Contact

E-mail: support@salusec.io

SALUS

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.

SALUS

Content

SALUS

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.

SALUS

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Centralization risk	Medium	Centralization	Pending
2	Incomplete initialization	Low	Code Quality	Pending
3	Use of floating pragma	Informational	Configuration	Pending
4	Redundant Code	Informational	Redundancy	Pending

SALUS

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Centralization risk

Severity: Medium	Category: Centralization
Target: - contracts/Avalon.sol	

Description

The contract has privileged addresses, such as `minter` and `burner`, which can use the `_mint()` and `_burn()` functions to increase or decrease the token balance of any address.

If the private keys of the privileged addresses are leaked, an attacker could arbitrarily mint tokens or burn tokens from other users' balances.

If the privileged accounts are plain EOA accounts, this can be worrisome and pose a risk to the other users.

Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

SALUS

2. Incomplete initialization

Severity: Low	Category: Code Quality
Target: - contracts/Avalon.sol	

Description

The contracts inherit a total of three upgradeable contracts, but during initialization, only one of them was initialized. The `AccessControlUpgradeable` and `PausableUpgradeable` contracts were not initialized.

Recommendation

It is recommended to add logic to initialise the remaining two contracts.

SALUS

2.3 Informational Findings

3. Use of floating pragma

Severity: Informational	Category: Configuration
Target: - All	

Description

```
pragma solidity ^0.8.22;
```

All contracts use a floating compiler version `^0.8.22`.

Using a floating pragma `^0.8.22` statement is discouraged, as code may compile to different bytecodes with different compiler versions. Use a locked pragma statement to get a deterministic bytecode. Also use the latest Solidity version to get all the compiler features, bug fixes and optimizations.

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.

SALUS

4. Redundant Code

Severity: Informational	Category: Redundancy
Target: - contracts/Avalon.sol - contracts/AvalonOFTAdapter.sol	

Description

Unused code should be removed before deploying the contract to mainnet. We have identified the following code are not being utilized:

contracts/Avalon.sol:L6

```
import {OwnableUpgradeable} from
"@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol";
```

contracts/AvalonOFTAdapter.sol:L4

```
import {Ownable} from "@openzeppelin/contracts/access/Ownable.sol";
```

Recommendation

Consider removing the redundant code.

SALUS

Appendix

Appendix 1 - Files in Scope

This audit covered the following files:

File	SHA-1 hash
contracts/Avalon.sol	251c5b987f78dbe529f67517ac9c6c672f72c8c1
contracts/AvalonOFTAdapter.sol	bec3485fbce65cfcc2751c37befeff1bab1c4f04

SALUS