

SALUS SECURITY

AUG 2024



CODE SECURITY ASSESSMENT

LISTA DAO

Overview

Project Summary

- Name: Lista Dao - AMO
- Platform: BNB Smart Chain
- Language: Solidity
- Repository:
 - <https://github.com/lista-dao/lista-dao-contracts>
- Audit Range: See [Appendix - 1](#)

Project Dashboard

Application Summary

Name	Lista Dao - AMO
Version	v2
Type	Solidity
Dates	Aug 08 2024
Logs	Aug 07 2024; Aug 08 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	2
Total informational issues	1
Total	3

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	4
1.1 About SALUS	4
1.2 Audit Breakdown	4
1.3 Disclaimer	4
Findings	5
2.1 Summary of Findings	5
2.2 Notable Findings	6
1. Centralization risk	6
2. Missing events for functions that change critical state	7
2.3 Informational Findings	8
3. Redundant code	8
Appendix	9
Appendix 1 - Files in Scope	9

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	Centralization risk	Low	Centralization	Mitigated
2	Missing events for functions that change critical state	Low	Logging	Acknowledged
3	Redundant code	Informational	Redundancy	Acknowledged

2.2 Notable Findings

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

1. Centralization risk	
Severity: Low	Category: Centralization
Target: <ul style="list-style-type: none">- contracts/Interaction.sol	

Description

In Interaction contracts, there exists a privileged owner role. This auth has authority over key operations such as ``setCollateralType``, ``setWhitelistOperator`` to change the permissions for some addresses and ``setCores`` to modify the dependent contract logic.

If the auth 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.

Status

This issue has been acknowledged by the team. The team has stated that after deployment the admin role will be migrated to a multi-sig wallet.

2. Missing events for functions that change critical state

Severity: Low

Category: Logging

Target:

- contracts/Interaction.sol

Description

Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes that allow users to evaluate them. Missing events do not promote transparency and if such changes immediately affect users' perception of fairness or trustworthiness, they could exit the protocol causing a reduction in protocol users.

In the Interaction contract, events are lacking in the privileged setter functions (e.g. ``addToWhitelist()`, `removeFromWhitelist()`, `addToBlacklist()`, `removeFromBlacklist()```).

Recommendation

It is recommended to emit events for critical state changes.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

3. Redundant code

Severity: Informational

Category: Redundancy

Target:

- contracts/Interaction.sol

Description

The `stringToBytes32()` function is never used in Interaction contract. Therefore, removing the function will cut down on gas during the contract deployment process.

Interaction.sol:L188-L197

```
function stringToBytes32(string memory source) public pure returns (bytes32 result) {
    bytes memory tempEmptyStringTest = bytes(source);
    if (tempEmptyStringTest.length == 0) {
        return 0x0;
    }

    assembly {
        result := mload(add(source, 32))
    }
}
```

Recommendation

Consider using the above suggestions to save gas.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit [8b83905](#):

File	SHA-1 hash
contracts/Interaction.sol	3edcdc412db8a60265974fca1b0ee830b30fd69a
contracts/amo/DynamicDutyCalculator.sol	64bc80854fb3156532bac00e7bb0a2d2150b1dd0
contracts/libraries/FixedMath0x.sol	ba9f7b15b00054ac652da7bb550888bf59bbc006