SALUS SECURITY

OCT 2024

CODE SECURITY ASSESSMENT

LISTA DAO

Overview

Project Summary

- Name: Lista token - Incremental audit
- Platform: BNB Smart Chain
- Language: Solidity
- Repository:
    - https://github.com/lista-dao/lista-token
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Lista token - Incremental audit
Version	v3
Type	Solidity
Dates	Oct 21 2024
Logs	Oct 10 2024; Oct 16 2024; Oct 21 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	0
Total informational issues	1
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 **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
2.3 Informational Findings 7
2. Gas optimization suggestions 7
Appendix **8**
Appendix 1 - Files in Scope 8

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	Medium	Centralization	Acknowledge
2	Gas optimization suggestions	Informational	Gas Optimization	Acknowledge

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/dao/EmissionVoting.sol - contracts/dao/ListaVault.sol - contracts/dao/ClisBNBLaunchPoolDistributor.sol	

Description

`ListaVault`, `EmissionVoting` and `ClisBNBLaunchPoolDistributor` contracts have privileged accounts. These privileged accounts can withdraw all tokens from the contracts using the `emergencyWithdraw()` or `adminTransfer()` functions. Additionally, it can vote on distributor's weights without needing governance tokens.

If privileged accounts' private key or admin's is compromised, an attacker can steal all the tokens in the contract and arbitrarily increase the distributor's weight.

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.

Status

This issue has been acknowledged by the team. The team states that all privileged functions will be operated under a multi-sig account.

2.3 Informational Findings

2. Gas optimization suggestions

Severity: Informational	Category: Gas Optimization

Target:
- contracts/dao/EmissionVoting.sol
- contracts/dao/ListaVault.sol
- contracts/dao/ClisBNBLaunchPoolDistributor.sol

Description

1. Memory reading saves more gas than storage reading multiple times when the state is not changed. So caching the storage variables in memory and using the memory instead of storage reading is effective. So cache array length outside of the loop can save gas.

contracts/dao/EmissionVoting.sol:L224

```
for (uint256 i = 0 ; i < distributorIds.length; ++i) {
```

contracts/dao/ListaVault.sol:L142

```
for (uint16 i = 0; i < ids.length; ++i) {
```

contracts/dao/ListaVault.sol:L189

```
for (uint16 i = 0; i < _distributors.length; ++i) {
```

contracts/dao/ListaVault.sol:L208

```
for (uint16 i = 0; i < distributors.length; ++i) {
```

contracts/dao/ClisBNBLaunchPoolDistributor.sol:L193

```
for (uint256 i = 0; i < _epochIds.length; i++) {
```

2.The uint256 type is always greater than or equal to 0, so the check is redundant.

contracts/dao/EmissionVoting.sol:L166

```
require(_adminVotePeriod >= 0 && _adminVotePeriod <= WEEK, "admin vote period should
within 0 to 1 week");
```

contracts/dao/EmissionVoting.sol:L228

```
require(weight >= 0, "weight should be equals to or greater than 0");
```

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 0fc2a7c:

File	SHA-1 hash
contracts/dao/EmissionVoting.sol	41149c4f6998d0032a04b8cb24eb5f1adf95d5a4
contracts/dao/ListaVault.sol	a4406a6dc4c2cbf2034400b12998d75f9303bb86
contracts/dao/ClisBNBLaunchPoolDistributor.sol	5e923823caa825a6ececc59bb4b6b2b98b990247