Smart Contract
Security Audit Report

Prepared for Lista DAO

Prepared by Supremacy

April 03, 2024

Contents

1 Introduction 3
1.1 About Client 4
1.2 Audit Scope 4
1.3 Changelogs 4
1.4 About Us 4
1.5 Terminology 5
2 Findings 6
2.1 Informational 6
3 Disclaimer 7

1 Introduction

Given the opportunity to review the design document and related codebase of the Lista DAO token, we outline in the report our systematic approach to evaluate potential security issues in the smart contract(s) implementation, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts can be further improved due to the presence of several issue related to either security or performance. This document outlines our audit results.

1.1 About Client

Lista DAO functions as the open-source decentralized stablecoin lending protocol powered by LSDfi. Users can undergo staking and liquid staking on Lista, as well as borrow lisUSD against a variety of decentralized collateral. Present on the BNB Chain, Lista aims to position lisUSD as the number one stablecoin in the crypto space, leveraging on innovative liquid staking solutions.

Item	Description
Client	Lista DAO
Website	https://lista.org
Type	Smart Contract
Languages	Solidity
Platform	EVM-compatible

1.2 Audit Scope

In the following, we show the Git repository of reviewed file and the commit hash used in this security audit:

- Repository: https://github.com/lista-dao/lista-token/tree/master/contracts
- Commit Hash: 68e0ade64b1d401117428fd77b2b594006c6db15

Below are the files in scope for this security audit and their corresponding MD5 hashes.

Filename	MD5
`./ListaToken.sol`	`4e6beed05a79ec04194aa7e43d46bb1c`

And this is the commit hash after all fixes for the issues found in the security audit have been checked in:

- Repository: https://github.com/lista-dao/lista-token/tree/master/contracts
- Commit Hash: 8451a9310429f0ba665f61baa61bcd298f2c7b92

Filename	MD5
`./ListaToken.sol`	`b079a72bc6bef4103a98dcd34cbe7696`

1.3 Changelogs

Version	Date	Description
0.1	March 25, 2024	Initial Draft
1.0	April 03, 2024	Final Release

1.4 About Us

Supremacy is a leading blockchain security firm, composed of industry hackers and academic researchers, provide top-notch security solutions through our technology precipitation and innovative research.

We are reachable at Twitter (https://twitter.com/SupremacyHQ), or Email (contact@supremacy.email).

1.5 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- Likelihood represents the likelihood of a finding to be triggered or exploited in practice
- Impact specifies the technical and business-related consequences of a finding
- Severity is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

Severity

Impact	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

Likelihood

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

2 Findings

The table below summarizes the findings of the audit, including status and severity details.

ID	Severity	Description	Status
1	Informational	Centralized risk	Fixed

2.1 Informational

1. Centralized risk [Informational]

`Status: Fixed`

Description:

In the Lista DAO token, there is an ownership privilege account that is defined directly by the constructor and receive a quantity of 1 Billion of Lista tokens.

Our analysis shows that privileged accounts need to be scrutinized. In the following, we will examine privileged accounts and the associated privileged access in the current contract.

Note that if the privileged owner account is a plain EOA, this may be worrisome and pose counter-party risk to the protocol users. A multi-sig account could greatly alleviate this concern, though it is still far from perfect. Specifically, a better approach is to eliminate the administration key concern by transferring the role to a community-governed DAO. In the meantime, a timelock-based mechanism can also be considered as mitigation.

```
36      // --- Functions ---
37      constructor(address owner) ERC20(_NAME, _SYMBOL) {
38          bytes32 hashedName = keccak256(bytes(_NAME));
39          bytes32 hashedVersion = keccak256(bytes(EIP712_VERSION));
40          DOMAIN_SEPARATOR = keccak256(
41              abi.encode(
42                  EIP712_DOMAIN,
43                  hashedName,
44                  hashedVersion,
45                  block.chainid,
46                  address(this)
47              )
48          );
49
50          // mint 1B tokens to the lista treasury account
51          _mint(owner, 1_000_000_000 * 10 ** decimals());
52      }
```

ListaToken.sol

Recommendation: From the comments, the ownership privileged account here to be `treasury`. A better solution is to directly hardcode the `owner` to the `treasury` address.

Feedback: The owner is a multisig wallet with 4/6 threshold.

3 Disclaimer

This security audit report does not constitute investment advice or a personal recommendation. It does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Any entity should not rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. This security audit report is not an endorsement of any particular project or team, and the report does not guarantee the security of any particular project. This audit does not give any warranties on discovering all security issues of the smart contracts, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues, also cannot make guarantees about any additional code added to the assessed project after the audit version. As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with independent audits and a public bug bounty program to ensure the security of smart contract(s). Unless explicitly specified, the security of the language itself (e.g., the solidity language), the underlying compiling toolchain and the computing infrastructure are out of the scope.