SALUS SECURITY

NOV 2024

CODE SECURITY ASSESSMENT

LISTA DAO

Overview

Project Summary

- Name: Lista DAO - PSM
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/lista-dao/lista-dao-contracts
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Lista DAO - PSM
Version	v2
Type	Solidity
Dates	Nov 22 2024
Logs	Nov 20 2024; Nov 22 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	2
Total informational issues	2
Total	5

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 withdraw fees 7

3. Forget to remove distributor 8

2.3 Informational Findings 9

4. Use of floating pragma 9

5. Redundant Code 10

Appendix **11**

Appendix 1 - Files in Scope 11

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	Mitigated
2	Missing events for functions that withdraw fees	Low	Logging	Resolved
3	Forget to remove distributor	Low	Business Logic	Resolved
4	Use of floating pragma	Informational	Configuration	Acknowledged
5	Redundant Code	Informational	Redundancy	Resolved

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/psm/LisUSDPoolSet.sol - contracts/psm/PSM.sol	

Description

The `manager` has privileged access. It can withdraw all tokens from the contract via the `emergencyWithdraw()` function, creating a single point of failure, potentially resulting in asset loss.

If the `manager`'s private key is compromised, an attacker can steal all tokens within the contract. 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 mitigated by the team. The team has stated that they will use a multisig wallet as the owner.

2. Missing events for functions that withdraw fees

Severity: Low	Category: Logging
Target: - contracts/psm/PSM.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 `PSM` contract, events are lacking in the `harvest()` function.

contracts/psm/PSM.sol:L307 - L313

```
function harvest() external {
    if (fees > 0) {
      uint256 _fees = fees;
      fees = 0;
      IERC20(lisUSD).safeTransfer(feeReceiver, _fees);
    }
}
```

Recommendation

It is recommended to emit events in the `harvest()` function.

Status

The team has resolved this issue in commit 3d037b9.

3. Forget to remove distributor

Severity: Low	Category: Business Logic
Target: - contracts/psm/LisUSDPoolSet.sol	

Description

The `removePool()` function needs to remove the distributor, but when emitting the `RemoveDistributor` event, it does not set the corresponding `distributor` to the zero address.

contracts/psm/LisUSDPoolSet.sol:L396-L406

```
function removePool(address pool) external onlyRole(MANAGER) {
    require(pools[pool].active, "pool not exists");

    pools[pool].active = false;

    emit RemovePool(pool);
    address distributor = pools[pool].distributor;
    if (distributor != address(0)) {
      emit RemoveDistributor(pool, distributor);
    }
 }
```

Recommendation

Consider updating the non-zero `distributor` address to a zero address in the `removePool()` function.

```
if (distributor != address(0)) {
    // pools[pool].distributor = address(0);  <------- set to zero
    emit RemoveDistributor(pool, distributor);
}
```

Status

The team has resolved this issue in commit 3d037b9.

2.3 Informational Findings

4. Use of floating pragma

Severity: Informational	Category: Configuration
Target: - All	

Description

```
pragma solidity ^0.8.10;
```

All contracts use a floating compiler version `^0.8.10`.

Using a floating pragma `^0.8.10` 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.

Status

This issue has been acknowledged by the team. The team has stated that they will use a fixed compiler version 0.8.19.

5. Redundant Code

Severity: Informational	Category: Redundancy
Target: - contracts/psm/VenusAdapter.sol	

Description

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

contracts/psm/VenusAdapter.sol:L24

```
event EmergencyWithdraw(address token, uint256 amount);
```

Recommendation

Consider removing the redundant code.

Status

The team has resolved this issue in commit 3d037b9.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 34d738a :

File	SHA-1 hash
contracts/psm/EarnPool.sol	6c689361eff762ff7c547606f655cd0f7d33e0e6
contracts/psm/LisUSDPoolSet.sol	f7fd3326574c3b1297579edad6e2273b8da15120
contracts/psm/PSM.sol	6cc38a9a164c17af87e08ff07dc6806c89bf860d
contracts/psm/VaultManager.sol	d35e98d689b9a057176a61bf3de73a15fe297fa2
contracts/psm/VenusAdapter.sol	5c28dcca281155785f75dd16edaecea5be4e21a6

And we audited the commit fca5f0c that introduced new features to the ListaDao repository :

File	SHA-1 hash
contracts/psm/LisUSDPoolSet.sol	eda76fa1b1fcf6daafd270324ad4aaa59e7c77ce