SALUS SECURITY

AUG 2024

CODE SECURITY ASSESSMENT

LISTA DAO

Overview

Project Summary

- Name: Lista Dao - Token emission
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/lista-dao/lista-token
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Lista Dao - Token emission
Version	v2
Type	Solidity
Dates	Aug 13 2024
Logs	Aug 09 2024; Aug 13 2024

Vulnerability Summary

Total High-Severity issues	1
Total Medium-Severity issues	2
Total Low-Severity issues	3
Total informational issues	2
Total	8

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

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	The onERC721Received function can be called directly	High	Business Logic	Resolved
2	Incorrect boundary usage in the loop causes the function to fail	Medium	Business Logic	Resolved
3	Centralization risk	Medium	Centralization	Mitigated
4	Rewards not fetched in time	Low	Business Logic	Acknowledged
5	Incomplete initialization	Low	Business Logic	Resolved
6	Third-party dependencies	Low	Dependency	Acknowledged
7	Typos	Informational	Code Quality	Resolved
8	Use of floating pragma	Informational	Configuration	Acknowledged

2.2 Notable Findings

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

1. The onERC721Received function can be called directly

Severity: High	Category: Business Logic
Target: - contracts/dao/ERC721LpListaDistributor.sol	

Description

In the `ERC721LpListaDistributor` contract, the normal deposit logic involves transferring the NFT to the contract first, then completing the remaining operations via the `onERC721Received()` function. However, the `onERC721Received()` function can be directly called by a malicious user to perform a deposit, and the caller can specify depositing to any account.

contracts/dao/ERC721LpListaDistributor.sol:L191-L1922

```
function onERC721Received(
    address operator,
    address from,
    uint256 tokenId,
    bytes calldata data
) override external returns (bytes4) {
    (bool isValid, uint256 liquidity) = checkNFT(tokenId);
    require(isValid, "invalid NFT");

    _addNFT(from, tokenId, liquidity);
    _deposit(from, liquidity);
    return IERC721Receiver.onERC721Received.selector;
}
```

This can result in the legitimate owner of a specific `tokenId` being unable to make a normal deposit, allowing a malicious user to perform deposits without consuming any NFTs.

Recommendation

Consider modifying the `onERC721Received` function logic to prevent it from being directly called by malicious users.

Status

The team has resolved this issue in commit 9569aa0.

2. Incorrect boundary usage in the loop causes the function to fail

Severity: Medium	Category: Business Logic
Target: - contracts/dao/ListaVault.sol	

Description

The `claimableList()` function in `ListaVault.sol` queries the distributor by id and calls its `claimableReward()` function.

contracts/dao/ListaVault.sol:L162-L168

```
function claimableList(address account, address[] memory distributors) external view
returns (uint256[] memory) {
    uint256[] memory claimable = new uint256[](distributors.length);
    for (uint16 i = 0; i <= distributors.length; ++i) {
        claimable[i] = IDistributor(idToDistributor[i]).claimableReward(account);
    }
    return claimable;
}
```

However, since the id starts at 1, `idToDistributor[0]` is the zero address, which has no callable functions. The loop includes the case where id is 0, causing the function to fail.

contracts/dao/ListaVault.sol:L103-L105

```
function registerDistributor(address distributor) external onlyRole(MANAGER) returns
(uint16) {
    ...
    ++distributorId;
    distributorUpdatedWeek[distributorId] = week;
    idToDistributor[distributorId] = distributor;
    ...
}
```

Recommendation

Consider using the correct starting value for the id in the loop.

Status

The team has resolved this issue in commit 9569aa0.

3. Centralization risk	
Severity: Medium	Category: Centralization
Target: - all	

Description

The `ListaVault` and `OracleCenter` contracts have privileged accounts(owner), the privileged accounts can call the `emergencyWithdraw()` function in `ListaVault.sol` and the `setOracle()` and `setFixedPrice()` functions in `OracleCenter.sol`. Other contracts have a _admin privileged address, which can grant permissions to any other address.

If the owner's or admin's private key is compromised, an attacker can withdraw all tokens in `ListaVault` contract, change the oracle address to a malicious contract address and grant roles to any other address.

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.

4. Rewards not fetched in time

Severity: Low	Category: Business Logic
Target: - contracts/dao/CommonListaDistributor.sol	

Description

When the current time is in the same week as `periodFinish`, the `fetchRewards()` function should be executed immediately after the `updateReward()` function is triggered to update key parameters.

However, the `_claimReward` function does not implement this logic, causing key variables such as `rewardRate` and `periodFinish` not to be updated, which may lead to unexpected contract behavior.

Recommendation

Consider calling the `fetchRewards()` function immediately after `updateReward` if the condition `getWeek(block.timestamp) >= getWeek(periodFinish)` is true.

Status

This issue has been acknowledged by the team. The bot will call the `fetchRewards()` function every week.

5. Incomplete initialization

Severity: Low	Category: Business Logic
Target: - contracts/dao/BorrowLisUSDListaDistributor.sol	

Description

The `BorrowLisUSDListaDistributor` contract inherits from `CommonListaDistributor`, which in turn inherits from `PausableUpgradeable`. However, the `initialize` function does not call the `__Pausable_init()` function.

contracts/dao/BorrowLisUSDListaDistributor.sol:L14

```
contract BorrowLisUSDListaDistributor is CommonListaDistributor
```

contracts/dao/CommonListaDistributor.sol:L15

```
abstract contract CommonListaDistributor is Initializable, AccessControlUpgradeable,
PausableUpgradeable
```

Recommendation

Consider calling the `__Pausable_init()` function to ensure proper initialization of the `PausableUpgradeable` functionality.

Status

The team has resolved this issue in commit e91306e.

6. Third-party dependencies

Severity: Low	Category:Dependency

Target:
- contracts/dao/OracleCenter.sol
- contracts/dao/ERC721LpListaDistributor.sol

Description

contracts/dao/OracleCenter.sol:L41-L60

```
function getPrice(address token0, address token1) external view returns (uint256) {
    ...
    if (price0 == 0) {
        price0 = oracle.peek(token0);
    }

    if (price1 == 0) {
        price1 = oracle.peek(token1);
    }
    ...
}
```

contracts/dao/ERC721LpListaDistributor.sol:L112-L148

```
function checkNFT(uint256 tokenId) public view returns (bool, uint256) {
    (
        ,
        ,
        address _token0,
        address _token1,
        uint24 _fee,
        ...
        ,
        ,
        ,

    ) = INonfungiblePositionManager(lpToken).positions(tokenId);
    ...
}
```

The `OracleCenter` and `ERC721LpListaDistributor` rely on third-party oracle contracts and NonfungiblePositionManager contracts, respectively, to obtain prices and check NFTs. The current audit treats third-party entities as black boxes and assumes they are working correctly. However, in reality, third parties could be compromised, resulting in abnormal contract operation.

Recommendation

We understand that the business logic requires interaction with the third parties. We encourage the team to regularly monitor the statuses of third parties to reduce the impacts when they are not functioning properly.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

7. Typos

Severity: Informational	Category: Code Quality
Target: - contracts/dao/SlisBnbDistributor.sol	

Description

The following code contains spelling errors.

contracts/dao/SlisBnbDistributor.sol:L88

```
// Rewards of current week will be exclued from the current epoch because they're not included in the merkle root
```

Recommendation

Consider fixing the spelling errors.

Status

The team has resolved this issue in commit 9569aa0.

8. 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.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 5a60674:

File	SHA-1 hash
contracts/dao/BorrowLisUSDListaDistributor.sol	a4eb67ecbce8f2e518cff25a9144fcd5a272a73e
contracts/dao/CommonListaDistributor.sol	70c6bc0e83ee884275f64c75f31b4104d9306f5f
contracts/dao/ERC20LpListaDistributor.sol	9221decf82d3fdfe72c75e3e2e7a88f708573f0d
contracts/dao/ERC721LpListaDistributor.sol	40627cca1f5d99433290711f05bbd6a22c09ed74
contracts/dao/ListaVault.sol	a5e65e11fe9627c17b7485c13407aeb2c83277fe
contracts/dao/OracleCenter.sol	f6e39789302138392256219fa131b1bc36d11778
contracts/dao/SlisBnbDistributor.sol	82ce4266eed5f16bfcc2b16f7ee61b9a07582fe1
contracts/dao/StakeLisUSDListaDistributor.sol	5e01493ecf3a47a504e7652a1149322d734ece69
contracts/library/TickMath.sol	ebc5c374e58df7ac02afd69b0eb2c7562813fd1a