TECHRIGHT

Security Assessment

# ZebraDAO

TechRight Verified on 15 Dec, 2023

## Disclaimer

## Description

Network

Base

Website

https://www.zebradao.finance/

## Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0

| Level | Value | Vulnerability | Risk<br>(Required Action) |
|---|---|---|---|
| Critical | 9 - 10 | A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken. | Immediate action to reduce risk level. |
| High | 7 - 8.9 | A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way. | Implementation of corrective actions as soon as possible. |
| Medium | 4 - 6.9 | A vulnerability that could affect the desired outcome of executing the contract in a specific scenario. | Implementation of corrective actions in a certain period. |
| Low | 2 - 3.9 | A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective. | Implementation of certain corrective actions or accepting the risk. |
| Informational | 0 - 1.9 | A vulnerability that has informational character but is not affecting any of the code. | An observation that does not determine a level of risk |

## Auditing Strategy and Techniques Applied

During the evaluation process, the repository was thoroughly examined to identify any security-related concerns, assess code quality, and ensure adherence to specifications and best practices. Our team of expert pentesters and smart contract developers reviewed the code line-by-line and documented any issues identified.

## Methodology

The auditing process follows a step-by-step routine:

1. Code review that includes:
   i. Review of the specifications, sources and instructions provided to TechRight to ensure a thorough understanding of the size, scope, and functionality of the smart contract's.

   ii. Manual review of code, which involves carefully reading the source code line-by-line to identify potential vulnerabilities.

   iii. Comparison to specification, which is the process of confirming whether the code performs as described in the specifications, sources, and instructions provided.

2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which involves assessing the degree to which test cases cover the code and how much of the code is executed while running those test cases.

   ii. Symbolic execution, which refers to the analysis of a program to identify the inputs that trigger each component of the program to execute.

3. Best practices review, which involves evaluating smart contracts to enhance efficiency, effectiveness, clarity, maintainability, security, and control in accordance with industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations that enable you to take necessary measures to secure your smart contracts.

## Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review

## Scope

This section lists files that are in scope for the metrics report.

- **Project:** `ZebraDAO`
- **Included Files:**
  - ``
- **Excluded Paths:**
  - ``
- **File Limit:** `undefined`
  - **Exclude File list Limit:** `undefined`
- **Workspace Repository:** `unknown` ( `undefined` @ `undefined` )

### Source Units in Scope

Source Units Analyzed: **`1`**
Source Units in Scope: **`1`** (**100%**)

| Type | File | Logic Contracts | Interfaces | Lines | nLines | nSLOC | Comment Lines | Complex. Score | Capabilities |
|---|---|---|---|---|---|---|---|---|---|
| 📝🔍 | ./src/rewardDistributor.sol | 2 | 1 | 486 | 476 | 285 | 135 | 265 | ✏️💰📤 |
| 📝🔍 | **Totals** | **2** | **1** | **486** | **476** | **285** | **135** | **265** | ✏️💰📤 |

Legend:

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

#### Out of Scope

##### Excluded Source Units

Source Units Excluded: **`0`**

| File |
|---|
| None |

##### Duplicate Source Units

Duplicate Source Units Excluded: **`0`**

| File |
|---|
| None |

##### Doppelganger Contracts

Doppelganger Contracts: **`0`**

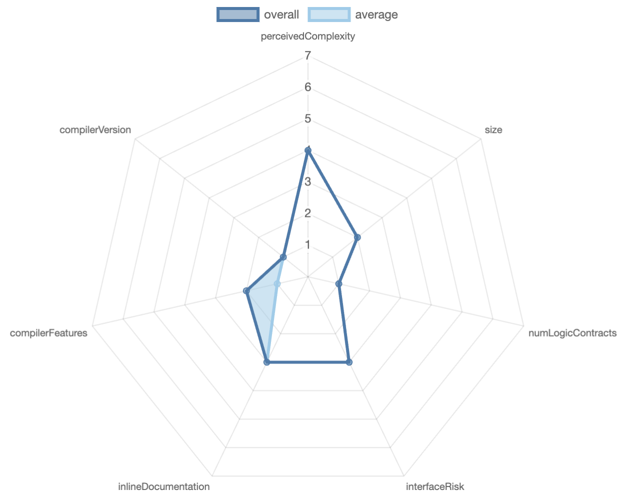
| File | Contract | Doppelganger |
|---|---|---|

## Report

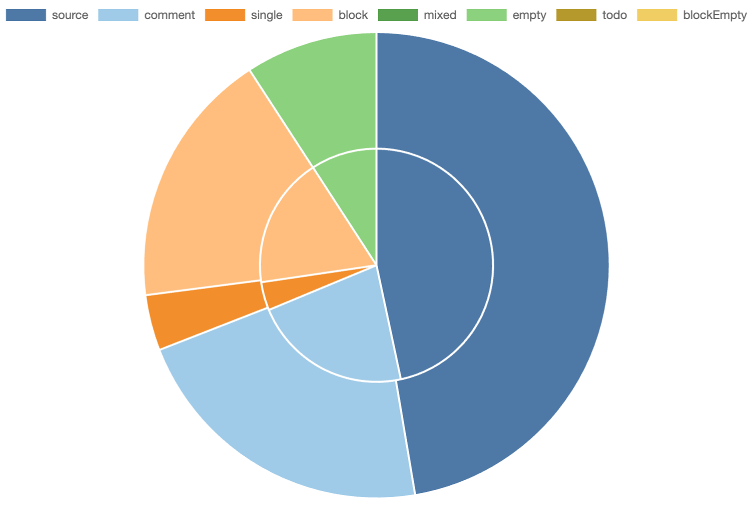
### Overview

The analysis finished with `0` errors and `0` duplicate files.

#### Risk



#### Source Lines (sloc vs. nsloc)



#### Inline Documentation

- **Comment-to-Source Ratio:** On average there are `2.18` code lines per comment (lower=better).
- **ToDo's:** `0`

#### Components

| Contracts | Libraries | Interfaces | Abstract |
|---|---|---|---|
| 2 | 0 | 1 | 0 |

#### Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

| Public | Payable |
|---|---|
| 20 | 2 |

| External | Internal | Private | Pure | View |
|---|---|---|---|---|
| 7 | 26 | 0 | 0 | 6 |

#### StateVariables

| Total | Public |
|---|---|
| 13 | 12 |

#### Capabilities

| Solidity Versions observed | Experimental Features | Can Receive Funds | Uses Assembly | Has Destroyable Contracts |
|---|---|---|---|---|
| `^0.5.16` | `ABIEncoderV2` | `yes` | ---- | ---- |

| Transfers ETH | Low-Level Calls | DelegateCall | Uses Hash Functions | ECRecover | New/Create/Create2 |
|---|---|---|---|---|---|
| `yes` | ---- | ---- | ---- | ---- | ---- |

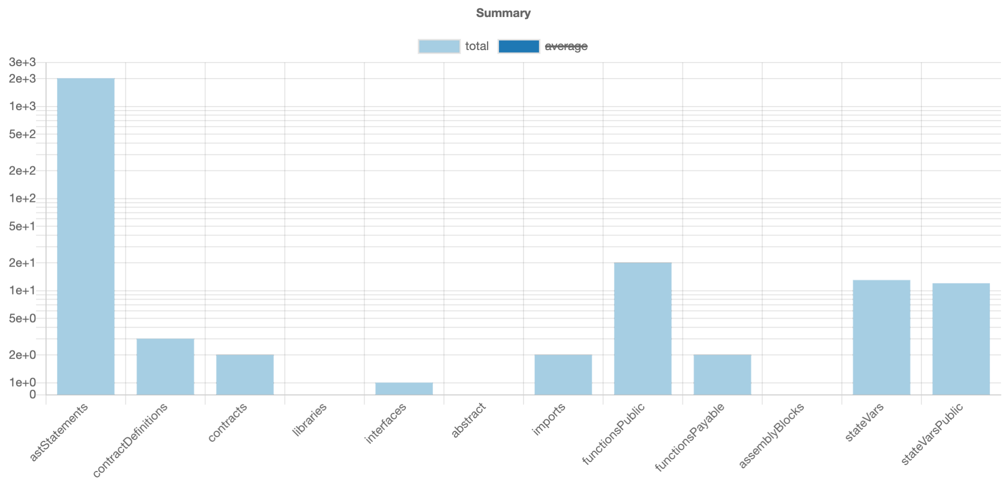
| ♻️ TryCatch | Σ Unchecked |
|---|---|
| | |

#### Dependencies / External Imports

| Dependency / Import Path | Count |
|---|---|

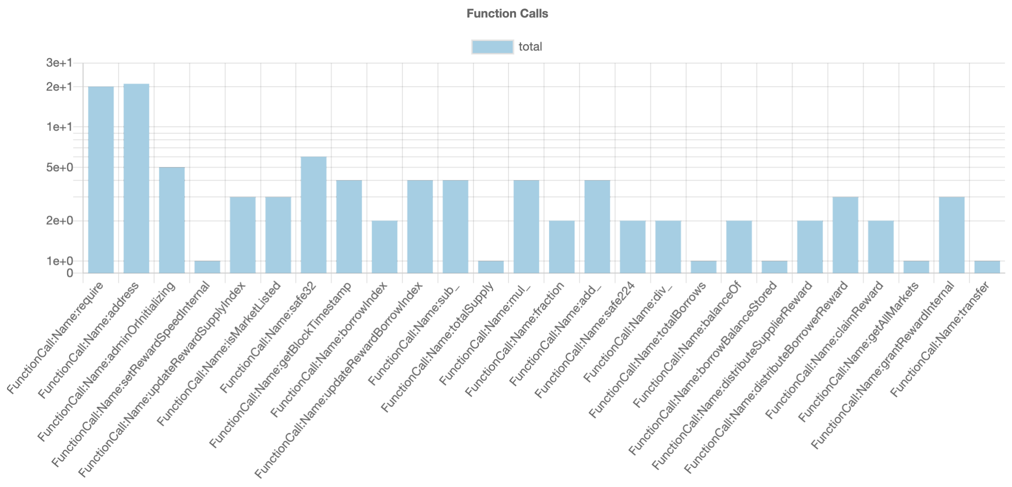
#### Totals

##### Summary

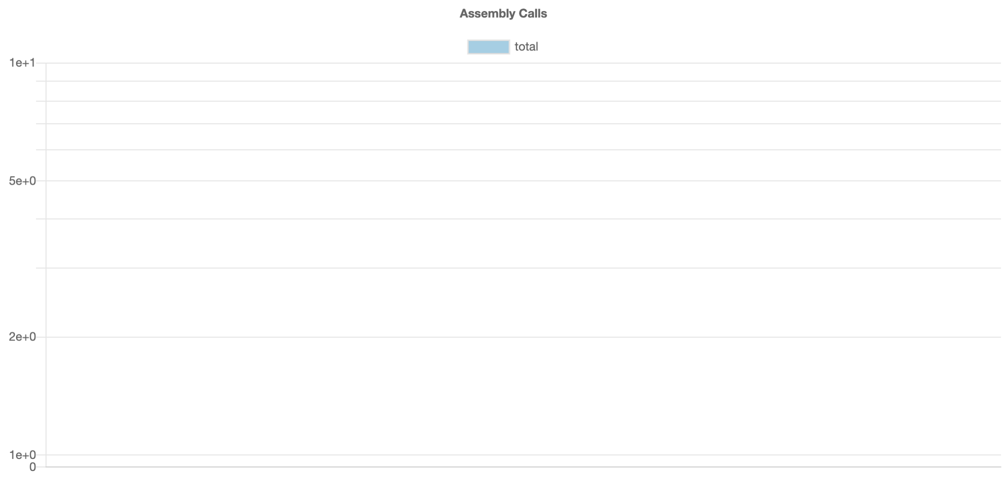


##### AST Node Statistics

#### Function Calls

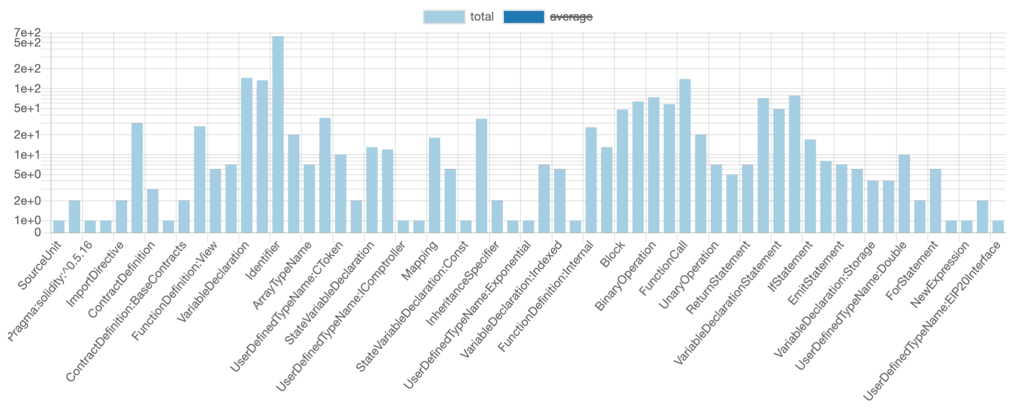


#### Assembly Calls



#### AST Total

AST Elements



**Inheritance Graph**

**Contract Summary**

Sūrya's Description Report Files Description Table

| File Name | SHA-1 Hash |
|---|---|
| ./src/rewardDistributor.sol | 3b64d52d10aa4d97cdc49fcf862596679642bdc7 |

Contracts Description Table

| Contract | Type | Bases | | |
|---|---|---|---|---|
| └ | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **IComptroller** | Interface | | | |
| └ | isMarketListed | External ❗️ | | NO❗️ |
| └ | getAllMarkets | External ❗️ | | NO❗️ |
| | | | | |
| **RewardDistributorStorage** | Implementation | | | |
| | | | | |
| **RewardDistributor** | Implementation | RewardDistributorStorage, Exponential | | |
| └ | | Public ❗️ | 🛑 | NO❗️ |
| └ | initialize | Public ❗️ | 🛑 | NO❗️ |
| └ | adminOrInitializing | Internal 🔒 | | |
| └ | _setRewardSpeed | Public ❗️ | 🛑 | NO❗️ |
| └ | setRewardSpeedInternal | Internal 🔒 | 🛑 | |
| └ | updateRewardSupplyIndex | Internal 🔒 | 🛑 | |
| └ | updateRewardBorrowIndex | Internal 🔒 | 🛑 | |
| └ | distributeSupplierReward | Internal 🔒 | 🛑 | |
| └ | distributeBorrowerReward | Internal 🔒 | 🛑 | |
| └ | updateAndDistributeSupplierRewardsForToken | External ❗️ | 🛑 | NO❗️ |
| └ | updateAndDistributeBorrowerRewardsForToken | External ❗️ | 🛑 | NO❗️ |
| └ | updateAndDistributeBorrowerRewardsForToken | External ❗️ | 🛑 | NO❗️ |
| └ | claimReward | Public ❗️ | 🛑 | NO❗️ |
| └ | claimReward | Public ❗️ | 🛑 | NO❗️ |
| └ | claimReward | Public ❗️ | 💵 | NO❗️ |
| └ | grantRewardInternal | Internal 🔒 | 🛑 | |
| └ | _grantReward | Public ❗️ | 🛑 | NO❗️ |
| └ | addRewardAddress | Public ❗️ | 🛑 | NO❗️ |
| └ | getRewardAddress | Public ❗️ | | NO❗️ |
| └ | getRewardAddressLength | External ❗️ | | NO❗️ |
| └ | setRewardAddress | Public ❗️ | 🛑 | NO❗️ |
| └ | setComptroller | Public ❗️ | 🛑 | NO❗️ |
| └ | setAdmin | Public ❗️ | 🛑 | NO❗️ |
| └ | | External ❗️ | 💵 | NO❗️ |
| └ | getBlockTimestamp | Public ❗️ | | NO❗️ |

Legend

| Symbol | Meaning |
|---|---|
| 🛑 | Function can modify state |
| 💵 | Function is payable |

## Detectors Issue

| Description | Check | Impact | Confidence |
|---|---|---|---|
| No outbound ETH transfer method available, potential ETH value loss. (rewardDistributor.sol#359) | logical | Medium | High |
| grantRewardInternal() ignores transfer return value. Extra value verification is recommended (rewardDistributor.sol#409) | logical | Medium | High |
| grantRewardInternal() mentions transfer of ETH, but no such method available. (rewardDistributor.sol#409) | logical | Medium | High |
| function claimReward(uint8 rewardType, address payable[] memory holders, CToken[] memory cTokens, bool borrowers,bool suppliers) is payable, but it is redundant. (rewardDistributor.sol#359) | logical | Medium | High |
| function setAdmin(address _newAdmin) missing zero address validation. (rewardDistributor.sol#473) | validation | Low | High |
| if (borrowers == true) is unoptimized. (rewardDistributor.sol#370) | optimization | Informational | High |
| function initialize() set to external is recommended. (rewardDistributor.sol#111) | logical | Informational | High |

## Summary

| CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL | OPTIMIZATION |
|---|---|---|---|---|---|
| 0 | 0 | 4 | 1 | 2 | 0 |

## Owner privileges

| No. | Issue | Description | Status |
|---|---|---|---|
| 1 | **No critical issues found** | The contract does contain issues of high or medium criticality (weak randomization) | **Passed** |
| 2 | **Contract owner cannot mint** | Contract owner does not have privilege on minting new tokens. | **Passed** |
| 3 | **Contract owner cannot blacklist addresses** | It is not possible to lock user NFTs by blacklisting addresses. | **Passed** |
| 4 | **Contract owner cannot set high fees** | The fees, if applicable, can be a maximum of 25% or lower. The contract can therefore not be locked. Please take a look in the comment section for more details. | **Passed** |
| 5 | **Contract cannot be locked** | Owner cannot lock any user funds. | **Passed** |
| 6 | **Token cannot be burned** | There is no burn function within the contract. | **Passed** |