yAudit Cozy safety module reward manager Review

Review Resources:

- Public Cozy documentation.

Auditors:

- Panda
- HHK

Table of Contents

Review Summary

Cozy safety module reward manager

The Cozy safety module reward manager provides staking contracts for the safety module receipt tokens. Depositors of any safety module can stake their tokens for a staked version of the receipt token and receive various token rewards. Managers can set a weight for each stackable token, multiple token rewards, and custom rates.

The contracts of the Cozy safety module reward manager Repo were reviewed over five days. Two auditors performed the code review between the 5th of May and the 9th of May 2025. The repository was under active development during the review, but the review was limited to the latest 5ca18ab03f8430be7285097eca83d86fb52e3700 for the Cozy safety module reward manager repo.

Scope

The scope of the review consisted of the following contracts at the specific commit:

```
src/
├── CozyManager.sol
├── interfaces
│   ├── IConfiguratorErrors.sol
│   ├── IConfiguratorEvents.sol
│   ├── ICozyManager.sol
│   ├── ICozyManagerEvents.sol
│   ├── IDepositorErrors.sol
│   ├── IDepositorEvents.sol
│   ├── IRewardsManager.sol
│   ├── IRewardsManagerFactory.sol
│   └── IStateChangerEvents.sol
├── lib
│   ├── Configurator.sol
│   ├── ConfiguratorLib.sol
│   ├── Depositor.sol
│   ├── RewardsDistributor.sol
│   ├── RewardsManagerBaseStorage.sol
│   ├── RewardsManagerCalculationsLib.sol
│   ├── RewardsManagerCommon.sol
│   ├── RewardsManagerInspector.sol
│   ├── RewardsManagerStates.sol
│   ├── Staker.sol
```

```
│   ├── StateChanger.sol
│   └── structs
│       ├── Configs.sol
│       ├── Pools.sol
│       └── Rewards.sol
├── RewardsManager.sol
├── RewardsManagerFactory.sol
└── StkReceiptToken.sol
```

After the findings were presented to the Cozy team, fixes were made and included in several PRs.

This review is a code review to identify potential vulnerabilities in the code. The reviewers did not investigate security practices or operational security and assumed that privileged accounts could be trusted. The reviewers did not evaluate the security of the code relative to a standard or specification. The review may not have identified all potential attack vectors or areas of vulnerability.

yAudit and the auditors make no warranties regarding the security of the code and do not warrant that the code is free from defects. yAudit and the auditors do not represent nor imply to third parties that the code has been audited nor that the code is free from defects. By deploying or using the code, Cozy safety module reward manager and users of the contracts agree to use the code at their own risk.

Code Evaluation Matrix

Category	Mark	Description
Access Control	Good	Role-based permissions well implemented with a clear ownership model.
Mathematics	Good	The mathematical calculations appear correct, and no arithmetic issues were identified.
Complexity	Average	Supporting multiple staked and reward tokens increases complexity. Certain architectural choices also contribute to this added complexity.
Libraries	Average	Some unnecessary imports and dependencies.
Decentralization	Good	Protocol allows permissionless operation with appropriate governance.
Code stability	Good	The codebase remained stable, with little to no new development introduced during the review.
Documentation	Good	Inline comments and function documentation are present.
Monitoring	Good	Good event coverage.
Testing and verification	Good	Good test coverage and invariant suite. Be cautious when using mock contracts for testing.

Findings Explanation

Findings are broken down into sections by their respective impact:

- Critical, High, Medium, Low impact
    - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements.
- Gas savings
    - Findings that can improve the gas efficiency of the contracts.
- Informational
    - Findings including recommendations and best practices.

Critical Findings

None.

High Findings

None.

Medium Findings

None.

Low Findings

1. Low - `updateUserRewardsForStkReceiptTokenTransfer()` doesn't drip rewards

Technical Details

The function `updateUserRewardsForStkReceiptTokenTransfer()` update rewards for users when they transfer their staked token to a new address.

However, it doesn't drip rewards and update the rewards pools before, which means users will only receive rewards up to the last `claimRewards()` or `stake()` call. Potentially, a small amount of rewards will be missing at the receiver address's benefit.

Impact

Low. Senders may miss a small amount of rewards.

Recommendation

Call `_dripAndApplyPendingDrippedRewards()` before updating a users rewards if the contract is not paused.

Developer Response

Acknowledged. We will leave the implementation unchanged. We had discussed this at time of design and wanted to keep it as is. One can obviously always explicitly claim rewards prior to transferring stake receipt tokens if desired.

Gas Saving Findings

1. Gas - Structs can be packed into fewer storage slots

Each slot saved can avoid an extra Gsset (**20000 gas**) for the first setting of the struct. Subsequent reads and writes have smaller gas savings.

Technical Details

```
File: src/lib/RewardsDistributor.sol

// @audit: 1 slot could be saved, by using a different order:
\*
 * uint256 amount; // (256 bits)
 * contract IERC20 rewardAsset; // (160 bits)
 * uint16 stakePoolId; // (16 bits)
 * uint16 rewardPoolId; // (16 bits)
 * uint16 claimFee; // (16 bits)
 * address owner; // (160 bits)
 * address receiver; // (160 bits)
 */

51: struct FinalizeClaimedRewardsArgs {
52:     uint16 stakePoolId;
53:     uint16 rewardPoolId;
54:     IERC20 rewardAsset;
55:     address owner;
56:     address receiver;
57:     uint256 amount;
58:     uint16 claimFee;
59:   }
```

[src/lib/RewardsDistributor.sol#L51]

```
File: src/lib/structs/Rewards.sol
```

```
// @audit: 1 slot could be saved, by using a different order:
\*
 * uint256 amount; // (256 bits)
 * uint256 claimFeeAmount; // (256 bits)
 * contract IERC20 asset; // (160 bits)
 * uint16 rewardPoolId; // (16 bits)
 */

41: struct PreviewClaimableRewardsData {
42:   // The ID of the reward pool.
43:   uint16 rewardPoolId;
44:   // The amount of claimable rewards.
45:   uint256 amount;
46:   // The claim fee amount.
47:   uint256 claimFeeAmount;
48:   // The reward asset.
49:   IERC20 asset;
50: }
```

src/lib/structs/Rewards.sol#L41

Impact

Gas savings.

Recommendation

Reorder the structure.

Developer Response

Fixed: https://github.com/Cozy-Finance/cozy-safety-module-rewards-manager/pull/61.

Note: I could not re-order the structs into exactly the order suggested above due to stack too deep issues, but both have been re-ordered to save 1 slot.

Informational Findings

1. Informational - Unused library `RewardsManagerCalculationsLib` and other imports

Technical Details

- The library `RewardsManagerCalculationsLib` is imported in multiple contracts but never used.

Since the stkToken is always worth 1 - 1 underlying asset, the library can be removed from imports and potentially from the codebase.

- Additionally, multiple contracts have unused imports. Consider removing them:

All of the imports missing:

```
File: src/interfaces/IDepositorEvents.sol

4: import {IReceiptToken} from "cozy-safety-module-
libs/interfaces/IReceiptToken.sol";
```

src/interfaces/IDepositorEvents.sol#L4

```
File: src/interfaces/IRewardsManager.sol

4: import {IDripModel} from "cozy-safety-module-libs/interfaces/IDripModel.sol";

6: import {IReceiptToken} from "cozy-safety-module-
libs/interfaces/IReceiptToken.sol";

7: import {IReceiptTokenFactory} from "cozy-safety-module-
libs/interfaces/IReceiptTokenFactory.sol";
```

src/interfaces/IRewardsManager.sol#L4, src/interfaces/IRewardsManager.sol#L6, src/interfaces/IRewardsManager.sol#L7

```
File: src/lib/Configurator.sol

9: import {StakePool, RewardPool} from "./structs/Pools.sol";
```

src/lib/Configurator.sol#L9

```
File: src/lib/Depositor.sol
```

```
4: import {IDripModel} from "cozy-safety-module-libs/interfaces/IDripModel.sol";

11: import {RewardsManagerCalculationsLib} from
"./RewardsManagerCalculationsLib.sol";
```

src/lib/Depositor.sol#L4, src/lib/Depositor.sol#L11

```
File: src/lib/RewardsDistributor.sol

12: import {StakePool, AssetPool} from "./structs/Pools.sol";
```

src/lib/RewardsDistributor.sol#L12

```
File: src/lib/RewardsManagerInspector.sol

5: import {RewardsManagerCalculationsLib} from
"./RewardsManagerCalculationsLib.sol";
```

src/lib/RewardsManagerInspector.sol#L5

```
File: src/lib/Staker.sol

11: import {RewardsManagerCalculationsLib} from
"./RewardsManagerCalculationsLib.sol";
```

src/lib/Staker.sol#L11

Impact

Informational.

Recommendation

Remove the imports.

Developer Response

Fixed: https://github.com/Cozy-Finance/cozy-safety-module-rewards-manager/pull/62.

2. Informational - Typos

Typos found on the code.

Technical Details

```
File: src/lib/Configurator.sol

// @audit: mantain should be maintain
29: // To mantain the invariant, before applying the update: we drip rewards,
update claimable reward indices and
```

src/lib/Configurator.sol#L29

```
File: src/lib/RewardsDistributor.sol

// @audit: accure should be accrue, occur, acquire
117: /// rewards they are entitled to as their new balance is smaller after the
transfer. Also, the `to_` user will accure

// @audit: accure should be accrue, occur, acquire
130: // Fully accure historical rewards for both users given their current
stkReceiptToken balances. Moving forward all
```

src/lib/RewardsDistributor.sol#L117, src/lib/RewardsDistributor.sol#L130

Impact

Informational.

Recommendation

Fix typos.

Developer Response

Fixed: https://github.com/Cozy-Finance/cozy-safety-module-rewards-manager/pull/59.

Final remarks

The Cozy Safety Module Reward Manager features a robust codebase with a comprehensive test suite; no critical vulnerabilities were found. However, the auditors caution against the extensive use of mock contracts in testing, as they may not always accurately reflect real

conditions and the additional complexity introduced by certain reward computation mechanisms. The team provided extra fixes that were not directly linked to findings, and these commits were also reviewed, up to PR#63.