Sentient Merkle Staking & Treasury

Security Review

Cantina Managed review by:
Om Parikh, Security Researcher
Red Swan, Security Researcher
Alireza Arjmand, Security Researcher Manager

November 26, 2025

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**
- 2.1 Scope . . . 3

3 Findings **4**
- 3.1 Low Risk . . . 4
  - 3.1.1 Hard-coded Gas Amount . . . 4
  - 3.1.2 Floating Pragma . . . 4
  - 3.1.3 Duplicate Code . . . 4
  - 3.1.4 Risk of Native Token Loss . . . 4
  - 3.1.5 Roots can't be denied when paused . . . 5
  - 3.1.6 `ADMIN_ROLE` can set bad timestamp . . . 5
  - 3.1.7 Unnecessary Computation . . . 5
- 3.2 Gas Optimization . . . 6
  - 3.2.1 Replace `ReentrancyGuard` with `ReentrancyGuardTransient` . . . 6
- 3.3 Informational . . . 6
  - 3.3.1 Duplicate Event Emissions . . . 6
  - 3.3.2 Code Clarity . . . 6
  - 3.3.3 Lack of Security Contact . . . 7
  - 3.3.4 Typo in `pendingMerkleProofs` variable naming . . . 7
  - 3.3.5 Unable to withdraw native token . . . 8
  - 3.3.6 Inactive reports are not reportable . . . 8
  - 3.3.7 Unchecked Loop Increments Are Now Optimized . . . 8

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings are a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Sentient's mission is to ensure that Artificial General Intelligence is open-source and not controlled by any single entity.

From Oct 21st to Oct 25th the Cantina team conducted a review of treasury-contracts, merkle-staking-contracts and artifact-contracts on commit hashes d12c4397, e5e4c8fe and 871f166d respectively. The team identified a total of **15** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	7	5	2
Gas Optimizations	1	0	1
Informational	7	3	4
Total	**15**	**8**	**7**

2.1 Scope

The security review had the following components in scope for Sentient Labs-merkle-staking-and-treasury on commit hash b81c5bf4:

```
├── artifact-contracts
│   └── contracts
│       └── src
│           ├── Artifact.sol
│           ├── ArtifactFactory.sol
│           └── ArtifactManager.sol
├── merkle-staking-contracts
│   └── src
│       ├── SentientRewardDistributor.sol
│       ├── SentientStakingDepositManager.sol
│       └── tokens
│           └── WSENT.sol
└── treasury-contracts
    └── src
        ├── AirdropVault.sol
        ├── Paymaster.sol
        ├── tokens
        │   └── WSENT.sol
        └── Treasury.sol
```

3 Findings

3.1 Low Risk

3.1.1 Hard-coded Gas Amount

Severity: Low Risk

Context: SentientRewardDistributor.sol#L418, SentientStakingDepositManager.sol#L299

Description: There are several locations that have hard-coded gas amounts:

- `emergencyWithdraw` in `SentientRewardDistributor`.
- `emergencyWithdraw` in `SentientStakingDepositManager`.

This is considered a bad practice in that gas costs can change or it might not be enough for smart contracts to utilize.

Recommendation: Consider forwarding all gas when making external calls.

Sentient Labs: Fixed in commit 973ce91c.

Cantina Managed: Fix verified.

3.1.2 Floating Pragma

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: On-chain code should have clear descriptions of what is does. That includes what compiler the developer intends the code to be compiled with so that future, valid but breaking compiler versions can be avoided.

Recommendation: Consider fixing the compiler version of ArtifactFactory.

Sentient Labs: Fixed in commit 7f4f973a.

Cantina Managed: Fix verified.

3.1.3 Duplicate Code

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: There are a few locations in the codebase where multiple functions provide the same functionality. Such code can make it harder to understand the features of a codebase or how it can be used.

- In `ArtifactManager` there are two functions that provide the active artifacts: `getArtifactList` and `artifactList()`.
- In `ArtifactManager` there are wrappers around the AccessControl `grantRole` and `revokeRole` functions just for a role bespoke to this contract: `grantFactoryRole` and `revokeFactoryRole`.

Recommendation: Consider whether these superfluous functions are critical to the contract's lifecycle and remove them if not.

Sentient Labs: Fixed in commit 03bdc3e1.

Cantina Managed: Fix verified.

3.1.4 Risk of Native Token Loss

Severity: Low Risk

Context: SentientStakingDepositManager.sol#L180-L200

Description: The `deposit` function in `SentientStakingDepositManager` is payable to allow users to stake SENT tokens on artifacts. It also allows users to stake ERC20 tokens. It does not, however, check, that if a user is staking ERC20 that they have not accidentally sent SENT with the transaction.

Recommendation: Consider adding such a check to protect a user from creating such an unintended, yet valid transaction.

Sentient Labs: Acknowledged. User should be careful. We will document this.

Cantina Managed: Acknowledged.

3.1.5 Roots can't be denied when paused

Severity: Low Risk

Context: SentientRewardDistributor.sol#L243

Description: If a merkle root was set just before a pause, then due to `whenNotPaused` modifier, authorized parties might not be able to deny it and window may pass before unpausing.

Given pauses are only executed during catastrophic times, and not allowing to deny roots during such periods might allow a malicious root being added. E.g:

- At t_0, root was added.
- At t_1, pause was triggered.
- Now at t_3 when sufficient time has passed, approve could be called (front-running deny if required).

Recommendation:

- Consider allowing `denyMerkleRoot` even if paused.
- This could be tackled on operational front by batching pausing with increase `merkleRootPendingPeriod` to a duration larger than for which contracts will be paused for.

Sentient Labs: Acknowledged. We will document this. We only would pause at extreme situations.

Cantina Managed: Acknowledged.

3.1.6 `ADMIN_ROLE` can set bad timestamp

Severity: Low Risk

Context: ArtifactManager.sol#L199

Description: The `reportArtifactUsage` function in the `ArtifactManager` contract allows the holder of the `ADMIN_ROLE` to storage usage data for any artifact. There is an implicit assumption that the `timestamp` used should increase over time and we believe this should be made explicit. It would effect all reporting on artifact usage if it were ever set to a previous time.

Recommendation: Revert if the timestamp is less or equal to a previous timestamp.

Sentient Labs: Fixed in commit 0059178e.

Cantina Managed: Fix verified.

3.1.7 Unnecessary Computation

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: In the `ArtifactManager` contract artifacts usage is stored and retrievable by iterating through the stored details. However, there is a top-level variable stored called `usageCount` that denotes the total usage of an artifact over time. This means then, that the function `_calculateTotalUsage` function is entirely superfluous as it iterates through to calculate this stored variable.

Recommendation: Remove this function and retrieve the stored value when needed.

Sentient Labs: Fixed in commit bc6f3269.

Cantina Managed: Fix verified.

3.2 Gas Optimization

3.2.1 Replace `ReentrancyGuard` with `ReentrancyGuardTransient`

Severity: Gas Optimization

Context: SentientStakingDepositManager.sol#L7

Description: In `SentientStakingDepositManager` and `SentientRewardDistributor`, `ReentrancyGuard` can be switched with `ReentrancyGuardTransient` for consistent gas savings across most of the user transactions.

Recommendation: Consider switching to `ReentrancyGuardTransient` if transient opcodes are supported.

Sentient Labs: Acknowledged.

Cantina Managed: Acknowledged.

3.3 Informational

3.3.1 Duplicate Event Emissions

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: In the codebase there are several functions that emit events to update off-chain systems of on-chain state changes. Some of these do not guard against the possibility of emitting an event even though the on-chain state has not changed. For example:

- `SentientRewardDistributor.setMerkleProofPendingPeriod`.
- `SentientStakingDepositManager.setDepositEnabled`.
- `SentientStakingDepositManager.setWithdrawEnabled`.
- `ArtifactFactory.setArtifactBeacon`.
- `ArtifactManager.setArtifactFactory`.

Recommendation: Check input parameters and revert if they are identical to the existing state they are assigning to.

Sentient Labs: Acknowledged. They are not used anywhere in the backend system.

Cantina Managed: Acknowledged.

3.3.2 Code Clarity

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: There are several locations where the code can be improved in its accuracy or clarity.

- The use of `unchecked` to increment loop counters in `ArtifactManager` when solidity now does this automatically as of version `0.8.22`.
- The use of `abi.encodeWithSelector` is not type safe and should be eschewed in favor or `abi.encodeCall`.
- Contracts that inherit `Context` have `_msgSender()` and `_msgData()` to allow chain-specific unwrapping of these values. Therefore, a consistent style of using these over `msg.sender` or `msg.data` is preferred. `ArtifactFactory`, `ArtifactManager`, `SentientRewardDistributor`, and `SentientStakingDepositManager` all are inconsistent in this style.
- In `SentientRewardDistributor`, the `SentientRewardDistributor__InvalidProofs` error is unused.

- In `SentientStakingDepositManager`, the `StakingTokenId` struct is unused.
- In `ArtifactManager`, there are two functions named `artifactList` that return different pieces of information.
- In `Artifact`, the `MAX_INITIAL_OWNERS` constant is unused.
- Both `IArtifact` and `IArtifactManager` define a `ArtifactInfo` struct which should be named differently as they represent different things.
- Consider documenting that, in `SentientRewardDistributor.claimRewards` for `keccak256(abi.encodePacked(msg.sender, amounts[i]));` holds valid only if there is atmost 1 entry per address in when constructing root.
- Using `codesize()` in `WSENT::withdraw` provides minimal gas optimizations at the expense of readbility and transparency. Consider replacing it with `0x00` for readability or even use mainline `WETH9`.
- The `_artifactList` state variable in the `ArtifactManager` contract is used to house only *active* artifacts. Its name however may mislead readers/developers to think that it houses all artifacts instead.
- The comments in `Artifact`'s `initialize` function describe storage slots and packing that is incorrect. `_owner` is stored in its own slot and the ArtifactInfo struct stores `_expectedApy` and `_commissionRate` as uint256 so there's no packing with those two either.

Recommendation: Consider updating the highlighted sections above to be more accurate or clear in their intent.

Sentient Labs: Acknowledged.

Cantina Managed: Acknowledged.

3.3.3 Lack of Security Contact

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: Adding a security contact into the smart contract allows good-faith actors to notify developers of issues in the code as quick as possible. Without such, it may be impossible for a found bug to be reported. None of the contracts in scope had such a security contact.

Recommendation: Consider adding a NatSpec comment containing a security contact at the top of the contract. At ethereum-lists is an instructive example.

Sentient Labs: Acknowledged.

Cantina Managed: Acknowledged.

3.3.4 Typo in `pendingMerkleProofs` variable naming

Severity: Informational

Context: SentientRewardDistributor.sol#L132

Description: consider renaming to `pendingMerkleProofs` to `pendingMerkleRoots` to reflect its correct purpose.

Recommendation:

```
- mapping(address => bytes32) public pendingMerkleProofs
+ mapping(address => bytes32) public pendingMerkleRoots
```

```
- mapping(address => bytes32) public pendingMerkleProofsTimestamp
+ mapping(address => bytes32) public pendingMerkleRootsTimestamp
```

```
- uint256 public merkleProofPendingPeriod = 1 hours;
+ uint256 public merkleRootPendingPeriod = 1 hours;
```

Sentient Labs: Acknowledged.

Cantina Managed: Acknowledged.

3.3.5 Unable to withdraw native token

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: In `SentientStakingDepositManager`, the `deposit` function allows users to use native tokens for their staking. The `withdraw` function returns these deposits as wrapped tokens.

Recommendation: Consider allowing the user to withdraw native tokens from the contract.

Sentient Labs: Fixed in commit bd442769.

Cantina Managed: Fix verified. `.transfer` is used which passes 2300 gas which is fine in this case, if receiver is not expected to do any computations in callback.

3.3.6 Inactive reports are not reportable

Severity: Informational

Context: ArtifactManager.sol#L274

Description: In the `ArtifactManager` contract, artifact reporting is retrieved using several functions. Though many of these only work if the artifact is active. As these are view functions and developed to provide value on chain, it seems curious that this reporting distinction would be made as the data is not private in any way.

Recommendation: Consider whether inactive artifacts should be reported on as well.

Sentient Labs: Fixed in commit bc6f3269.

Cantina Managed: Fix verified.

3.3.7 Unchecked Loop Increments Are Now Optimized

Severity: Informational

Context: ArtifactManager.sol#L182-L184

Description: In the codebase there are several locations where loop variables are incremented using unchecked variables. This was a widespread practice as the Solidity Compiler would add expensive bounds checking in loops. But as of Solidity v0.8.22, the compiler no longer does this and instead uses the method as coded here. An overview of the feature is explained in the Solidity blog release 0.8.22 announcement.

Recommendation: Consider removing unchecked loop increments.

Sentient Labs: Fixed in commit 9fb9f6da.

Cantina Managed: Fix verified.