ChainLight

OpenCohort Security Audit

：Ozys OpenCohort

October 7, 2024

Revision 1.0

ChainLight@Theori

Table of Contents

Executive Summary

Starting July 5, 2024, ChainLight of Theori audited Ozys's OpenCohort for three weeks. In the audit, we primarily considered the issues/impacts listed below.

- Theft of funds
- Permanent freeze of funds
- Abuse of authority by contract instance deployers
- Accounting issues in airdrop/vesting

As a result, we identified the issues listed below.

- Total: 3
- Medium: 2
- Informational: 1 (Minor issues including defense-in-depth suggestions)

Audit Overview

Scope

Name	OpenCohort Security Audit
Target / Version	• Git Repository (`0xSilicon/opencohort-contracts`): commit (after patch) `00cd357e400923f3989bdaf8130c5fed10f735af`
Application Type	Smart contracts
Lang. / Platforms	Smart contracts [Solidity]

Code Revision

N/A

Severity Categories

Severity	Description
Critical	The attack cost is low (not requiring much time or effort to succeed in the actual attack), and the vulnerability causes a high-impact issue. (e.g., Effect on service availability, Attacker taking financial gain)
High	An attacker can succeed in an attack which clearly causes problems in the service's operation. Even when the attack cost is high, the severity of the issue is considered "high" if the impact of the attack is remarkably high.
Medium	An attacker may perform an unintended action in the service, and the action may impact service operation. However, there are some restrictions for the actual attack to succeed.
Low	An attacker can perform an unintended action in the service, but the action does not cause significant impact or the success rate of the attack is remarkably low.
Informational	Any informational findings that do not directly impact the user or the protocol.
Note	Neutral information about the target that is not directly related to the project's safety and security.

Status Categories

Status	Description
Reported	ChainLight reported the issue to the client.
WIP	The client is working on the patch.
Patched	The client fully resolved the issue by patching the root cause.
Mitigated	The client resolved the issue by reducing the risk to an acceptable level by introducing mitigations.
Acknowledged	The client acknowledged the potential risk, but they will resolve it later.
Won't Fix	The client acknowledged the potential risk, but they decided to accept the risk.

Finding Breakdown by Severity

Category	Count	Findings
Critical	**0**	• N/A
High	**0**	• N/A
Medium	**2**	• `OPENCOHORT-002` • `OPENCOHORT-003`
Low	**1**	• `OPENCOHORT-001`
Informational	**1**	• `OPENCOHORT-004`
Note	**0**	• N/A

Findings

Summary

#	ID	Title	Severity	Status
1	OPENCOHORT-001	Tokens can be forcefully airdropped to the old beneficiary address in `OpenCohortAirdrop`	Low	Patched
2	OPENCOHORT-002	JSON injection in `MinimalNameTag.tokenURI()`	Medium	Patched
3	OPENCOHORT-003	Cohort metadata should be attested or provable by users	Medium	WIP
4	OPENCOHORT-004	Minor suggestions	Informational	Patched

#1 `OPENCOHORT-001` Tokens can be forcefully airdropped to the old beneficiary address in `OpenCohortAirdrop`

ID	Summary	Severity
`OPENCOHORT-001`	Since the identity-address mapping signature can be replayed, an attacker can trigger the airdrop claim to the old beneficiary address even if the beneficiary address for the identity has been changed.	Low

Description

When a `config.signer` creates multiple `OpenCohortAirdrop` contracts, a signature used in one `OpenCohortAirdrop.claimBySignature()` function could be reused in other `OpenCohortAirdrop` contracts. Although the ability to reuse a single signature across multiple `OpenCohortAirdrop` contracts is intentional, even if the beneficiary address (mapped from the identity `uniqueKey`) has changed (off-chain), tokens may still be claimed by the old beneficiary address through the reused signature.

Impact

Low

Airdrop can be claimed to the old beneficiary address even if the beneficiary address for the identity has been changed. The attacker is unlikely to profit from this. However, it may lead to the loss or theft of funds if the private key of the old address is lost or compromised.

Recommendation

Implement EIP-712 with replay protection to require a signature from the beneficiary address or have a mapping for revoked beneficiary address.

Remediation

Patched

When the beneficiary address associated with an identity changes, a mapping in the `OpenIdentityRegistry` contract can be updated to prevent using the old beneficiary address.

The `OpenCohortAirdrop` contract has been modified to refer to the `OpenIdentityRegistry` contract to verify that the beneficiary address has not been revoked.

#2 `OPENCOHORT-002` JSON injection in `MinimalNameTag.tokenURI()`

ID	Summary	Severity
`OPENCOHORT-002`	`MinimalNameTag.tokenURI()` is vulnerable to JSON injection since it does not escape special characters in property keys or values during JSON generation.	Medium

Description

When generating JSON that includes properties using `MinimalNameTag.tokenURI()`, special characters like `\` and `"` within property keys or values are not properly handled. The presence of these characters can cause the parsing of the generated JSON to fail. Moreover, a crafted key/value would allow the manipulation of fields that should not be affected by specific key/value settings.

Impact

Medium

The parsing of the generated JSON may fail, or fields that cannot be set through properties may be manipulated.

Recommendation

Restrict the character set for property keys and values or implement JSON string escaping mechanisms.

Remediation

Patched

The issue has been resolved as recommended.

#3 `OPENCOHORT-003` Cohort metadata should be attested or provable by users

ID	Summary	Severity
`OPENCOHORT-003`	Merkle Tree's contents may not match the corresponding fields in the `cohortMetadata`, which may lead to unexpected token disbursement in the `OpenCohortAirdrop` contract.	Medium

Description

The cohort item owner can set up `cohortMetadata` using `Cohort.mint()` or `Cohort.rollupWithSignature()`. The `totalWeight` and `totalCount` fields of `cohortMetadata` determine the airdrop amount of certain reward types. However, they may have discrepancies with the Merkle Tree's content, and there is no way to ensure it is not tampered with.

Impact

Medium

Cohort item owners may provide a false sense of token allocation to users and the market.

Recommendation

A trusted entity other than the cohort item owners should attest to the correctness of the Merkle Tree. Also, the entire content of the Merkle Tree should be provided by an off-chain service to prove it.

Remediation

WIP

An off-chain backend module providing the required features is currently under development.

#4 OPENCOHORT-004 Minor suggestions

ID	Summary	Severity
OPENCOHORT-004	The description includes multiple suggestions for preventing incorrect settings caused by mitigating potential issues, improving code maturity and readability, and other minor issues.	Informational

Description

Code Maturity

1. Consider renaming `cohortId` to `cohortTokenId` to clarify that it refers to the token's ID.
2. Add an underscore to the names of all internal and private functions for consistency.
3. Modify `CohortMetadata.parentCohort` in `CohortConfiguration` to match the relationship outlined in the documentation or remove the property if unnecessary.
4. Contracts like `OpenCohortAirdrop` and `MinimalNameTag` revert in the fallback function. Since it is possible to have the same behavior by not implementing a fallback function, consider removing it to enhance code maturity.

Other Recommendations

1. In `MinimalNameTag`, define acceptable max length for Key, Value, Name, and Description attributes in `_setNameTagMetadata()` and `_addProperty()` to avoid resource consumption issues in IPFS indexing and frontend.
2. Introduce logic in `OpenCohortAirdrop` for the owner to reclaim unclaimed tokens after a certain period and validate that the contract holds enough tokens during `_cohortTime` setup.
3. To modify a property in `MinimalNameTag`, a property must be first deleted through `removeProperty()` and added back through `addProperty()`. However, it may lead to a race condition, where a property is accessed after it is deleted but before it is added back. Consider adding a `modify` function to support atomic updates.
4. Strengthen the Merkle Tree validation in `OpenCohortAirdrop._claim()` by hashing leaf nodes twice to avoid potential exploits with empty proofs.
5. Add a public `totalClaimedAmount` variable to `OpenCohortAirdrop` to allow users to monitor remaining token balances.
6. Optimize `tokenOfOwnerByIndex()` by creating a mapping that tracks token issuance for improved efficiency.

7. In `Cohort.getSnapShot()`, replace `require(timestamp < block.timestamp);` with `require(timestamp <= block.timestamp);`.
8. Adjust the behavior of `Cohort.getExactTimeSnapShot()` to return an empty struct when no matching snapshot is found since the function looks for the snapshot at the specific time.

Missing or Confusing Events

1. In `MinimalNameTag`, remove the inappropriate `event Transfer` in non-asset transfer functions.

Recommendation

Consider applying the suggestions in the description above.

Remediation

Patched

Most suggestions are applied as recommended.

Revision History

Version	Date	Description
1.0	October 7, 2024	Initial version

Theori, Inc. (“We”) is acting solely for the client and is not responsible to any other party. Deliverables are valid for and should be used solely in connection with the purpose for which they were prepared as set out in our engagement agreement. You should not refer to or use our name or advice for any other purpose. The information (where appropriate) has not been verified. No representation or warranty is given as to accuracy, completeness or correctness of information in the Deliverables, any document, or any other information made available. Deliverables are for the internal use of the client and may not be used or relied upon by any person or entity other than the client. Deliverables are confidential and are not to be provided, without our authorization (preferably written), to entities or representatives of entities (including employees) that are not the client, including affiliates or representatives of affiliates of the client.