OZYS

OpenCohort (3Q)

Security Analysis Report

Prepared by

78ResearchLab

RedSpider

Sep 13, 2024

TABLE OF CONTENTS

PROJECT OVERALL

About Project

OpenCohort is a separate closely integrated with the Silicon network that categorizes users to support business applications, manage Web3 communities, and enhance social interactions among like-minded individuals. By forming hyper-personalized, decentralized cohorts, it enables partners and builders to manage memberships and communities within the broader Web3 ecosystem. User cohort data is securely stored on the blockchain, allowing individuals to expand their social activities into Web3 and connect with others who share similar interests.

Target Summary

Name	OpenCohort
Website	https://ozys.io/
Repository	https://github.com/0xSilicon/opencohort-contracts
Commit	ba04712bfb0318a7c6bb5a485f8482dd5e6d7a3f
Network	Silicon
Languages	Solidity
Method	Source code auditing
Timeline	June 19, 2024 ~ June 31, 2024

SCOPE

The `MinimalNameTag.sol` in this report is currently changed to `OpenNameTag.sol`.

Source code

Name	commit
OpenCohort	ba04712bfb0318a7c6bb5a485f8482dd5e6d7a3f

```
├── airdrop
│   ├── OpenCohortAirdrop.sol
├── configuration
│   ├── CohortConfiguration.sol
│   └── OpenCohortAirdropConfiguration.sol
├── deployer
│   └── OpenCohortAirdropDeployer.sol
├── interface
│   ├── ICohort.sol
│   ├── IERC5192.sol
│   ├── IMinimalNameTag.sol
│   ├── IOpenCohortAirdrop.sol
│   ├── IOpenCohortAirdropDeployer.sol
│   └── IProxy.sol
├── proxy
│   ├── SimpleProxy.sol
│   └── StandardProxyUnreceivable.sol
├── token
│   ├── Cohort.sol
│   ├── MinimalNameTag.sol
└── utils
    └── StringEscape.sol
```

RISK CLASSIFICATION

Severity

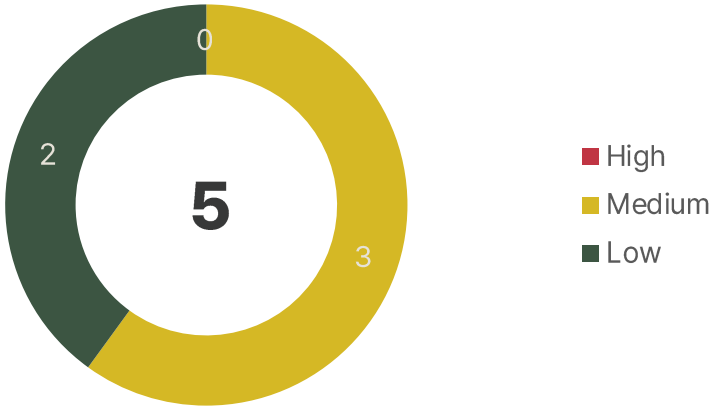
Our risk classification is based on Severity Categorization of code4ena.

> High ●
>
> Assets can be stolen, lost, compromised directly or indirectly via a valid attack path (e.g. Malicious Input Handling, Escalation of privileges, Arithmetic).

> Medium ●
>
> Assets not at direct risk, but the function of the protocol or its availability could be impacted, or leak value with a hypothetical attack path with stated assumptions, but external requirements.

> Low ●
>
> Assets are not at risk. User mistake, misuse of privileges, governance risk fall under this grade.

FINDINGS BREAKDOWN



Severity	Acknowledged	fixed	Total
● High	0	0	0
● Medium	1	2	3
● Low	1	1	2
			5

* Fixed : Risk is fixed by Ozys.
* Acknowledged : Ozys has recognized the risk but has not addressed it, as it poses only a minor impact.

FINDINGS

MEDIUM

M-01. OpenCohortAirdrop: Signature replay attack can be used to force an old beneficiary to receive a token

Acknowledged

IMPACT

The previously used beneficiary can be forced to receive the next airdrop tokens.

DESCRIPTION

The `claimBySignature` function allows tokens to be airdropped to the beneficiary address instead of the uniqueKey used to generate the Merkle tree. It verifies the beneficiary address with a signature from the registered signer. But the hash being signed does not include the address of the OpenCohortAirdrop.

Suppose user X called `claimBySignature` on OpenCohortAirdrop A and received an airdrop. OpenCohortAirdrop B is deployed on the same chain with the same signer, and the uniqueKey of the same user X is used to generate the Merkle tree. User X wants to receive the airdrop on OpenCohortAirdrop B to a different address. However, if an attacker reuses the signature used in OpenCohortAirdrop A and calls `claimBySignature`, they can make the airdrop be received to the beneficiary address used in OpenCohortAirdrop A.

```
function claimBySignature(uint256 index, address uniqueKey, uint256 weight, bytes32[]
calldata proof, address beneficiary, bytes memory signature) external {
    OpenCohortAirdropConfig memory config = openCohortAirdropConfig();
    require(config.signer != address(0));

@>  bytes32 hash = keccak256(abi.encode(cohort(), getChainId(), uniqueKey,
beneficiary));
    if(config.signer.code.length == 0) {
        require(signature.length == 65);

        bytes32 signingHash = MessageHashUtils.toEthSignedMessageHash(hash);
        require(config.signer == ECDSA.recover(signingHash, signature));
    }
    else require(IERC1271(config.signer).isValidSignature(hash, signature) ==
IERC1271.isValidSignature.selector);

    _claim(config, index, uniqueKey, weight, proof, beneficiary);
}
```

File 1 : airdrop/OpenCohortAirdrop.sol#267-281 Function: claimBySignature

RECOMMENDATIONS

Include `address(this)` in the hash to prevent signature replay attacks between different OpenCohortAirdrop contracts.

```
function claimBySignature(uint256 index, address uniqueKey, uint256 weight, bytes32[]
calldata proof, address beneficiary, bytes memory signature) external {
    OpenCohortAirdropConfig memory config = openCohortAirdropConfig();
    require(config.signer != address(0));

-   bytes32 hash = keccak256(abi.encode(cohort(), getChainId(), uniqueKey,
beneficiary));
+   bytes32 hash = keccak256(abi.encode(address(this), cohort(), getChainId(),
uniqueKey, beneficiary));
    if(config.signer.code.length == 0) {
        require(signature.length == 65);

        bytes32 signingHash = MessageHashUtils.toEthSignedMessageHash(hash);
        require(config.signer == ECDSA.recover(signingHash, signature));
    }
    else require(IERC1271(config.signer).isValidSignature(hash, signature) ==
IERC1271.isValidSignature.selector);

    _claim(config, index, uniqueKey, weight, proof, beneficiary);
}
```

STATUS

Acknowledged

Ozys: We think that it is inconvenient to request a signature from the signer for each airdrop, so the reuse of signatures across various airdrops is an intended implementation. We are currently discussing the policy on whether to allow the beneficiary address to be changed, so for now, we will maintain the current implementation.

M-02. MinimalNameTag, Cohort: Third parties cannot detect metadata changes because supportsInterface does not indicate ERC4906 support

Fixed

IMPACT

Third parties cannot detect ERC4906 support

DESCRIPTION

ERC4906 is a standard that notifies when the metadata of an ERC721 NFT has been updated. The MinimalNameTag contract triggers the `MetadataUpdate` event in the `setNameTagMetadata`, `addProperty`, `addPropertyBatch`, and `removeProperty` functions, which seems to be intended to follow the ERC4906 standard. According to https://eips.ethereum.org/EIPS/eip-4906#specification, to comply with the ERC4906 standard, it must return `true` when queried with `0x49064906` in `supportsInterface`.

> *The supportsInterface method MUST return true when called with 0x49064906.*

ERC4906 is generally used to allow third parties like Opensea to detect changes in metadata. To make third parties aware that the NFT supports ERC4906, it must be queryable in `supportsInterface`.

However, the MinimalNameTag contract does not indicate support for ERC4906 in `supportsInterface`.

```
function supportsInterface(bytes4 interfaceId) public view virtual override(ERC165)
returns (bool) {
    return
        interfaceId == type(IERC721).interfaceId ||
        interfaceId == type(IERC721Metadata).interfaceId ||
        interfaceId == type(IERC5192).interfaceId ||
        super.supportsInterface(interfaceId);
}
```

File 2 : token/MinimalNameTag.sol#L66-72 Function: supportsInterface

The Cohort contract also has the same issue.

RECOMMENDATIONS

Indicate support for ERC4906 in the `supportsInterface` of the MinimalNameTag and Cohort contracts.

```
function supportsInterface(bytes4 interfaceId) public view virtual override(ERC165)
returns (bool) {
    return
        interfaceId == type(IERC721).interfaceId ||
        interfaceId == type(IERC721Metadata).interfaceId ||
        interfaceId == type(IERC5192).interfaceId ||
+       interfaceId == bytes4(0x49064906) ||
        super.supportsInterface(interfaceId);
}
```

STATUS Fixed

Ozys: Modified to indicate support for ERC4906 in `supportsInterface`.

Fixed in commit `5a6e65aeaaca45e1dd8e2724dc13557fea5f23f9`.

M-03. MinimalNameTag: JSON injection can inject unintended metadata or payloads

Fixed

IMPACT

Unintended metadata fields can be added or used as an attack trigger point.

DESCRIPTION

MinimalNameTag NFT allows the owner of the NFT to freely set their NFT metadata. Users can set the name, description, image fields, and attributes field. `tokenURI` creates and returns a JSON using the metadata registered by the user.

There is no verification to check if the value entered by the user is correct when registering metadata. Therefore, users can inject payloads into the metadata to escape JSON fields and create fields other than the specified ones.

```
function tokenURI(uint256 tokenId) public view virtual returns (string memory) {
    require(tokenId != 0);
    require(_owners[tokenId] != address(0));

    NameTagMetadata memory nameTagMetadata = _metadata[tokenId];

    bytes memory attributes;
    uint256 tokenPropertyCount = _propertyCount[tokenId];
    if(tokenPropertyCount == 0){
        attributes = bytes('"}');
    }
    else{
        attributes = bytes('","attributes":[');
        for(uint256 i = 0; i < tokenPropertyCount; i++){
            string memory key = _propertyKeyList[tokenId][i];
            string memory value = _properties[tokenId][key];
            string memory end = i == (tokenPropertyCount - 1) ? '"}]}' : '"},';
            bytes memory attribute = abi.encodePacked(
                '{"trait_type":"',
@>              key,
                '","value":"',
@>              value,
@>              end
            );

            attributes = abi.encodePacked(
                attributes,
                attribute
            );
        }
```

```
    }

    return string(abi.encodePacked(
        'data:application/json;base64,',
        Base64.encode(
            abi.encodePacked(
                '{"name":"',
@>              nameTagMetadata.name,
                '", "description":"',
@>              nameTagMetadata.description,
                '", "image":"',
@>              nameTagMetadata.image,
@>              attributes
            )
        )
    ));
}
```

File 3 : token/MinimalNameTag.sol#L138-184 Function: tokenURI

Due to this, it can become an attack trigger point whenever there are vulnerabilities in third-party or Quickstarter's front/back end that display or collect metadata.

For example, let's say the `nameTagMetadata.name` value is set to `"nft_name", "external_url": "malicious_url`. This would add an unintended external_url field. The external_url is linked to the view website button in third parties like Opensea.

Additionally, according to the Opensea documentation, the animation_url field can utilize HTML or Javascript. Arbitrarily adding an animation_url field could cause issues such as displaying a malicious page.

The image field can also include SVGs to attempt XSS attacks.

RECOMMENDATIONS

Filter out special characters, empty strings, and other characters that can cause issues. You can use a library like Solady's LibString.escapeJSON. Alternatively, you can add an admin function to modify malicious user metadata and prevent users from changing it.

STATUS Fixed

Ozys: Used Solady library's `escapeJSON` to prevent injection.

Fixed in commit `a37306a2b61aede430c1628eef8d8d99716b965f` .

LOW

L-01. Cohort, OpenCohortAirdropDeployer : Implemented to be upgradeable, but inherited contracts are not upgradeable

Acknowledged

IMPACT

In an upgradeable implemented contract, a non-upgradeable contract is inherited.

DESCRIPTION

The Cohort, OpenCohortAirdropDeployer contracts are implemented as UUPSUpgradeable. However, these contracts inherit Ownable instead of OwnableUpgradeable. Since storage gaps are not registered, there is a possibility that the storage layout could be broken if variables are added during future library version upgrades.

RECOMMENDATIONS

Use OwnableUpgradeable instead of Ownable, or be cautious with storage during upgrades.

STATUS

Acknowledged

Ozys: We will not fix this issue but will be cautious during upgrades.

L-02. MinimalNameTag: Implemented to prevent receiving native tokens, but proxies can still receive native tokens

Fixed

IMPACT

Although the logic contract is implemented to not receive native tokens, the proxy can still receive native tokens.

DESCRIPTION

The MinimalNameTag contract is intended to not receive native tokens by reverting in the `receive` and `fallback` functions.

```
receive() external payable { revert(); }
fallback() external payable { revert(); }
```

File 4 : token/MinimalNameTag.sol#L52-53 Function: receive, fallback

From the deployment script, the contract is used via StandardProxy, which implements the `receive` function so it can receive native tokens.

```
contract StandardProxy is TransparentUpgradeableProxy {
    ...

@>  receive() external payable {}
}
```

File 5 : proxy/StandardProxy.sol#L23 Function: receive

When native tokens are sent to the proxy of MinimalNameTag, the proxy's fallback is called, which invokes `StandardProxy.receive` before `MinimalNameTag.receive` is called. Therefore, the proxy of MinimalNameTag can receive native tokens.

RECOMMENDATIONS

If you want to prevent the proxy of MinimalNameTag from receiving native tokens, use a proxy without a `receive` function.

STATUS

Fixed

Ozys: Modified to use a proxy with the `receive` function disabled.

Fixed in commit `aeff59c566309ba4f02cd9ee298125fca0b0e9fb`.

ABOUT 78ResearchLab

78ResearchLab is a offensive security corporation offering security auditing, penetration testing, education to enterprises, national organizations, and laboratories with the goal of making safe and convenience digital world. We have our own proprietary technology from system/security analysis and projects on various industries. We are working with the top technical experts who have won prizes in global Realword Hacking Competition/CTF, reported numerous security vulnerabilities, and have 10 years of experience in the information security.
Learn more about us at https://www.78researchlab.com/.

ABOUT RED SPIDER

Red Spider offers cyber security research and auditing service with our new R&D technologies and customized solution in IoT, OS, Web3.
Learn more about red spider at https://www.78researchlab.com/redSpider.html.