OZYS

Open-Wallet & Tokenized-Vault Security Analysis Report

Prepared by

78ResearchLab

RedSpider

Jan 20, 2025

TABLE OF CONTENTS

PROJECT OVERALL

About Project

The Open-Wallet enables the creation of virtual wallets by signers, even in the absence of an assigned owner. This system allows tokens to be securely stored in the wallet until ownership is transferred to a designated owner. Once the owner is authenticated, the wallet ownership is transferred, granting them full control over the stored tokens. For example, a signer (e.g., Riot) can create a wallet for an owner (e.g., Faker). Upon proper authentication, Riot transfers the wallet ownership to Faker, who can then utilize the tokens as desired.

Tokenized-Vault is a reward token vault used by Vennie Service. The reward token in this vault is shared across multiple airdrops. It was implemented to address the need to set a unified budget for all airdrops collectively when creating multiple airdrops. By setting Tokenized-Vault as the reward token, the availability of claims across all airdrop events can be controlled based on the amount held in the vault contract. This approach improves convenience compared to the previous method, which required managing each airdrop separately.

Target Summary

Name	Open-Wallet, Tokenized-Vault
Website	
Repository	https://github.com/0xSilicon/opencohort-contracts/
Commit	Tokenized-Vault : 84a8f33c63c1484721bde75ec56c0f5450ee38a6 Open-Wallet : 56aaff9c49fc380416bc6cb674dabc79e91d319c
Network	Silicon
Languages	Solidity
Method	Source code auditing
Timeline	Jan 13, 2025 ~ Jan 17, 2025

SCOPE

The audit will focus on the Open-Wallet and Tokenized-Vault, ensuring their functionality, security, and compliance with specified requirements. Key objectives include verifying the ownership transfer and wallet deployment processes in Open-Wallet, including token and native token handling with proper fee mechanisms. For the Tokenized-Vault, the review will ensure seamless integration as a reward token repository for multiple airdrop contracts, particularly the claim control mechanism that halts claims when the vault is depleted. Both contracts' administrative roles, ERC20 compliance, and upgradeability will be assessed, alongside their security and behavior under edge cases.

Source code

Name	commit
Tokenized-Vault	84a8f33c63c1484721bde75ec56c0f5450ee38a6
Open-Wallet	56aaff9c49fc380416bc6cb674dabc79e91d319c

```
contracts/
├── deployer
│   └── SiliconProtocolManagerDeployer.sol
├── interface
│   ├── INamedWallet.sol
│   ├── ITokenizedVault.sol
│   └── IWalletFactory.sol
├── namedWallet
│   ├── NamedWallet.sol
│   └── WalletFactory.sol
└── utils
    ├── SiliconProtocolManager.sol
    └── TokenizedVault.sol

4 directories, 8 files
```

RISK CLASSIFICATION

Severity

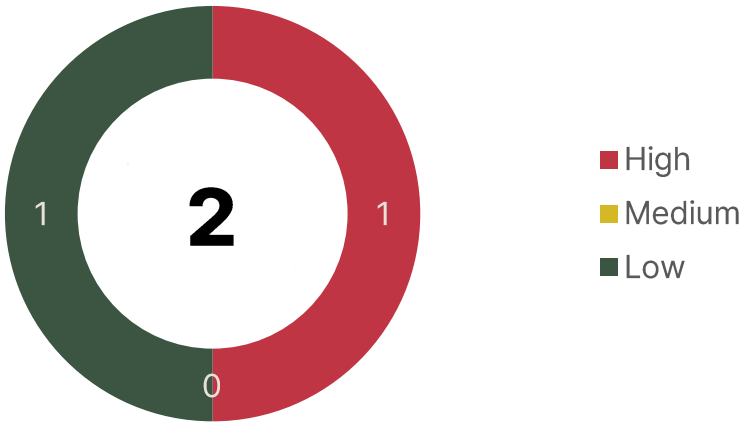
Our risk classification is based on Severity Categorization of code4ena.

> 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	1	1
● Medium	0	0	0
● Low	0	1	1
			2

* 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

HIGH

H-01. OpenWallet : The Owner can bypass tax obligations by upgrading OpenWallet. Fixed

IMPACT

If the owner of OpenWallet has acquired ownership from the signer, they can upgrade the contract to avoid paying the tax owed to the signer when transferring tokens.

DESCRIPTION

OpenWallet allows users to deposit token even when the wallet deos not have an owner. These deposited tokens can be transferred to external addresses using the `transferTo` or `transferTokenTo` functions, which are exclusively callable by the owne after they acquire ownership from the signer.

```
function transferTokenTo(address token, address to, uint256 amount) public onlyOwner
nonReentrant{
    uint256 tax = amount * rate / 10000;
    amount -= tax;
    require(to != address(0), "Invalid address");
    if(amount > 0){
        IERC20(token).safeTransfer(to, amount);
        emit TransferByOwner(token, to, amount);
    }
    if(tax > 0){
        IERC20(token).safeTransfer(signer, tax);
        emit TransferByOwner(token, signer, tax);
    }
}
```

File 1 : OpenWallet.sol

The transfer process is implemented to allocate a portion of the amount as tax to the signer, but this can be easily bypassed if the owner upgrades the contract.

```
function _authorizeUpgrade(address newImplementation) internal override view {
    if(owner != address(0)){
        require(msg.sender == owner, "Unauthorized");
    } else {
        require(msg.sender == signer, "Unauthorized");
    }
    require(newImplementation != address(0), "Invalid address");
}
```

File 2 : OpenWallet.sol

The `_authorizeUpgrade` function in UUPS, which handles upgrade authorization, permits contract upgrades if an owner exists and the `msg.sender` initiating the upgrade is the owner. This enables a malicious owner to upgrade the contract and remove the tax-imposing logic for token transfers, preventing the signer from receiving any tax.

RECOMMENDATIONS

The owner should be restricted from upgrading the contract, and any necessary upgrades should be conducted by the signer.

STATUS Fixed

Ozys : We plan to modify the system so that the signer, who is responsible for authorizing the transfer of the wallet to the owner, also holds the authority for upgrades. However, there are concerns about potential cost issues if the signer is required to directly handle upgrades for all wallets.

Therefore, we plan to develop a method where the owner obtains a signature provided by the signer (which includes the address of the new implementation) and directly requests the upgrade during `_authorizeUpgrade`. What are your thoughts on this approach?

78 : There doesn't seems to be any significant issues. Once the implmentation is complete, we can conduct another audit on this part to ensure its integrity.

Ozys : Fixed in commit `ba67e495da422bd465286451fce10a5575aeb142`.

We also included the data value in the hash to ensure that only authorized data can be executed : commit `f9ca4aac510415f1185b775d115fc54a2d80f49c`.

● LOW

L-01. OpenWallet : Lack of validation for the rate value

Fixed

IMPACT

There is no validation for the `rate` value used to caculate the tax . If the `rate` is set outside the range of 10,000, `tranfer*` functions will become unusable until the `rate` is adjusted to a valid range.

DESCRIPTION

The `rate` has an invariant of from 0 to 10,000. If the value is out of this range, the contract may fail to function properly when transferring assets.

`rate` can be set through `initialize` and `changeTaxRate` functions.

```
function initialize(WalletInfo calldata walletInfo, address _openNameTag, address _signer)
    public initializer{
        signer = _signer;
        factory = msg.sender;
        rate = walletInfo.rate;
        openNameTag = _openNameTag;
        IOpenNameTag.NameTagMetadata memory nameTagMetadata =
IOpenNameTag.NameTagMetadata(walletInfo.name, walletInfo.description, walletInfo.image);
        mintNameTag(nameTagMetadata, new string[](0), new string[](0));
        emit InitializedInfo(walletInfo.name, walletInfo.image, walletInfo.description,
walletInfo.rate, _openNameTag, _signer);
        }

function changeTaxRate(uint256 rate_) public onlySigner{
    rate = rate_;
    emit ChangeTaxRate(rate_);
}
```

File 3 : OpenWallet.sol

RECOMMENDATIONS

Add a `require` statement to validate the range of the `rate`.

STATUS

Fixed

Ozys : Restricted the range so that the `rate` value can only be set up to 10,000 : Fixed in commit `185c2c72ad6df86e27c8753df0258cabb25b19ba`.

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.