CD SECURITY

AUDIT REPORT

Libree

September 2023

Introduction

A time-boxed security review of the **Libree** protocol was done by **ddimitrov22** and **chrisdior4**, with a focus on the security aspects of the application's implementation.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or if even the review will find any problems with your smart contracts.

About **Libree**

Libree is a DeFi ecosystem that allows seamless finance and DAO interactions for Web 3 creators, their subscribers, and other revenue makers. The focus of the audited scope was the subscription service which provides the ability to the users to subscribe to their favorite creators and to have access to their private content by paying periodic fees.

A new `subscription` is created by the `Creator` with `createSubscription()` in `FactoringManager.sol`. Thus, a `Vault` is created where the payment fees collected from the users are stored. Every creator receives a unique `Flow Token` which is an NFT and enables the `Creator` to perform different actions regarding their `Subscription` and `Vault` like withdrawing funds, changing fees, and even changing the manager of the `Vault`.

The `Flow Token` can be sold on Libree's marketplace or third-party marketplaces, as well as used as collateral for debt issuance. In such a scenario, a `Debt Token` is received by the borrower which has specific terms and conditions agreed upon creation.

Threat Model

Privileged Roles & Actors

- Creator - the creator of a subscription service which is the `manager` of a `Vault` and the holder of a unique `Flow Token`.
- Creator's Subscription - the actual exclusive content for which the users can subscribe.
- Subscribers - the regular users that pay monthly to have access to the Creator's Subscription
- `Flow Token` - an NFT which represents the authority over the `Vault` and the Creator's Subscription.
- `Debt Token` - a separate token which represents a debt when the `Flow Token` is used as collateral.

Security Interview

Q: What in the protocol has value in the market?

A: The monthly payments stored in the `Vault`, the `Flow Token` and the `Debt Token`.

Q: In what case can the protocol/users lose money?

A: The `Creator` can lose money if he lose access to the `Vault`.

Q: What are some ways that an attacker achieves his goals?

A: An attack that drains the vaults or steal the `Flow Token`.

Severity classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Scope

The following smart contracts were in scope of the audit:

- `LibreFlowConfig.sol`
- `LibreeHub.sol`
- `Constants.sol`
- `Errors.sol`
- `NFTActions.sol`
- `Manager.sol`
- `FactoringManger.sol`
- `NFTToken.sol`
- `Vault.sol`
- `VaultFactory.sol`

The following number of issues were found, categorized by their severity:

- Critical & High: 0 issues
- Medium: 1 issues
- Low: 1 issues
- Informational: 2

Findings Summary

ID	Title	Severity
[M-01]	Insufficient input validation	Medium
[L-01]	Use `Ownable2Step` rather than `Ownable`	Low

ID	Title	Severity
[I-01]	Redundant code	Informational
[I-02]	Missing event emission	Informational

[M-01] Insufficient input validation

Severity

Impact: Medium, because a protocol can be broken and the code could give a false calculations

Likelihood: Medium, as it can be gamed but it needs compromised / malicious owner

Description

The `_fee` parameter in `setFee()` is missing any constrains and if we have malicious or compromised owner there might be a serious problem. The comments above the declaration of `fee` variable says that a value of 100 is 1% fee:

```
     * @notice Fee value in basis points.
     * @dev Value of 100 is 1% fee.
     */
    uint16 public fee;
```

but since we are missing any input validation the `fee` can be set to uint16 max value which is 65535. This value will be equal to ~ 650% fee. The same can happen in the `constructor` as well as the `setFee()` method is called during construction time.

Recommendation

Set a reasonable upper constrain for `fee`. For example 5%, or as much as you decide:

```
if(_fee > 500) {
revert MaxFeeIs5Percent();
}
```

Client

Fixed

[L-01] Use `Ownable2Step` rather than `Ownable`

Description

Several important methods in scope are using the `onlyOwner` modifier which shows that the owner role is an important one. Make sure to use a two-step ownership transfer approach by using `Ownable2Step` from OpenZeppelin as opposed to `Ownable` as it gives you the security of not unintentionally sending the owner role to an address you do not control.

Client

Fixed

[I-01] Redundant code

The getter functions in `LibreFlowConfig.sol` are redundant since the return variables that they are returning are public storage variables that have automatically generated getters.

Also: `managerNames` mapping in `LibreeHub.sol` is not updated anywhere in the whole codebase. Despite that we have the following check in `LibreeHub.sol`:

```
if (bytes(managerNames[_manager]).length == 0) {
            managers.push(_manager);
            emit ManagerAdded(_manager, name, active);
        }
```

The check is redundant because `bytes(managerNames[_manager]).length` will always be 0 as the mapping is never updated.

`InsufficientDebtBalance` error in `Errors.sol` can be removed as it is not used anywhere.

The `address private manager` in `Vault.sol` is not used anywhere and can be removed

Client

Acknowledged

[I-02] Missing event emission

An event emission is missing in `initialize()` which is located in `Vault.sol` as well as in the constructor of `VaultFactory.sol`. It's a best practice to emit events on every state changing method for off-chain monitoring.

Client

Acknowledged