Centrifuge Vaults
Security Review

Cantina Managed review by:

Gerard Persoon, Lead Security Researcher
0xDjango, Associate Security Researcher

February 22, 2025

Contents

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	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

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 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

Centrifuge empowers asset managers to tokenize, manage, and distribute their funds onchain, while providing investors access to a diversified group of tokenized assets.

From Feb 11th to Feb 16th the Cantina team conducted a review of vaults-internal on commit hash 70c00ee6. The team identified a total of **8** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	1	1	0
Gas Optimizations	1	1	0
Informational	6	5	1
Total	**8**	**7**	**1**

3 Findings

3.1 Low Risk

3.1.1 Checks in `_processRedeem()` can be circumvented and is missing in `claimCancelDepositRequest()`

Severity: Low Risk

Context: InvestmentManager.sol#L97-L109, InvestmentManager.sol#L524-L533, InvestmentManager.sol#L543-L554, RestrictedRedemptions.sol#L67-L71, RestrictionManager.sol#L64-L67

Description: In `_processRedeem()`, `receiver` can be arbitrarily selected via `redeem()` and `withdraw()`. `RestrictionManager` and `RestrictedRedemptions` both have the following restrictions on `from`:

```
if (uint128(hookData.from).getBit(FREEZE_BIT) == true && !root.endorsed(from)) {
    // Source is frozen and not endorsed
    return false;
}
```

The caller of `redeem()` and `withdraw()` can choose a `receiver` that doesn't have the `FREEZE_BIT` set, so the `_canTransfer()` check can be circumvented.

Note: this assumes `address(0)` doesn't have the `FREEZE_BIT` set, but if that would be the case `_processRedeem()` would never work.

Additionally `claimCancelDepositRequest()` also transfers assets but has no checks and `receiver` can also be arbitrarily selected.

Recommendation: Consider adding a `_canTransfer()` check to both `_processRedeem()` where `controller` is used as the `from`. Also add similar checks in `claimCancelDepositRequest()`.

Note: this requires adding `controller` as a parameter to `_processRedeem()`.

Note: this is similar to the check on `controller` in `InvestmentManager::requestRedeem()`.

A possible solution as suggested by the project is adding the following:

```
if (controller != receiver) {
    require(
        _canTransfer(vault, controller, receiver, convertToShares(vault, assetsDown)),
        // ...
    )
}
```

Centrifuge: Fixed in PR 96.

Cantina Managed: Fix verified.

3.2 Gas Optimization

3.2.1 Functions in ERC7540Vault can be made external

Severity: Gas Optimization

Context: ERC7540Vault.sol#L26, ERC7540Vault.sol#L341

Description: Several functions in `ERC7540Vault` as public while they are not called from the contract itself:

```
function requestDeposit(uint256 assets, address controller, address owner) public returns (uint256) {
function pendingDepositRequest(uint256, address controller) public view returns (uint256 pendingAssets) {
function requestRedeem(uint256 shares, address controller, address owner) public returns (uint256) {
function pendingRedeemRequest(uint256, address controller) public view returns (uint256 pendingShares) {
function pendingCancelDepositRequest(uint256, address controller) public view returns (bool isPending) {
function claimableCancelDepositRequest(uint256, address controller) public view returns (uint256
↪  claimableAssets) {
function pendingCancelRedeemRequest(uint256, address controller) public view returns (bool isPending) {
function claimableCancelRedeemRequest(uint256, address controller) public view returns (uint256
↪  claimableShares) {
function setOperator(address operator, bool approved) public virtual returns (bool success) {
function setEndorsedOperator(address owner, bool approved) public virtual {
function convertToShares(uint256 assets) public view returns (uint256 shares) {
function maxDeposit(address controller) public view returns (uint256 maxAssets) {
function maxMint(address controller) public view returns (uint256 maxShares) {
function mint(uint256 shares, address receiver) public returns (uint256 assets) {
function maxWithdraw(address controller) public view returns (uint256 maxAssets) {
function withdraw(uint256 assets, address receiver, address controller) public returns (uint256 shares) {
function onRedeemRequest(address controller, address owner, uint256 shares) public auth {
function onDepositClaimable(address controller, uint256 assets, uint256 shares) public auth {
function onRedeemClaimable(address controller, uint256 assets, uint256 shares) public auth {
function onCancelDepositClaimable(address controller, uint256 assets) public auth {
function onCancelRedeemClaimable(address controller, uint256 shares) public auth {
```

Note: the following functions are called from the contract, so should stay public:

```
function DOMAIN_SEPARATOR() public view returns (bytes32) {
function convertToAssets(uint256 shares) public view returns (uint256 assets) {
function deposit(uint256 assets, address receiver, address controller) public returns (uint256 shares) {
function mint(uint256 shares, address receiver, address controller) public returns (uint256 assets) {
function maxRedeem(address controller) public view returns (uint256 maxShares) {
```

Recommendation: Consider making these functions `external` to save some gas.

Centrifuge: Fixed in PR 96.

Cantina Managed: Fix verified.

3.3 Informational

3.3.1 Unused bits are erased in `updateMember()`

Severity: Informational

Context: RestrictedRedemptions.sol#L129-L138, RestrictionManager.sol#L125-L133

Description: The functions `updateMember()` in `RestrictedRedemptions` and `RestrictionManager` update the timestamp and the `FREEZE_BIT`. However they don't retreive and maintain the remaining 63 bits of the `hookData`, so the unused bits are erased. If more bits would be used in the future it might be useful to keep them.

Recommendation: Consider keeping the unused bits from `hookData`.

Centrifuge: This shouldn't be an issue, since the contract is non-upgradeable, any future implementation using more bits would use other logic.

Cantina Managed: Acknowledged.

3.3.2 Different order of parameters in events `Cancel...Claim`

Severity: Informational

Context: ERC7540Vault.sol#L179, ERC7540Vault.sol#L206

Description: The following events start with `receiver` and then `controller`. However most events start with `controller`, followed by `receiver`.

```
emit CancelDepositClaim(receiver, controller, REQUEST_ID, msg.sender, assets);
emit CancelRedeemClaim(receiver, controller, REQUEST_ID, msg.sender, shares);
```

Recommendation: Consider using the same order in all events.

Centrifuge: Fixed in PR 96.

Cantina Managed: Fix verified.

3.3.3 Similar checks done in different places

Severity: Informational

Context: ERC7540Vault.sol#L119-L132, InvestmentManager.sol#L97-L109

Description: Function `ERC7540Vault::requestRedeem()` calls `checkTransferRestriction()` and `InvestmentManager::requestRedeem()`. Function `InvestmentManager::requestRedeem()` calls `_canTransfer()`, which is similar to `checkTransferRestriction()`. So two similar checks are done in different places.

Recommendation: Consider putting the checks in the same location, which is easier to read and maintain.

Centrifuge: Fixed in PR 96.

Cantina Managed: Fix verified.

3.3.4 Comment in `_canTransfer()` can be improved to be easier to understand

Severity: Informational

Context: InvestmentManager.sol#L641-L646

Description: The following comment in `InvestmentManager::_canTransfer()` can be improved to make it more readable:

```
- Sender (from) and receiver (to) have both to pass
+ Sender (from) and receiver (to) have to both pass
```

Centrifuge: Fixed in PR 96.

Cantina Managed: Fix verified.

3.3.5 Comment can be updated to provide more detail

Severity: Informational

Context: ERC7540Vault.sol#L340-L341, ERC7540Vault.sol#L353-L356

Description: Comments in the `withdraw()` and `redeem()` functions state: `DOES NOT support controller != msg.sender`. However, this is not completely accurate. `msg.sender` can be different than the controller if `msg.sender` happens to be an `operator` of `controller`.

Recommendation: Consider updating the comment to make this clear.

Centrifuge: Fixed in PR 96.

Cantina Managed: Fix verified.

3.3.6 `maxRedeem()` and `maxWithdraw()` don't have `_canTransfer()` check

Severity: Informational

Context: InvestmentManager.sol#L267, InvestmentManager.sol#L281-L282, InvestmentManager.sol#L389-L390, InvestmentManager.sol#L400-L401, InvestmentManager.sol#L406, InvestmentManager.sol#L411, InvestmentManager.sol#L454, InvestmentManager.sol#L469, InvestmentManager.sol#L524-L535

Description: The functions `Redeem()` and `Withdraw()` call `_processRedeem()`, which now has a `_canTransfer()` check. The related functions `maxRedeem()` and `maxWithdraw()` don't have such a check. However the functions `maxMint()` and `maxDeposit()` , which are similar to `maxRedeem()` and `maxWithdraw()`, do have a `_canTransfer()` check.

Recommendation: Consider adding a `_canTransfer()` check to `maxRedeem()` and `maxWithdraw()`.

Centrifuge: Fixed in PR 96.

Cantina Managed: Fix verified.