Centrifuge CFG
Security Review

Cantina Managed review by:

Gerard Persoon, Lead Security Researcher

Cccz, Security Researcher
0xDjango, Associate Security Researcher

April 1, 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 *(formed by **Gerard Persoon** and **0xDjango**)* team conducted a review of `src/InflationMinter.sol` from inflation-minter on commit hash bb8739dc and `IouCfg.sol` from vaults-internal on commit hash 70c00ee6. The team identified a total of **10** 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	2	2	0
Informational	7	4	3
Total	**10**	**7**	**3**

In addition, from Mar 27th to Mar 28th the Cantina team *(formed by **Gerard Persoon** and **Cccz**)* conducted a review of `src/CFG.sol` and `src/DelegationToken.sol` from cfg-token on commit hash 0b51f63d. The team identified a total of **10** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	1	1	0
Low Risk	2	2	0
Gas Optimizations	2	2	0
Informational	5	3	2
Total	**10**	**8**	**2**

3 Findings *(february 2025)*

3.1 Low Risk

3.1.1 `transferFrom()` uses incorrect emit

Severity: Low Risk

Context: IouCfg.sol#L47-L59

Description: The `emit Transfer(receiver, address(0), amount)` in `transferFrom()` simulates burning `IOU` tokens from the `receiver`. However the "*I Owe You*" promise is settled from the `sender` perspective, so more logical to use the `sender`.

This can confuse the offchain logic.

Recommendation: Consider changing the code to:

```
- emit Transfer(receiver, address(0), amount);
+ emit Transfer(sender, address(0), amount);
```

Centrifuge: Fixed in PR 92.

Cantina Managed: Fix verified.

3.2 Gas Optimization

3.2.1 `InflationMinter` can be optimized

Severity: Gas Optimization

Context: InflationMinter.sol#L11

Description: Function `mint()` of `InflationMinter` will be used regularly (daily) on mainnet. It can be optimized to same gas costs.

Recommendation: Consider using one or more of the following optimizations:

- Replace `uint64` with `uint256`;
- Replace `_rpow()` with solady rpow , which will save ~73 gas and saves duplicating code;
- Perform the calculation in periods, which will save ~1100 gas. This can be done in the following way:

```
uint256 public immutable period = ...;
uint256 public lastMintingInPeriods = block.timestamp/period;
function mint() external {
    uint256 elapsedPeriods = block.timestamp/period;
    uint256 cached = lastMintingInPeriods;
    require(elapsedPeriods > cached, "InflationMinter/already-inflated");
    uint256 mintingsElapsed = elapsedPeriods - cached;
    lastMintingInPeriods = elapsedPeriods;
    // ... // use lastMintingInPeriods
}
```

Centrifuge: Fixed in PR 1. We will keep the current `rpow` implementation as the gas savings are minimal.

Cantina Managed: Fix verified.

3.2.2 Function `decimals()` can be optimized

Severity: Gas Optimization

Context: IouCfg.sol#L78-L80

Description: Function `decimals()` queries the `decimals()` from `legacyCfg` every time. This could be optimized.

Recommendation: Consider retrieving the value for `IERC20Metadata(legacyCfg).decimals()` in the `constructor` and store the value in an `immutable` variable.

Centrifuge: Fixed in PR 92.

Cantina Managed: Fix verified.

3.3 Informational

3.3.1 `IERC20` imported from `forge-std`

Severity: Informational

Context: InflationMinter.sol#L4

Description: `InflationMinter` uses an import from `forge-std`, which make the source depend on the used toolset (e.g. foundry).

Recommendation: If you don't want to rely on the toolset, consider importing `IERC20` from one of the following:

- OpenZeppelin's IERC20.sol.
- Centrifuge's `liquidity-pools`' IERC20.sol.

Centrifuge: Acknowledged.

Cantina Managed: Acknowledged.

3.3.2 Interface for `burn` isn't used

Severity: Informational

Context: InflationMinter.sol#L8

Description: The interface for function `burn()` isn't used in `InflationMinter` so it could be removed.

Recommendation: Consider removing the interface for `burn()`:

```
- function burn(address user, uint256 value) external;
```

Centrifuge: Fixed in PR 1.

Cantina Managed: Fix verified.

3.3.3 Constructor of `InflationMinter` doesn't have sanity checks

Severity: Informational

Context: InflationMinter.sol#L22-L31

Description: The constructor of `InflationMinter` doesn't have sanity checks. Its very unlikely that the contract will be deployed with incorrect parameters and if it is, it can be redeployed, so the risk is low.

Recommendation: If you do want to add sanity checks, the following can be checked:

- `target != address(0)`.
- `token != address(0)`.
- `decimals()` exists (isn't mandatory in ERC20).
- `period != 0` (that would result in a revert).
- `rate` is far smaller than `ONE` (a high inflation rate could result in overflows).

Centrifuge: Acknowledged.

Cantina Managed: Acknowledged.

3.3.4 The interface for function `burn()` is different than OZ version

Severity: Informational

Context: IouCfg.sol#L10

Description: The interface for function `burn()` in contract `IouCfg` is different than the standards from OpenZeppelin.

Recommendation: Consider using the standards from OZ ERC20Burnable:

- `function burn(uint256 value) external;`
- `function burnFrom(address account, uint256 value) external;`

Note: `burnForm()` requires an `allowance` and `burn()` doesn't.

Note: this assumes the contract for `newCfg` still has to be designed/isn't final yet.

Centrifuge: Fixed in PR 92.

Cantina Managed: Fix verified.

3.3.5 `IouCfg` and `InflationMinter` use different names for same interface

Severity: Informational

Context: InflationMinter.sol#L6-L9, IouCfg.sol#L7-L11

Description: The contracts `IouCfg` and `InflationMinter` use different names for the same interface:

- `IouCfg` uses `InflationMinter`.
- `InflationMinter` uses `IERC20Mintable`.

Recommendation: Consider using the same names in both contracts.

Centrifuge: Acknowledged.

Cantina Managed: Acknowledged.

3.3.6 `Auth` and `auth` could be confused

Severity: Informational

Context: Auth.sol#L13, IouCfg.sol#L18

Description: The `constructor` in contract `IouCfg` uses: `Auth(auth)`. It is not immediately clear what the difference between `Auth` and `auth` is.

Recommendation: Consider using `initialWard` instead of `auth`:

```
- constructor(address auth, address escrow_, address newCfg_, address legacyCfg_) Auth(auth) {
+ constructor(address initialWard , address escrow_, address newCfg_, address legacyCfg_) Auth(initialWard) {
```

Centrifuge: Fixed in PR 92.

Cantina Managed: Fix verified.

3.3.7 Comment in `IouCfg::transferFrom()` can be improved to be easier to understand

Severity: Informational

Context: IouCfg.sol#L47-L49

Description: A comment in `IouCfg::transferFrom()` can be improved to make it more readable:

```
- "Ensures that only for Centrifuge to Eth tokens are minted"
+ "Ensures that only for the direction "Centrifuge to Eth", tokens are minted"
```

Centrifuge: Fixed in PR 92.

Cantina Managed: Fix verified.

4 Findings *(march 2025)*

4.1 Medium Risk

4.1.1 Function `CFG::burn()` doesn't call `_moveDelegateVotes()`

Severity: Medium Risk

Context: CFG.sol#L15-L26, DelegationToken.sol#L78-L81

Description: `CFG::burn()` doesn't call `_moveDelegateVotes()` unlike the similar functions like `DelegationToken::burn()`. This could leave token delegations, while the underlying tokens are burnt and could lead to incorrect results in voting. This function will mostly be used by tokens bridges to the impact seems low.

Recommendation: Consider adding a call to `_moveDelegateVotes()`.

Centrifuge: Fixed in PR 3.

Cantina Managed: Fix verified.

4.2 Low Risk

4.2.1 Invalid signatures not detected in `delegateWithSig()`

Severity: Low Risk

Context: DelegationToken.sol#L38-L48

Description: Invalid signatures not detected in `delegateWithSig()`. In that situation `delegator` will be `address(0)`. The risk is limited because `address(0)` won't contain tokens. However, the following unwanted effects are present:

- Setting a delegate for `address(0)`, that normally shouldn't have a delegate.
- Emitting `DelegateeChanged` for `address(0)`.

Also future changes/bugs could potentially allow tokens to exist on `address(0)`.

Recommendation: Consider reverting when `delegator == address(0)`. Alternatively use a variation of `isValidSignature()`, which is also used used in `ERC20:Permit()` and already contains this check.

Centrifuge: Fixed in PR 3.

Cantina Managed: Fix verified.

4.2.2 `msg.sender` of `CFG.sol` might hold unwanted authorizations

Severity: Low Risk

Context: CFG.s.sol#L28, CFG.sol#L8-L12

Description: The `msg.sender` of `CFG.sol` is authorized via auth to be able to do `file()`. Then `ward` is also authorized. Assuming `ward` is not equal to `msg.sender`, `msg.sender` stays authorized and can still to `mint()`.

Recommendation: Make sure only one address is authorized after the contract is deployed. This can be done in several ways:

- Add something like: `if (ward != msg.sender) deny(msg.sender);` in the constructor of `CFG.sol`;
- Change the code so only `ward` is authorized;
- Uncomment the `deny` statement in the deployment script.

Centrifuge: Fixed in the deploy script (see PR 3).

Cantina Managed: Fix verified.

4.3 Gas Optimization

4.3.1 Use of `balanceOf()` in `burn()` not optimal

Severity: Gas Optimization

Context: CFG.sol#L15-L26

Description: Function `burn()` uses both `balanceOf()` and `_balanceOf()`. The second call could be replaced with `balance`, which cantains the value of the first version.

Recommendation: Consider changing the code to:

```
  function burn(uint256 value) external {
      uint256 balance = balanceOf(msg.sender);
      require(balance >= value, InsufficientBalance());
      // ...
-     _setBalance(msg.sender, _balanceOf(msg.sender) - value);
+     _setBalance(msg.sender, balance - value);
      // ...
  }
```

Note: these changes can also be done in `ERC20.sol` for functions `transfer()`, `_transferFrom()` and `burn()`.

Centrifuge: Fixed in PR 3.

Cantina Managed: Fix verified.

4.3.2 Optimization possible for calculation of `totalSupply`

Severity: Gas Optimization

Context: CFG.sol#L22

Description: A small optimization can be done in the calculation of `totalSupply`.

Note: similar optimizations can also be done in `ERC20.sol` for `mint()` and `burn()`.

Recommendation:

```
- totalSupply = totalSupply - value;
+ totalSupply -= value;
```

Centrifuge: Fixed in PR 3.

Cantina Managed: Fix verified.

4.4 Informational

4.4.1 `transfer()` and `transferFrom()` don't check result of their `super` functions

Severity: Informational

Context: DelegationToken.sol#L60-L69

Description: The functions `transfer()` and `transferFrom()` don't explicitly check the result of their `super` functions. The risk is limited because the underlying implementation is known and allways returns `true`. However it is safer not to rely on this implicit information.

Recommendation: Consider check the return values of the `super` functions, or at least add a comment.

Centrifuge: Acknowledged.

Cantina Managed: Acknowledged.

4.4.2 URL in comment has a newer version

Severity: Informational

Context: DelegationToken.sol#L17

Description: The URL `https://github.com/morpho-org/morpho-token-upgradeable` resolves to `https://github.com/morpho-org/morpho-token`. So the resulting url could also be references in `DelegationToken.sol`.

Recommendation: Consider changing the comment in the following way:

```
- /// @author Modified from https://github.com/morpho-org/morpho-token-upgradeable
+ /// @author Modified from https://github.com/morpho-org/morpho-token
```

Centrifuge: Fixed in PR 3.

Cantina Managed: Fix verified.

4.4.3 The repository `protocol-v3` might not be accessible

Severity: Informational

Context: DelegationToken.sol#L4

Description: The `protocol-v3` repository might not be accessible by everyone, which makes reviewing the code more difficult.

Recommendation: Consider adding the underlying code to the `cfg-token` repository.

Centrifuge: Fixed in PR 1.

Cantina Managed: Fix verified.

4.4.4 `delegateWithSig()` doesn't support signatures from smart contract accounts

Severity: Informational

Context: DelegationToken.sol#L38-L48

Description: `ERC20::permit()` uses `SignatureLib.isValidSignature()`, which also allow smart contract accounts to sign. However `delegateWithSig()` doesn't support this. The result is that smart contract accounts can't used. This is increasingly relevant with `ERC 4337/EIP 7702`.

Recommendation: Consider using a variation of `SignatureLib.isValidSignature()` to also allow smart contract accounts.

Note: `SignatureLib.isValidSignature()` currently requires `signer` as an input parameter, which isn't passed to `delegateWithSig()`.

Centrifuge: Acknowledged.

Cantina Managed: Acknowledged.

4.4.5 No interface file for `CFG.sol` exists

Severity: Informational

Context: CFG.sol#L7

Description: There is no interface file for `CFG.sol`. This makes using the token more complicated.

Recommendation: Consider creating an interface file `ICFG.sol`. Also inherit from it to make sure the interface is correct.

Centrifuge: Fixed in PR 3.

Cantina Managed: Fix verified.