Permissioned ERC20 Wrapper

Security Review

Cantina Managed review by:
0xRajeev, Lead Security Researcher
Cccz, Security Researcher

June 16, 2024

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 is the infrastructure that facilitates the decentralized financing of real-world assets natively on-chain, creating a fully transparent market which allows borrowers and lenders to transact without unnecessary intermediaries. The protocol aims to lower the cost of borrowing for businesses around the world, while providing DeFi users with a stable source of collateralized yield that is uncorrelated to the volatile crypto markets.

On Jun 12th the Cantina team conducted a review of PermissionedUSDCWrapper.sol on commit hash e0a9287e. The team identified a total of **3** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 0
- Gas Optimizations: 0
- Informational: 3

3 Findings

3.1 Informational

3.1.1 Missing event emit for privileged operation

Severity: Informational

Context: PermissionedERC20Wrapper.sol#L64-L70

Description: Function `PermissionedERC20Wrapper.file(bytes32 what, address data)` allows authorized addresses to change EAS indexer/service and `Memberlist` management contracts. However, it lacks an event emit for `emit File(what, data);` for this privileged operation.

Recommendation: Consider adding `emit File(what, data);` in `file(bytes32 what, address data)`.

3.1.2 `recover()` inconsistently allows recovering tokens to the contract address

Severity: Informational

Context: PermissionedERC20Wrapper.sol#L95-L97

Description: `ERC20Wrapper.depositFor()` does not allow minting tokens to `address(this)`. However, this check is not applied in `recover()`.

```
function depositFor(address account, uint256 value) public virtual returns (bool) {
    address sender = _msgSender();
    if (sender == address(this)) {
        revert ERC20InvalidSender(address(this));
    }
    if (account == address(this)) {
        revert ERC20InvalidReceiver(account);
    }
    SafeERC20.safeTransferFrom(_underlying, sender, address(this), value);
    _mint(account, value);
    return true;
}
```

Recommendation: Consider checking that `account` is not `address(this)` in `recover()`:

```
  function recover(address account) public auth returns (uint256) {
+     if (account == address(this)) {
+         revert ERC20InvalidReceiver(account);
+     }
      return _recover(account);
  }
```

3.1.3 `IERC4626` interface is incorrectly used instead of `IERC7575`

Severity: Informational

Context: VaultOracle.sol#L42, VaultOracle.sol#L8-L12

Description: In `VaultOracle`, `vault` is treated as an ERC4626 instance. However, the declared `IERC4626` does not match ERC4626 specification e.g. `function share()`. Team clarified that `vault` should be an ERC7575 instance.

Recommendation: Consider changing `IERC4626` interface to `IERC7575`.