Byzantine
Security Review

Cantina Managed review by:

Chris Smith, Lead Security Researcher
Sujith Somraaj, Security Researcher

September 10, 2025

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**

3 Findings **4**
- 3.1 Medium Risk . . . 4
  - 3.1.1 Anyone can overwrite factory mapping . . . 4
  - 3.1.2 Inconsistent skim protection between adapters . . . 4
  - 3.1.3 Missing slippage protection in claim functions . . . 5
- 3.2 Low Risk . . . 5
  - 3.2.1 Adapters should protect from resetting approval to parent vault . . . 5
- 3.3 Informational . . . 5
  - 3.3.1 Inefficient data encoding in `ERC4626MerklAdapter.claim()` . . . 5
  - 3.3.2 Increase test coverage . . . 6
  - 3.3.3 Dust accumulation due to time delay between swap data generation and execution . . . 6

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 level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

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 are 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

Byzantine is a trustless and efficient restaking layer with permissionless strategy creation.

From Aug 31st to Sep 2nd the Cantina team conducted a review of debt-fund-vault-v2 on commit hash a0916794. The team identified a total of **7** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	3	3	0
Low Risk	1	1	0
Gas Optimizations	0	0	0
Informational	3	3	0
Total	**7**	**7**	**0**

3 Findings

3.1 Medium Risk

3.1.1 Anyone can overwrite factory mapping

Severity: Medium Risk

Context: CompoundV3AdapterFactory.sol#L23

Description: The `CompoundV3AdapterFactory` is responsible for deploying multiple instances of CompoundV3Adapters, each configured with different parentVault and comet address parameters. After deployment, the address of each new adapter is stored in the `compoundV3Adapter[parentVault][comet]` mapping. However, this mapping only considers `parentVault` and `comet` as keys, and does not include the `cometRewards` parameter.

This creates a critical vulnerability where:

- Any user can deploy a new adapter with the same parentVault and comet, but different cometRewards.
- The new deployment will overwrite the existing mapping entry.
- Downstream systems relying on this mapping will be affected by the malicious overwrite.

Likelihood explanation: The issue has a HIGH likelihood since it can be triggered by a random user simply by paying minimal gas costs.

Impact explanation: The contract is not directly used by other contracts within the scope of the audit. Therefore, the assumption is that any external integrators relying on the mapping could be affected, which in some cases might lead to fund loss. Due to the uncertain nature of this impact, it is assessed as LOW.

Recommendation: Consider adding validation to prevent overwrites or including cometRewards in the mapping key structure:

```
// Option 1: Prevent overwrites
require(compoundV3Adapter[parentVault][comet] == address(0), "Adapter already exists");

// Option 2: Use three-dimensional mapping
mapping(address => mapping(address => mapping(address => address))) public compoundV3Adapter;
```

Byzantine Finance: Fixed in commit ff52cb66.

Cantina Managed: Fix verified.

3.1.2 Inconsistent skim protection between adapters

Severity: Medium Risk

Context: CompoundV3Adapter.sol#L55-L60

Description: The CompoundV3Adapter lacks essential protection in its `skim()` function that prevents the withdrawal of underlying protocol tokens, while the `ERC4626MerklAdapter` correctly implements this safeguard. This inconsistency creates a significant risk where the skim recipient could maliciously or accidentally withdraw the adapter's Compound V3 shares (cTokens), leading to vault insolvency. In the `CompoundV3Adapter.skim()` function, there is no validation to prevent skimming of the underlying Compound V3 tokens (cTokens) that represent the adapter's allocated assets:

```
// CompoundV3Adapter.sol - VULNERABLE
function skim(address token) external {
    if (msg.sender != skimRecipient) revert NotAuthorized();
    uint256 balance = IERC20(token).balanceOf(address(this));
    SafeERC20Lib.safeTransfer(token, skimRecipient, balance);
    emit Skim(token, balance);
}
```

In contrast, the `ERC4626MerklAdapter.skim()` correctly implements protection:

```
// ERC4626MerklAdapter.sol - SECURE
function skim(address token) external {
    require(msg.sender == skimRecipient, NotAuthorized());
    require(token != erc4626Vault, CannotSkimERC4626Shares()); //  Protection
```

```
    uint256 balance = IERC20(token).balanceOf(address(this));
    SafeERC20Lib.safeTransfer(token, skimRecipient, balance);
    emit Skim(token, balance);
}
```

Likelihood explanation: The issue has a LOW likelihood since it is a protected role and the odds of this happening has slightly lower chance.

Impact explanation: Since the issue can make the parent vault insolvent, the impact from this issue is HIGH.

Recommendation: Add the exact protection mechanism used in `ERC4626MerklAdapter` to prevent skimming of the underlying Compound V3 tokens.

Byzantine Finance: Fixed in commit 40666029.

Cantina Managed: Fix verified.

3.1.3 Missing slippage protection in claim functions

Severity: Medium Risk

Context: CompoundV3Adapter.sol#L94

Description: Both `CompoundV3Adapter` and `ERC4626MerklAdapter` lack slippage protection in their reward claiming and swapping mechanisms. The claim functions execute token swaps without minimum output validation, exposing users to potential MEV attacks, sandwich attacks, and unfavorable swap rates that could result in significant value loss during reward liquidation.

The function only checks if the receiver received at least 1 wei amount of tokens (which is highly dangerous in some instances).

Likelihood explanation: By using Flashbot RPC / other private mempool solution for `claim()`, this issue can be avoided to a certain degree. Provided that the problem is given a MEDIUM likelihood.

Impact explanation: Since the reward value can be significantly reduced and can be severe depending on the claim value, the issue has MEDIUM to HIGH impact.

Recommendation: Consider implementing comprehensive slippage protection by adding minimum output validation to ensure accurate results.

Byznatine: Fixed in commits f7c3cf52 and 31049706.

Cantina Managed: Fix verified.

3.2 Low Risk

3.2.1 Adapters should protect from resetting approval to parent vault

Severity: Low Risk

Context: CompoundV3Adapter.sol#L101, ERC4626MerklAdapter.sol#L125

Description: The adapters prevent the `claim` function from interacting with the underlying vaults which is critical; however, it would also be good to ensure that the `claim` functions do not call the `parentVault` as well. If they could successfully do this it would reset the approval between the adapter and the `parentVault`.

Recommendation: Add a require check that the swapper is not the `parentVault`. For extra safety it might also make sense to block calling the reward contracts in these functions.

Byzantine Finance: Fixed in commit a52ce795.

Cantina Managed: Fix verified.

3.3 Informational

3.3.1 Inefficient data encoding in `ERC4626MerklAdapter.claim()`

Severity: Informational

Context: ERC4626MerklAdapter.sol#L106, ERC4626MerklAdapter.sol#L109-L112

Description: The `claim()` function in `ERC4626MerklAdapter` uses an unnecessary intermediate data structure (`ClaimParams`) that wraps `MerklParams` and `SwapParams[]`, resulting in additional encoding/decoding overhead and increased gas consumption. The function could be optimized by directly encoding the required parameters without the wrapper struct.

Recommendation: Consider replacing the wrapper struct with direct parameter encoding:

```
function claim(bytes calldata data) external {
    require(msg.sender == claimer, NotAuthorized());

    // Direct decoding - OPTIMIZED
    (
        MerklParams memory merklParams,
        SwapParams[] memory swapParams
    ) = abi.decode(data, (MerklParams, SwapParams[]));

    // Claim data checks
    require(swapParams.length == merklParams.tokens.length, InvalidData());

    // ... rest of function remains the same
}
```

Byzantine Finance: Fixed in commit 79364af7.

Cantina Managed: Fix verified.

3.3.2 Increase test coverage

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: There is less than complete test coverage of two key contracts under review (`CompoundV3Adapter` & `ERC4626MerklAdapter`). Adequate test coverage and regular reporting are essential processes in ensuring the codebase works as intended. Insufficient code coverage may lead to unexpected issues and regressions arising due to changes in the underlying smart contract implementation.

File	Line Coverage			Branch Coverage			Function Coverage		
	Rate	Total	Hit	Rate	Total	Hit	Rate	Total	Hit
CompoundV3Adapter.sol	84.6 %	65	55	26.7 %	15	4	70.0 %	10	7
CompoundV3AdapterFactory.sol	100.0 %	6	6	-			100.0 %	1	1
ERC4626MerklAdapter.sol	84.6 %	65	55	41.4 %	29	12	80.0 %	10	8

Recommendation: Add to test coverage ,ensuring all execution paths are covered.

Byzantine Finance: Fixed in commit a115dd32.

Cantina Managed: Fix verified.

3.3.3 Dust accumulation due to time delay between swap data generation and execution

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: During the review, the project team brought up this issue.

On `CompoundV3Adapter`, the reward tokens are supposed to accrue every second. As a result, some tokens might accumulate between the generation of the claim and swap data (off-chain) and the transaction being executed on-chain. These dust tokens are locked in the adapter contractor, unless the off-chain mechanism is accurate.

To fix this issue, the adapter now approves the swapper for the entire balance of the adapter, rather than the claimed amount.

Byzantine Finance: This specific issue is fixed in two commits: 14671d87 and 50509b65.

Cantina Managed: Verified fixes. This change allows the swapper to handle donated tokens rather than having to call `skim()`.