CANTINA

Morpho Bundler3

Security Review

Cantina Managed review by:

MiloTruck, Lead Security Researcher
Om Parikh, Security Researcher

February 17, 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

Morpho is a trustless and efficient lending primitive with permissionless market creation.

From Feb 7th to Feb 8th the Cantina team conducted a review of bundler3 on commit hash 99e4a495. The team identified **1** issue:

Issues Found

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

3 Findings

3.1 Medium Risk

3.1.1 Non-whitelisted users can mint vault shares with permissioned tokens through the bundler

Severity: Medium Risk

Context: GeneralAdapter1.sol#L57-L68, GeneralAdapter1.sol#L224-L246

Description: In PR 233, `GeneralAdapter1.erc20WrapperDepositFor()` was modified such that wrapped tokens are minted to the `initiator()` instead of any `receiver` address:

```
function erc20WrapperDepositFor(address wrapper, uint256 amount) external onlyBundler3 {
    IERC20 underlying = ERC20Wrapper(wrapper).underlying();
    if (amount == type(uint256).max) amount = underlying.balanceOf(address(this));

    require(amount != 0, ErrorsLib.ZeroAmount());

    SafeERC20.forceApprove(underlying, wrapper, type(uint256).max);

    require(ERC20Wrapper(wrapper).depositFor(initiator(), amount), ErrorsLib.DepositFailed());

    SafeERC20.forceApprove(underlying, wrapper, 0);
}
```

This change was made to prevent users who are not whitelisted by a permissioned token from minting vault shares through the bundler. Using verUSDC as an example, a non-whitelisted user could previously do the following:

- Send USDC to `GeneralAdapter1`.
- Call `erc20WrapperDepositFor()` to wrap USDC → verUSDC, with `receiver` as `GeneralAdapter1`.
- Call `morphoSupply()` to deposit verUSDC, with `onBehalf` as himself.

However, by directly minting permissioned tokens to the adapter, it is still possible for a non-whitelisted user to mint vault shares. More specifically, a non-whitelisted user can:

- Directly call `verUSDC.depositFor()` with `account` as `GeneralAdapter1`, which mints verUSDC to `GeneralAdapter1`.
- Call `morphoSupply()` to deposit verUSDC, with `onBehalf` as himself.

This attack is possible as `depositFor()` only checks if `account` is whitelisted; there is no restriction on the caller. As a result, attackers will still be able to use the bundler to bypass any form of whitelist in permissioned tokens.

Recommendation: Consider moving the `erc20WrapperDepositFor()` and `erc20WrapperWithdrawTo()` functions to a standalone adapter only meant for `ERC20Permissioned` tokens.

To block non-whitelisted users from minting vault shares, the adapter should check that the initiator is whitelisted in `morphoSupply()` and all other deposit functions (e.g. `erc4626Mint()`).

To block non-whitelisted users from withdrawing vault shares (e.g. Morpho shares → verUSDC → USDC), check that the initiator is whitelisted in `erc20WrapperWithdrawTo()`. This prevents users from withdrawing permissioned tokens to the adapter and using the adapter to unwrap them.

Morpho: Fixed in commit 263f75d by:

- Restricting `withdrawTo()` so that the initiator must be whitelisted. With this change, users can bypass the `depositFor()` restriction by depositing directly to the adapter, but they cannot later unwrap their tokens.
- Move wrap and unwrap actions from `GeneralAdapter1` to `ERC20WrapperAdapter`, because we want to develop a better solution for permissioned tokens later and do not want to redeploy `GeneralAdapter1`.
- Add a morpho-specific `morphoWrapperDepositFor()` function to `GeneralAdapter1`.

Cantina Managed: Verified, the issue has been fixed by creating a standalone adapter. Although non-whitelisted users are still able to mint vault shares with permissioned tokens, they will not be able to unwrap the tokens after withdrawing shares.

Note that a non-whitelisted user is still able to wrap tokens and send them to whitelisted addresses through `ERC20Wrapper.depositFor()` → `CoreAdapter.erc20Transfer()`.