CANTINA

Coinbase: Morpho Loan Servicing Fee Policy Round 1

Security Review

Cantina Managed review by:
Sujith Somraaj, Lead Security Researcher
Rvierdiiev, Security Researcher

June 27, 2026

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**
- 2.1 Scope . . . 3

3 Findings **4**
- 3.1 Low Risk . . . 4
  - 3.1.1 Uninstall + reinstall resets the billing anchor, letting a user skip accrued fees . . . 4
- 3.2 Gas Optimization . . . 4
  - 3.2.1 Remove unnecessary market check inside `_onSingleExecutorExecute()` function . 4
- 3.3 Informational . . . 4
  - 3.3.1 Delete unused mapping entries for replaced policies . . . 4
  - 3.3.2 `_requireMarketParams` rejects valid interest-free Morpho markets . . . 5
  - 3.3.3 Borrower can defer fee collection by holding LTV at or above the collection cap . . . 5
  - 3.3.4 Bypass of fee payment using config parameters . . . 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

Base is a secure, low-cost, builder-friendly Ethereum L2 built to bring the next billion users onchain.

From Jun 23rd to Jun 24th the Cantina team conducted a review of morpho-loan-servicing-fee-policy on commit hash 4ee44966. The team identified a total of **6** 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	0	1
Gas Optimizations	1	0	1
Informational	4	2	2
Total	**6**	**2**	**4**

2.1 Scope

The security review had the following components in scope for morpho-loan-servicing-fee-policy on commit hash 4ee44966:

```
src/policies/MorphoLoanServicingFeePolicy.sol
```

3 Findings

3.1 Low Risk

3.1.1 Uninstall + reinstall resets the billing anchor, letting a user skip accrued fees

Severity: Low Risk

Context: MorphoLoanServicingFeePolicy.sol#L336-L343

Description: Fee progress is tracked by `lastCollectedEndBlockByPolicyId[policyId]`. The contract carefully preserves this anchor across a `PolicyManager.replace` (`_onUninstallForReplace` retains it, `_onInstallForReplace` inherits it), but the equivalent plain uninstall + fresh install does not: the account owner can uninstall permissionlessly (account sovereignty), `_onSingleExecutorUninstall()` deletes the anchor, and a reinstall re-seeds it to `block.number - 1`.

Every block between the last `FeeCollected` and the uninstall is orphaned, and the contiguity rule then bars the reinstalled policy from ever reaching them. So a user skips fees simply by uninstalling before collection (or staying uninstalled and reinstalling on demand). The carry-forward doesn't help because a fresh install mints a new policyId with no link to the old retained slot.

Recommendation: Because the uninstall is unpreventable on-chain and the fee is computed off-chain, handle this operationally rather than in the contract: detect the pattern off-chain - Coinbase has full visibility through the `PolicyInstalled` / `FeeCollected` / `PolicyManager` uninstall events and compute the unbilled span for any account that uninstalls with fees outstanding, and reconcile it by folding the arrears into the next collection's `feeAmount` on reinstall and/or gating re-onboarding. (and continued integration access) on settling the balance.

Alternatively, if on-chain state is required, consider retaining the values after uninstall and re-use them while installing again.

Coinbase: Intended behavior, good note for threat models.

Cantina Managed: Acknowledged.

3.2 Gas Optimization

3.2.1 Remove unnecessary market check inside `_onSingleExecutorExecute()` function

Severity: Gas Optimization

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

Description: The `_onSingleExecutorExecute()` function calls `_requireMarketParams()` for the policy's market. This function verifies that the market is a valid Morpho market. However, this check is already performed during policy installation, and there is no way for a Morpho market to become suspended or removed afterward. As a result, the check in `_onSingleExecutorExecute()` is redundant and can be removed.

Recommendation: Remove the `_requireMarketParams()` check from the `_onSingleExecutorExecute()` function.

Coinbase: Checks are useful especially if Morpho or a forked-protocol creates new market that is mutable. So leaving it as such.

Cantina Managed: Acknowledged.

3.3 Informational

3.3.1 Delete unused mapping entries for replaced policies

Severity: Informational

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

Description: Delete `marketKeyByPolicyId[otherPolicyId]` and `lastCollectedEndBlockByPolicyId[otherPolicyId]` variables associated with the replaced policy to keep the state clean, as they will not be used anymore after the replacement.

Recommendation:

```
   function _onInstallForReplace(
  //...
   ) internal virtual override {
       _onInstall(policyId, account, policyConfig);
       if (otherPolicy == address(this)) {
           bytes32 oldMarketKey = marketKeyByPolicyId[otherPolicyId];
           bytes32 newMarketKey = marketKeyByPolicyId[policyId];
           if (oldMarketKey != bytes32(0) && oldMarketKey == newMarketKey) {
               lastCollectedEndBlockByPolicyId[policyId] =
                ↪  lastCollectedEndBlockByPolicyId[otherPolicyId];
           }
       }
+      delete marketKeyByPolicyId[otherPolicyId];
+      delete lastCollectedEndBlockByPolicyId[otherPolicyId];
   }
```

Coinbase: Acknowledged, good practice to retain information in case of v2 or future reference.

Cantina Managed: Acknowledged.

3.3.2 `_requireMarketParams` rejects valid interest-free Morpho markets

Severity: Informational

Context: MorphoLoanServicingFeePolicy.sol#L549

Description: Morpho Blue supports interest-free markets, for which `marketParams.irm` is legitimately `address(0)`. `createMarket()` accepts any enabled IRM (`address(0)` is enabled currently), and `_accrueInterest()` explicitly skips accrual via `if (marketParams.irm != address(0))`. Such a market is fully valid and createable.

However, `_requireMarketParams()` folds a zero `irm` into its ”market does not exist” check and reverts with `MarketNotFound`. The rejection is also unnecessary: the policy has no functional dependency on irm being non-zero.

Recommendation: Drop the `marketParams.irm == address(0)` clause from the rejection condition.

Coinbase: Fixed in commit 15fff454.

Cantina Managed: Fix verified.

3.3.3 Borrower can defer fee collection by holding LTV at or above the collection cap

Severity: Informational

Context: MorphoLoanServicingFeePolicy.sol#L110

Description: Fee collection borrows `feeAmount` against the borrower's Morpho position, raising the position's LTV, and `_onPostExecute()` reverts when the resulting LTV exceeds the per-install `maxLtvAfterCollection`.

A borrower whose LTV already sits at or above the cap blocks every collection: any non-zero fee borrow pushes the post-collection LTV past the cap and reverts the entire call.

Since `maxLtvAfterCollection` is configured below the market `lltv`, the borrower can occupy this range without being liquidatable and does not need to approach liquidation to do so.

The deferral can be sustained two ways:

- Passively - keep standing LTV at or above the cap.
- Actively - raise LTV above the cap immediately before a collection (extra borrow or collateral withdrawal) and reverse it after the collection reverts. The temporary increase only needs to reach the cap, which is below `lltv`, so the position is not exposed to liquidation during the window.

This defers but does not waive the fee. Only the in-flight action reverts, `lastCollectedEndBlock` is not advanced, `feeAmount` is supplied off-chain per execution, and the unbilled block span stays collectable once the borrower deleverages.

Recommendation: No code changes are required. Monitor the `MAX_LTV_AFTER_COLLECTION_RATIO_WAD` post deployment and flag rogue configurations.

Tighten the off-chain fee collection mechanism to:

- Collect on an unpredictable, privately submitted schedule, which removes the window the active front-run-and-repay variant relies on.
- Retry failed collections, so a passive deferrer must hold elevated LTV continuously rather than briefly.
- Accrue the deferred span off-chain so the deferral carries a cost.
- Re-collect opportunistically when the borrower deleverages; the borrow-or-withdraw-then-reverse pattern around a reverted collection is observable on-chain and can be flagged.

Coinbase: Intended behavior, good note for threat models.

Cantina Managed: Acknowledged.

3.3.4 Bypass of fee payment using config parameters

Severity: Informational

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

Description: `feeRecipient` and `maxLtvAfterCollection` values are provided by the user during policy installation. This allows them to bypass fee collection by setting an own fee recipient or lowering `maxLtvAfterCollection` to a value that prevents fees from being collected (like 1%).

Recommendation: Consider restricting fee recipients to a predefined whitelist. The policy should reference a whitelist contract and validate that the provided fee recipient is included in the allowed list.

Additionally, avoid allowing users to provide an arbitrary `maxLtvAfterCollection` value. Instead, either enforce a safe minimum LTV threshold, such as 60%, for example, or remove the `maxLtvAfterCollection` parameter entirely and calculate the maximum allowed LTV from the market configuration:

```
maxAllowedLtv = Math.mulDiv(marketParams.lltv, MAX_LTV_AFTER_COLLECTION_RATIO_WAD, 1e18);
```

This ensures the value is derived from the market LTV with a safe buffer and cannot be manipulated by users to avoid fees.

Coinbase: Fixed in commit 2930ead.

Cantina Managed: The team addressed the `maxLtvAfterCollection` issue. However, the `feeRecipient` validation suggestion is not implemented, and is accepted as acknowledged.