Security Review

Cantina Managed

COINBASE: FIXED-RATE-LOAN-PROTECTION-POLICY

Cantina Managed review by:
0x52, Lead Security Researcher
R0bert, Lead Security Researcher
Jay, Security Researcher

CANTINA

September 17, 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 Policy can activate after its maturity cutoff . . . 4
  - 3.1.2 One-shot protection can be consumed without improving position health . . . 4
  - 3.1.3 Take callbacks can force premature top ups . . . 5
  - 3.1.4 Top ups disable unrelated Midnight buy offers . . . 6
- 3.2 Informational . . . 6
  - 3.2.1 Approval failures can go undetected when an existing allowance permits the top-up 6
  - 3.2.2 Policy does not enforce Midnight per borrower active collateral limit . . . 7
  - 3.2.3 Top up activates unused collateral and breaks single collateral rolling . . . 7
  - 3.2.4 Misleading NatSpec on `marketKeyByPolicyId` . . . 8

1 Introduction

1.1 About Cantina

Cantina is a security platform that pairs a world-class network of security researchers with agentic AI to help teams building onchain find and fix real risk. Through managed reviews, competitions, and bounties, Cantina secures protocols before vulnerabilities can be exploited.
Learn more at cantina.security

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 Sep 7th to Sep 8th the Cantina team conducted a review of fixed-rate-loan-protection-policy on commit hash b2db912b. The team identified a total of **8** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	4	4	0
Gas Optimizations	0	0	0
Informational	4	4	0
Total	**8**	**8**	**0**

2.1 Scope

The security review had the following components in scope for fixed-rate-loan-protection-policy on commit hash b2db912b:

```
src/policies/MorphoFixedRateLoanProtectionPolicy.sol
```

3 Findings

3.1 Low Risk

3.1.1 Policy can activate after its maturity cutoff

Severity: Low Risk.

Context: MorphoFixedRateLoanProtectionPolicy.sol#L249-L252, MorphoFixedRateLoanProtectionPolicy.sol#L320-L323.

Description: The policy supports bindings that are installed before the manager's `validAfter` timestamp. Such a binding should become executable while collateral can still protect the Midnight position.

During installation, `_onSingleExecutorInstall()` passes the current `block.timestamp` to `_requireBeforeMaturityBuffer()` but does not inspect the `validAfter` value already stored by `PolicyManager`. The manager deliberately accepts early installation and enforces `validAfter` only when execution is attempted. An account or signing integration can therefore set `validAfter` to `market.maturity - maturityBuffer` or any later timestamp while installing the binding earlier outside the buffer.

The installation succeeds in this configuration. Once `validAfter` is reached, the manager considers the policy active but `_onSingleExecutorExecute()` rejects every call with `WithinMaturityBuffer`. At maturity and afterward it rejects every call with `MarketMatured`. The account can consequently rely on an apparently installed protection policy that never has a valid execution timestamp, which can delay detection until the position needs a top up.

Recommendation: Validate the binding's effective execution start during installation. Read the stored `validAfter` from `PolicyManager` and apply the maturity and buffer checks to the later of `block.timestamp` and `validAfter`. Reject the installation when that timestamp is at or beyond the configured cutoff.

Coinbase: Fixed in commit a76e1f19.

Cantina: Fix verified. `_onSingleExecutorInstall()` now reads the binding's stored `validAfter` from `PolicyManager` and applies the maturity and buffer checks to the later of `block.timestamp` and `validAfter`. Installation rejects bindings whose earliest possible execution is at or beyond the configured cutoff. The same validation runs during replacement while execution continues to check the current block timestamp. This prevents a binding from being installed or replaced with an execution window that starts only after collateral top ups are refused.

3.1.2 One-shot protection can be consumed without improving position health

Severity: Low Risk.

Context: MorphoFixedRateLoanProtectionPolicy.sol#L313-L315, MorphoFixedRateLoanProtectionPolicy.sol#L325-L346, MorphoFixedRateLoanProtectionPolicy.sol#L377-L381.

Description: The execution trigger allows a top up when the health factor is at or below `triggerHealthFactor`, which must exceed the `1e18` liquidation boundary. The post execution check only requires the position to be healthy, meaning a health factor of at least `1e18`. It never requires recovery to the trigger.

So any position where `1e18 <= healthFactor <= triggerHealthFactor` can execute while it already passes the postcondition. The amount check rejects zero but accepts one base unit, which adds zero borrowing capacity after downward rounding. A relayer cannot change a signed amount, so the executor must authorize the ineffective top up.

Example:

- Two collateral legs of 100 tokens at 80% LLTV give 160 capacity against 150 debt, so the health factor is about 1.0667 and a 1.10 trigger permits execution.

- A one base unit top up adds nothing, so the health factor is unchanged and the post check passes.
- The one shot flag stays consumed, so further top ups revert with almost all authorized capacity unused.
- The account must install fresh protection to recover.

Recommendation: Consider adding an account configured `minRecoveryHealthFactor` set strictly above `triggerHealthFactor`, commit it in the config, pass it to the post execution check, and require the post top up health factor to reach it, else revert so the one shot flag and nonce are preserved. A strict inequality is needed because requiring recovery only to the trigger still lets a position already at the trigger burn protection with a negligible top up.

Alternatively, if health only success is intentional and recovery sizing is delegated to the executor, document that trust assumption and the risk that a negligible signed top up can consume protection.

Coinbase: Addressed through documentation in commit 4e02d25f.

Cantina: Documentation fix verified. `triggerHealthFactor` is now explicitly documented as a reaction threshold rather than a recovery target. The `_onPostExecute` documentation states that the postcondition only requires `maxDebt >= debt`, delegates top-up sizing to the trusted executor and acknowledges that a nonzero executor-signed top-up can consume the one-shot while only marginally improving health.

3.1.3 Take callbacks can force premature top ups

Severity: Low Risk.

Context: MorphoFixedRateLoanProtectionPolicy.sol#L316-L329.

Description: The policy is intended to release a signed collateral top up only when the account's aggregate health factor has reached the configured threshold. `_onSingleExecutorExecute()` derives the current factor directly from Midnight debt and collateral, but it never checks whether Midnight has placed the account under `liquidationLocked`.

Midnight increases a seller's debt before calling the buyer callback from `take()`. It sets `liquidationLocked` during that callback and postpones seller health validation until the callback returns. A taker filling an account's sell offer can choose a buyer callback that relays a valid executor signature. The policy then reads the provisional debt, passes the trigger, deposits the authorized collateral and consumes the protection. The added collateral makes the outer fill pass the deferred health check even when the same fill would revert without the policy.

This lets a taker use the account's protection budget as extra borrowing capacity. The account ends with more debt, its capped collateral moved into Midnight and its one shot protection consumed even though the stable position before `take()` was above the trigger.

Recommendation: Add `liquidationLocked(bytes32,address)` to the local `IMidnight` interface. In `_onSingleExecutorExecute()`, after deriving the market key and before reading the health factor, reject execution when `IMidnight(MIDNIGHT).liquidationLocked(marketKey, account)` is true. This confines trigger evaluation to stable position states and prevents the protection from completing a debt increase that Midnight has not yet validated.

Coinbase: Fixed in commit 87c51e02.

Cantina: Fix verified. `IMidnight` now exposes `liquidationLocked(bytes32,address)`. `_onSingleExecutorExecute()` checks the lock for the derived market key and protected account before computing the health factor and reverts with `PositionLiquidationLocked` when it is set. A `take()` callback therefore cannot trigger protection against provisional debt or use the authorized collateral to complete an otherwise unhealthy fill. The revert preserves the collateral, executor nonce and one shot protection so the same signed action can execute later when a stable position reaches the trigger.

3.1.4 Top ups disable unrelated Midnight buy offers

Severity: Low Risk.

Context: MorphoFixedRateLoanProtectionPolicy.sol#L443-L471.

Description: The policy gives Midnight an exact temporary allowance before supplying each selected collateral. `_buildTopUpCalls()` first sets the account's existing token allowance to zero and then approves only `topUpAssets`. It does not read or restore the value that existed before execution. A standard ERC20 consumes the exact temporary allowance during `supplyCollateral()`, so the account's allowance to Midnight is zero after a successful top up.

ERC20 allowances are scoped by owner and spender rather than by Midnight market or operation. The same token can be collateral in the protected market and the loan token in another Midnight market. For a maker funded buy offer without a buyer callback, Midnight uses the maker account as the payer and pulls the loan token from that account when a taker fills the offer.

If the account has approved Midnight to keep such an offer fillable, an executor can erase that standing authorization by executing an otherwise valid top up with the same token. The protected position becomes healthy and the policy succeeds but every affected offer reverts until the account submits another wallet transaction to restore the allowance.

Recommendation: Read the account's existing allowance for each selected token before building the wallet call plan. After `supplyCollateral()`, reset the temporary allowance and restore the recorded value. Check the return value of every approval so the batch preserves the prior authorization while retaining compatibility with tokens that require a zero reset.

Coinbase: Fixed through documentation in commit 5719d18f.

Cantina: Documentation fix verified. `_buildTopUpCalls()` now explicitly documents that a top-up clears the account's standing Midnight allowance for the selected token, including an allowance funding maker buy offers in other markets. It states that prior allowances are intentionally not restored and that accounts must avoid relying on those allowances and re-approve after a top-up.

3.2 Informational

3.2.1 Approval failures can go undetected when an existing allowance permits the top-up

Severity: Informational.

Context: MorphoFixedRateLoanProtectionPolicy.sol#L444.

Description: `_buildTopUpCalls()` constructs a wallet batch that resets Midnight's allowance to zero, approves the selected top-up amount, and supplies collateral.

The wallet checks whether each low-level call reverted but does not decode its return value. An `approve` call that returns `false` therefore appears successful to the batch executor. ERC20 explicitly requires callers to handle `false` returns; returning `false` does not itself make a token noncompliant.

If both approval calls fail without changing a sufficient pre-existing allowance, Midnight can still transfer the selected collateral amount using that allowance. For example:

1. The wallet initially approves Midnight to spend 1,000 tokens.
2. Both `approve(Midnight, 0)` and `approve(Midnight, 100)` return `false`, leaving the allowance unchanged.
3. Midnight supplies 100 tokens using the existing allowance.
4. Assuming ordinary allowance consumption, 900 tokens remain approved instead of zero.

If the position passes the health postcheck, execution succeeds and consumes the one-shot protection despite the failed allowance reset and bounded approval.

This does not, by itself, allow the supply call to exceed the authorized top-up amount. The demonstrated impact is failure to establish the intended allowance limit and cleanup. Without sufficient existing allowance, the transfer would ordinarily revert the entire transaction.

Recommendation: In `_buildTopUpCalls()`, encode approval calls that revert on failure rather than raw `approve` calls, since the wallet's batch executor ignores a `false` return. Run the approval from the wallet so it updates the wallet's allowance, validating the return value and resetting the allowance to zero before setting it for tokens that require this, and reverting if it does not succeed.

Alternatively, if this is the designated behavior, update the NatSpec to document the assumption that only tokens whose approvals revert on failure are supported.

Coinbase: Addressed through documentation in commit e78cfb55.

Cantina: Documentation fix verified. `_buildTopUpCalls()` now documents that the smart wallet ignores the boolean returned by `approve` and assumes collateral tokens revert on approval failure or return `true`. It explicitly marks tokens that return `false` without reverting as unsupported and describes how a sufficient existing allowance can let the batch succeed without applying the intended allowance reset or exact amount bound.

3.2.2 Policy does not enforce Midnight per borrower active collateral limit

Severity: Informational.

Context: MorphoFixedRateLoanProtectionPolicy.sol#L254-L267.

Description: Midnight allows up to 128 collateral definitions per market but at most 16 simultaneously active legs per borrower, and its supply function reverts with `TooManyActivatedCollaterals` when activating a new leg would exceed that ceiling. The policy validates only that a selected index is within market bounds and is authorized. It never consults the borrower active collateral count. An account already holding 16 active legs can install protection that authorizes an inactive seventeenth leg, and that protection cannot execute while the position remains at the ceiling. A batch that crosses the ceiling also rolls back any earlier top ups in the same execution. The ceiling only applies when activating a currently zero leg, so topping up already active collateral is unaffected.

Recommendation: At install and at execution, cross check the borrower active collateral bitmap so authorizing or selecting a currently inactive leg cannot push the active count past the maximum, and surface a clear error when a selection would activate a leg that Midnight would reject. If this behavior is intended, document the limitation clearly so integrators understand that protection for an inactive collateral leg may be temporarily unexecutable while the borrower is at the active collateral limit.

Coinbase: Fixed through documentation in commit c930e098.

Cantina: Fix verified as a documentation change. `_onSingleExecutorExecute()` now explicitly states that enforcement of Midnight's limit of 16 active collateral legs per borrower is intentionally delegated to Midnight. A top up that would activate a seventeenth leg reverts the entire execution, including earlier top ups in the batch, preserving the account's collateral and the one shot. Top ups of already active legs are unaffected.

3.2.3 Top up activates unused collateral and breaks single collateral rolling

Severity: Informational.

Context: MorphoFixedRateLoanProtectionPolicy.sol#L336-L351.

Description: The policy restores health by supplying any collateral leg the executor selects and it does not require that leg to already be active in the borrower's collateral bitmap. Because the health computation counts every activated collateral, supplying a previously unused but authorized collateral raises the aggregate health factor and lets execution succeed, leaving a position with more than one active collateral. If the borrower relies on an external fallback rolling mechanism that requires exactly one active collateral, that mechanism reverts after the top up, and removing the added collateral returns the position to an unhealthy state.

Recommendation: Document the composition constraint so borrowers authorize only their one active leg. If the systems are meant to compose as a product guarantee, add an optional install time flag that restricts top ups to legs already active in the borrower's bitmap.

Coinbase: Addressed through documentation in commit e95436c4.

Cantina: Documentation fix verified. `LoanProtectionPolicyConfig.allowedCollaterals` now documents that topping up an inactive leg activates it and that Midnight's fallback rolling into Morpho Blue requires exactly one active collateral. It instructs accounts relying on that rollover to authorize only their single active leg and warns that topping up additional authorized legs can block a later roll.

3.2.4 Misleading NatSpec on `marketKeyByPolicyId`

Severity: Informational.

Context: MorphoFixedRateLoanProtectionPolicy.sol#L88.

Description: The NatSpec for `marketKeyByPolicyId` states that the mapping stores the market key "to validate uninstallation." However, the mapping performs no validation. It is populated during installation and read by `_clearInstallState()` to locate and delete the corresponding `activePolicyByMarket` entry before deleting the reverse lookup itself. Uninstall authorization is handled by the inherited policy and manager flow. This is a documentation discrepancy, not an authorization vulnerability.

Recommendation: Change the NatSpec to describe the real role, a reverse lookup used to clear per install state on uninstall, with no validation. Example: `/// @notice Reverse lookup from policyId to marketKey used to clear per install state on uninstall.`

Coinbase: Fixed in commit 66478cdd.

Cantina: Fix verified. The NatSpec on `marketKeyByPolicyId` now describes its actual role as a reverse lookup from `policyId` to `marketKey` used by `_clearInstallState` to locate and delete the matching `activePolicyByMarket` entry on uninstall. It explicitly states that the mapping performs no authorization and that uninstall access is enforced by the inherited policy and `PolicyManager` flow.