Morpho Midnight Bundles

Security Review

Cantina Solo review by:
StErMi, Lead Security Researcher

August 10, 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 Informational . . . 4
  - 3.1.1 Bulk Informational Finding . . . 4
  - 3.1.2 Bulk natspec issues . . . 5
  - 3.1.3 `MidnightBundles` documentation improvement . . . 6

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

A security review is 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 the review 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 a 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

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

From May 19th to May 26th the security researchers conducted a review of midnight on commit hash a9988d99. A total of **3** issues were identified:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	0	0	0
Gas Optimizations	0	0	0
Informational	3	2	1
Total	**3**	**2**	**1**

2.1 Scope

The security review had the following components in scope for midnight on commit hash a9988d99:

```
src/periphery/MidnightBundles.sol
src/periphery/interfaces/IMidnightBundles.sol
```

3 Findings

3.1 Informational

3.1.1 Bulk Informational Finding

Severity: Informational.

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

(This finding contains elements that refer to `Midnight.sol`, whose additional findings were delivered on a different report).

Description:

1. Consider surrounding with the """ character in the Midnight's contract natspec everything that is code related (values, variables, function names, contract names and so on). Plenty of times it's challenging to read the natspec, particularly when you use the function's name take which can be confused with the actual verb.
2. Midnight.sol#L330-L338: Consider adding additional sanity checks to the `take` function to validate the correctness and coherence of the inputs. For example when `offer.buy=TRUE` we should have `receiverIfMakerIsSeller == address(0)`.
3. Midnight.sol#L184: Consider adding a natspec comment to the `consumed` state variable to clarify that such value could be misleading to monitor the funds consumed by a "group" of offers. The user, or an authorized account, can manually increase it.
4. Consider creating periphery contracts to deploy pre-configured Midnight markets. The `setMarketTickSpacing`, `setMarketTradingFee` and `setMarketContinuousFee` functions **require** the market to have already been created.
5. Midnight.sol#L343: Consider reverting the `take` action when **both** `offer.maxAssets` and `offer.maxUnits` are equal to zero. The `take` function will **always** revert with the `ConsumedUnits` error unless `units` is equal to zero.
6. Midnight.sol#L512: The `onRepay` function does not pass `msg.sender` as part of the input parameter which could be an important piece of information for the `callback` logic.
7. Midnight.sol#L539: Consider tracking in the `SupplyCollateral` event if the user has activated a new collateral or emit a custom new event.
8. Midnight.sol#L566: Consider tracking in the `WithdrawCollateral` event if the user has deactivated an existing collateral or emit a custom new event.
9. In `Midnight` th `collateral` function returns a `uint128` value given that the collateral amount of a user is stored in `uint128`. The `creditOf` function is instead returning the credit of a user in `uint256` even if the credit is also stored as a `uint128` value. Morpho should consider aligning all these functions to return the same integer type.
10. EventsLib.sol#L24: The `callback` input parameter of the `Repay` event should be named `payer` and not `callback`.
11. MidnightBundles.sol: Apply the same order of `_forceApproveMax` and `_pullToken` execution across all the bundler's functions.
12. MidnightBundles.sol: Consider refactoring the functions in the `MidnightBundles` contract to apply, where possible and needed, a common and coherent code and logic. For example, execute the `_forceApproveMax` and `_pullToken` functions in the same order or define the `loanToken = takes[0].offer.market.loanToken` local variable to be used in the code instead of always accessing it with the long-form reference. There are multiple places where the code can be improved for better maintainability and reading experience.
13. MidnightBundles.sol#L357: Align the `_safeApprove` function logic to the one used by similar functions defined in `SafeTransferLib`. Add a custom error when the `require` statement fails.

Recommendation: Morpho should consider implementing all the suggestions listed in the above section.

Stermi:

1) Acknowledged by Morpho.
2) Fixed by PR 962.
3) Fixed by PR 962.
4) Acknowledged by Morpho.
5) Fixed by PR 962.
6) Acknowledged by Morpho.
7) Acknowledged by Morpho.
8) Acknowledged by Morpho.
9) Fixed by PR 935. Now `creditOf`, `debtOf`, `totalUnits`, `withdrawable` and `continuousFeeCredit` returns `uint128`.
10) Fixed by PR 909.
11) Fixed by PR 919.
12) Fixed by PR 919.
13) Fixed by PR 951.

3.1.2 Bulk natspec issues

Severity: Informational.

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

(This finding contains elements that refer to `Midnight.sol`, whose additional findings were delivered on a different report).

Description:

1. Midnight.sol#L105-L107: The natspec of the "AUTHORIZATIONS" sections should be updated to specify that the authorized account/sc can **fully** manage the authorizations for the `onBehalf` user. The authorized account can **add** other accounts but **also remove** existing accounts. The natspec does not mention the removal part of the logic.
2. Midnight.sol#L160: consider adding to the list of the assumptions in the "MISC" section that Midnight assumes that both the `buyerCallback` and `sellerCallback` executions will consume as little gas as possible. Theoretically those callback calls could be used to grief `msg.sender` to pay for "extra" gas and make the tx expensive.
3. Midnight.sol#L164: `lossFactor` in the MISC natspec comment should be replaced with `lastLossFactor`.
4. Midnight.sol#L164: `continuousFeeCredit` and `lastAccrual` should be included in the list of values/functions that could be not up-to-date when fetched.
5. Midnight.sol#L160: Consider expanding the "no-op" disclosure in the "MISC" natspec section documenting that operations like `withdrawCollateral` or `repay` can be executed even on non-existing markets (that have no collateral to be withdrawn or debt to be repaid) as long as the `assets` / `units` (depending on the function signature) is equal to zero. Those "impossible" operations will not revert, create a new market and emit an event.
6. TickLib.sol#L54: The natspec of the `TickLib.priceToTick` function contains a typo: "Among the ticks **than** are multiples" → "Among the ticks **that** are multiples".
7. MidnightBundles.sol: Consider adding to the general assumption of the `MidnightBundles` contract that the order of the `takes` and the amount `units` configured for each of them matters quite a

lot for both the success of the operation and also for how much the operation is going to cost/be profitable (depending on if you are a borrower or a supplier).

8. MidnightBundles.sol: Consider adding to the general assumption of the `MidnightBundles` contract that "No-ops are allowed" and "Zero checks are not systematically performed".
9. MidnightBundles.sol: The `buyWithUnitsTargetAndWithdrawCollateral` and `buyWithUnitsTargetAndWithdrawCollateral` functions mention the usage of `TakeAmountsLib` saying "@dev Reverts if TakeAmountsLib or ConsumableUnitsLib reverts." but they **do not** use it internally. Remove the mention from the natspec comments.
10. MidnightBundles.sol: Consider rewriting the natspec section about "Total receipt". The term and concept are not clear as they are right now.
11. MidnightBundles.sol#L363: Consider documenting the special allowance check performed by `MidnightBundles._forceApproveMax`.
12. HashLib.sol#L51-L52: The comment "In practice the height is capped at 20 by offerTreeTypeHash" is misleading. The comment is true for the `EcrecoverRatifier` because `HashLib.offerTreeTypeHash(proof.length)` would revert with `length > 20` but it's **not** true for the `SetterRatifier`. The `SetterRatifier` has **NO cap** on the height of the three.
13. Midnight.sol#L111-L118: There are scenarios where the `Minight.take` operation could end up (depending on the `tick`, `price`, `_tradingFee` and rounding) with `buyerAssets` or `sellerAssets` equal to zero. This means that (depending on the `offer` configuration) one of the parts will lose something and the other will gain something. The "ROUNDING" section of the natspec should be updated to cover these scenarios.
14. Midnight.sol#L27-L28: The "When some assets become withdrawable before maturity (after a repayment or a liquidation), there is an incentive to take resting sell offers with price < 1 and withdraw instantly" natspec should also include the fee that the `taker` (the one that is buying the units) has to pay to withdraw those units instantly at face value.

Recommendation: Morpho should consider implementing all the suggestions listed in the above section.

Morpho: Acknowledged.

Stermi: Acknowledged.

3.1.3 `MidnightBundles` documentation improvement

Severity: Informational.

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

Description: The documentation / natspec of the `MidnightBundles` contract could be widely improved. While the core logic of the functions offered is easy to understand and follow, the usage and the roles of the actors (depending on the context and the inputs) are not straight forward.

Let's take for example the `supplyCollateralAndSellWithUnitsTarget` function. Let's assume that `taker != msg.sender != receiver != referralFeeRecipient`. In such a scenario.

- `msg.sender` is providing the collateral funds that will be accounted to the `taker` input.
- `taker` is taking the debt (sell the units) accounted for in the Midnight contract.
- `receiver` is the account that will receive the loan tokens from the debt accounted to `taker`.
- `referralFeeRecipient` will receive part of the loan tokens that `receiver` would take as part of a "referral fee" concept.

Without a proper explanation of each role for each function exposed by `MidnightBundles` the usage and understanding of these functions remain unclear and confusing.

Recommendation: Morpho should consider documenting and explaining the usage (with explicit examples and scenarios) of these ”utility functions” and the role that each actor represents in the context of each function.

In the specific context of the `supplyCollateralAndSellWithUnitsTarget` function, Morpho should also document that the `msg.sender` could end up overcollateralizing the `taker` (the borrower) if it provides more collateral than needed to perform the `take` action on Midnight.

Morpho: Fixed in PR 965.

Stermi: Fix verified.