Blend PR 80
Security Review

Solo review by:
Sujith Somraaj, Lead Security Researcher

October 10, 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

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

Blend's Intent Engine delivers sustainable yield and auto-derisks your capital.

From Oct 6th to Oct 7th the security researchers conducted a review of Blend PR80 on commit hash f128acad. A total of **4** 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	4	4	0
Total	**4**	**4**	**0**

3 Findings

3.1 Informational

3.1.1 Missing visibility identifier for MIN_SECONDS_BETWEEN_OPERATIONS variables

Severity: Informational

Context: StrategyManager.sol#L44

Description: The state variable `MIN_SECONDS_BETWEEN_OPERATIONS` lacks an explicit visibility modifier. In Solidity, state variables default to internal visibility when no modifier is specified. This can lead to confusion and make the code less explicit about intended access patterns.

Recommendation: Explicitly declare the visibility modifier for `MIN_SECONDS_BETWEEN_OPERATIONS` (e.g., private, internal, or public) to improve code clarity and maintainability.

BlendMoney: Fixed in PR 83.

Sujith Somraaj: Fix verified.

3.1.2 Virtual function allows rate limit bypass in derived contracts

Severity: Informational

Context: StrategyManager.sol#L278

Description: The `executeRebalance()` function is marked virtual, allowing derived contracts to override it. A badly implemented derived contract could:

- Override without calling `super.executeRebalance()`.
- Skip the `_verifyRateLimit()` check entirely.
- Execute unlimited operations.

This is also in contrast to the `executeVaultAction()` function, which is external.

Recommendation: Consider doing either of the following fixes:

- Remove the `virtual` keyword if inheritance isn't needed.
- Add comprehensive documentation warning about maintaining rate limits in overrides.

BlendMoney: Fixed in PR 83.

Sujith Somraaj: Fix verified.

3.1.3 Off-by-one error in rate limiting allows operations one second earlier than intended

Severity: Informational

Context: StrategyManager.sol#L259

Description: The `_verifyRateLimit()` function contains an off-by-one error in its rate-limiting check, allowing operations to be executed one second earlier than the intended minimum interval.

If the `MIN_SECONDS_BETWEEN_OPERATIONS` is set to 1, then there is no rate limiting between operations (breaking the entire functionality).

Recommendation: Change the comparison operator from >= to > to ensure that more than `MIN_SECONDS_BETWEEN_OPERATIONS` have elapsed:

```
  function _verifyRateLimit(address safe, address vault) internal {
      bytes32 safeVaultKey = keccak256(abi.encodePacked(safe, vault));
-     require(block.timestamp - lastOperationTimestamp[safeVaultKey] >= MIN_SECONDS_BETWEEN_OPERATIONS,
↪  RateLimited());
+     require(block.timestamp - lastOperationTimestamp[safeVaultKey] > MIN_SECONDS_BETWEEN_OPERATIONS,
↪  RateLimited());
      lastOperationTimestamp[safeVaultKey] = block.timestamp;
  }
```

BlendMoney: Fixed in PR 83.

Sujith Somraaj: Fix verified.

3.1.4 Add upper bound limit for `_minSecondsBetweenOperations`

Severity: Informational

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

Description: The `_minSecondsBetweenOperations` is validated to be not equal to zero. But there is no upper bound limit enforced.

Recommendation: Consider enforcing an upper bound for the `_minSecondsBetweenOperations` as follows:

```
  constructor(uint256 _minSecondsBetweenOperations) {
-    require(_minSecondsBetweenOperations > 0, InvalidMinSecondsBetweenOperations());
+   require(_minSecondsBetweenOperations > 0 && _minSecondsBetweenOperations <= 100,
↪  InvalidMinSecondsBetweenOperations());
    MIN_SECONDS_BETWEEN_OPERATIONS = _minSecondsBetweenOperations;
  }
```

BlendMoney: Fixed in PR 83.

Sujith Somraaj: Fix verified.