SHERLOCK

Security Review For Blend Money

Collaborative Audit Prepared For:	Blend Money
Lead Security Expert(s):	PUSH0
Date Audited:	November 27 - November 28, 2025

Introduction

Blend is an intent-driven, non-custodial Yield Coordination Engine. It functions as a savings account for your digital assets: you keep full custody in a personal Gnosis Safe while Blend's automation allocates capital between stable base yields and sophisticated, delta-neutral strategies.

This architecture provides the simplicity of a "set-and-forget" yield app. Blend achieves this by building on top of trusted DeFi primitives, not by replacing them.

Scope

Repository: BlendMoney/contracts

Audited Commit: ea1e3a16cba4f484d3fe6607f0849f16f8f231f5

Final Commit: 822824e1b951462d28d130e1c13429b7a58f8742

Files:

- src/adapters/SwapAdapter.sol
- src/libraries/PriceLib.sol

Final Commit Hash

822824e1b951462d28d130e1c13429b7a58f8742

Findings

Each issue has an assigned severity:

- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	0	2

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue L-1: Material amount of intermediate token can get stuck due to price action [RESOLVED]

Source: https://github.com/sherlock-audit/2025-11-blend-money-nov-22nd/issues/4

Summary

Swap adapter does not refund intermediate tokens. Due to price action, a material amount of intermediate can get stuck (and not dust amount).

This is a revisit of issue 3.1.2 in a previous audit report. We further investigate the extent of the potential impact

Vulnerability Detail

From Cantina's audit report of the same scope, issue 3.1.2, it was identified that the intermediary token dust is not refunded in multi-hop swaps. The issue was acknowledged, and refunding was not implemented, prioritizing gas efficiency and simplicity of the design.

However, intermediary tokens can also be stuck due to price actions rather than just rounding or AMM math properties. Supposing the price action moves *in favor* of the swap, then the extra benefits are not captured for the rebalancing.

Let's say the swap is A --> B --> C. Price actions/slippage (in the swap's favor) can cause either:

- For the same A input, we get more B output than we expect
- For the same C output, we need not as much B input as originally expected

This causes the potential amount stuck to be possibly material, and not limited to dust amounts.

Impact

Material amount of intermediate tokens in multi-hop swaps can get stuck

Code Snippet

https://github.com/sherlock-audit/2025-11-blend-money-nov-22nd/blob/main/contracts/src/adapters/SwapAdapter.sol#L248-L256

Tool Used

Manual Review

Recommendation

Implement a sweep token functionality, if it is desired to retain the original design.

Discussion

abg4

How we execute swaps using routers is explained in this issue. However, we plan on using the `WhitelistedSwapAdapter` going forward and while funds could technically be recovered during a rebalance, it would be convenient to allow the owner of the `RolesReceiver` to sweep funds to a recipient if funds are stuck in the contract for an reason.

Here is the PR but let me know if you have any questions:

- https://github.com/BlendMoney/contracts/pull/87

midori-fuse

Adjusting the severity to Low/Info. It is technically possible to sweep tokens through a rebalancing. Hence even with the current code as-is, careful and proper usage should enable an easy recovery scenario regardless.

Issue L-2: Wrong code comment [RESOLVED]

Source: https://github.com/sherlock-audit/2025-11-blend-money-nov-22nd/issues/5

In `SwapAdapter.sol`, there is an incorrect code comment:

```
// Transfer only the swapped amount to the recipient
loanToken.safeTransfer(recipient, loanBalanceAfterSwap);
```

https://github.com/sherlock-audit/2025-11-blend-money-nov-22nd/blob/main/contracts/src/adapters/SwapAdapter.sol#L233-L234

```
// Transfer only the swapped amount to the recipient
collateralToken.safeTransfer(recipient, collateralBalanceAfterSwap);
```

https://github.com/sherlock-audit/2025-11-blend-money-nov-22nd/blob/main/contracts/src/adapters/SwapAdapter.sol#L182-L183

These actually transfers the entire output balance to the recipient, and not just the output amount.

Discussion

james-a-morris

Great catch: https://github.com/BlendMoney/contracts/pull/86

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users’ sole risk and is the users’ responsibility.