Pendle Boros Router & AMM Audit Report

Jul 29, 2025

Table of Contents

Summary

This report has been prepared for Pendle Boros smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Overview

Project Summary

Project Name	**Pendle Boros**
Codebase	**https://github.com/pendle-finance/pendle-core-v3**
Commit	**a905b3788f13edba3bbdb7e49b5594ae51f28b31**
Language	**Solidity**

Audit Summary

Delivery Date	**Jul 29, 2025**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**7**

[WP-M1] Swap-out amounts should round in favor of the market instead of the user

Medium

Issue Description

Because both invariant-keeping steps round down, the pool's computed `newNormFixedAmount` is always at least 1 wei smaller than the mathematically correct value and always favourable to the trader:

1. Float → Fixed (`floatOut < 0` , user deposits float and receives fixed)
   - `newNormFixedAmount < ideal`
   - Change ΔNormFixedAmount = new – old is negative and its magnitude is **larger** than the ideal one.
   - After scaling by 1/t, the user receives **more fixed tokens** than the fair-curve amount.
2. Fixed → Float (`floatOut > 0` , user withdraws float and pays fixed)
   - `newNormFixedAmount < ideal`
   - Change ΔNormFixedAmount is positive but **smaller** than it should be.
   - After scaling by 1/t, the user is asked to transfer **fewer fixed tokens** than fair value.

So the bias works in the trader's favour in both directions: they either get extra fixed when they take fixed out, or pay too little fixed when they bring fixed in.

```
function calcSwapOutput(AMMState memory state, int256 floatOut) internal pure returns (int256 fixedIn) {
    uint256 normalizedTime = calcNormalizedTime(state);

    uint256 newTotalFloatAmount;
    uint256 floatOutAbs = floatOut.abs();
    if (floatOut > 0) {
        // totalFloatAmount.pow(normalizedTime) does not work when totalFloatAmount = 1
        require(state.totalFloatAmount > floatOutAbs + 1, Err.AMMInsufficientLiquidity());
        unchecked {
            newTotalFloatAmount = state.totalFloatAmount - floatOutAbs;
        }
    } else {
```

```
        newTotalFloatAmount = state.totalFloatAmount + floatOutAbs;
    }

    uint256 liquidity = state.totalFloatAmount.pow(normalizedTime).mulDown(state.normFixedAmount);
    uint256 newNormFixedAmount = liquidity.divDown(newTotalFloatAmount.pow(normalizedTime));
    require(
        newNormFixedAmount * PMath.ONE >= state.minAbsRate * newTotalFloatAmount,
        Err.AMMInsufficientLiquidity()
    );
    require(
        newNormFixedAmount * PMath.ONE <= state.maxAbsRate * newTotalFloatAmount,
        Err.AMMInsufficientLiquidity()
    );
    int256 normFixedIn = newNormFixedAmount.Int() - state.normFixedAmount.Int();

    state.totalFloatAmount = newTotalFloatAmount;
    state.normFixedAmount = newNormFixedAmount;

    return normFixedIn.divDown(normalizedTime.Int());
}
```

Response from Pendle team:
The impact of this rounding is in the order of 1 wei of the collateral asset for each trade, which we deem as too insignificant compared to the AMM fees.

Status

ⓘ **Acknowledged**

[WP-D2] In section 3.6 of the whitepaper, the ”amount of fix stream tokens” might need to be changed from ”decreased” to ”increased”

Issue Description

The comment is made on boros-amm-whitepaper-25-05-2025.pdf

Section 3.6 on page 3:

> the amount of fix stream tokens ==decreased== during the duration t_1 to t_2, from y_1 to $y_1^{'}$ such that $y_1^{'} t_2 = y_1 t_1$.

Expected:

> the amount of fix stream tokens ==increased== during the duration t_1 to t_2, from y_1 to $y_1^{'}$ such that $y_1^{'} t_2 = y_1 t_1$.

Status

✓ **Fixed**

[WP-D3] Missing a in the formulas for $x^{'}$ and dx in Whitepaper Section 3.6.1

Issue Description

The comment is made on boros-amm-whitepaper-25-05-2025.pdf

Section 3.6.1 on Page 4

x$^{'} = \left(\frac{(x+a)^t yt}{r^{'}}\right)^{\frac{1}{t+1}}$
dx = x - x$^{'}$

Should be corrected to:

x$^{'} + a = \left(\frac{(x+a)^t yt}{r^{'}}\right)^{\frac{1}{t+1}}$
dx = x - x$^{'}$
= (x + a) - (x$^{'} + a$)
= (x + a) - $\left(\frac{(x+a)^t yt}{r^{'}}\right)^{\frac{1}{t+1}}$

This is consistent with the code implementation:

https://github.com/pendle-finance/pendle-core-v3/blob/5d3a330e47ea75e561644a53359bcad8a6a9f1ca/contracts/core/amm/PositiveAMMMath.sol#L169-L178

```
    function calcSwapSize(AMMState memory state, int256 targetRateInt) internal pure returns (int256 swapSize) {
        uint256 targetRate = clampRate(state, targetRateInt).Uint();
        uint256 normalizedTime = calcNormalizedTime(state);
        uint256 normalizedTimePlusOne = normalizedTime + PMath.ONE;
        uint256 liquidityMul1E18 = state.totalFloatAmount.pow(normalizedTime) * state.normFixedAmount;
        uint256 newTotalFloatAmount = (liquidityMul1E18 / targetRate).pow(PMath.ONE.divDown(normalizedTimePlusOne)).max(
            2
        );
        swapSize = state.totalFloatAmount.Int() - newTotalFloatAmount.Int();
    }
```

Derivation process:

$y^{'}t_{\overline{x^{'}+a=r^{'}}}$

$(x + a)^t yt = (x^{'} + a)^t y^{'} t = r^{'} \times (x^{'} + a)^{t+1}$

$(x + a)^t yt_{\overline{r^{'}=(x^{'}+a)^{t+1}}}$

$\left(\frac{(x+a)^t yt}{r^{'}}\right)^{\frac{1}{t+1}} = x^{'} + a$

Status

✓ **Fixed**

[WP-D4] OTC swap is yet to be implemented

Issue Description

The OTC Swap feature described in the whitepaper currently has no entry points that actually allow "A user u_i can trigger an OTC swap with another user u_j (with their consent) to create a new pair of swaps of size s_{new} between the two of them."

3.3.5 OTC swap

This must be triggered by "Execute a batch of orders"

A user u_i can trigger an OTC swap with another user u_j (with their consent) to create a new pair of swaps of size s_{new} between the two of them. A state change "Create a new pair of swaps" is triggered.

The initiator will pay an OTC fees of:

$$\text{otcFees} = F_{\text{otc}} \times \text{fee}_{u_{\text{otc}}} \times |s_{\text{new}}| \times \left(t_{\text{end}} - t\right)$$

Status

Acknowledged

[WP-I5] `AuthModule` assumes there must not be any reentrancy vulnerabilities within function calls

Informational

Issue Description

If any function has a reentry point, it can become an attack vector.

Note that anyone who captures the signature can execute the call.

When `account` is set, `setNonAuth` will not reset the account.

If the Token in TradeModule.solL51 allows reentrant, any methods with the `setNonAuth` modifier called within this reentry will operate on the original signer's account.

```
modifier setAuth(Account acc) {
    _setUnchecked(acc);
    _;
    _setUnchecked(AccountLib.ZERO_ACC);
}

// * Set the account without any message authentication. If an account is already set, it won't be overridden.
// The case for an account already set is when the call runs through the AuthModule and gets delegated to the
// TradeModule or AMMModule
modifier setNonAuth() {
    Account old = _account();
    if (old.isZero()) _setUnchecked(AccountLib.from(msg.sender, 0));
    _;
    if (old.isZero()) _setUnchecked(AccountLib.ZERO_ACC);
}
```

```
function vaultDeposit(
    VaultDepositMessage memory message,
    bytes memory signature
) external setAuth(message.root.toMain()) {
    _verifySignerSigAndIncreaseNonce(message.root, message.nonce, _hashVaultDepositMessage(message), signature);
```

```

    address(this).functionDelegateCall(
        abi.encodeCall(
            ITradeModule.vaultDeposit,
            (message.accountId, message.tokenId, message.marketId, message.amount)
        )
    );
}
```

```
function vaultDeposit(uint8 accountIdRcv, TokenId tokenId, MarketId marketId, uint256 amount) external setNonAuth {
    Account account = _account();
    require(account.isMain(), Err.TradeOnlyMainAccount());

    MarketAcc acc = AccountLib.from(account.root(), accountIdRcv, tokenId, marketId);

    address token = _MARKET_HUB.tokenIdToAddress(tokenId);

    IERC20(token).safeTransferFrom(acc.root(), address(this), amount);
    IERC20(token).forceApprove(address(_MARKET_HUB), type(uint256).max);

    _MARKET_HUB.vaultDeposit(acc, amount);
}
```

Status

ⓘ Acknowledged

[WP-L6] Router admin cannot disable deprecated/flawed AMMs

Low

Issue Description

There is no admin interface to execute `_setAMMIdToAcc(ammId, AccountLib.ZERO_MARKET_ACC);` .

For example, after an AMM's DelevLiqNonce becomes non-zero (such as after `forceDeleverage()` / `liquidate()` is called), it can no longer be used for swaps or receive additional liquidity.

The other case is that if one AMM's implementation is found to be flawed, continuing to use it for swaps may pose a security risk; we will need a way to disable the AMM.

https://github.com/pendle-finance/pendle-core-v3/blob/202a68a5fb1f03aed2579b5052d2251f2cfda8c2/contracts/core/router/modules/MiscModule.sol#L113-L120

Note that `amm.burnByBorosRouter()` only allows `ROUTER` as the `msg.sender` . The router removing an AMM could prevent remaining liquidity providers from exiting their positions.

Consider adding a method to disable adding liquidity and swapping through a given AMM while continuing to allow the withdrawal of liquidity.

```
    function setAMMIdToAcc(address amm) external onlyAuthorized {
        AMMId ammId = IAMM(amm).AMM_ID();
        MarketAcc ammAcc = IAMM(amm).SELF_ACC();
        require(!ammId.isZero(), Err.InvalidAMMId());
        require(!ammAcc.isZero(), Err.InvalidAMMAcc());
        _setAMMIdToAcc(ammId, ammAcc);
        emit AMMIdToAccUpdated(ammId, ammAcc);
    }
```

https://github.com/pendle-finance/pendle-core-v3/blob/202a68a5fb1f03aed2579b5052d2251f2cfda8c2/contracts/core/router/trade-base/TradeStorage.sol#L45-L47

```
    function _setAMMIdToAcc(AMMId ammId, MarketAcc amm) internal {
        _getTradeStorage().ammIdToAcc[ammId] = amm;
    }
```

https://github.com/pendle-finance/pendle-core-v3/blob/202a68a5fb1f03aed2579b5052d2251f2cfda8c2/contracts/core/router/trade-base/TradeStorage.sol#L13-L19

```
    struct TradeStorageStruct {
        mapping(MarketId marketId => MarketCache cache) marketIdCache;
        mapping(AMMId ammId => MarketAcc amm) ammIdToAcc;
        uint16 numTicksToTryAtOnce;
        uint256 maxIterationAddLiquidity;
        uint256 epsAddLiquidity;
    }
```

```
    function swapByBorosRouter(
        int256 sizeOut
    ) external onlyRouterWithOracleUpdate notWithdrawOnly returns (int256 costOut) {
@@ 135,137 @@
    }
```

```
    modifier notWithdrawOnly() {
        require(_isSizeInSync(), Err.AMMWithdrawOnly());
        _;
    }
```

```
    function _isSizeInSync() internal view returns (bool) {
        return IMarket(MARKET).getDelevLiqNonce(SELF_ACC) == 0;
    }
```

Response from Pendle team:
For users who interact with Boros through frontend app, we will make sure to only send transactions with valid AMM ID.

Status

ⓘ **Acknowledged**

[WP-I7] Router admin needs to be cautious when calling `setAMMIdToAcc(address amm)` to prevent accidentally overwriting existing AMM mappings.

Informational

Issue Description

`ammId` comes from `IAMM(amm).AMM_ID()` rather than being assigned by the router, which could lead to unintended duplicates or conflicts.

Furthermore, `setAMMIdToAcc()` lacks safety checks like:

```
MarketAcc oldAmmAcc = _getTradeStorage().ammIdToAcc[ammId];
require(oldAmmAcc.isZero() || oldAmmAcc == toBeReplaceAmmAcc, ...);
```

```
    function setAMMIdToAcc(address amm) external onlyAuthorized {
        AMMId ammId = IAMM(amm).AMM_ID();
        MarketAcc ammAcc = IAMM(amm).SELF_ACC();
        require(!ammId.isZero(), Err.InvalidAMMId());
        require(!ammAcc.isZero(), Err.InvalidAMMAcc());
        _setAMMIdToAcc(ammId, ammAcc);
        emit AMMIdToAccUpdated(ammId, ammAcc);
    }
```

```
    function _setAMMIdToAcc(AMMId ammId, MarketAcc amm) internal {
        _getTradeStorage().ammIdToAcc[ammId] = amm;
    }
```

```
    struct TradeStorageStruct {
        mapping(MarketId marketId => MarketCache cache) marketIdCache;
        mapping(AMMId ammId => MarketAcc amm) ammIdToAcc;
        uint16 numTicksToTryAtOnce;
        uint256 maxIterationAddLiquidity;
```

```
        uint256 epsAddLiquidity;
    }
```

Status

Acknowledged

Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.