F Y E O

Bio

Repo: https://github.com/bio-xyz/launchpad-agent-evm

Security Review Update: October 7, 2025

Reviewer: balthasar@gofyeo.com

Bio Launchpad EVM Security Review Update

New security issues, 2

After the development team implemented the latest updates, FYEO conducted a review of the modifications. The primary goal of this evaluation was to ensure the continued robustness of the program's security features, safeguarding user funds and maintaining the overall integrity of the program.

General Updates:

The update modernizes and reworks the project's token launch and factory flows to move from a simpler liquidity model to a more flexible, concentrated-liquidity approach. The factory, token, and launch components were changed so new launches can include fee levels and sets of concentrated liquidity ranges; those ranges and fee choices are carried through proposal, creation, and initial-liquidity steps. Several pieces of plumbing were shifted to support the new flow — including how tokens and positions are created, how initial liquidity is provided, and how launch reserves are allocated to project treasuries and liquidity. Overall, the change broadens the system's ability to create more precisely-configured liquidity positions at launch.

At the same time, the update replaces older Uniswap V2–style integrations with code that targets a different swap/positioning infrastructure and simplifies or removes legacy tax-and-liquidity bookkeeping that was previously scattered through the token code. Launch and execute flows were adjusted so launch finalization now passes additional financial parameters and the execution component accepts new swap parameters and treasury routing. Several utility pieces and type definitions were added to support the new architecture. The net effect is a refactor toward a more feature-rich launch process with concentrated-liquidity support and a slimmer token contract, at the cost of changing many assumptions and integration points across the system.

Missing fallback for unsupported fee values

Finding ID: FYEO-BIO-01
Severity: **Informational**
Status: **Remediated**

Description

The `_getTick` function only sets tick bounds for explicit fee values and returns default zeros for any other fee, which can lead to unintended tick ranges (0,0) being used downstream without an explicit revert or validation.

Proof of Issue

File name: src/AgentToken.sol
Line number: 623

```
function _getTick(uint24 fee) internal view returns (int24 tickLower, int24
tickUpper) {
    if (fee == 10000) { tickLower = -887200; tickUpper = 887200; }
    if (fee == 3000)  { tickLower = -887220; tickUpper = 887220; }
    if (fee == 500)   { tickLower = -887270; tickUpper = 887270; }
    if (fee == 100)   { tickLower = -887272; tickUpper = 887272; }
    // no else / revert for unsupported fee -> returns (0,0)
}
```

Severity and Impact Summary

Using (0,0) tick bounds unintentionally can cause incorrect or failed position mints or unexpected liquidity placement. Downstream code may assume valid ticks and proceed, producing subtle runtime failures or economic errors.

Recommendation

Validate `fee` and revert on unsupported values.

ProposeAgent lacks validation for ranges

Finding ID: FYEO-BIO-02
Severity: **Informational**
Status: **Remediated**

Description

The `proposeAgent` function accepts a `ConcentratedRange[]` array but does not validate length or per-range amounts. `LaunchFactory.createLaunch` enforces a max ranges length, but `proposeAgent` is another entry point that must also enforce bounds and that each `amount > 0`.

Proof of Issue

File name: src/AgentFactory.sol
Line number: 198

```
function proposeAgent(string memory name, string memory symbol, uint24 fee,
ConcentratedRange[] memory ranges)
    public
    whenNotPaused
    returns (uint256)
{
    AgentFactoryStorage storage $ = _getAgentFactoryStorage();

    uint256 bioForLpPositions;
    address sender = _msgSender();
    uint256 applicationThreshold_ = $.applicationThreshold;
```

Severity and Impact Summary

Unbounded or zero-amount ranges could cause gas exhaustion.

Recommendation

Enforce a maximum `ranges.length` consistent with `createLaunch` and validate each range (`amount0 > 0`, `amount1 > 0`, and `tickLower < tickUpper`) before transferring funds. Compute totals and validate bounds **before** `safeTransferFrom`.

Commit Hash Reference:

For transparency and reference, the security review was conducted on the specific commit hash for the Bio repository. The commit hash for the reviewed versions is as follows:

92af0e6e99e1e85ff7b849839e6593973106d00a

Remediations were made at commit: 9dda2d1c940ef326d8dc5403bbf63e0442606417

Conclusion:

In conclusion, the security aspects of the Bio program remain robust and unaffected by the recent updates. Users can confidently interact with the protocol, assured that their funds are well-protected. The commitment to security exhibited by the development team is commendable, and we appreciate the ongoing efforts to prioritize the safeguarding of user assets.