LinumLabs

Flagship Vesting- Audit

Prepared by Linum Labs AG

2025-06-13

linumlabs.com
info@linumlabs.com
+1 (650) 534-2011

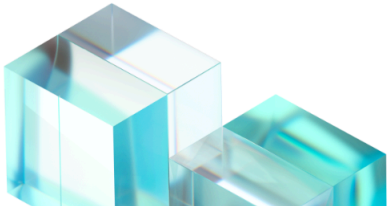
Executive Summary

This document outlines the security review of the Flagship smart contracts. Linum Labs Auditing has identified several vulnerabilities that should be addressed before deploying smart contracts to any live chain. The report outlines these vulnerabilities and advises on how to fix them.

Protocol Summary

Overview

Flagship is an innovative platform creating AI-managed crypto portfolios, aiming for mass adoption with its mobile-first approach. At its core is a "Crypto Brain" that leverages autonomous AI managers, adaptive learning, and real-time data to provide 24/7 crypto portfolio strategizing.



linumlabs.com
info@linumlabs.com
+1 (650) 534-2011

Audit Scope

./src/Vesting.sol

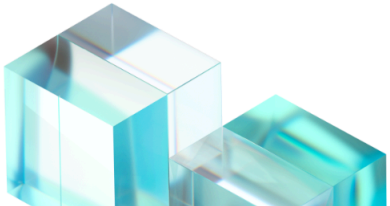
Audit Results

Summary

Repository	https://github.com/LinumLabs/flagship-contracts
Commit	883611a8cb17b342ee0a462a6953a70ebbc3b02d
Timeline	June 12th - June 13th

Issues Found

Bug Severity	Count
Critical	0
High	0
Medium	1



linumlabs.com
info@linumlabs.com
+1 (650) 534-2011

Low	0
Informational	3
Total Findings	4

Summary of Issues

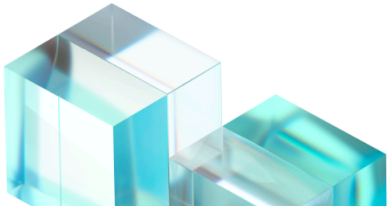
Description	Severity	Status
Vested but Unclaimed Tokens Become Stuck After Revocation.	Medium	Resolved
Redundant beneficiaryScheduleExists Mapping.	Informational	Resolved
Redundant Assignments in createVestingSchedule.	Informational	Resolved
Revocation Mechanism Does Not Refund Payment Tokens.	Informational	Acknowledged

Issues

Medium Severity

1. Vested but Unclaimed Tokens Become Stuck After Revocation.

Description: When a vesting schedule is revoked by the owner, any tokens that have already vested (passed the cliff and linearly vested up to the revocation time) but have not yet been claimed by the beneficiary become permanently inaccessible. The `revoke` function sets `schedule.revoked = true`, and the `claim` function explicitly reverts if `schedule.revoked` is true. This prevents the beneficiary from ever claiming their rightfully vested tokens after the revocation.



linumlabs.com
info@linumlabs.com
+1 (650) 534-2011

Potential Risk: Loss of funds for users. Tokens they are legitimately entitled to, having met their vesting criteria, become locked within the contract and cannot be withdrawn.

Suggested Mitigation: Modify the `revoke` function to immediately transfer any vested but unreleased tokens to the beneficiary *before* marking the schedule as revoked or performing any other transfers. This ensures the beneficiary receives all tokens they are due at the time of revocation. The `TokensReleased` event should also be emitted for these tokens.

Flagship: Fixed in PR.

Linum Labs: Verified.

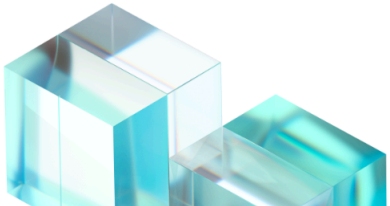
Informational Severity

2. Redundant **beneficiaryScheduleExists** Mapping.

Description: The `beneficiaryScheduleExists` mapping (`mapping(address beneficiary => mapping(uint256 scheduleId => bool)) public beneficiaryScheduleExists;`) is created and updated in `createVestingSchedule`. However, in the `claim` function, the check `require(schedule.beneficiary == msg.sender, "Not schedule beneficiary");` already verifies that the `msg.sender` is the legitimate beneficiary for the given `scheduleId`.

Potential Risk: This mapping adds a small amount of gas cost for storage and updates when a new vesting schedule is created, without providing a clear functional benefit in the current contract logic for on-chain verification.

Suggested Mitigation: Consider removing this mapping to optimize gas usage, unless there is a specific off-chain indexing or future on-chain functionality that



explicitly relies on iterating or querying schedules by beneficiary (e.g., a `getSchedulesForBeneficiary` function).

Flagship: Fixed in PR.

Linum Labs: Verified.

3. Redundant Assignments in `createVestingSchedule`.

Description: In the `createVestingSchedule` function, the lines `schedule.releasedAmount = 0;` and `schedule.revoked = false;` are explicitly setting values that are already the default for `uint256` and `bool` types, respectively, when a new `VestingSchedule` struct is initialized in storage via `vestingSchedules[scheduleId]`. Since `nextScheduleId` ensures a unique, previously unassigned ID for each new schedule, these explicit assignments are unnecessary.

Potential Risk: This adds a small, albeit negligible, amount of gas cost for these storage writes, which are effectively redundant.

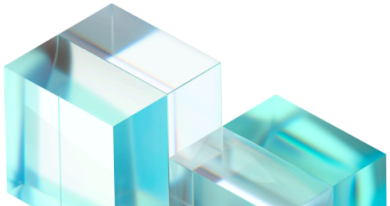
Suggested Mitigation: These lines can be removed to slightly optimize gas usage in the `createVestingSchedule` function without affecting functionality.

Flagship: Fixed in PR.

Linum Labs: Verified.

4. Revocation Mechanism Does Not Refund Payment Tokens.

Description: The `revoke` function, callable only by the contract owner, allows for the cancellation of a revocable vesting schedule. When a schedule is revoked, any unreleased `FYIToken` is transferred back to the `owner()` of the `Vesting` contract. The users pay a `paymentToken` in the `Presale.sol` contract to eventually receive `FYIToken` through this `Vesting` contract. However, the `revoke` function *only* returns `FYIToken` to the contract owner and provides no mechanism to refund the



`paymentToken` (or any equivalent value) to the beneficiary whose vesting schedule was revoked.

Potential Risk: This design creates a significant centralization risk and potential for substantial loss of funds for beneficiaries. If the owner revokes a schedule, beneficiaries lose their claim to future `FYIToken` releases, and crucially, they do not receive back the `paymentToken` they initially used to acquire the `FYIToken` through the presale. This effectively allows the contract owner to confiscate purchased tokens without compensation if a schedule is marked as revocable. This could lead to a complete loss of investment for users in revoked schedules, even if they had fully paid for the tokens.

Suggested Mitigation:

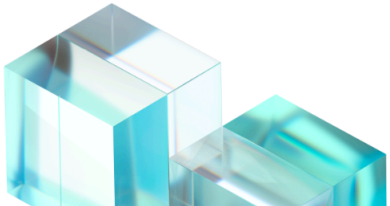
- Explicitly disclose to users that if a schedule is revocable and subsequently revoked, they will not be refunded their initial payment token and will lose claim to the unvested `FYIToken`.
- Refund the `paymentToken` (or a pro-rata equivalent) to the beneficiary for the `revokableAmount` of `FYIToken`. This would require the `Vesting` contract to either hold `paymentToken` or have a mechanism to exchange `FYIToken` back into `paymentToken`.

Flagship: For this, I think we can go with Option 1 and not amend the current logic. In case of any refunds, we will process these manually.

Linum Labs: Acknowledged.

Disclaimer

This report is based on the materials and documentation provided to Linum Labs Auditing for the purpose of conducting a security review, as outlined in the Executive



linumlabs.com
info@linumlabs.com
+1 (650) 534-2011

Summary and Files in Scope sections. It's important to note that the results presented in this report may not cover all vulnerabilities. Linum Labs Auditing provides this review and report on an as-is, where-is, and as-available basis. By accessing and/or using this report, along with associated services, products, protocols, platforms, content, and materials, you agree to do so at your own risk. Linum Labs Auditing disclaims any liability associated with this report, its content, and any related services and products, to the fullest extent permitted by law. This includes implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Linum Labs Auditing does not warrant, endorse, guarantee, or assume responsibility for any third-party products or services advertised or offered through this report, its content, or related services and products. Users should exercise caution and use their best judgment when engaging with third-party providers. It's important to clarify that this report, its content, access, and/or usage thereof, including associated services or materials, should not be considered or relied upon as financial, investment, tax, legal, regulatory, or any other form of advice.