DayHub FairLaunch Security Review

Duration: September 25, 2024 - September 27, 2024

October 8, 2024

Conducted by **KeySecurity**
Georgi Krastenov, Lead Security Researcher

Table of Contents

1 About KeySecurity 3

2 Introduction 3

3 About DayHub 3

4 Disclaimer 3

5 Risk classification 3

5.1 Impact . . . 3
5.2 Likelihood . . . 4
5.3 Actions required by severity level . . . 4

6 Executive summary 5

7 Findings 6

7.1 Medium . . . 6
7.1.1 Cancel contribution logic does not work properly . . . 6
7.2 Low . . . 6
7.2.1 Owner can scam all users . . . 6
7.2.2 DAY tokens can be stuck in the contract . . . 6
7.3 Information . . . 7
7.3.1 Use a custom error in the onlyOwner modifier . . . 7
7.3.2 Emit event in crucial places . . . 7
7.3.3 A wrong value is used in the TokensClaimed event . . . 7
7.4 Code Improvement . . . 8
7.4.1 Use uint256 instead of uint128 where applicable . . . 8

1 About KeySecurity

KeySecurity is an innovative Web3 security company that hires top talented security researchers for your project. We have conducted over 25 security reviews for different projects which hold over $300,000,000 in TVL. For security audit inquiries, you can reach out to us on Twitter/X or Telegram @gkrastenov, or check our previous work here.

2 Introduction

Users are allowed to contribute to the FairLaunch smart contract by depositing USDT tokens for a duration of 7 days, until the sale ends. After the soft cap is reached, 35% of the DAY token supply will be available for claiming. If the soft cap is not reached, users can withdraw their allocation.

3 About DayHub

Dayhub is the first ecosystem that revolutionizes funded trading using Blockchain Transparency and Fairness Dayhub leverages cutting-edge blockchain technology to ensure fair and transparent trading opportunities. Participate in custom challenges, access substantial trading capital, and experience a new era of decentralized funded trading.

4 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

5 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

5.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

5.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

6 Executive summary

Overview

Project Name	DayHub
Repository	https://github.com/Dayhub-io/fairlaunch-simulation
Commit hash	b27a95c51c06c20422375ecfd9c9c735b39f16c7
Review Commit hash	9702cb4eae094bde43e51963780223b10a12b9ba
Documentation	https://docs.dayhub.io/
Methods	Manual review

Scope

FairLaunch.sol

Timeline

September 25, 2024	Audit kick-off
September 27, 2024	Preliminary report
October 8, 2024	Mitigation review

Issues Found

Severity	**Count**
High	0
Medium	1
Low	2
Information	3
Code Improvement	1
Total	**10**

7 Findings

7.1 Medium

7.1.1 Cancel contribution logic does not work properly

Severity: *Medium*

Context: FairLaunch.sol#L85

Description: Currently, the `cancelContribution` function has a `requireSaleEnded` modifier, which means that the sale must be ended before the user is able to withdraw their USDT tokens from the contract. The correct logic should be the opposite: the user should only be able to cancel their contribution before the sale ends.

Recommendation: Add new modifier, `requireSaleActive` where if `block.timestamp >= saleEnd` will revert with `SaleNotActive`error and instead of `requireSaleEnded` use `requireSaleActive` for `cancelContribution` function.

Resolution and Client comment: Resolved. Fixed at `9702cb4eae094bde43e51963780223b10a12b9ba` commit.

7.2 Low

7.2.1 Owner can scam all users

Severity: *Low*

Context: FairLaunch.sol#L122

Description: Currently, the owner can withdraw all USDT tokens sent to the `FairLaunch` contract when the sale has ended. If the soft cap is not reached, the sale has failed, and the user should be able to claim their USDT allocation.

If the owner withdraws all USDT from the contract when the soft cap is not reached, the users should not be able to claim their USDT tokens and will be scammed out of their funds.

Recommendation: In the case where the soft cap is not reached, allow the owner to wait 3-4 days before being allowed to withdraw USDT tokens from the contract. If the soft cap is reached, allow him to withdraw tokens whenever he want.

Resolution and Client comment: Resolved. Fixed at `9702cb4eae094bde43e51963780223b10a12b9ba` commit.

7.2.2 DAY tokens can be stuck in the contract

Severity: *Low*

Context: Global

Description: In the case, when the sale is ended and the soft cap is not reached user should be able to withdraw only their USDT allocation and to not be able to claim `DAY` tokens.

Currently, the owner can withdraw only USDT tokens in the contract but can not withdraw `DAY` tokens. If the above scenario is happen(the soft cap to not be reached) all `DAY` tokens will be stuck in the contract. The `FairLaunch` contract is expected to hold 65% of the total `DAY` token supply and if the campaign of selling fails then all of this tokens will be forever frozen.

Recommendation: Allow the owner to withdraw `DAY` tokens from the contract when the sale has ended and the soft cap has not been reached.

Resolution and Client comment: Resolved. Fixed at `9702cb4eae094bde43e51963780223b10a12b9ba` commit.

7.3 Information

7.3.1 Use a custom error in the onlyOwner modifier

Severity: *Information*

Context: FairLaunch.sol#L60

Description: In the onlyOwner modifier, instead of using a direct revert, you can use a custom error. It is recommended to follow the same approach (as in other modifiers) throughout the codebase.

Recommendation: Use a custom error instead of a direct revert.

Resolution and Client comment: Resolved. Fixed at `9702cb4eae094bde43e51963780223b10a12b9ba` commit.

7.3.2 Emit event in crucial places

Severity: *Information*

Context: FairLaunch.sol#L85

Description: Emit an event in crucial places, such as in the cancelContribution() function, when the user withdraws their USDT tokens and the total amount is decreased.

Recommendation: Emit event in `cancelContribution` function

Resolution and Client comment: Resolved. Fixed at `9702cb4eae094bde43e51963780223b10a12b9ba` commit.

7.3.3 A wrong value is used in the TokensClaimed event

Severity: *Information*

Context: FairLaunch.sol#L118

Description: A wrong value is used in the `TokensClaimed` event when the user gets their DAY tokens. Instead of using the `_amount` variable, which holds the deposited USDT tokens, use the `tokenAmount` variable, which indicates how many DAY tokens will be claimed.

Recommendation: Use `tokenAmount` instead of `_amount`.

Resolution and Client comment: Acknowledged.

7.4 Code Improvement

7.4.1 Use uint256 instead of uint128 where applicable

Severity: *Code Improvement*

Context: Global

Description: Use `uint256` instead of `uint128` where applicable. This will avoid unnecessary casting to `uint256` when the variable in the `allocations` mapping is changed, as well as in events.

Also, using `uint128` will not be more gas efficient because only one slot is used in storage for storing the `totalAmount`. The constant and immutable variables will be stored directly in the bytecode of the contract.

Recommendation: Use `uint256` instead of `uint128` where applicable.

Resolution and Client comment: Acknowledged.