yHair Vesting Security Review

Pashov Audit Group

Conducted by: pashov

January 4th, 2024

Contents

1. About pashov

Krum Pashov, or **pashov**, is an independent smart contract security researcher. Having found numerous security vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by putting time and effort into security research & reviews. Check his previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **token-vesting-contracts** was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About yHair Vesting

The protocol is a fork of the Molecule vesting protocol. It is an on-chain token vesting implementation with cliff and vesting periods. On top of the forked implementation, a functionality for vesting schedule purchasing as well as a cost for claiming tokens has been added.

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 material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

review commit hash **-** **39635f44b219d9fe95649b8208a98f86d8dd7ebd**

fixes review commit hash **-** **f14edf844e443ea227b70f20d4dc4f7da888052b**

7. Executive Summary

Over the course of the security review, pashov engaged with yHair Vesting to review yHair Vesting. In this period of time a total of **8** issues were uncovered.

Protocol Summary

Protocol Name	yHair Vesting
Date	January 4th, 2024

Findings Count

Severity	**Amount**
High	1
Low	7
Total Findings	**8**

Summary of Findings

ID	Title	Severity	Status
[H-01]	createVestingSchedule can be front-ran by another holder of ROLE_CREATE_SCHEDULE role	High	Resolved
[L-01]	Flawed access control	Low	Resolved
[L-02]	Pausing not implemented correctly	Low	Partially Resolved
[L-03]	Possible DoS in getVestingSchedulesIds	Low	Resolved
[L-04]	Missing input validation in token price setters	Low	Resolved
[L-05]	Using the transfer function of address payable is discouraged	Low	Resolved
[L-06]	Protocol is using a vulnerable library version	Low	Resolved
[L-07]	Missing INVALID value in enum	Low	Resolved

8. Findings

8.1. High Findings

[H-01] `createVestingSchedule` can be front-ran by another holder of `ROLE_CREATE_SCHEDULE` role

Severity

Impact: High, as vesting token balance can be stolen

Likelihood: Medium, as it requires front-running

Description

The `createVestingSchedule` method of `TokenVestingV2` expects to have a pre-transferred balance before initializing a vesting schedule. The problem with the current contract version is that multiple accounts can hold the `ROLE_CREATE_SCHEDULE` role. Since two transactions are expected to create a vesting schedule (transferring funds to the `TokenVestingV2` contract and then calling `createVestingSchedule`) this means that between them another holder of the role can come in and create a vesting schedule of his own (with himself as beneficiary for example, non-revokable with just 7 days of duration) and in this way steal the funds of the other role holder.

Recommendations

Either change `createVestingSchedule` to itself transfer the vesting schedule tokens from the caller to the contract or make it callable by just 1 address

8.2. Low Findings

[L-01] Flawed access control

The `setVTokenCost` and `setTokenCost` methods are callable only by `ROLE_CREATE_SCHEDULE` role holder. The methods decide the cost of purchasing vesting schedules. This should be a function of the `DEFAULT_ADMIN_ROLE` instead, since a role that creates schedules shouldn't decide their pricing.

[L-02] Pausing not implemented correctly

Currently the `setPaused` method of `TokenVestingV2` says the following in its NatSpec - "Pauses or unpauses the release of tokens and claiming of schedules". The problem is that no method in the contract has the `whenNotPaused` or `whenPaused` modifiers, so the comment is wrong. Remove the `setPaused` method altogether.

Discussion

Pashov Audit Group: the client put a `whenNotPaused` modifier on `createVestingSchedule` and `purchaseVSchedule` methods instead as a fix, which brings some centralization to schedule purchasing - the owner can block users from doing so.

[L-03] Possible DoS in `getVestingSchedulesIds`

The `getVestingSchedulesIds` method copies the whole `vestingSchedulesIds` array to memory, which due to memory expansion costs can cost a huge amount of gas. Pushing to the array is unbounded, so if it gets too big then the gas needed for the method call can be more than the block gas limit or just too expensive to execute. Make sure to limit the size of the array so that such error can't happen.

[L-04] Missing input validation in token price setters

The `setVTokenCost` and `setTokenCost` methods are missing lower and upper bounds, meaning the caller of them can set for example huge values so tokens are not actually purchasable. Make sure to put a sensible upper bound and possibly a lower bound on the values that you can set in those methods.

[L-05] Using the `transfer` function of `address payable` is discouraged

The `release`, `releaseAvailableTokensForHolder` and `purchaseVSchedule` methods in `TokenVestingV2` use the `transfer` method of `address payable` to transfer native asset funds to an address. If the `paymentReceiver` address is a smart contract that has a `receive` or `fallback` function that takes up more than the 2300 gas (which is the limit of `transfer`), then the methods will revert every time until the `paymentReceiver` is changed. Examples are some smart contract wallets or multi-sig wallets, so usage of `transfer` is discouraged. To fix this, use a `call` with value instead of `transfer`. There is also no reentrancy risk as the three methods all use the `nonReentrant` modifier.

[L-06] Protocol is using a vulnerable library version

In `.gitmodules` file in the repository we can see this:

```
url = https://github.com/openzeppelin/openzeppelin-contracts
branch = release-v4.8
```

This version contains multiple vulnerabilities as you can see here. While the problems are not present in the current codebase, it is strongly advised to upgrade the version to v4.9.5 which has fixes for all of the vulnerabilities found so far after v4.8.

[L-07] Missing `INVALID` value in enum

The current default value of the `Status` enum is `INITIALIZED`. This is error-prone as even for a vesting schedule that doesn't exist, when the schedule is a value in a mapping it has default values and its `Status` will be `INITIALIZED`. Make sure to add `INVALID` as a Status value with index 0 (the default one) to protect from subtle errors with default values.