Molecule Vesting Security Review

Pashov Audit Group

Conducted by: pashov

April 18th, 2023

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 **Molecule Vesting** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Molecule Vesting

The protocol represents an on-chain vesting scheme. It uses a typical vesting scheme that includes a cliff and vesting period. Multiple vesting schedules per user are allowed, while non-vested token amounts are revokable by a centralized entity (the contracts' owner). The vesting schedules are represented by a non-transferable ERC20 balance for users, which can also be utilized for governance purposes.

More docs here.

Observations

Only 18 decimal tokens are allowed.

The contract uses a `nonReentrant` modifier for many of its methods so it is protected against ERC777-type reentrancy attacks.

The contract does not use `transferFrom` to receive tokens to vest, but expects tokens will be directly transferred to it.

When admin revokes a vesting schedule, all of the already vested tokens are transferred to the user.

Threat Model

Privileged Roles & Actors

- Contract Owner - can create vesting schedules, revoke them, pause the contract, withdraw excessive balance and also release vested tokens to users
- Vesting user - can claim his vested tokens from the contract

Security Interview

Q: What in the protocol has value in the market?

A: The contract's balance in terms of the tokens that will be vested.

Q: In what case can the protocol/users lose money?

A: If the vesting calculations are incorrect or if the contract gets into a state of DoS.

Q: What are some ways that an attacker achieves his goals?

A: By making other user's claim transactions always revert or by exploiting a calculations error in the vested tokens math.

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 **- 88c1cda0e77d881fffb8a9deab1f65585503d504**

Scope

The following smart contracts were in scope of the audit:

- `TokenVesting`
- `TokenVestingMerkle`

7. Executive Summary

Over the course of the security review, pashov engaged with Molecule Vesting to review Molecule Vesting. In this period of time a total of **6** issues were uncovered.

Protocol Summary

Protocol Name	Molecule Vesting
Date	April 18th, 2023

Findings Count

Severity	**Amount**
Medium	4
Low	2
Total Findings	**6**

Summary of Findings

ID	Title	Severity	Status
[M-01]	The revoke mechanics are not compatible with tokens that implement a block list feature	Medium	Resolved
[M-02]	Insufficient input validation in function createVestingSchedule	Medium	Resolved
[M-03]	Contract can receive ETH but has no withdraw function for it	Medium	Resolved
[M-04]	Users won't be able to claim vested tokens when contract is paused	Medium	Resolved
[L-01]	Limit the max size of the vestingSchedulesIds array and holdersVestingScheduleCount	Low	Resolved
[L-02]	The onlyIfVestingScheduleNotRevoked modifier will not revert even if the given vestingScheduleId is non-existent	Low	Resolved

8. Findings

8.1. Medium Findings

[M-01] The `revoke` mechanics are not compatible with tokens that implement a block list feature

Severity

Impact: High, as important functionality in the protocol won't work

Likelihood: Low, as a special type of ERC20 token has to be used as well as the attacker's address has to be in a block list

Description

Some tokens, for example `USDC` and `USDT` implement an admin controlled address block list. All transfers to a blocked address will revert. Since the `revoke` functionality forcefully transfers the claimable vested tokens to an address with a `vestingSchedule`, all calls to `revoke` will revert if such an address has claimable balance and is in the token's block list.

Recommendations

Use the Pull over Push pattern to send tokens out of the contract in a `revoke` scenario.

Discussion

pashov: Acknowledged, as the team listed in the protocol docs that such tokens won't be supported.

[M-02] Insufficient input validation in function `createVestingSchedule`

Severity

Impact: High, as it can lead to users never vesting their tokens

Likelihood: Low, as it requires a malicious/compromised admin or an error on his side

Description

The input arguments of the `createVestingSchedule` function are not sufficiently validated. Here are some problematic scenarios:

1. `_start` can be a timestamp that has already passed or is too far away in the future
2. `_cliff` can be too big, users won't be able to claim
3. 1 is a valid value for `duration`, the `!= 0` check is insufficient
4. If `_slicePeriodSeconds` is too big then the math in `_computeReleasableAmount` will have rounding errors

Recommendations

Add sensible lower and upper bounds for all arguments of the `createVestingSchedule` method.

Discussion

pashov: Resolved in 8f8f786d95ee8db1e3d3ae96e26a86b7e250de0f.

[M-03] Contract can receive ETH but has no withdraw function for it

Severity

Impact: High, as value can be stuck forever

Likelihood: Low, as it should be an error that someone sends ETH to the contract

Description

The `TokenVesting` contract has `receive` and `fallback` functions that are `payable`. If someone sends a transaction with `msg.value != 0` then the ETH will be stuck in the contract forever without a way for anyone to withdraw it.

Recommendations

Remove the `receive` and `fallback` functions since the ETH balance is not used in the contract anyway.

Discussion

pashov: Resolved in 8f8f786d95ee8db1e3d3ae96e26a86b7e250de0f.

[M-04] Users won't be able to claim vested tokens when contract is paused

Severity

Impact: High, as owner has the power to make it so that users can't claim any vested tokens

Likelihood: Low, as it requires a malicious or a compromised owner

Description

The owner can currently execute the following attack:

1. Call `setPaused` with `paused == true`, so pause the contract
2. Now all user calls to `releaseAvailableTokensForHolder` will fail, since it has the `whenNotPaused` modifier
3. He can not unpause the contract forever or even renounce ownership

This is a common centralization problem which means the contract owner can "rug" users.

Recommendations

Remove the `whenNotPaused` modifier from `releaseAvailableTokensForHolder`, so users can claim vested tokens even if admin pauses the contract.

Discussion

pashov: Resolved in 8f8f786d95ee8db1e3d3ae96e26a86b7e250de0f.

8.2. Low Findings

[L-01] Limit the max size of the `vestingSchedulesIds` array and `holdersVestingScheduleCount`

If too many vesting schedules are added for a user it is possible that the `getVestingSchedulesIds` method will take too much gas and won't be executable (if it gets over the block gas limit, for example). Also in `releaseAvailableTokensForHolder` there is a `for` loop that loops `vestingScheduleCount` number of times, which can also be problematic, as it can lead to a DoS state with the function. Limit the max size of both, for example up to 500 vesting schedules created from the contract.

Discussion

pashov: Resolved in 8f8f786d95ee8db1e3d3ae96e26a86b7e250de0f.

[L-02] The `onlyIfVestingScheduleNotRevoked` modifier will not revert even if the given `vestingScheduleId` is non-existent

The modifier will pass successfully when the `vestingScheduleId` passed is of a non-existent vesting schedule, because the default `Status` of a vesting schedule is `INITIALIZED` anyway. Validate that the `vestingSchedules` exists, by checking that `vestingSchedules[vestingScheduleId].duration != 0`.

Discussion

pashov: Resolved in 8f8f786d95ee8db1e3d3ae96e26a86b7e250de0f.