CoinFabrik

1Inch Audit

StepVesting

May 2021

By CoinFabrik

Introduction

CoinFabrik was asked to audit modifications on the contracts for 1Inch's Governance project, and in particular the StepVesting contract. This is a report of CoinFabrik's fundings. It includes a summary of findings, what is their current status, and then details on these findings.

Summary

We analyzed the StepVesting contracts from the 1Inch repository:

https://github.com/1inch/governance-contracts

This audit is based on the following pull:

https://github.com/1inch/governance-contracts/pull/1

The step vesting contract allows an owner to define, by using the contract's constructor, a receiver and vesting parameters. Tokens are vested periodically, according to the timestamp published in new blocks, to this receiver until a given moment.

Once deployed, the contract is started (i.e. `started = block.timestamp`), it allows the receiver to claim `stepAmount` tokens every time the `block.timestamp` increases in `stepDuration`. At any time, the receiver can call the `claim()` function and get all the tokens available transferred to his/her account.

The code is simple and clear. No documentation was made available. There are public functions that can be executed by anyone, which are view functions and cannot introduce threats.

Special Considerations

The functions kill(), setReceiver() and claim() can only be called by the owner or the receiver. There are two caveats worth mentioning related to these functions.

1. The function

   ```
   setReceiver(address _receiver)
   ```

   allows the owner to change the receiver at any time at his discretion. After this happens, the new receiver can claim whatever tokens were made available and the old receiver loses the option to claim tokens, even if they were available before `setReceiver()` was called.

2. The function

    ```
    kill(address target)
    ```

    allows the owner, at his/her discretion, to immediately stop the vesting and have all the unclaimed funds transferred to an address of his choice (`target`).

In these two cases, the receiver loses the chance to claim all the tokens which had already vested. This is a relevant fact which should be clear to users.

Contracts

The audited contracts are:

- contracts/StepVesting.sol

Analyses

The following analyses were performed:

- Misuse of the different call methods
- Integer overflow errors
- Division by zero errors
- Outdated version of Solidity compiler
- Front running attacks
- Reentrancy attacks
- Misuse of block timestamps
- Softlock denial of service attacks
- Functions with excessive gas cost
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Failure to use a withdrawal pattern
- Insufficient validation of the input parameters

- Incorrect handling of cryptographic signatures

Issues Found

No issues found.

Conclusion

The StepVesting contract is concise and well written. It includes a small number of functions which are public, but declared view functions, and only three functions may change the state but may be called by the owner or the viewer. Lacking documentation, these functions warrant special consideration since a receiver could lose the possibility to claim, through the contract, tokens which had been available for claiming. No security issues were found.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the Governance project (StepVesting contract) since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.