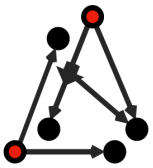
FundRequest: Vesting Wallet Smart Contracts



Least Authority
Freedom Matters

Updated Security Audit Report

Overview

FundRequest has requested Least Authority perform a security audit of the FundRequest Vesting Wallet contracts. The audit was performed the weeks of March 5 and 12, 2018.

Target Code and Revision

For this audit, we reviewed the following repositories:

- https://github.com/FundRequest/vesting-wallets/tree/master/contracts
- Third party vendor code is considered out of scope.

Specifically, we examined the Git revisions:

```
4ae4191a32f83017eb0502b5b333b7ec44f88046
```

For verification, we examined the Git revisions:

```
216713cf257a006501887f7ef239fdc80bb6ec46
```

All file references in this document use Unix-style paths relative to the project's root directory.

Code Review

In manually reviewing all of the contract code, we looked for any potential issues with code logic, error handling, and interaction with contracts that are dependencies. We also considered areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus was on the contract code, we examined some dependency code and behavior when it was relevant to a particular line of investigation.

Findings

The FundRequest Vesting Wallet contract is structured clearly and functions as designed. The contract is short and simple and is paired with adequate test coverage. Given the simplicity of the contract, the fact that it does not make calls to other contracts, and it's simplistic nature, we have no critical security recommendations at this time. However, we did

find some areas where more defensive programming techniques should be used to prevent usage mistakes that could lead to loss of funds.

Issues

We list the issues we found in the code in the order we reported them.

Issue / Suggestion	Status
Issue A: No percentage validation in `VestingWallet#registerVestingScheduleWithPercentage`	Reported: 14.03.2018
Issue B: `VestingWallet#registerVestingSchedule` assumes an implied follow up transaction	Reported: 14.03.2018
Issue C: Recovering from accidental over deposit	Reported: 14.03.2018

Issue A: No percentage validation in `VestingWallet#registerVestingScheduleWithPercentage`

Reported: 14.03.2018

Synopsis:
https://github.com/FundRequest/vesting-wallets/blob/4ae4191a32f83017eb0502b5b333b7ec44f88046/contracts/vesting/VestingWallet.sol#L85-L103

There are no guards to prevent the `_percentage` argument from exceeding 100%. A mistake or typo by the contract owner will not be caught, and will allocate more tokens than intended. As a result, the recipient of the created schedule will be able to withdraw more tokens than intended or allotted. Other vesting recipients may be unable to withdraw tokens as a result.

Impact: Potential loss of funds

Feasibility: High, can occur with simple typo.

Mitigation: Add `require(_percentage <= 100);` between lines 98 and 99 in `VestingWallet#registerVestingScheduleWithPercentage`

Verification: Verified on contracts deployed in Remix's JS EVM.

Issue B: `VestingWallet#registerVestingSchedule` assumes an implied follow up transaction

Reported: 14.03.2018

Synopsis:
https://github.com/FundRequest/vesting-wallets/blob/4ae4191a32f83017eb0502b5b333b7ec44f88046/contracts/vesting/VestingWallet.sol#L105-L142

`VestingWallet#registerVestingSchedule` registers the vesting schedule and assumes the `_depositor` will transfer tokens to vest. If this second implied transaction does not occur or it transfers fewer tokens than the vesting schedule defines, some token recipients will be unable to withdraw funds.

Impact: Potential loss of funds

Feasibility: High, can occur due to poor internal communication.

Mitigation: Redesign this into an approval flow where a vesting schedule does not become valid/active until a corresponding deposit is made (or there already exists an appropriate deposit).

Verification: Verified. Note that registering a schedule now transfers the tokens automatically, and tokens that are accidentally transferred to the contract cannot be recovered by registering an unfunded schedule.

Issue C: Recovering from accidental over deposit

Reported: 14.03.2018

Synopsis: When registering a vesting schedule and subsequently performing a corresponding deposit, funds that are deposited over the sum of all existing vesting schedules become inaccessible. Recovering those funds requires a new vesting schedule for the extraneous amount to be created then calling `VestingWallet#endVesting` to recover the funds.

Impact: Temporary locking of funds with reconciliation workflow possibly leading to additional user input mistakes.

Feasibility: High, can occur with simple typo.

Mitigation: Implement a method that explicitly balances the deposit amount to match the sum of all vesting schedules, refunding the remainder to the owner of the vesting wallet in order to reduce the complexity of recovering from over-deposit and mitigate further mistakes.

Verification: Partially mitigated by the new `approve/transferFrom` flow. Over depositing is now unlikely, because manual deposits are no longer necessary.