CLIENT

PROJECT

Decentralized Vision

Smart Contract Audit

DATE

01.07.2020

Version 0.3

ELECTI

Table of Contents

Introduction

Electi Consulting Ltd was assigned to perform a smart contract audit against a recently developed smart contract called **multi-vesting**. The source code and documentation of the smart contract can be found under the *pumapayio* official Github [1]repository.

The commit for which the smart contract has been audited is 8d8b61b[2].

Objectives

The objectives of this work are to:

- Find potential flaws or vulnerabilities that might result to either exploitation by malicious users.
- Find potential bugs that might cause undefined behavior.
- Verify the behavior of the source code against the business logic.
- Suggest potential fixes or improvements based on the findings.

Update

The smart contract has been revisited[3] after the issues identified in the first audit have been addressed.

[1] https://github.com/pumapayio/multi-vesting

[2] https://github.com/pumapayio/multi-vesting/commit/8d8b61b647ac120031f714ad5b057cc5593df3ea

[3] https://github.com/pumapayio/multi-vesting/commit/342b28c26012e6ed689b3af20bced1077151e3b2

Executive Summary

In this code review we have identified one high, one low and one informational issue. The most important issue is the one which may allow a vesting to start in the future given a bad input. Given that the function in question can be only executed by the contract owner, it reduces the attack surface but still does not prevent the owner from inserting a bad input.

Overall, the smart contract is adequately tested and the functionality that it provides fulfills the business logic.

Update

The high severity issue has been addressed, as advised in the initial audit, by checking the upper bound of the date inserted not be bigger than 180 days from the current date.

Findings

In this section, we will list all the findings identified, the best of our knowledge, for the smart contract under audit and suggest potential fixes or improvements.

Description

MutliVesting.sol

The purpose of this contract is to perform a standard type of vesting with fixed deadlines and amounts. Specifically, upon creation of a vesting object, the beneficiary can expect to receive 4% of the amount in 25 steps (each step corresponds to 1 months of 30 days). The beneficiary can have multiple vestings assigned and can claim the amount either per vesting or aggregated.

Start date not sanitized (fixed)

The starting date of the vesting is only checked against the inbuilt functionality of solidity which provides the block timestamp. In case of a bad input date (e.g. 10 years from now), the vesting start date could be in the very far future. There is no functionality which can "correct" the start date. In such a case the impact would be irreversible. A potential check against an upper bound of the start date could be put in place.

Method	Description	Remedy	Impact
addVesting	Start date is only checked against the lower bound (current time) and not against the future.	Place a check against an upper bound which complies with the business logic.	High - The start date could potentially land in the future.

```
function addVesting(address _beneficiary, uint256 _amount, uint256 _startedAt) public onlyOwner {
    require(_startedAt >= now, "TIMESTAMP_CANNOT_BE_IN_THE_PAST");
    require(_amount >= STEPS_AMOUNT, "VESTING_AMOUNT_TO_LOW");
```

Remedy

Commit 342b28c [4] fixes the issue by checking the value of **_startedAt** to be not bigger than 180 days after the current date.

```
57  +         require(_startedAt <= (now + 180 days), "TIMESTAMP_CANNOT_BE_MORE_THAN_A_180_DAYS_IN_FUTURE");
```

[4] https://github.com/pumapayio/multi-vesting/commit/342b28c26012e6ed689b3af20bced1077151e3b2

Uneven withdrawal of funds

The calculation of the available amount at a given point is calculated as follows: Given the total amount, as allocated in the Vesting structure, the reward per month is calculated by diving the total amount by the number of steps (25). If the total amount is not a multiple of 25 then there is a remainder of tokens which is transferred only during the last installment. For example, if the total amount is 124, then the reward per month is 4.96 (124/25) but since we are doing integer division the quotient is 4 and the remainder 24. 24 out of 25 installments will be 4 tokens but the last one will be 28.

A realistic scenario would be the following:

If the amount to be vested, even with 18 decimals, cannot be divided with 25 then the first 24 withdrawals will not be the same as the last one (25th) as the last one will include the remainder of the division.

Method	Description	Remedy	Impact
getAvailableAmountAtTimestamp	Calculation of the reward per month does not account the remainder of the division. The remainder of the tokens are only transferred in the last installment.	N\A	Low – Beneficiaries have to wait for their last installment.

```
uint256 rewardPerMonth = vesting.totalAmount.div(STEPS_AMOUNT);

// 25 Month (%4 per month)
uint256 monthPassed = _timestamp
    .sub(vesting.startedAt)
    .div(30 days); // We say that 1 month is always 30 days

uint256 alreadyReleased = vesting.releasedAmount;

// In 25 month 100% of tokens is already released:
if (monthPassed >= STEPS_AMOUNT) {
    monthPassed = STEPS_AMOUNT;
    return vestingMap[_beneficiary][_vestingId].totalAmount.sub(alreadyReleased);
}

return rewardPerMonth.mul(monthPassed).sub(alreadyReleased);
```

Invalid assignment (fixed)

Line 178 assigns to the local variable *monthPassed* the value *STEPS_AMOUNT* which is then never used as the routine returns a value that does not depend on it.

Method	Description	Remedy	Impact
getAvailableAmountAtTimestamp	Invalid assignment	Remove line 178	Informational.

```
176         // In 25 month 100% of tokens is already released:
177         if (monthPassed >= STEPS_AMOUNT) {
178             monthPassed = STEPS_AMOUNT;
179             return vestingMap[_beneficiary][_vestingId].totalAmount.sub(alreadyReleased);
180         }
```

Remedy

Commit 509f9ec[5] addresses the issue by removing line 178.

[5] https://github.com/pumapayio/multi-vesting/commit/509f9ec4c5351a821e7ec399f0e7e87e4c0a1dbc

contact

ai@electiconsulting.com

visit

www.electiconsulting.com