Tend ICO Audit - Revision

by MonteLabs

January 17, 2018

1 Introduction

The Tend ICO smart contracts were modified since our first audit, therefore this document reports our findings regarding the changes applied to the smart contracts.

The most significant changes are in the `IcoCrowdsale` smart contract which is the focus of this audit.

The code was written by Validity Labs, and is hosted at `github.com/validitylabs/ico-tend`. The verified version is commit d0a2b8f6e73f1a318afb36a4004d10fc8b21f1d0 from January 12, 2018.

2 Issues

2.1 Low severity

Function `mintDevelopmentTeamTokens` does not update storage variable `developmentTeamTokensMinted`.

In the same way that function `mintIcoEnablersTokens` updates variable `icoEnablersTokensMinted`, function `mintDevelopmentTeamTokens` should update variable `developmentTeamTokensMinted`, otherwise the amount minted may be greater than `DEVELOPMENT_TEAM_CAP` over time.

Function `buyTokens` does not check `ICO_TOKEN_CAP`.

The suggested solution is to add the following line after line 172, similar to an existing one from function `mintTokenPreSale`:

```
require(tokensToMint.add(tokenAmount) <= ICO_TOKEN_CAP);
```

Discount may be given to more tokens than planned.

There is a comment already in the code stating that a different reasoning would be necessary to achieve the exact desired discount distribution. In the current version, a group of people could collude and buy more than 3 million tokens at once, and would receive the high discount for the entire buy.

The given discount is higher than the specified.

The 20% discount, for example, is given by increasing the amount of bought tokens in the following way: `tokenAmount = tokenAmount.mul(10).div(8);`. By doing this, the actual increase is 25% (10/8 = 1.25) instead of 20%. The suggested solution is to multiply by 12 and divide by 10. The 10% discount case follows analogously.

`rateTokenPerChf` is misleading.

In `Ico.cnf.json`, `rateTokenPerChf` is set to 1. Since the constructor parameter `_rateTokenPerChf` is of type `uint` in the smart contract, it has to be an integer. The specification states that the standard rate is 0.1 (1 token = 10 CHF). To correct the ratio `rateWeiPerChf` could be adapted accordingly. Side effects of this approach include adjusting also the `MIN_CONTRIBUTION_CHF` constant.

2.2 Minor suggestions

- Storage var `tokensBoughtWithEther` declared but never used.
- Variables `_companyAddress` and `_teamAddress` in `IcoCrowdsale` constructor are never used.

3 Conclusion

We have found few low severity issues in the revised code that can be quickly fixed by the developers.