PeerityCoin Token Contract: Final Review

Mikhail Vladimirov and Dmitry Khovratovich

10th October, 2017

We were asked to audit the first draft of Peerity Token Contracts. We have found several issues, from minor to moderate, and described them in a separate report. The report was communicated privately to Peerity. Based on our comments, Peerity developers fixed most of the issues we mentioned, whereas the others were deemed non-critical and left untouched. We reviewed several intermediate versions of the contracts until the last one available to us.

We certify that to the best of our knowledge the code does not contain any major flaws that would prevent a secure and proper interaction with this contract.

The other issues are mentioned below.

1. Introduction

We were asked to review a set of contracts:

- PeerityCoin Token Contract (the contract is contained in PeerityToken.sol).
- SafeMath Token Contract (the contract is contained in SafeMath.sol).
- Standard Token Contract (the contract is contained in StandardToken.sol).

We tried to match the code with the Readme file.

2. Peerity Token

In this section we describe issues still relevant to the token contract defined in PeerityToken.sol.

Most of this contract is about selling tokens, and this code will stay in the contract even after sale will be finalized, making token contract complicated and token transactions harder to mine. Probably it would be better to move sale logic into separate smart contract that will transfer tokens to buyers.
The sale logic has been improved significantly since our first review and mainly matches with the readme file with the sole exclusion that `ETH_SOFT_CAP` variable is not binded to the

second phase even though it is highly unlikely this bound can be matched in the second phase. The funding phases `One, Two, Finalized, Redeeming` and contract states `Running, Paused, Emergency` look consistent with the sale procedure though we did not apply any automata analysis to make this claim rigorous.

3. SafeMath Token

In this section we describe issues related to the token contract defined in SafeMath.sol.

3.1 Arithmetic Overflow Issues

This section lists issues of token smart contract related to arithmetic overflows.

1. The inequality checks (lines 6, 10, 14) rely on undocumented behavior of overflow in Solidity. We recommend to prevent overflow rather than to detect it afterwards.

4. Standard Token

In this section we describe issues related to the token contract defined in StandardToken.sol.

4.1 Documentation Issues

This section lists documentation issues found in the token smart contract.

1. Mapping in line 31 has two keys and their meaning is not clear without documentation.

4.2 Arithmetic Overflow Issues

The Peerity token behaves wierdly (check will signal and overflow and `transfer` method will return false) when user has N tokens and wants to transfer M tokens to himself if M <= N and M + N >= 2^{256}. The Peerity authors communicated to us that this behaviour is part of business requirements and are not of concern as such balances can never be obtained.

4.3 Other Issues

This section lists stylistic and other minor issues found in the token smart contract.

1. The “short address” protection delivered by `onlyPayloadSize` modifier looks redundant. There are too many ways to call a smart contract incorrectly (for example, confusing the order of parameters), of which the “short address” issue is the most known and thus usually fixed. Moreover, it is a dangerous practice: if another function from the same contract calls one equipped with such check will fail.

5. Security provisions

We recommend the smart contract designers to claim explicitly security provisions in addition to the intended functionality of the contract. Such provisions should be non-trivial falsifiable claims, i.e. they should declare certain property of the contract for which an attack can be demonstrated if implemented incorrectly. The security provisions serve not only for the purpose of confidence of future users in the contract's behaviour, but also as a platform for a potential bug bounty program. The contract authors are encouraged to offer various bounties binded to some provision being violated.

Here we provide a list of security provisions which are appropriate for these contracts and would be expected by ordinary users.

Name	**Description**	**Current status**
	Peerity Token	
KnownRate	It is always possible to figure out how many tokens will be delivered for the supplied Ether	TRUE
RefundOrBuy	Refunds are always provided if the minimum cap is not reached	TRUE
RefundOrTransfer	If refunds are not allowed token transfers will be eventually allowed	TRUE
EtherRetrieval	Deposited Ether can be withdrawn entirely by the contract owner	TRUE