By Sergio Demian Lerner

1. Executive Summary

In August 2017, REAL engaged Coinfabrik to perform a security audit of the REAL crowdsale smart contracts. The purpose of the audit was to evaluate the security of the smart contracts. The source code was retrieved by Coinfabrik from https://github.com/RealEstateAssetLedger/real_contract commit `e8af021445785ad0e36c54adf6568b079ec6293d` on August 11th 2017. During the assessment, we identified no high-risk issues, one medium risk issue, and one low risk issue.

2. Introduction

The REAL smart contracts implement the REAL crowdsale, the withdrawal limited developer and team wallets and a clonable ERC20 token to support trading. A whitebox security audit was conducted to detect Solidity vulnerabilities, cryptographic flaws, and logical errors. To detect logical errors, we compared the code with the intended behavior as described in code comments. The present report, completed on August 22th, presents the results of the audit.

3. Summary Of Findings

Two issues were found as described in the following sections.

3.1. Flaw in Spam Prevention (medium risk)

The REAL crowdsale uses an anti-spam system to prevent contracts from buying REAL tokens during the crowdsale several times in a row. This is to prevent contracts to “game the system”. A single smart contract could cheat by placing a high number of buy orders for third parties, until the crowdsale cap is reached, preventing other users from having a fair chance to buy. This has the additional benefit for the cheater that he pays only 3000 gas units per buy (internal transaction), instead of the normal 21K gas of an external transaction.
The REAL crowdsale imposes a limit on the buy frequency from a single source to deter the cheat. A single source account must wait 100 blocks (about 30 minutes) to be able to buy again. However, a single smart contract can call a high number of

proxy smart contracts to buy tokens in behalf of third parties using a single external transaction to trigger the batch buy. This requires pre-creating the proxy contracts, with 31K gas cost per proxy contract, which is close to the cost of an external transaction, and the investment can be amortized by re-using the proxy contracts for several crowdsales. Because of this, the REAL crowdsale tries to prevent smart contracts from buying tokens. REALCrowdsale buyNormal() performs this check by reading the buyer address associated code size using EXTCODESIZE. If the code size is non-zero, then it is supposed to mean that the caller is a smart contract, and not a ECDSA-controlled account. But this is not always true.
During initialization, any Ethereum smart-contract has empty code, and it executes a temporary construction code that returns the actual code that must be installed. This temporary code is hidden from other contracts. But during initialization this temporary code can call other contracts, and therefore it can also buy REAL tokens. Therefore, the REALCrowdsale contract does not prevent smart contracts from buying tokens. In general, EXTCODESIZE can be used to prevent interacting with accounts but it cannot be used to prevent selling tokens to a smart-contract. A cheater can dynamically create thousands of contracts that buy REAL tokens on behalf of third parties, and trigger the batch buy by a single external transaction. Creating a new contract for each token buy costs 31K gas each, but about 19K can be refunded using the SUICIDE opcode, so the net cost for each buy can be 12K gas, almost half of the cost of an external transaction.

There are three possible on-chain methods to distinguish an account from a contract:

a. require in a crowdsale Buy() method a signature of a message using the sender's account ECDSA private key.
b. split the crowdsale (or each buy order) into two phases, each phase must be performed in a different block. In the first phase, the order is stored in a temporary buffer, and the caller address is stored along each order. In the second phase, each caller must send a confirmation message to the crowdsale contract from the same caller address. This second time the caller is checked not to be a smart contract using EXTCODESIZE. Since the creation of a smart contract can only occur in a single block, a contract that is being initialized cannot pass the second stage check.
c. On each buy, check that the origin of each transaction is equal to the sender (using the ORIGIN and CALLER opcodes; tx.origin and msg.sender fields in Solidity).

The (a) and (b) methods complicate significantly buying REAL tokens. The third (c) approach is simple and effective. It must be noted that the use of the ORIGIN opcode is not recommended past the Serenity hard-fork, so the REAL crowdsale should be performed before this event.

3.2. MiniMeToken contract outdated (low risk)

The MiniMeToken in the REAL repository does not correspond to the last published MiniMeToken. Although no bug was found in the audited version, the last version of the MiniMeToken contract uses the "require" method instead of throws, which makes the code cleaner.

Concluding Remarks

We were hired to check the security of the REAL crowdsale contracts. We reviewed them and concluded they implement correctly the functions according to the requirements specified in the source code comments. We can also attest that the source code is good quality and well commented.

Note: our security audit is limited to smart contract code. We are not auditing the technologies were this smart contract are based neither the general operational security. This document should not be read as investment advice or an offering of tokens.