MagicCoin Smart Contracts Security Review

2017

TABLE OF CONTENTS

Executive summary

NB | The section contains overall summary of the security audit. Detailed information used for the conclusion is listed further in the document.

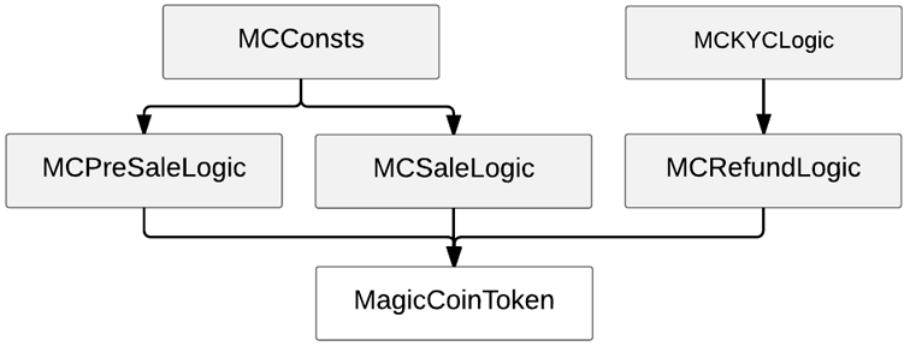
Review performed by:	Distributed Ledgers Inc. info@ledgers.world
Review Object:	*MagicCoin* crowdsale smart contracts
Source:	https://github.com/MagicCoin/token
Version:	https://github.com/MagicCoin/token/commit/624deb60de08fff6a9d4e5bc7c4b070bf724b152
Conclusion:	The smart contract cannot be used in production environment
Rating:	2 of 5 (0 is the lowest score, 5 is the highest score)

Technical details

NB	The section contains technical details of smart contract set. It may include: smart contract inheritance, Ether and Token flow, token distribution, smart contract life cycle, special conditions, whitepaper and smart contract logic comparison and so on.

Class structure of the smart contracts set

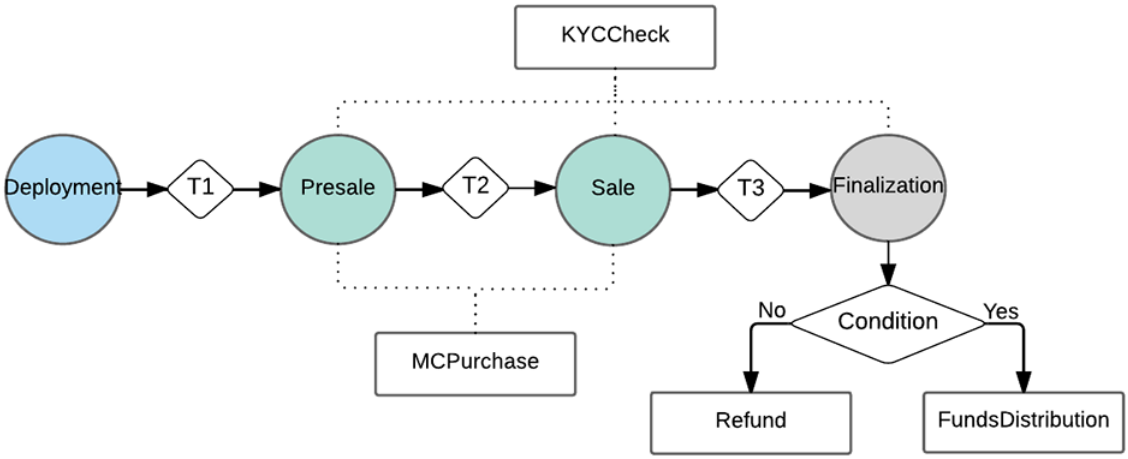
Scheme 1: *Inheritance scheme of the reviewed smart contacts*



MagicCoinToken smart contract is deployed to the Blockchain.

Smart contract lifecycle

Scheme 2: *State transitions of smart contract lifecycle.*



Smart contract states are circled on the scheme 2.
T1 (beginning of presale stage) > **T2** (end of presale and beginning of sale stage) > **T3** (end of sale stage)

Ether flow

Investors purchase MagicCoin Tokens for Ethers during pre-sale and sale stages (method *MCPurchase*). The Ethers are collected into *MCCollectingWallet*. If KYC is rejected, the investor can get a refund with the method *KYCRejectedRefund*. There is no access to the funds until finalization, only investors whose KYCs are rejected can request refunds.
After ICO finalization:

1. The investors can request refunds in the case of unsuccessful ICO (method *MCRefund*).
2. The owner gains access to the funds in any case.

MagicCoin token flow

When the smart contract is deployed a fixed amount of MagicCoin Tokens are emitted: MC_Emission (78,000,000,000 Tokens). Additional emissions are possible during pre-sale and sale stages. The created MagicCoin Tokens are distributed as follows:

- Owner_Pool_Tokens: 5%;
- Dev_Pool_Tokens: 5%;
- Sale_Pool_Tokens: 90%.

Exchange rate ETH:MagicCoin Tokens for presale and sale stages can be set before presale and sale stages accordingly. ICO owners and investors cannot manage (transfer from one to another wallet) MagicCoin Tokens until ICO finalization. MagicCoin Tokens located in Sale_Pool_Tokens are purchasing during presale and sale stages.

After ICO finalization (in the case of ICO success):

1. Unsold MagicCoin Tokens are distributed 50/50 between Owner_Pool_Tokens and Sale_Pool_Tokens.
2. MagicCoin Tokens located in Owner_Pool_Tokens and Sale_Pool_Tokens become available after 6 months.
3. Purchased MagicCoin Tokens become available for investors.

KYC model

KYC (Know Your Customer) key points:

- Owner of the smart contracts approves or rejects KYC.
- Approved KYC can be cancelled.
- If a KYC is rejecter, the investor should request a refund (no automatic refund).

Detected issues

NB | The section contains issues detected during the security audit. The issues are divided to two main groups: critical issues and other (or non-critical issues). Each critical issue is classified by type and severity. The type can be Trust model / Vulnerability / Bug / Other. The severity can be Critical / High / Medium.

Critical issues

1. No Refund, method *MCRefund*

The method *MCRefund* is intended to allow customers to receive back Ethers in case of unsuccessful ICO. It does not work because:

- When customers buy tokens, Ethers are send directly to an address which is defined in the variable *wallet*. No Ether is enrolled to smart token address.
- The method *MCRefund* uses smart contract balance to perform refunds.

Type: Logic Bug	Severity: Critical
Source: *<link to an appropriate part of the source code>*	

2. Additional Token Emission(s), method *addTokens*

The smart contract owner can call the method *addTokens* to emit tokens. The method is available until ICO finalization.

Type: Trust model	Severity: Critical
Source: *<link to an appropriate part of the source code>*	

3. Token purchasing after finalization, method *MCPurchase*

The method *finalize* does not suspend token sale. After the finalization the method *MCPurchase* can be still used to purchase tokens. Actually, the method can be used while the wallet *Sale_Pool_Tokens* contains tokens. In additional, unsold tokens are distributed between *Owner_Pool_Tokens* and *Dev_Pool_Tokens*. According to the whitepaper unsold tokens should be destroyed after ICO finalization.

Type: Trust model	Severity: High
Source: *<link to an appropriate part of the source code>*	

4. *fundContibuted* not decreasing when *KYCRejectedRefund*

When KYC is rejected the investor can request a refund with the method *KYCRejectedRefund.* The method is called, the investor receives a refund but the variable *fundContributed* is not decreased.

Type: Logic Bug	Severity: High
Source: *<link to an appropriate part of the source code>*	

Other issues

	Source	Method	Description
1	*<link to an appropriate part of the source code>*	*addPrecommitement*	The method increases *fundContribution* and *fundRaised* but Ethers are not transferred to the smart contract. The *Vault*'s balance will be less than *fundRaised*
2	*<link to an appropriate part of the source code>*	*MCRefund*	Variable *balanceEth* is not decreased when the method is called. If the method *MCRefund* worked, it would be possible to do multiple refunds.
3	*<link to an appropriate part of the source code>*	*addPrecommitment*	Smart contract owner can exceed HARD_CAP limit by calling the method addPrecommitement (until START_DATE).
4	*<link to an appropriate part of the source code>*	*KYCApprove*	Anyone can call the method (no restriction).
5	*<link to an appropriate part of the source code>*	*MCPurchase*	Repetitive purchasing from the same wallet sets KYC flag to false. It means KYC has to be performed for every token purchase.

Recommendations

NB	The section contains improvement recommendations.

Recommendations can be divided to two groups: Trust model revision and Code revision.

Trust model revision

Currently the ICO may look suspicious for potential investors due to the following reasons:

1. The smart contract owner can emit tokens at their discretion.
2. Investors cannot receive refund in case of unsuccessful ICO.
3. The whitepaper and smart contract logic does not match.
4. The funds become available to the smart contract owner right away after ICO finalization. If the funds are transferred to another wallet, investors will not get any refunds in case of KYC reject or unsuccessful ICO.
5. Transition to the state when investors can use tokens depends on the smart contract owner.

It is advisable to modify the current trust model to make the ICO safer for potential investors.

Code revision

To make work with programming code easier and to reduce possibilities of issues, please do the following:

1. Fix reported bugs;
2. Resolve all contradiction with the whitepaper;
3. Do the code refactoring that includes:
    - Removing excessive (duplicated) code;
    - Bringing the code to a single common style;
    - Following Solidity security guideline https://consensys.github.io/smart-contract-best-practices/