Positive com

trade.io

TRADE.IO SMART CONTRACT SECURITY ASSESSMENT

London, 2017

SUMMARY

This document presents the path that was taken by the consultants in an effort to simulate a real world attack scenario on TRADE.IO. The assessment included smart contract analysis.

The smart contract proved to be in a very good state. No vulnerabilities were identified. A few best-practice recommendations are provided in the relevant section.

TradeToken Smart Contract Audit

Positive Technologies experts have assessed the source code of TradeToken smart contracts.

The audited contracts (TIOToken, TIOCrowdsale) are in the trade-io/Tradeio-TokenSale-Contract Github repository. The version used for this report is the commit f5c14561dcf37c0508554499a8de23fd134e0528.

The goal of the audit is to find vulnerabilities, logical flaws and other code errors.

TIOToken

The contract is built on top of Majoolr's TokenLib library with minimal additions.

Contract Profiling

Contract	Function	Visibility	Constant	Returns	Modifiers
TIOToken	TIOToken(address,string,string,uint8,uint256,bool)	public	false		
TIOToken	owner()	public	true	address	
TIOToken	name()	public	true	string	
TIOToken	symbol()	public	true	string	
TIOToken	decimals()	public	true	uint8	
TIOToken	initialSupply()	public	true	uint256	
TIOToken	totalSupply()	public	true	uint256	
TIOToken	balanceOf(address)	public	true	uint256	
TIOToken	allowance(address,address)	public	true	uint256	
TIOToken	transfer(address,uint)	public	false	ok	
TIOToken	transferFrom(address,address,uint)	public	false	ok	
TIOToken	approve(address,uint)	public	false	ok	
TIOToken	changeOwner(address)	public	false	ok	
TIOToken	burnToken(uint256)	public	false	ok	

Vulnerabilities

No vulnerabilities have been found.

Notes & Additional Information

- Keep in mind that ERC20 standard has a possible attack vector on approve/transferFrom methods described here; consider making `approveChange` from `TokenLib` callable in `TIOToken` to avoid possible double spend attack;

- Consider checking that `_to` argument is not equal to `0x0` in `transfer` and `transferFrom` functions in TokenLib.sol to avoid accidental token transfer to undefined addresses;
- Consider raising the minimum version of Solidity compiler (currently 0.4.15) to the latest.

TIOCrowdsale

The contract is built on top of Majoolr's CrowdsaleLib library with minimal additions.

Contract Profiling

Contract	Function	Visibility	Constant	Returns	Modifiers
TIOCrowdsale	TIOCrowdsale(address,uint256,uint256,uint256,uint256,uint8,CrowdsaleToken)	public	false		
TIOCrowdsale	()	public	false		payable
TIOCrowdsale	sendPurchase()	public	false	bool	payable
TIOCrowdsale	activateGreenshoe()	private	false	bool	
TIOCrowdsale	withdrawTokens()	public	false	bool	
TIOCrowdsale	withdrawLeftoverWei()	public	false	bool	
TIOCrowdsale	withdrawOwnerEth()	public	false	bool	
TIOCrowdsale	crowdsaleActive()	public	true	bool	
TIOCrowdsale	crowdsaleEnded()	public	true	bool	
TIOCrowdsale	setTokenExchangeRate(uint256)	public	false	bool	
TIOCrowdsale	setTokens()	public	false	bool	
TIOCrowdsale	getOwner()	public	true	address	
TIOCrowdsale	getTokensPerEth()	public	true	uint256	
TIOCrowdsale	getExchangeRate()	public	true	uint256	
TIOCrowdsale	getCapAmount()	public	true	uint256	
TIOCrowdsale	getStartTime()	public	true	uint256	
TIOCrowdsale	getEndTime()	public	true	uint256	
TIOCrowdsale	getEthRaised()	public	true	uint256	
TIOCrowdsale	getContribution(address)	public	true	uint256	
TIOCrowdsale	getTokenPurchase(address)	public	true	uint256	
TIOCrowdsale	getLeftoverWei(address)	public	true	uint256	
TIOCrowdsale	getSaleData(uint256)	public	true		
TIOCrowdsale	getTokensSold()	public	true	uint256	
TIOCrowdsale	getPercentBurn()	public	true	uint256	

Vulnerabilities

No vulnerabilities have been found.

Notes & Additional Information

- Consider updating the number of tokens that are given per 1 eth to 625 as is stated by the whitepaper (currently 150);
- Consider updating fallback exchange rate to 400 dollars per 1 eth as of 23.11.2017 (currently 300 dollars);
- Consider checking error in `receivePurchase()` function in `DirectCrowdsaleLib` when subtracting `_numTokens` from owner's tokens in `withdrawTokensMap`;
- Consider calling `setTokens()` in `setTokenExchangeRate()` in `CrowdsaleLib` to avoid code duplication;
- Consider raising the minimum version of Solidity compiler (currently 0.4.15) to the latest.