MixBytes()

Audit of LTO Token Sale Contracts

Introduction

General Provisions

The LTO Network team asked MixBytes to audit their token sale contracts. The code is located in the hidden github repository.

Scope of the Audit

The primary scope of the audit is smart contracts at https://github.com/legalthings/tokensale/tree/02fa2620aef4c854675230b6544461961d47f968/contracts. Migrations at https://github.com/legalthings/tokensale/tree/02fa2620aef4c854675230b6544461961d47f968/migrations were also included in the scope as they contain mission critical deployment and configuration code which ties the system together. Audited commit is 02fa2620aef4c854675230b6544461961d47f968. Subsequently we were also asked to audit the revised token at https://github.com/legalthings/tokensale/blob/98ba921251fb4989d18f8b99a2bfc732f2056937/contracts/LTOToken.sol.

Security Assessment Principles

Classification of Issues

- CRITICAL: Bugs that enable theft of ether/tokens, lock access to funds without possibility to restore it, or lead to any other loss of ether/tokens to be transferred to any party (for example, dividends).
- MAJOR: Bugs that can trigger a contract failure, with further recovery only possible through manual modification of the contract state or contract replacement altogether.
- WARNINGS: Bugs that can break the intended contract logic or enable a DoS attack on the contract.
- COMMENTS: All other issues and recommendations.

Security Assessment Methodology

The audit was performed with triple redundancy by three auditors.

Stages of the audit were as follows:

- “Blind” manual check of the code and model behind the code
- “Guided” manual check of the code
- Check of math balance
- Check of adherence of the code to requirements of the client
- Automated security analysis using internal solidity security checker
- Automated security analysis using public analysers
- Manual by-checklist inspection of the system
- Discussion and merge of independent audit results
- Report execution

Detected Issues

CRITICAL

None found

MAJOR

None found

WARNING

1.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOToken.sol#L23-L24

Generation of two events in the form `emit Transfer(address(0), receiver address, amount);` should be added. Otherwise, some client software won't recognize the arrival of the tokens to `msg.sender` and `_bridgeAddress` recipients.

Fixed in 686101d

2.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOToken.sol#L52

The recipient address specified in the `Transfer` event does not match factual address of the recipient (which can be seen here). In case this was made on purpose we should note that it'll hinder diagnosing of the contract operation.We recommend to do actual transfer using `super.transfer` function.

Fixed in 686101d

3.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOToken.sol#L64

In case `_intermediate` is being used at the moment by some user `addIntermediateAddress` call will cause all token transfers to this user to fail to reach them. It'll be difficult to diagnose this situation because of the problem with the `Transfer` event described above. Access to the `addIntermediateAddress` function is limited to the address stored in the `bridgeAddress` field, but for the moment, code that will be used is unavailable to the audit. It's a common security approach to assume that the problem described above will take place, accidentally or as a result of more complex attack vector. A minimal, but not complete, solution is to add a check that the token balance of `_intermediate` address is zero at the moment of `addIntermediateAddress` call. A definitive solution can be designed only with the bridge mechanics in mind.

Fixed in 686101d

4.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOToken.sol#L59

Obviously, `totalSupply_` is being maintained as a difference between total amount of tokens (`internalTotalSupply`) and amount of tokens in `bridgeAddress` possession. However, `totalSupply_` value won't be properly updated in case of `transferFrom` function usage, which is inherited from the `StandardToken` .

Fixed in 686101d

5.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOToken.sol#L46

There are mechanics which allows users to transfer tokens from some predefined addresses (enumerated in `intermediateAddresses`) to the bridge balance. However, mechanics can be bypassed, and tokens can be transferred, to an address. This applies even if this address is listed in intermediate addresses in case of `transferFrom` function usage, which is inherited from the `StandardToken`.

Fixed in 686101d

6.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L215

Users sending ether from contracts which default function performs nontrivial computations (e.g. writing value to a new storage variable) won't be able to receive their tokens, while the project won't be able to receive transferred ether. This is caused by limited gas stipend of the `transfer` function. It's recommended to use [withdrawal pattern] (https://solidity.readthedocs.io/en/v0.4.24/common-patterns.html#withdrawal-from-contracts) to send change and to untie sending of change from other actions. I.e. we recommend offering change to users during `withdrawal` or `withdrawalFor` transaction processing via `send` or, failing that, with a separate transaction via a dedicated function call.

Fixed in PR #18

7.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L215

Malicious contract (or contracts which default function performs nontrivial computations) can block a change to it and as a result any transaction which calls `_withdrawal` for this contract will be reverted. Therefore `withdrawn` will never reach `purchaserList.length` and the `clear` function will be blocked, which will result in blocking some tokens and ether on the balance of `LTOTokenSale`. The ether vulnerable to this attack includes only ether deposited to the contract bypassing `buy()` and default functions, which is why it's not a serious problem. It is worth noting that tokens transferred to the contract by creators exceeding `totalSaleAmount` are vulnerable to the attack. It's recommended using withdrawal pattern to send change and to untie sending of change from other actions. I.e. we recommend to send change to users during `withdrawal` or `withdrawalFor` transaction processing via `send` or, failing that, with a separate transaction via a dedicated function call.

Fixed in PR #14

8.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L223

Payments are not expected to be sent to this function. We therefore recommend removing the `payable` keyword to prevent accidental ether transfer to the contract which won't be processed by the token sale logic.

Fixed in PR #14

9. https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L227

Payments are not expected to be sent to this function. We therefore recommend removing `payable` keyword to prevent accidental ether transfer to the contract which won't be processed by the token sale logic.

Fixed in PR #14

10. https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L233

Payments are not expected to be sent to this function. We therefore recommend removing `payable` keyword to prevent accidental ether transfer to the contract which won't be processed by the token sale logic.

Fixed in PR #14

11. https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L199

An attacker can send some value of ether in the first transaction, receive maximum bonus and then send 99 transactions of minimal value `minimumAmount`, depriving other token sale participants of their bonuses. Essentially this is griefing of a transaction counter. This makes economic sense given some conditions, because the fewer participants receive their bonuses, the lower the price is to the attacker compared to the average price, making the following sale more profitable for the attacker.

Acknowledged

12. https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L179

Individual cap mechanics are subject to Sybil attack. An user can distribute his `N` ethers to `uint(N / maximumCapAmount) + 1` distinct addresses, and perform purchase from each of these addresses, effectively bypassing `maximumCapAmount` limit.

Acknowledged

13. The burning of unsold tokens was mentioned by the client. However, in the contracts code there is no sign of such mechanics and `LTOToken` is not burnable.

Fixed in PR #16

14. https://github.com/legalthings/tokensale/blob/98ba921251fb4989d18f8b99a2bfc732f2056937/contracts/LTOToken.sol#L19

Generation of an event in the form `emit Transfer(address(0), _bridgeAddress, _bridgeSupply);` should be added. Otherwise, some client software won't recognize the arrival of the tokens to the `_bridgeAddress`.

Fixed in PR #18

COMMENT

1\.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOToken.sol#L67

The only value besides 0, which a key X can have in the `intermediateAddresses` mapping, is the X itself. It means that logically this mapping has boolean value type, i.e. `mapping (address => bool)`. We recommend using this type explicitly and making appropriate code changes to simplify reasoning about the code.

2\.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOToken.sol#L60

We recommend using `SafeMath.sub` here because this code fragment has a high probability of underflow error.

3\.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L249

The only value besides 0, which a key X can have in the `capFreeAddresses` mapping, is the X itself. It means that logically this mapping has boolean value type, i.e. `mapping (address => bool)`. We recommend using this type explicitly and making appropriate code changes to simplify reasoning about the code.

4\.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L218

This branch is unachievable in practise (it requires ~5e14 ether to achieve), we recommend replacing it with `assert(false);`.

5\.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L144

Expression `totalWannaBuyAmount == 0` here is unachievable, we recommend removing it from if-operator and introduce `assert(totalWannaBuyAmount > 0);` instead.

6\.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L120

In `startSale` function it would be helpful to ensure that token balance of the contract is equal to `totalSaleAmount`. Otherwise, if the token balance is less than `totalSaleAmount` some ether will be locked in the contract forever.

7.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L105

Right after deployment of the contract `isEnded` function will return `true`. This won't cause any troubles in this version of the code, but it's better to return `false` in case the sale was never started (similar check is present in `isStarted`). The same holds for `isUserWithdrawalTime`, `isClearTime` functions.

8.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L145

It's better to use the `ethDecimals` constant, instead of the `1 ether` value to prevent possible inconsistencies.

9.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L191

Condition `<=` is excessive and can be replaced with a strict comparison (`<`).

10.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L176

Keeping a purchasers list in `purchaserList` and further processing in `withdrawalFor` incurs a high gas consumption. At this moment, calling `withdrawalFor` is required to get all possible ether from the contract and to send tokens and ether to purchasers. These two processes can be separated and gas spendings by the project can be minimized. Amount of ether to be withdrawn by the project can be calculated based on `globalAmount`, `totalSaleAmount` and `totalWannaBuyAmount` and withdrawn in one transaction.

11.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L123

We recommend creating the explicit view function `isSaleInitialized` and use it here to prevent double initialization, as well as throughout `isStarted`, `isEnded` and other state querying functions.

12.
https://github.com/legalthings/tokensale/blob/02fa2620aef4c854675230b6544461961d47f968/contracts/LTOTokenSale.sol#L235

We recommend checking the result of token transfer call.

13. There are a lot of states and state transitions involved in the token sale contract, e.g. sale started, sale ended, user withdrawal time etc. We recommend using state machine to remove some code complexity and ensure correct operation, as can be seen in this example.

14. https://github.com/legalthings/tokensale/blob/98ba921251fb4989d18f8b99a2bfc732f2056937/contracts/LTOToken.sol#L35

Tokens which were accidentally sent to the bridge address could not be recovered. We recommend adding `require(to != bridgeAddress);` check.

15. https://github.com/legalthings/tokensale/blob/98ba921251fb4989d18f8b99a2bfc732f2056937/contracts/LTOToken.sol#L38

If `value` is expected to be greater than `bridgeBalance` in some cases we recommend adding check `require(value <= bridgeBalance);` . The reason is that `assert` inside the `bridgeBalance.sub` should never be reached during an expected path of execution of the code.

16. https://github.com/legalthings/tokensale/blob/98ba921251fb4989d18f8b99a2bfc732f2056937/contracts/LTOToken.sol#L56

The sum of balances is not equal to `totalSupply` . This behavior could be unexpected for some tools or clients.

CONCLUSION

The overall security level of the system was rated “High”. No major flaws were found. However, there were many issues about which we warned the client. Some of them can be accepted as a known expected behavior, but some, in our opinion, required fixes, e.g., ones related to withdrawal functions. All necessary changes were made and existed in pool request #19 branch.