BANKEX Token-Timelock Security Review

REVISION 1

2018

TABLE OF CONTENTS

Executive summary

Review performed by:	Distributed Ledgers Inc. info@ledgers.world
Review Object:	BankEx Token-Timelock
Source:	https://github.com/BANKEX/token-timelock/blob/e3403d07f932e6a74a9f893c79fbc3692acb90ed/onepager/SafeERC20TimelockOnepager.sol
Version:	e3403d07f932e6a74a9f893c79fbc3692acb90ed
Conclusion:	**The original version of BankEX smart contract is NOT secure for token holders.**
Rating:	2 of 5 (0 is the lowest score, 5 is the highest score)

Evaluation method

The logic of the smart contract is examined to evaluate possible impacts on interests of various types of users. Users can be divided to two types:

1. Owner of Token-Timelock smart contract. A risk for this type of users is:
    a. Loss of control of administrative functions.
2. Token holders. A risk for this type of users is:
    a. Reserved funds can be stolen.

Technical details

Token-Timelock smart contract allows to reserve ERC20-compatible tokens. An address of a token smart contract for reservation is defined in Token-Timelock smart contract constructor. The address cannot be changed later. The smart contract does not work with the basic cryptocurrency (Ether).

ERC20-compatible token smart contract	
Address1	balance
Address2	balance
TTL-address	balance
…	…

Token-Timelock smart contract		
Beneficiary1	balance	timestamp
Beneficiary2	balance	timestamp
…	…	…

Token reservation

1. Sender approves withdrawing some amount of tokens to Token-Timelock smart contract account in the ERC20-compatible token smart contract (function *approve* in the ERC20-compatible token smart contract).
2. The sender calls Token-Timelock's function *accept* with the following parameters: a beneficiary address, an amount of tokens to reserve and a timestamp (timestamp is a date and time, the beneficiary can receive the reserved tokens after the timestamp).
    - The sender can specify a several reservations for each beneficiary address with different timestamps.
    - Any other sender can add tokens to an existing reservation.
3. Token-Timelock smart contract transfers the approved amount of tokens (step 1) from the sender's account to Token-Timelock smart contract account in the ERC20-compatible token smart contract.

Token release

1. A beneficiary calls Token-Timelock's function *release* with the following parameters: a timestamp, an amount of tokens to release. Both parameters are arrays. It allows to release tokens for several reservations by one transaction.
    - The amount to release cannot be greater than the reserved amount. A partial token release is possible.

- The owner of Token-Timelock smart contract can force token release. It this case the owner specifies the beneficiary address (function *releaseForce*).

2. Token-Timelock smart contract performs necessary checks and transfers the specified amount(s) of tokens from Token-Timelock smart contract account to the beneficiary account in the ERC20-compatible token smart contract. If any of checks fails all changes (if any) are rolled back.

Detected issues

1. Backdoor, function *execute*

There is function *execute* defined in Token-Timelock smart contract. The function is not enough documented, and its purpose is unclear. Function *execute* allows the owner to call functions of other smart contracts. The peculiarity of its usage is that the smart contract called through it gets Token-Timelock smart contract address, not the actual address of the sender. It gives full control over all tokens that are reserved through Token-Timelock smart contract to the owner of Token-Timelock smart contract.

An example below explains the attack.

1. The owner of Token-Timelock smart contract creates a new address, for example:

   ```
   0xd1d4e623d10f9fba5db95830f7d3839406c6af2
   ```

2. The owner sends a transaction:

   ```
   lock.execute(token,
   '0x095ea7b300000000000000000000000d1d4e623d10f9fba5db95830f7d3839406
   c6af2000000000000000000000000000000000000000000000000000000000000fa'
   , { from: lockOwner })
   ```

   The binary string is a call of ECR20 function *approve*, which allows the address specified in step 1 to retrieve 250 (0xfa) tokens from Token-Timelock smart contract account in the ERC20-compatible token smart contract.

3. The owner transfers 250 tokens from Token-Timelock smart contract account to the account of the address specified in step 1 (controlled by the owner) by calling function *transferFrom* in the ERC20-compatible token smart contract:

   ```
   `token.transferFrom(lock.address,
   0xd1d4e623d10f9fba5db95830f7d3839406c6af2, 250, { from:
   0xd1d4e623d10f9fba5db95830f7d3839406c6af2})`
   ```

Type: Trust Model	Severity: High
Source: https://github.com/BANKEX/token-timelock/blob/e3403d07f932e6a74a9f893c79fbc3692acb90ed/onepager/SafeERC20TimelockOnepager.sol#L212	

2. Outdated SafeMath version

The current implementation of Token-Timelock smart contract uses an outdated version of SafeMath library.

Type: Trust model	Severity: Low
Source: https://github.com/BANKEX/token-timelock/blob/e3403d07f932e6a74a9f893c79fbc3692acb90ed/onepager/SafeERC20TimelockOnepager.sol#L8	

Enhancements

The suggested improvements are listed below.

1. Removing function *execute*

The current implementation allows the owner of Token-Timelock smart contract to use the reserved tokens. Since another use of the function is not documented and not detected, it is advisable to remove the function completely from the code.

2. Removing owner

In the current implementation the owner:

- Has access to the backdoor (function *execute*),
- Can change the owner address,
- Can force token release (function *releaseForce*).

If function *execute* is removed and *releaseForce* is changed to be called by anonymous, the owner role is not necessary.

3. Renewing SafeMath

It is better to use the latest version of SafeMath. In the new version *assert* is replaced by *require.* It minimizes gas usage in cases of erroneous calls.

The latest version of SafeMath can be obtained here:
https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/math/SafeMath.sol

4. Input validation, function *accept*

It is advisable to add input validation when calling *accept*:

```
require(_for != address(0));
require(_for != address(this));
require(_timestamp > block.timestamp);
require(_tvalue > 0)
```

Conclusion

The current version of Token-Timelock smart contract has a backdoor, which gives a full control over the reserved tokens to the owner of the smart contract. This can be considered as a direct threat to token holders.