September, 2018

— BANKEX Token Timelock Smart Contract Audit by Ambisafe Inc.


YAROSLAV TUMANOV


ULYANA GROMOVA


OLEKSII MATIIASEVYCH

Ambisafe

Confidential

This document and the Code Review it references is strictly private, confidential and personal to its recipients and should not be disclosed, copied, distributed or reproduced inwhole or in part, nor passed to any third party.

YOU MAY NOT ISSUE ANY PRESS RELEASE ABOUT THIS CODE REVIEW. AMBISAFE IS PROVIDING THIS CODE REVIEW AS PART OF OUR QUALITY ASSURANCE PROCESS ON A “BEST EFFORTS”, “AS IS” AND “AS AVAILABLE” BASIS AT THIS PARTICULAR POINT AND TIME, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

WITHOUT LIMITING THE FOREGOING, AMBISAFE EXPRESSLY DISCLAIM ANY AND ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, TITLE, ACCURACY AND COMPLETENESS, OR FUNCTIONING OF THIS CODE.

BY ACCESSING THIS DOCUMENT YOU ACKNOWLEDGE, ACCEPT AND AGREE TO THE FOREGOING. THIS DOCUMENT IS NOT FOR PUBLICATION OR ANY DISTRIBUTION, DIRECTLY OR INDIRECTLY, IN WHOLE OR IN PART. THESE MATERIALS ARE NOT AN OFFER OF SECURITIES FOR SALE. SECURITIES MAY NOT BE OFFERED OR SOLD IN THE UNITED STATES IN ABSENCE OF REGISTRATION WITH THE U.S. SECURITIES AND EXCHANGE COMMISSION OR AN EXEMPTION FROM REGISTRATION UNDER THE U.S. SECURITIES ACT OF 1933, AS AMENDED.

1. Introduction

BANKEX requested Ambisafe to perform an audit of the contract implementing timelock logic. The contract in question are hosted at:

https://github.com/BANKEX/token-timelock/blob/e3403d07f932e6a74a9f893c79fbc3692acb90ed/onepager/SafeERC20TimelockOnepager.sol

There are 7 contracts in scope which consist of 299 lines of code:

- IERC20
- SafeMath
- TimeMachineP
- ITimeMachine
- Ownable
- SafeERC20Timelock
- SafeERC20TimelockProd

7 contracts in scope

299 lines of code

2. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts for any specific purpose, or their bugfree status. The audit documentation below is for internal management discussion purposes only and should not be used or relied upon by external parties without the express written consent of Ambisafe.

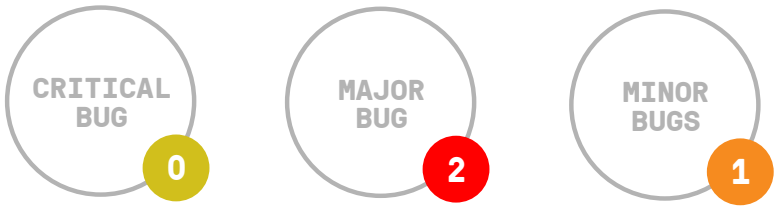
3. Executive Summary

Critical issues were not found in timelock contract. Token transfer during the release might fail, in which case tokens will be locked until recovered through an **execute()** function call. We recommend checking the result of all external calls to avoid undesired results. Contract owner is able to withdraw all tokens from timelock contract address at any time (described in 5.7). We recommend to restrict **execute()** function from making calls to the **token** contract, and adding another function for accidentally sent tokens recovery. Timelock contract is not affected by known Solidity 0.4.24 compiler bugs. **SafeMath** is used to protect calculations from overflows. Care should be taken by contract owner when transferring ownership, mistake in the process will leave **onlyOwner** functions unaccessible. We recommend using two-step, failsafe, ownership transfer. Tokens can be released only when lock time passes, by request from the receiver or the contract owner.

Not passed

4. Critical Bugs and Vulnerabilities.

No places in code were identified as critical issues.



0 critical bug

2 major bug

1 minor bugs

5. Line By Line Review

5.1 Note: it's not possible to foresee which solidity version will be used in deployed contract. Consider specifying strict compiler version.

5.2 Line 106. Note: control over token contract might be lost if owner call **transferOwnership()** with wrong **newOwner** parameter (e.g. receiver lost private key, wrong address used etc.). Possible solution: new owner should accept ownership through additional function call, thus confirming that he has access to his address.

5.3 Line 141. Minor: Consider adding a check for **_timestamp** in order to avoid setting a zero timestamp or some past time.

5.4 Line 145. Note: Consider checking for successful **transferFrom()** result and reverting otherwise to avoid emitting **Lock** event in case of fail.

5.5 Line 174. Note: consider the situation when sender try to release 5 tokens and four of them are not locked anymore but just one is locked. During the check on this line all the tokens will not be released in this transaction and it will not be clear which token is still locked. Consider making the statement to return **false** in case **_curTimestamp** <= **_now** and emit an error event with information about timestamp. In this case all tokens which wasn't locked will be released.

5.6 Line 180. Major: **token.transfer()** can return **false** but changes have been already made. In this case address **_for** will not get his tokens released and will not has them on his balance in the current contract. Consider checking for successful **transfer()** and revert if it is not.

5. Line By Line Review (cont'd)

5.7 Line 212 Major: Contract owner is able to withdraw all tokens from timelock contract address via **execute()** function by following the next steps:

- 5.7.1 There are some tokens on timelock contract.
- 5.7.2 Get balance via **contractBalance**_().
- 5.7.3 Contract owner calls **token.approve(ownerAddress, timelockContractBalance)** via **execute()** function.
- 5.7.4 Contract owner calls directly **token.transferFrom(timelockContractAddress, ownerAddress, contractBalance)**.
- 5.7.5 Contract owner gets all the tokens from the timelock contract.

5.8 Note: there is function/event arguments naming inconsistencies, some are using underscore prefixed names, while others aren't, in some places **uint** is used while in others **uint256** is used. Also **returns** variable names declared in some functions while left undeclared in the others. We recommend using a unified code style.

Oleksii Matiiasevych,
Solidity Engineer