DepositTomo Contracts Audit

Yaroslav Tumanov

November, 2024

Scoring

6/10

Email: info@4irelabs.com

Website: 4irelabs.com

CONFIDENTIAL

This document and the Code Review it references are strictly private, confidential and personal to its recipients and should not be disclosed, copied, distributed, or reproduced in whole or in part, not passed to any third party.

1. **Introduction.** Tomo requested 4irelabs a review of their contracts.

   There is 1 contract source code in scope:

   - DepositTomo

2. **Warranty.** Audit is provided on an "as is" basis, without warranty of any kind, express or implied. The Auditor does not guarantee that the Code Review will identify all instances of security vulnerabilities or other related issues.

3. **Executive Summary.** The DepositTomo contract is designed to manage staking contributions with 90-day soft tokens lock. It allows to get bonuses for stakers or get withdrawal penalties if the user withdraws earlier. The contract has some not implemented parts (check Major and Low issues). Also, the contract administrator can stop the withdrawal process by pausing the contract. There are no known compiler bugs, for the specified compiler versions, that might affect the logic of the contracts.

4. **Critical Bugs and Vulnerabilities.** 4 Major issues have been found. More details are described in Line by Line review in 5.1.10, 5.1.21, 5.1.22, and 5.1.24.

5. Line By Line Review.

5.1. DepositTomo:

5.1.1. Low: Floating pragma.

The current version of solc in contract DepositTomo is **^0.8.7** which allows the code to be compiled with any compiler version from **0.8.7** up to (but not including) **0.9.0**. This can lead to inconsistencies due to changes in the compiler's behavior in future versions.

Contracts should be deployed with the same compiler version and flags they have tested thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation: Lock pragma to a specific version.

5.1.2. Note: Not necessary Library Imported.

The **SafeMath** library is imported, but Safe Math works out of the box from 0.8.x version.

Recommendation: Remove the library import statement to reduce bytecode size and improve code readability.

5.1.3. Low/Optimization: Simplify depositedAmounts mapping.

Since **coin** is immutable, **depositedAmounts** could be simplified by removing 2nd level mapping by **coinAddress**.

5.1.4. Low/Optimization: Simplify depositedAmounts mapping.

Since **coin** is immutable, **currentDates** could be simplified by removing 2nd level mapping by **coinAddress**.

5.1.5. Note: Explicitly mark visibility of state.

Explicitly mark visibility of state **totalDeposit** variable.

5.1.6. Note: Explicitly mark visibility of state.

Explicitly mark visibility of state **lockPeriod** variable.

5.1.7. Note: Explicitly mark visibility of state.

Explicitly mark visibility of state **precision1000** variable.

5.1.8. Note: Explicitly mark visibility of state.

Explicitly mark visibility of state **withdrawalPenalty** variable.

5.1.9. Note: Explicitly mark visibility of state.

Explicitly mark visibility of state **addressFee variable**.

5.1.10. Major: Deposits are always opened, stopDeposit will never be changed.

Deposits are always opened, **stopDeposit** variable never changed and always **false**.

*Recommendation: Add **stopDeposit()** functionality or use **whenNotPaused()** modifier modifier instead of **whenNoStopDeposit()**.*

5.1.11. Note: Not used event.

Event **RemoveAllTokensFromContract** is not used and could be removed.

5.1.12. Optimization: Change variables to immutable.

Variable **bonusAmounts**, **coin**, **lockPeriod** could be set as immutable to save gas.

5.1.13. Low: Use one base contract to manage admin roles.

It is recdommended to use just one of the base contracts AccessControl or **Ownable**.

*Recommendation: Remove **Ownable** contract and use **onlyRole(OWNER_ROLE)** instead of **onlyOwner** modifier.*

5.1.14. Optimization: Remove getanotherStakingWithoutFee() getter.

No need for **getanotherStakingWithoutFee** getter, variable **anotherStakingWithoutFee** is accessible as getter by default.

5.1.15. Optimization: Change some public functions to external.

Functions **deposit**, **setAddressFee**, **setAnotherStakingAdresses**, **setPenalty** could be marked as external to save gas.

5.1.16. Low: Use ReentrancyGuard contract or remove as not used.

Main contract has Reentrancy guard. Consider to add **nonReentrant** modifier to all user functions or remove base contract as not used.

5.1.17. Note: Admin tokens deposit does not transfer real tokens.

Function **depositOnlyManager** does not transfer real user tokens to the Deposit contract. If it's and valid business logic, then not an issue.

5.1.18. Optimization: Write one time during transaction to the totalDeposit and depositedAmounts storages.

It is possible to write 1 time to the **totalDeposit** and **depositedAmounts** storages in **deposit**/ **checkBalanceAndAddRewards** functions. it would save 5k * 3 = 15k gas per users's transaction.

*Recommendation: Do calculations in **checkBalanceAndAddRewards** function and update variables **totalDeposit** and **depositedAmounts** only in **deposit**/ **depositOnlyManager** functions.*

5.1.19. Low: Some users may not withdraw due to insufficient tokens on the Deposit contract.

During the withdrawal process, users could claim all tokens on the contract if everyone waited 90 days and would have bonuses. Then, some of the depositors will not be able to withdraw, therefor Admin should top-up the contract with the tokens to let all users withdraw.

5.1.20. Note/optimization: Use one global precision variable.

Global **precision1000** variable could be used instead of the **denominator variable** inside **withdraw()** function.

5.1.21. Major: User address is not removed from allUsersAddresses after withdrawal.

*Recommendation: Remove user address from **allUsersAddresses** after withdraw.*

5.1.22. Major: restakeAvailable() function is not linked with deposit() function.

RestakeAvailable() validations will not stop the user from calling deposit() function multiple times for restaking.

*Recommendation: Link **restakeAvailable** verifications with **deposit()** rules.*

5.1.23. Low: Add events to the admin functions.

Add events when **setAddressFee()**, **setAnotherStakingAdresses()**, **or setPenalty()** functions are executed.

5.1.24. Major: anotherStakingWithoutFee array is not used in the contract logic.

AnotherStakingWithoutFee is not used in the contract logic. Potentially some logic with no fee stakers missed.

*Recommendation: Add setting **anotherStakingWithoutFee** array and exclude these stakers from fees on **withdraw()**.*

5.1.25. Low: Possible to set penalty more than precision1000 variable.

It's possible to **setPenalty** more than **precision1000**/**denominator** variable. This will break **withdraw()** logic and the user will not be able to withdraw.

*Recommendation: Add verification that **withdrawalPenalty** could not be more than **precision1000**/**denominator**.*