Guard

Smart contracts security assessment

Final report

Tariff: Top

Kokomo Finance Token

March 2023

0xguard.com

hello@0xguard.com

Contents

Introduction

ERC-20 token with snapshots for voting (forked from Compound Finance COMP token) with an initial mint of 55% of the maximum total supply (100M with 18 decimals).

Name	Kokomo Finance Token
Audit date	2023-03-16 - 2023-03-20
Language	Solidity
Platform	Optimism Network

Contracts checked

Name	Address
Token	0x7Da25Bc4cFAed3F29414C6779676e53B19a356f5
TokenVesting	
ERC20	
SafeMath	

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities

- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	not passed
Message call with hardcoded gas amount	passed
Typographical Error	not passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed

Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Mint is open for owner (Token)
Status: Open

The owner has a one-time ability to mint 45% of MAX_SUPPLY, i.e. 45e24 tokens, to an arbitrary address.

```
    function mintToFarm(address _farm) external onlyOwner {
        require(!doOnce, "only can mint once");
```

```
        doOnce = true;
        __mint(_farm, 45 * 1e6 * 1e18);
    }
```

Recommendation: This function must be triggered before the public launch, the `_farm` recipient must be a contract with a clear and documented distribution policy.

Team response: As mentioned in the audit, the owner possesses a one-time ability to mint 45% of MAX_SUPPLY, i.e. 45e24 tokens, to an arbitrary address. This ability is to mint and distribute the needed tokens to the mining contract at the start of fair launch. The contract will then distribute those tokens linearly according to the emmission schedule.

Medium severity issues

1. Vesting benificiary (Token)

Status: Open

All vesting contracts are set to have a single beneficiary despite having different purposes from naming. The beneficiary address is the deployer:
EOA 0x41BE327a34D5D2f0855fF7e4FB3f6F1748B3310f.

```
    constructor(string memory name_, string memory symbol_) Ownable()
ERC20(name_,symbol_){

        teamLockToken = new TokenVesting(msg.sender, block.timestamp + 360 days, 0
days, 300 days);
        communityToken = new TokenVesting(msg.sender, block.timestamp, 0 days, 0
days);
        seedRoundLockToken = new TokenVesting(msg.sender, block.timestamp + 180 days, 0
days, 150 days);
        teamVaultToken = new TokenVesting(msg.sender, block.timestamp, 0 days, 0 days);
        __mint(msg.sender, 10 * 1e6 * 1e18);
        __mint(address(teamLockToken), 10 * 1e6 * 1e18);
        __mint(address(communityToken), 5 * 1e6 * 1e18);
        __mint(address(seedRoundLockToken), 20 * 1e6 * 1e18);
        __mint(address(teamVaultToken), 10 * 1e6 * 1e18);
```

```
    }
```

Recommendation: Justify the choice of a single vesting beneficiary for all tokens in the documentation.

Low severity issues

1. Votes delegating for minted tokens (Token)

Status: Open

Minted votes are transferred to the token recipient, i.e. vesting contracts or a farm contract, where it would be locked until tokens can be transferred further.

```
    function __mint(address _to, uint256 _amount) internal {
        _mint(_to, _amount);
        _moveDelegates(address(0), _to, _amount);
    }
```

Recommendation: Correct usage to avoid votes' locking:

```
_moveDelegates(address(0), _delegates[_to], _amount);
```

2. Gas optimisation (Token)

Status: Open

The state variables `teamLockToken`, `communityToken`, `seedRoundLockToken`, and `teamVaultToken` may be declared with an immutable keyword to reduce gas on reads from storage.

Recommendation: Declare all non-changeable variables immutable.

3. Gas optimisation (TokenVesting)

Status: Open

a. The state variables `beneficiary`, `cliff`, `start`, and `duration` may be declared with an immutable keyword to reduce gas on reads from storage.

b. Excessive reads of the `released[token]` variable from the storage in the `release` function may be reduced by bubbling its value from the `vestedAmount` function to `releasableAmount` and to the `release` function.

Recommendation: a. Declare all non-changeable variables immutable.

b.

```
function vestedAmount(address token) public view returns (uint256) {
    uint256 released = released[token];
    return _vestedAmount(token, released);
}

function _vestedAmount(address token) internal view returns (uint256) {
    ...
}

function releasableAmount(address token) public view returns (uint256) {
    uint256 released = released[token];
    return _vestedAmount(token, released).sub(released);
}

function release(address token) external {
    uint256 released = released[token];
    uint256 unreleased = _vestedAmount(token, released).sub(released);
    ...
    released[token] = released.add(unreleased);
    ...
}
```

4. Lack of error messages (TokenVesting)
Status: Open

Require statements in the constructor section and the `release` function have no error messages, which complicates failed transactions debugging.

Recommendation: Add error messages or codes to require statements.

5. Incomplete event (TokenVesting)

Status: Open

The `Released()` event doesn't contain information about the ERC20 token being released.

```
event Released(uint256 amount);

function release(address token) external {
  ...
  IERC20(token).safeTransfer(beneficiary, unreleased);
  emit Released(unreleased);
}
```

Recommendation: Add an address of the ERC-20 token to be transferred to the event in order to ease tracking on the front- or backend.

6. Gas optimisation (ERC20)

Status: Open

The initializable contract (forked from OpenZeppelin) is inherited but not used anywhere.

```
contract ERC20 is Initializable, Context, Ownable, IERC20 {
  ...
}
```

Recommendation: Remove excessive inheritance.

7. Typos (SafeMath)

Status: Open

Typo in 'substraction'.

Conclusion

Kokomo Finance Token Token, TokenVesting, ERC20, SafeMath contracts were audited. 1 high, 1 medium, 7 low severity issues were found.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability)set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Slither output

```
Token._writeCheckpoint(address,uint32,uint256,uint256) (contracts/Kokomo.sol#1139-1157)
uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock ==
blockNumber (contracts/Kokomo.sol#1149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

ERC20.allowance(address,address).owner (contracts/Kokomo.sol#583) shadows:
        - Ownable.owner() (contracts/Kokomo.sol#59-61) (function)
ERC20._approve(address,address,uint256).owner (contracts/Kokomo.sol#731) shadows:
        - Ownable.owner() (contracts/Kokomo.sol#59-61) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-
shadowing

Reentrancy in TokenVesting.release(address) (contracts/Kokomo.sol#857-867):
        External calls:
        - IERC20(token).safeTransfer(beneficiary,unreleased) (contracts/Kokomo.sol#864)
        Event emitted after the call(s):
        - Released(unreleased) (contracts/Kokomo.sol#866)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

TokenVesting.release(address) (contracts/Kokomo.sol#857-867) uses timestamp for
comparisons
        Dangerous comparisons:
        - require(bool)(unreleased > 0) (contracts/Kokomo.sol#860)
TokenVesting.vestedAmount(address) (contracts/Kokomo.sol#883-895) uses timestamp for
comparisons
        Dangerous comparisons:
        - block.timestamp < cliff (contracts/Kokomo.sol#888)
        - block.timestamp >= start.add(duration) (contracts/Kokomo.sol#890)
Token.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/
Kokomo.sol#1005-1046) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,FarmToken::delegateBySig:
signature expired) (contracts/Kokomo.sol#1044)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp
```

```
Initializable.isConstructor() (contracts/Kokomo.sol#152-162) uses assembly
        - INLINE ASM (contracts/Kokomo.sol#160)
Token.getChainId() (contracts/Kokomo.sol#1164-1168) uses assembly
        - INLINE ASM (contracts/Kokomo.sol#1166)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['>=0.6.0', '>=0.6.0<0.9.0', '^0.8.0']
        - >=0.6.0 (contracts/Kokomo.sol#771)
        - >=0.6.0<0.9.0 (contracts/Kokomo.sol#3)
        - >=0.6.0<0.9.0 (contracts/Kokomo.sol#28)
        - >=0.6.0<0.9.0 (contracts/Kokomo.sol#95)
        - >=0.6.0<0.9.0 (contracts/Kokomo.sol#169)
        - >=0.6.0<0.9.0 (contracts/Kokomo.sol#462)
        - ^0.8.0 (contracts/Kokomo.sol#247)
        - ^0.8.0 (contracts/Kokomo.sol#804)
        - ^0.8.0 (contracts/Kokomo.sol#899)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used

Context._msgData() (contracts/Kokomo.sol#20-23) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Kokomo.sol#708-716) is never used and should be
removed
ERC20._setupDecimals(uint8) (contracts/Kokomo.sol#746-748) is never used and should be
removed
Initializable.isConstructor() (contracts/Kokomo.sol#152-162) is never used and should
be removed
SafeERC20._safeApprove(IERC20,address,uint256) (contracts/Kokomo.sol#775-779) is never
used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (contracts/Kokomo.sol#780-783) is never
used and should be removed
SafeERC20.safeTransferETH(address,uint256) (contracts/Kokomo.sol#797-800) is never used
and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (contracts/
Kokomo.sol#791-795) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/Kokomo.sol#430-435) is never used and
should be removed
SafeMath.mod(uint256,uint256) (contracts/Kokomo.sol#394-396) is never used and should
be removed
SafeMath.mod(uint256,uint256,string) (contracts/Kokomo.sol#452-457) is never used and
```

```
should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/Kokomo.sol#265-271) is never used and
should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/Kokomo.sol#307-312) is never used and
should be removed
SafeMath.tryMod(uint256,uint256) (contracts/Kokomo.sol#319-324) is never used and
should be removed
SafeMath.tryMul(uint256,uint256) (contracts/Kokomo.sol#290-300) is never used and
should be removed
SafeMath.trySub(uint256,uint256) (contracts/Kokomo.sol#278-283) is never used and
should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.6.0<0.9.0 (contracts/Kokomo.sol#3) is too complex
Pragma version>=0.6.0<0.9.0 (contracts/Kokomo.sol#28) is too complex
Pragma version>=0.6.0<0.9.0 (contracts/Kokomo.sol#95) is too complex
Pragma version>=0.6.0<0.9.0 (contracts/Kokomo.sol#169) is too complex
Pragma version^0.8.0 (contracts/Kokomo.sol#247) allows old versions
Pragma version>=0.6.0<0.9.0 (contracts/Kokomo.sol#462) is too complex
Pragma version>=0.6.0 (contracts/Kokomo.sol#771) allows old versions
Pragma version^0.8.0 (contracts/Kokomo.sol#804) allows old versions
Pragma version^0.8.0 (contracts/Kokomo.sol#899) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Low level call in SafeERC20._safeApprove(IERC20,address,uint256) (contracts/
Kokomo.sol#775-779):
        - (success,data) =
address(token).call(abi.encodeWithSelector(0x095ea7b3,to,value)) (contracts/
Kokomo.sol#777)
Low level call in SafeERC20.safeTransfer(IERC20,address,uint256) (contracts/
Kokomo.sol#785-789):
        - (success,data) =
address(token).call(abi.encodeWithSelector(0xa9059cbb,to,value)) (contracts/
Kokomo.sol#787)
Low level call in SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (contracts/
Kokomo.sol#791-795):
        - (success,data) =
address(token).call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (contracts/
Kokomo.sol#793)
```

```
Low level call in SafeERC20.safeTransferETH(address,uint256) (contracts/
Kokomo.sol#797-800):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/Kokomo.sol#798)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Variable Initializable._____gap (contracts/Kokomo.sol#165) is not in mixedCase
Parameter Token.mintToFarm(address)._farm (contracts/Kokomo.sol#926) is not in
mixedCase
Function Token.__mint(address,uint256) (contracts/Kokomo.sol#931-934) is not in
mixedCase
Parameter Token.__mint(address,uint256)._to (contracts/Kokomo.sol#931) is not in
mixedCase
Parameter Token.__mint(address,uint256)._amount (contracts/Kokomo.sol#931) is not in
mixedCase
Variable Token._delegates (contracts/Kokomo.sol#947) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

Redundant expression "this (contracts/Kokomo.sol#21)" inContext (contracts/
Kokomo.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

Token.communityToken (contracts/Kokomo.sol#905) should be immutable
Token.seedRoundLockToken (contracts/Kokomo.sol#906) should be immutable
Token.teamLockToken (contracts/Kokomo.sol#904) should be immutable
Token.teamVaultToken (contracts/Kokomo.sol#907) should be immutable
TokenVesting.beneficiary (contracts/Kokomo.sol#820) should be immutable
TokenVesting.cliff (contracts/Kokomo.sol#822) should be immutable
TokenVesting.duration (contracts/Kokomo.sol#824) should be immutable
TokenVesting.start (contracts/Kokomo.sol#823) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-immutable
. analyzed (9 contracts with 84 detectors), 55 result(s) found
```

0x Guard