



SMART CONTRACT SECURITY AUDIT

Koji

July, 2021

Table of Contents

Table of Contents	2
Disclaimer	3
Procedure	4
Terminology	5
Limitations	5
Token Contract Details for 22.07.2021	6
Audit Details	6
Koji Token Distribution	7
Contract Function Details	8
Vulnerabilities checking	11
Security Issues	12
Conclusion	14

Disclaimer

This is a comprehensive report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws. We took into consideration smart contract based algorithms, as well. Reading the full analysis report is essential to build your understanding of project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on the our research and cannot claim what it states or how we created it. Before making any judgments, you have to conduct your own independent research. We will discuss this in more depth in the following disclaimer - please read it fully.

DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Soken and its affiliates shall not be held responsible to you or anyone else, nor shall Soken provide any guarantee or representation to any person with regard to the accuracy or integrity of the report. Without any terms, warranties or other conditions other than as set forth in that exclusion and Soken excludes hereby all representations, warrants, conditions and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills). The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Soken disclaims all responsibility and responsibilities and no claim against Soken is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential or pure economic losses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent).

Security analysis is based only on the smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

Procedure

Our analysis contains following steps:

1. Project Analysis;
2. Manual analysis of smart contracts:
 - Deploying smart contracts on any of the network(Ropsten/Rinkeby) using Remix IDE
 - Hashes of all transaction will be recorded
 - Behaviour of functions and gas consumption is noted, as well.
3. Unit Testing:
 - Smart contract functions will be unit tested on multiple parameters and under multiple conditions to ensure that all paths of functions are functioning as intended.
 - In this phase intended behaviour of smart contract is verified.
 - In this phase, we would also ensure that smart contract functions are not consuming unnecessary gas.
 - Gas limits of functions will be verified in this stage.
4. Automated Testing:
 - Mythril
 - Oyente
 - Manticore
 - Solgraph

Terminology

We categorize the finding into 4 categories based on their vulnerability:

- Low-severity issue — less important, must be analyzed
- Medium-severity issue — important, needs to be analyzed and fixed
- High-severity issue — important, might cause vulnerabilities, must be analyzed and fixed
- Critical-severity issue — serious bug causes, must be analyzed and fixed.

Limitations

The security audit of Smart Contract cannot cover all vulnerabilities. Even if no vulnerabilities are detected in the audit, there is no guarantee that future smart contracts are safe. Smart contracts are in most cases safeguarded against specific sorts of attacks. In order to find as many flaws as possible, we carried out a comprehensive smart contract audit. Audit is a document that is not legally binding and guarantees nothing.

Token Contract Details for 22.07.2021

Contract Name: **Koji**

Deployer address: **0x1c8266a4369af6d80df2659ba47b3c98f35cb8be**

Total Supply: **1,000,000,000,000**

Token Tracker: **KOJ**

Decimals: **18**

Token holders: **3981**

Transactions count: **13,378**

Top 100 holders dominance: **78.96%**

Contract deployer address:
0x1c8266a4369af6d80df2659ba47b3c98f35cb8be

Audit Details



Project Name: **Koji**

Language: **Solidity**

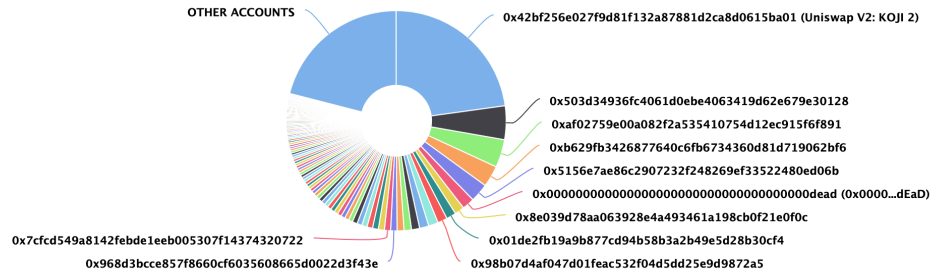
Blockchain: **Ethereum**

Project Website: **koji.earth**

Koji Token Distribution

Koji Top 100 Token Holders

Source: Etherscan.io



Koji Top 10 Holders

Rank	Address	Quantity (Token)	Percentage
1	Uniswap V2: KOJI 2	229,787,274,673.51573949498094589	22.9787%
2	0x503d34936fc4061d0ebe4063419d62e679e30128	49,431,872,427.983539094650205762	4.9432%
3	0xaf02759e00a082f2a535410754d12ec915f6f891	41,170,998,298.421583063037457918	4.1171%
4	0xb629fb3426877640c6fb6734360d81d719062bf6	31,357,143,345.303493687536689624	3.1357%
5	0x5156e7ae86c2907232f248269ef33522480ed06b	27,217,862,572.016460905349794238	2.7218%
6	0x0000...dEaD	17,910,079,815.582900845054700116	1.7910%
7	0x8e039d78aa063928e4a493461a198cb0f21e0f0c	14,247,430,085.442946925013569336	1.4247%
8	0x01de2fb19a9b877cd94b58b3a2b49e5d28b30cf4	14,242,709,351.493900379904083337	1.4243%
9	0x98b07d4af047d01feac532f04d5dd25e9d9872a5	14,241,380,400.093279532713370562	1.4241%
10	0x520aa93968396fc22de3f49038c1e9f6b4a026e0	14,235,518,885.397538931935646045	1.4236%

Contract Function Details

- + Context.sol
 - [Int] _msgSender
 - [Int] _msgData
- + IERC20.sol
 - [Ext] transfer #
 - [Ext] allowance #
 - [Ext] approve
 - [Ext] transferFrom #
 - [Ext] totalSupply
 - [Ext] balanceOf
- + [Lib] SafeMath
 - [Int] add
 - [Int] sub
 - [Int] sub
 - [Int] mul
 - [Int] div
 - [Int] div
 - [Int] mod
 - [Int] mod
- + [Lib] Address.sol
 - [Int] isContract
 - [Int] sendValue #
 - [Int] functionCall #
 - [Int] functionCall #
 - [Int] functionCallWithValue #
 - [Int] functionCallWithValue #
 - [Int] functionStaticCall #
 - [Int] functionStaticCall #
 - [Int] _verifyCallResult #
- + Ownable.sol (Context)
 - [Pub] <Constructor> #
 - [Pub] owner
 - [Pub] onlyOwner
 - [Pub] renounceOwnership #
 - modifiers: onlyOwner
 - [Pub] transferOwnership #
 - modifiers: onlyOwner

+ Kojitoken.sol

- [Ext] name
- [Ext] symbol
- [Ext] decimals
- [Ext] totalSupply
- [Pub] balanceOf
- [Pub] allowance
- [Pub] approve #
- [Pub] transfer #
- [Pub] transferFrom #
- [Pub] increaseAllowance
- [Pub] decreaseAllowance
- [Pub] isExcludedFromRewards
- [Pub] isExcludedFromFees
- [Pub] totalFees
- [Pub] totalHolderFees
- [Pub] totalCharityFees
- [Pub] totalBurnFees
- [Pub] totalAdminFees
- [Pub] distribute
- [Pub] excludeFromFees
- [Pub] includeInFees
- [Pub] excludeFromRewards
- [Pub] includeInRewards
- [Pub] rewardsFromToken
- [Pub] tokenWithRewards

- [Prv] _approve
- [Prv] _transfer
- [Prv] _transferWithRewards
- [Prv] _transferWithRecipientRewards
- [Prv] _transferWithoutSenderRewards
- [Prv] _transferWithoutRewards
- [Prv] _updateHolderFee
- [Prv] _updateCharityFee
- [Prv] _updateBurnFee
- [Prv] _updateAdminFee
- [Prv] _getValues
- [Prv] _getActualValues
- [Prv] _getRewardValues
- [Prv] _getRewardsRate
- [Prv] _getCurrentSupply
- [Prv] _getFee
- [Prv] _getHolderFee

- [Prv] _getCharityFee
- [Prv] _getBurnFee
- [Prv] _getAdminFee

- [Ext] setTaxPercentage
- [Ext] setTaxAllocations
- [Ext] setCharityAddress
- [Ext] setBurnAddress
- [Ext] setAdminAddress

(\$) = payable function

= non-constant function

Vulnerabilities checking

Issue Description	Checking Status
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed
Timestamp Dependence	Completed
Shadowing State Variables	Completed
DoS with Failed Call	Completed
DoS with Block Gas Limit	Low-issues
Outdated Compiler Version	Completed
Assert Violation	Completed
Use of Deprecated Solidity Functions	Completed
Integer Overflow and Underflow	Completed
Function Default Visibility	Completed
Malicious Event Log	Completed
Math Accuracy	Completed
Design Logic	Completed
Fallback Function Security	Completed
Cross-function Race Conditions	Completed
Safe Zeppelin Module	Completed

Security Issues

1) Owner Privileges over tax allocations:

Owner can change tax allocations using the function setTaxAllocations.

```
function setTaxAllocations(
    uint256 _holderTaxAlloc,
    uint256 _charityTaxAlloc,
    uint256 _burnTaxAlloc,
    uint256 _adminTaxAlloc
) external onlyOwner {
    totalTaxAlloc = _holderTaxAlloc.add(_charityTaxAlloc).add(_burnTaxAlloc).add(_adminTaxAlloc);
}
```

2) Out of Gas issue:

```
function includeInRewards(address _account) public onlyOwner() {
    require(excludedFromRewards[_account], "Account is already included in rewards");

    for (uint256 i = 0; i < rewardExcluded.length; i++) {
        if (rewardExcluded[i] == _account) {
            rewardExcluded[i] = rewardExcluded[rewardExcluded.length - 1];
            actual[_account] = 0;
            excludedFromRewards[_account] = false;
            rewardExcluded.pop();
            break;
        }
    }
}
```

The function includeInRewards() uses the loop to find and remove addresses from the _excluded list. Function will be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.

3) Out of Gas issue:

```
function _getCurrentSupply() private view returns (uint256, uint256) {
    uint256 rewardsSupply = rewardsTotal;
    uint256 actualSupply = ACTUAL_TOTAL;

    for (uint256 i = 0; i < rewardExcluded.length; i++) {
        if (rewards[rewardExcluded[i]] > rewardsSupply || actual[rewardExcluded[i]] > actualSupply) {
            return (rewardsTotal, ACTUAL_TOTAL);
        }

        rewardsSupply = rewardsSupply.sub(rewards[rewardExcluded[i]]);
        actualSupply = actualSupply.sub(actual[rewardExcluded[i]]);
    }

    if (rewardsSupply < rewardsTotal.div(ACTUAL_TOTAL)) {
        return (rewardsTotal, ACTUAL_TOTAL);
    }
}
```

The function _getCurrentSupply also uses the loop for evaluating total supply. It also could be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.

Recommendation:

Use EnumerableSet instead of array or do not use long arrays.

Conclusion

Low-severity issues exist within smart contracts. Smart contracts are free from any critical or high-severity issues.

NOTE: Please check the disclaimer above and note, that audit makes no statements or warranties on business model, investment attractiveness or code sustainability.