



Security Assessment

# KokomoSwap

May 26th, 2021



# Table of Contents

## Summary

### Overview

Project Summary

Audit Summary

Vulnerability Summary

Audit Scope

### Findings

KPK-01 : Divide by Zero

KRS-01 : Incompatibility With Deflationary Tokens

MCK-01 : Variable Naming Convention

MCK-02 : Missing Emit Events

MCK-03 : Centralized Control of Bonus Multiplier

MCK-04 : add() Function Not Restricted

MCK-05 : Recommended Explicit Pool Validity Checks

MCK-06 : Incompatibility With Deflationary Tokens

MCK-07 : Public 'premint' function

### Formal Verification Requests

### Appendix

### Disclaimer

### About

# Summary

This report has been prepared for KokomoSwap smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

# Overview

## Project Summary

Project Name	KokomoSwap
Platform	BSC
Language	Solidity
Codebase	1. <a href="https://github.com/KokomoSwap/kokomo-swap-core">https://github.com/KokomoSwap/kokomo-swap-core</a> 2. <a href="https://github.com/KokomoSwap/kokomo-swap-periphery">https://github.com/KokomoSwap/kokomo-swap-periphery</a> 3. <a href="https://github.com/KokomoSwap/kokomo-farm">https://github.com/KokomoSwap/kokomo-farm</a>
Commits	KokomoSwap

## Audit Summary

Delivery Date	May 26, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	Farming, Router, Pool, Factory

## Vulnerability Summary

Total Issues	9
● Critical	0
● Major	1
● Medium	0
● Minor	3
● Informational	5
● Discussion	0

## Audit Scope

ID	file	SHA256 Checksum
KTk	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/KokomoToken.sol	4b945f3be319c55e8845ec83f1933a870f1b9d5e43c74bae068ded2c7d85b5f3
MCK	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/MasterChef.sol	7d3a45216872e7270f2571ae5ed60824e4a6b4f309c9d7d5b5ddc8b4b1ab4e19
SBK	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/SyrupBar.sol	46a90ae8fdcf41bb4f9577d443cb796061481009083f101b5f5d3095c98e9d15
TKS	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/Timelock.sol	cf5acefe3031a90d38017656bac9042404b8d35243533a09ad9ccdad2d2823a42
MKS	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/libs/Migrations.sol	7a30322962f749bd265e1d30f4d723034f9122663c9d94d623beffd4df867dfd
MBE	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/libs/MockBEP20.sol	4f2043f31a1e6e1418a78028b187672db977471f70219998e3bb0eeb3034f570
MKK	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/libs/Multicall.sol	3f247d59d93c1f7b6ab346154294384ec1ef160eba60acfd3d19bd9f1df702e6
WBN	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/libs/WBNB.sol	92020f2413c24891fa83b1e47ce5ff0471eb16618322bb2cb2af8ec5ede8d660
KER	projects/KokomoSwap/contracts/kokomo-swap-core-main/contracts/KokomoERC20.sol	c3e7839071ad0e4653c716cc6b42b97a948abec00b0ee02e9b94e5d340cf7f33
KFK	projects/KokomoSwap/contracts/kokomo-swap-core-main/contracts/KokomoFactory.sol	2d66a36d5081eecb13d50941082a8342bdd6cacf3c118769eb51d43203e8fbc8
KPK	projects/KokomoSwap/contracts/kokomo-swap-core-main/contracts/KokomoPair.sol	7d9e74148ec12dbe28d236100edb3d35dfcc54fcd0a23028716a4a8c2800955a
MKC	projects/KokomoSwap/contracts/kokomo-swap-core-main/contracts/Migration.sol	610064fcc750b02061bf4e6b7666a4ac791fa137069ea7cf25848a9aeb3ed048
IER	projects/KokomoSwap/contracts/kokomo-swap-core-main/contracts/interfaces/IERC20.sol	2b63f199f838028184efefbcfd6cf2b9192624c3dae5dc1116ecbb15c36a67e8

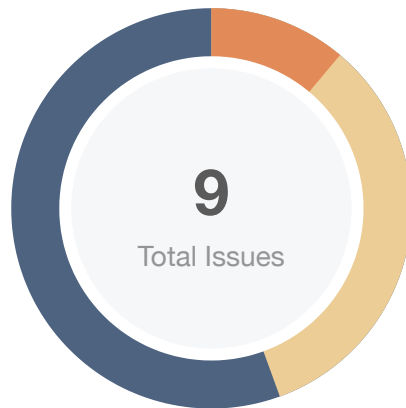
ID	file	SHA256 Checksum
IKC	projects/KokomoSwap/contracts/kokomo-swap-core-main/contracts/interfaces/IKomooCallee.sol	a7ce92b219d84f7e9026aae769befbe849fbcf459ad63d3a935bde55697bfbd7
IKE	projects/KokomoSwap/contracts/kokomo-swap-core-main/contracts/interfaces/IKomooERC20.sol	c0f80e06adf15191ac92244b2b8e90d425fac8d16c9cf93508fcbd33eebc810d
IKF	projects/KokomoSwap/contracts/kokomo-swap-core-main/contracts/interfaces/IKomooFactory.sol	96d16ac8ae01f4f0a2f91b8d19ee2a5eabba108bea12c4c16806a8607eb87a83
IKP	projects/KokomoSwap/contracts/kokomo-swap-core-main/contracts/interfaces/IKomooPair.sol	7605cb335d852b104fd0aa75ae875237c1c354149f10e7f7c2cf266cd90aa672
KMK	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/KokomoMigrator.sol	8de4988c8ab184c91bdc6ea22809ef0c25f317e34ea5340220d37308f97804c2
KRK	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/KokomoRouter.sol	5c9c79428f1278f39ad3194dace1d4781fa5f03de2116a3557db85cda898c1b3
KRS	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/KokomoRouter01.sol	1a0a7f1f09c8ecc6e1df72f0ce490de8f3481759509fdaa5128d13d6782eab13
MKP	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/Migrations.sol	610064fcc750b02061bf4e6b7666a4ac791fa137069ea7cf25848a9aeb3ed048
BEP	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/interfaces/BEP20.sol	95039c0e35d707c20dd10c4322b6140a877c45f62c4e8ec65483d8890930ad88
IEC	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/interfaces/IERC20.sol	2b63f199f838028184efefbcfd6cf2b9192624c3dae5dc1116ecbb15c36a67e8
IKK	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/interfaces/IKokomoFactory.sol	96d16ac8ae01f4f0a2f91b8d19ee2a5eabba108bea12c4c16806a8607eb87a83

ID	file	SHA256 Checksum
IKM	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/interfaces/IKokomoMigrator.sol	a48a891e7ca2b51219e4e99e75a49fff50361de6fb10658944615d5b36026ce6
IKS	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/interfaces/IKokomoPair.sol	7605cb335d852b104fd0aa75ae875237c1c354149f10e7f7c2cf266cd90aa672
IKR	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/interfaces/IKokomoRouter01.sol	f6fce7088fd1390381724c36dc0e9d7893a2d5c9c6237fcb5fb9d777733d6833
IRK	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/interfaces/IKokomoRouter02.sol	dcb178022f209d2e9ea0e1c019e9ad1ae92ef7b754efa66d62c9ac23252c8178
IWE	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/interfaces/IWETH.sol	60760053849b916b72386fef1126ab69c7482c2aa6b5fb836368eff42905cb30
IUV	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/interfaces/V1/IUniswapV1Exchange.sol	af0f040756cfdad159024068160e3d29974963eea113bf79debe6e0d9bfc5f4
IUF	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/interfaces/V1/IUniswapV1Factory.sol	757faaa23cac0f415640c16c009e92458271ece8876c0b739f2a3bbad6338aee
BKS	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/libraries/Babylonian.sol	55f7f97f332b408835ff07a374bef0a8ef698abe282de76a259f2ea6b7d210b4
BMK	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/libraries/BitMath.sol	7ff2edbf176ee81b32675d2eda5936bf7dcae3d23ea1dabd3a8006ec9a565677
FPK	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/libraries/FixedPoint.sol	6d36a038b5961c5ad887fdbf48dd1a3c83b54e3fdcbe9149f3e027bae374d8df
FMK	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/libraries/FullMath.sol	d42f6802b4f03e14047e3ef38a68a5b98a756b1ee051c92dbc1dabc6ba68f9a3

ID	file	SHA256 Checksum
KLK	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/libraries/KokomoLibrary.sol	4f592b4b5b7637b5bc8d96a99777e038aa6c5ca8e993a03f41bc2adea332e9d7
KOL	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/libraries/KokomoOracleLibrary.sol	06f4ef2b3c52c16a41b3aaed269728159bb9cad9e09d44dbdc86b31d2e260459
SMK	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/libraries/SafeMath.sol	39e5b4c0bc19b72fa59c2d2177ba5ed6cfadcd76f3f804563011e579367530fb
THK	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/libraries/TransferHelper.sol	22b87fd425d590e533ab7e52478cf72bdc4bde2672e0977c7eff7742e8f0737d



# Findings



Critical	0 (0.00%)
Major	1 (11.11%)
Medium	0 (0.00%)
Minor	3 (33.33%)
Informational	5 (55.56%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
KPK-01	Divide by Zero	Logical Issue	Minor	ⓘ Acknowledged
KRS-01	Incompatibility With Deflationary Tokens	Logical Issue	Minor	ⓘ Acknowledged
MCK-01	Variable Naming Convention	Coding Style	Informational	ⓘ Acknowledged
MCK-02	Missing Emit Events	Gas Optimization	Informational	ⓘ Acknowledged
MCK-03	Centralized Control of Bonus Multiplier	Logical Issue	Informational	ⓘ Acknowledged
MCK-04	add() Function Not Restricted	Volatile Code	Major	ⓘ Acknowledged
MCK-05	Recommended Explicit Pool Validity Checks	Logical Issue	Informational	ⓘ Acknowledged
MCK-06	Incompatibility With Deflationary Tokens	Logical Issue	Minor	ⓘ Acknowledged
MCK-07	Public 'premint' function	Control Flow	Informational	ⓘ Acknowledged

## KPK-01 | Divide by Zero

Category	Severity	Location	Status
Logical Issue	Minor	projects/KokomoSwap/contracts/kokomo-swap-core-main/contracts/KokomoPair.sol: 144	ⓘ Acknowledged

### Description

The call to `burn()` function will fail if the value of `totalSupply` is 0.

### Recommendation

We advise the client to add the following validation in the function `burn()`

```
1 require(totalSupply != 0, "The value of totalSupply must not be 0");
```

### Alleviation

**[KokomoSwap Team]:** KokomoSwap Team has confirmed that this is a minor issue. This is not a common case, and if it happens, the transaction will be reverted.

## KRS-01 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Minor	projects/KokomoSwap/contracts/kokomo-swap-periphery-main/contracts/KokomoRouter01.sol: 70, 71, 91, 109, 188, 200, 226, 240	ⓘ Acknowledged

### Description

The users add, remove or swap LP tokens into the router, and the `mint`, `burn` and `swap` operations are performed. When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. As a result, the amount inconsistency will occur and the transaction may fail due to the validation checks.

### Recommendation

We advise the client to regulate the set of LP tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

### Alleviation

**[KokomoSwap Team]:** KokomoSwap Team has confirmed. This is an issue of KokomoRouter01.sol, KokomoSwap currently does not use it.

## MCK-01 | Variable Naming Convention

Category	Severity	Location	Status
Coding Style	● Informational	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/MasterChef.sol: 66	ⓘ Acknowledged

### Description

The linked variables do not conform to the standard naming convention of Solidity whereby functions and variable names utilize the format unless variables are declared as constant in which case they utilize the format.

### Recommendation

We advise that the naming conventions utilized by the linked statements are adjusted to reflect the correct type of declaration according to the Solidity style guide.

### Alleviation

**[KokomoSwap Team]:** KokomoSwap Team has confirmed that this is a minor issue.

## MCK-02 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	● Informational	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/MasterChef.sol: 114, 166	ⓘ Acknowledged

### Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `team()`
- `setMigrator()`
- `updateMultiplier()`

### Recommendation

Consider adding events for sensitive actions, and emit them in the function.

```
1 event Setteam(address indexed user, address indexed _teamaddr);
2
3 function team(address _devaddr) public {
4     require(msg.sender == devaddr, "team: wut?");
5     teamaddr = _teamaddr;
6     emit Setteam(msg.sender, _teamaddr);
7 }
```

### Alleviation

**[KokomoSwap Team]:** The ownership of the MasterChef belongs to the Timelock. There is at least a 6-hour timelock on the MasterChef contract with regards to all pool reward changes. The team will notify the community if there are any changes. Additionally, `team()` and `setMigrator()` will be used under rare cases. `updateMultiplier()` will not be used after June 2021.

## MCK-03 | Centralized Control of Bonus Multiplier

Category	Severity	Location	Status
Logical Issue	● Informational	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/MasterChef.sol: 115	ⓘ Acknowledged

### Description

The function can alter the `BONUS_MULTIPLIER` variable and consequently the output of which is directly utilized for the minting of new tokens.

### Recommendation

This is the intended functionality of the protocol, however, users should be aware of this functionality.

### Alleviation

**[KokomoSwap Team]:** The ownership of the MasterChef belongs to the Timelock. There is a 6-hour timelock on the MasterChef contract with regards to all pool reward changes. The team will notify the community if there are any changes. `updateMultiplier()` will not be used after June 2021

## MCK-04 | add() Function Not Restricted

Category	Severity	Location	Status
Volatile Code	● Major	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/MasterChef.sol: 124	ⓘ Acknowledged

### Description

The comment in line L123, mentioned `// XXX DO NOT add the same LP token more than once`. Rewards will be messed up if you do.

The total amount of KOKOMO Reward in function `updatePool()` will be incorrectly calculated if the same LP token is added into the pool more than once in function `add()`.

However, the code is not reflected in the comment behaviors as there isn't any valid restriction on preventing this issue.

The current implementation is relying on the trust of the owner to avoid repeatedly adding the same LP token to the pool, as the function will only be called by the owner.

### Recommendation

Detect whether the given pool for addition is a duplicate of an existing pool. The pool addition is only successful when there is no duplicate. Using a mapping of `addresses -> bools`, which can restricted the same address being added twice.

### Alleviation

**[KokomoSwap Team]:** It will be deployed with a 6-hour Timelock from the MasterChef, `add()` will be deployed with double-checking of the function. In case of incorrect input, there will be 6 hours to revert.

## MCK-05 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	● Informational	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/MasterChef.sol: 140	ⓘ Acknowledged

### Description

There's no sanity check to validate if a pool is existing.

### Recommendation

We advise the client to adopt following modifier `validatePoolByPid` to functions `set()`, `migrate()`, `deposit()`, `withdraw()`, `emergencyWithdraw()`, `pendingKokomo()` and `updatePool()`.


```
1 modifier validatePoolByPid(uint256 _pid) {  
2     require (_pid < poolInfo . length , "Pool does not exist") ;  
3     _;  
4 }
```

### Alleviation

**[KokomoSwap Team]:** It will be deployed with a 6-hour Timelock from the MasterChef, `set()` will be deployed with double-checking of the function. In case of incorrect input, there will be 6 hours to revert.



## MCK-06 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Minor	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/MasterChef.sol: 232, 254	 Acknowledged

### Description

The MasterChef contract operates as the main entry for interaction with staking users. The staking users deposit LP tokens into the Kokomo Swap pool and in return get proportionate share of the pool's rewards. Later on, the staking users can withdraw their own assets from the pool. In this procedure, `deposit()` and `withdraw()` are involved in transferring users' assets into (or out of) the Kokomo Swap protocol. When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged (and burned) transaction fee. As a result, this may not meet the assumption behind these low-level asset-transferring routines and will bring unexpected balance inconsistencies.

### Recommendation

Regulate the set of LP tokens supported in Kokomoswap and, if there is a need to support deflationary tokens, add necessary mitigation mechanisms to keep track of accurate balances.

### Alleviation

**[KokomoSwap Team]:** `deposit()`, `withdraw()` function only handle LP Tokens. They are not deflationary tokens. Thus, the KokomoSwap team has acknowledged that it is not a related issue. Also, our team has no plan for a Deflationary token listing.

## MCK-07 | Public 'premint' function

Category	Severity	Location	Status
Control Flow	● Informational	projects/KokomoSwap/contracts/kokomo-farm-main/contracts/MasterChef.sol: 344	ⓘ Acknowledged

### Description

The `premint` function is public, which means everyone can call this function. Although the function is restricted by the `premintCompleted` variable, functions that involve token minting and burning should be restarted to high privilege users.

### Recommendation

Add `onlyOwner` modifier to the `premint` function.

### Alleviation

**[KokomoSwap Team]:** The `premint` function is only allowed once via `1) require(premintCompleted == false, 'already preminted');` code. It is already fixed to a certain address for minting. Therefore no risk of the function is public.

# Appendix

## Finding Categories

### Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

### Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

### Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

### Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

### Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

## Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux `"sha256sum"` command against the target file.

# 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 CertiK's 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 CertiK 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.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

## About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

