CERTIK

# CertiK Audit Report for Bakery Swap

# Contents

# 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 Verification Services Agreement between CertiK and Bakery Swap (the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, 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.

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are 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 review.

CertIk Reports do 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.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing 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 CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, CertiK's mission of every audit is to apply different approaches and detection methods, ranging from manual, static, and dynamic analysis, to ensure that projects are checked against known attacks and potential vulnerabilities. CertiK leverages a team of seasoned engineers and security auditors to apply testing methodologies and assessments to each project, in turn creating a more secure and robust software system.

CertiK has served more than 100 clients with high quality auditing and consulting services, ranging from stablecoins such as Binance's BGBP and Paxos Gold to decentralized oracles such as Band Protocol and Tellor. CertiK customizes its engineering tool kits, while applying cutting-edge research on smart contracts, for each client on its project to offer a high quality deliverable.  For more information: https://certik.io.

# Executive Summary

This report has been prepared for **Bakery Swap** to discover issues and vulnerabilities in the delta of their source code in comparison to the Uniswap and SushiSwap implementations. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

**This audit should not be treated as an audit of the parent implementations of Uniswap and SushiSwap** as its sole purpose is to **verify the security of the source code differences** rather than the original source code that was carried over from those two projects. **We neither endorse nor guarantee the security of the Uniswap and SushiSwap mechanisms in any way.**

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.

# Testing Summary

Smart Contract Audit

This report has been prepared as a product of the Smart Contract Audit request by Bakery Swap.

This audit was conducted to discover issues and vulnerabilities in the source code of Bakery Swap's Smart Contracts.

| | |
|---|---|
| TYPE | Smart Contract |
| SOURCE CODE | https://github.com/BakeryProject/bakery-swap-periphery<br>https://github.com/BakeryProject/bakery-swap-core<br>https://github.com/BakeryProject/bakery-swap-lib |
| PLATFORM | EVM |
| LANGUAGE | Solidity |
| REQUEST DATE | Sept 11, 2020 |
| DELIVERY DATE | Sept 17, 2020 |
| METHODS | A comprehensive examination has been performed using Dynamic Analysis, Static Analysis, and Manual Review. |

# Review Notes

## Introduction

CertiK team was contracted by the Bakery Swap team to audit the design and implementation of the newly introduced functionalities they have coded in the smart contracts of Uniswap and SushiSwap.

The audited source code links are:

- https://github.com/BakeryProject/bakery-swap-periphery/tree/57302536083c08ec5cde90ea2b3e2cf3d4d6149c
- https://github.com/BakeryProject/bakery-swap-core/tree/904f7dc210ed45f30b602068efc94b277d01fa0e
- https://github.com/BakeryProject/bakery-swap-lib/tree/113a2c18165a63074c8ec1ef260f41ba61c1e855

The goal of this audit was to review the Solidity implementation for its business model, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

The findings of the initial audit have been conveyed to the team behind the contract implementations and the source code is expected to be re-evaluated before another round of auditing has been carried out.

**The security analysis was constrained to the delta of the source codes rather than the overall implementation of Uniswap and SushiSwap and this report in no shape or form guarantees or**

**endorses the safety and security of the Uniswap and SushiSwap protocol code. The audit's goal was to ensure the source code deltas introduced by BakerySwap are safe.**

## Documentation

The sources of truth regarding the operation of the contracts in scope were derived from their parent implementations by Uniswap and SushiSwap. To help aid our understanding of each contract's functionality we referred to in-line comments and naming conventions.

## Summary

The codebase of the project is a DeFi fork of the Uniswap and SushiSwap implementations with altered reward calculation mechanisms.

**Certain optimization steps** that we pinpointed in the source code mostly referred to coding standards and inefficiencies, however no flaws were identified in the delta of the source code. The delta of the source code of BakerySwap and SushiSwap / Uniswap was minimal and concerned the reward mechanisms rather than any core operation of the protocol and as such, the core operations were left out of scope such as the swapping of tokens etc.

The BakerySwap team decided to launch their product prior to the completion of the audit and as such, it was deemed impractical to update the codebase according to our optimizational findings as the project was already live by the time the audit concluded.

## Recommendations

Overall, the codebase of the contracts should be refactored to assimilate the findings of this report **to achieve a high standard of code quality and security**.

# Findings

## Exhibit 1

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Variable Mutability Specifier | Language Specific Issue | Informational | BakeryMaster.sol: L37-L38, L41-L56, L59-L60 |

**[INFORMATIONAL] Description:**

The linked variable declarations are only assigned to once during the constructor of the contract.

**Recommendations:**

The linked variable declarations can safely be declared as `immutable` to greatly reduce the gas cost of operating the overall contract.

## Exhibit 2

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Variable to `constant` | Optimization | Informational | BakeryMaster.sol: L57-L58 |

**[INFORMATIONAL] Description:**

The linked variable is only assigned to once during its instantiation at the contract level.

**Recommendations:**

The linked variable can safely be declared as a constant and have its naming convention adjusted as such, greatly optimizing the gas cost involved in utilizing it in the process.

## Exhibit 3

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Redundant `bool` Variable | Optimization | Informational | BakeryMaster.sol: L35 |

**[INFORMATIONAL] Description:**

The newly added boolean variable is meant to represent whether an entry to the new poolInfoMap mapping exists to prevent duplicate calls to the add function. While this functions as intended, it is possible to utilize the already-existent non-zero variable `lastRewardBlock` for this particular purpose.

**Recommendations:**

As the `lastRewardBlock` assignment will always be non-zero if a pool is added due to L123, it is possible to require that the variable is zero before assigning to it thus making the existing variable redundant.

This would affect all `pool.exists` lookups which should be replaced with `pool.lastRewardBlock != 0`.

## Exhibit 4

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Inefficient Per-Member Assignment | Optimization | Informational | BakeryMaster.sol: L121-L128 |

**[INFORMATIONAL] Description:**

Each instruction on the EVM costs gas to execute and in this case, each member of the struct is independently written to instead of conducting a single mass-write by instantiating a `PoolInfo` declaration in `memory` and subsequently assigning it to `poolInfoMap[_pair]`.

**Recommendations:**

We advise the linked code segment to instead only lookup, depending on Exhibit-3, the correct member of the struct necessary from `storage` and conduct the rest of the operations in `memory` before writing the final result to `storage`.

## Exhibit 5

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Inefficient Greater-Than Comparison w/ Zero | Optimization | Informational | BakeryMaster.sol: L169, L175, L194, L322, L326, L345, L348 |

**[INFORMATIONAL] Description:**

The linked statements conduct a greater-than (>) comparison between an unsigned integer and the value literal 0.

**Recommendations:**

As unsigned integers are restricted to the non-negative range, it is possible to convert these greater-than comparisons to inequality comparisons which are more gas efficient.

## Exhibit 6

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Inefficient Conditional | Optimization | Informational | BakeryMaster.sol: L171, L177, L209 |

**[INFORMATIONAL] Description:**

The conditionals check whether the uint256 variables are less than the literal 1. This is only the case when the value is 0.

**Recommendations:**

Those `if` conditionals can be converted to equalities with the value literal 0.

## Exhibit 7

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Inefficient Case | Optimization | Informational | BakeryMaster.sol: L194 |

**[INFORMATIONAL] Description:**

The linked case ensures that `fromBulkNumber` is greater-than (>) the literal 0 before multiplying it with a value in the ternary operator after being subtracted by 1. Since the value is subtracted by 1, in the case that `fromBlockNumber` is 1 the multiplication would again yield 0.

**Recommendations:**

It is more optimal to change the conditional to a greater-than (>) comparison with the literal 1.

## Exhibit 8

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Inconsistent Transaction Ordering | Optimization | Informational | BakeryMaster.sol: L212, L217-L219 |

**[INFORMATIONAL] Description:**

The linked mathematical statements are conducted in a different order than each other, with the former being a subtraction - multiplication - division and the latter being a subtraction - division (w/ subtraction rounding) - multiplication. In Solidity, the order of mathematical execution is important as there is no concept of a decimal number and divisions are rounded down.

**Recommendations:**

We advise that the proper function of the mathematical formula defined here is evaluated.

## Exhibit 9

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Function Plot | Optimization | Informational | BakeryMaster.sol: L232-L268 |

**[INFORMATIONAL] Description:**

We have taken note of an extensive test suite located in the repository, however we advise that fuzz-based testing is introduced to ensure that the values retrieved from the reward calculation follow the plot you have set within the calculations.

**Recommendations:**

It may be optimal to plot the reward line as well by testing _from and _to values linearly.

## Exhibit 10

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Manual Zeroing | Optimization | Informational | BakeryMaster.sol: L361-L362 |

**[INFORMATIONAL] Description:**

The manual zeroing operations could instead be replaced by a delete operation on `poolUserInfoMap[_pair][msg.sender]` which is more readable and gas efficient.

**Recommendations:**

See Description.

## Exhibit 11

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Utilization of `storage` Variable Instead of `memory` | Optimization | Informational | BakerySwapFactory.sol: L18 |

**[INFORMATIONAL] Description:**

The linked newly introduced assignment should utilize the already existent `_feeToSetter` variable in `memory` rather than retrieving the newly stored `feeToSetter` variable from `storage`, optimizing the gas cost involved in deploying the contract.

**Recommendations:**

See Description.

## Exhibit 12

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Fork Leftovers | Optimization | Informational | PairNamer.sol: L9, L31 |

**[INFORMATIONAL] Description:**

The linked segments contain the unicode icon 🦄 as a leftover from Uniswap.

**Recommendations:**

We advise that another unicode icon is utilized for the BakerySwap project, such as 🍩 or 🥐.

## Exhibit 13

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | BakeryToken.sol |

**[INFORMATIONAL] Description:**

No findings were found at this stage of the audit as the delta between SushiSwap and this implementation is inexistent apart from a function rename from `mint` to `mintTo` to avoid the collision. We should note that this results in a token that exposes both a `mint` and `mintTo` function to the owner of the contract, however these functions are identical and only invokable by the Owner of the contract. Regardless, **we advise that the function is renamed to `mint`** to ensure that no unintended consequences occur with the token.

## Exhibit 14

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | BakerySwapRouter.sol |

**[INFORMATIONAL] Description:**

No findings were found at this stage of the audit as the delta between “UniswapV2” and this implementation is inexistent apart from an omitted superfluous variable to `IBakerySwapPair` swap function calls and a renaming of all `ETH` type function and variable names to `BNB` type i.e. `WBNB` instead of `WETH`.

## Exhibit 15

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | BakerySwapBEP20.sol |

**[INFORMATIONAL] Description:**

No findings were found at this stage of the audit as the delta between “UniswapV2” and this implementation is inexistent apart from brand renaming in error messages as well as the `symbol` and `name` variables.

## Exhibit 16

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | BakerySwapFactory.sol |

**[INFORMATIONAL] Description:**

The delta between “UniswapV2” and this implementation is inexistent apart from brand renaming in error messages as well as the inclusion of a default value for the `feeTo` variable being set to the same value as `feeToSetter`.

## Exhibit 17

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | BakerySwapPair.sol |

**[INFORMATIONAL] Description:**

No findings were found at this stage of the audit as the delta between “UniswapV2” and this implementation is inexistent apart from brand renaming in the codebase as well as an adjustment of the swap function to not accept a data argument as it was optional and deemed unnecessary.

## Exhibit 18

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | PairNamer.sol |

**[INFORMATIONAL] Description:**

The delta between “UniswapV2” and this implementation is inexistent apart from brand renaming.

## Exhibit 19

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | SafeBEP20Namer.sol |

**[INFORMATIONAL] Description:**

No findings were found at this stage of the audit as the delta between “UniswapV2” and this implementation is inexistent apart from brand renaming.

## Exhibit 20

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | TransferHelper.sol |

**[INFORMATIONAL] Description:**

No findings were found at this stage of the audit as the delta between “UniswapV2” and this implementation is inexistent apart from brand renaming.

## Exhibit 21

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | FixedPoint.sol |

**[INFORMATIONAL] Description:**

No findings were found at this stage of the audit as the delta between “UniswapV2” and this implementation is inexistent apart from linting style.

## Exhibit 22

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | EnumerableSet.sol |

**[INFORMATIONAL] Description:**

This contract is out of scope as it is a direct copy of the homonymous OpenZeppelin EnumerableSet.sol library.

## Exhibit 23

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | Create2.sol |

**[INFORMATIONAL] Description:**

This contract is out of scope as it is a direct copy of the homonymous OpenZeppelin Create2.sol library.

## Exhibit 24

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | AddressStringUtil.sol |

**[INFORMATIONAL] Description:**

No findings were found at this stage of the audit as the delta between “UniswapV2” and this implementation is inexistent apart from linting style.

## Exhibit 25

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | Address.sol |

**[INFORMATIONAL] Description:**

This contract is out of scope as it is a direct copy of the homonymous OpenZeppelin Address.sol library. A delta was observed in the way an address is checked whether it contains a contract whereby extcodehash is utilized instead of `extcodesize`. The documentation around it is valid and as such we assume that this is a yet-to-be-released version of OpenZeppelin's library.

## Exhibit 26

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | BEP20.sol |

**[INFORMATIONAL] Description:**

No findings were found at this stage of the audit as the contract is an ERC20 copy of the homonymous OpenZeppelin ERC20.sol implementation with omitted functionality.

## Exhibit 27

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | IBEP20.sol |

**[INFORMATIONAL] Description:**

No findings were identified at this stage of the audit, the contract represents all functions that exist within BEP20.sol in `interface` format.

## Exhibit 28

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | SafeBEP20.sol |

**[INFORMATIONAL] Description:**

This contract is out of scope as it is a direct copy of the homonymous OpenZeppelin SafeERC20.sol library with Binance BEP20 compliant naming.

## Exhibit 29

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | SafeMath.sol |

**[INFORMATIONAL] Description:**

No findings were identified at this stage of the audit, this contract is a combination of the SafeMath.sol and Math.sol libraries of “UniswapV2” with an inexistent delta.

## Exhibit 30

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | Ownable.sol |

**[INFORMATIONAL] Description:**

This contract is out of scope as it is a direct copy of the homonymous OpenZeppelin Ownable.sol library with an internalized `_transferOwnership` function.

## Exhibit 31

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | Manageable.sol |

**[INFORMATIONAL] Description:**

No findings were identified at this stage of the audit, the contract is an Ownable.sol implementation with renamed variables and events.

## Exhibit 32

| TITLE | TYPE | SEVERITY | LOCATION |
| --- | --- | --- | --- |
| Contract Description | Contract Info | Informational | Context.sol |

**[INFORMATIONAL] Description:**

This contract is out of scope as it is a direct copy of the homonymous OpenZeppelin Context.sol library with the abstract contract modifier omitted and a constructor defined.