



CERTIK

CheeseSwap Token

Security Assessment

March 23th, 2021

For :
CheeseSwap





Disclaimer

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.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product’s IT infrastructure and or source code.



Overview

Project Summary

Project Name	CheeseSwap
Description	DeFi
Platform	Binance Smart Chain; Solidity
Codebase	Github Repository
Commit	027fcc1ecf33850b52d27ae25a933980410f5e1d

Audit Summary

Delivery Date	Mar. 23th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Mar. 8th, 2021 - Mar. 14th, 2021, Mar. 19, 2021, Mar. 23, 2021

Vulnerability Summary

Total Issues	1
● Total Critical	0
● Total Major	0
● Total Medium	0
● Total Minor	1
● Total Informational	0



Executive Summary

This report has been prepared for **CheeseSwap Token** smart contract 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 Dynamic Analysis, 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.

To initial setup project correctly, improve overall project quality, preserve the upgradability, the following functions, which are owner role, are adopted in the codebase:

owner , is adopted to control following functionality in CheeseMakerFarm.sol :

- mint() to mint a specific amount of CHS token to the address _to in smart contract CheeseMakerFarm.sol .

The advantage of the above functions in the codebase is that the client reserves the ability to adjust the project according to the runtime require to best serve the community. It is also worthy of note the potential drawbacks of these functions, which should be clearly stated through client's action/plan on how to prevent abuse of these functionalities

To improve the trustworthiness of the project, any dynamic runtime updates in the project should be notified to the community. Any plan to invoke abovementioned functions should be also considered to move to the execution queue of Timelock contract.



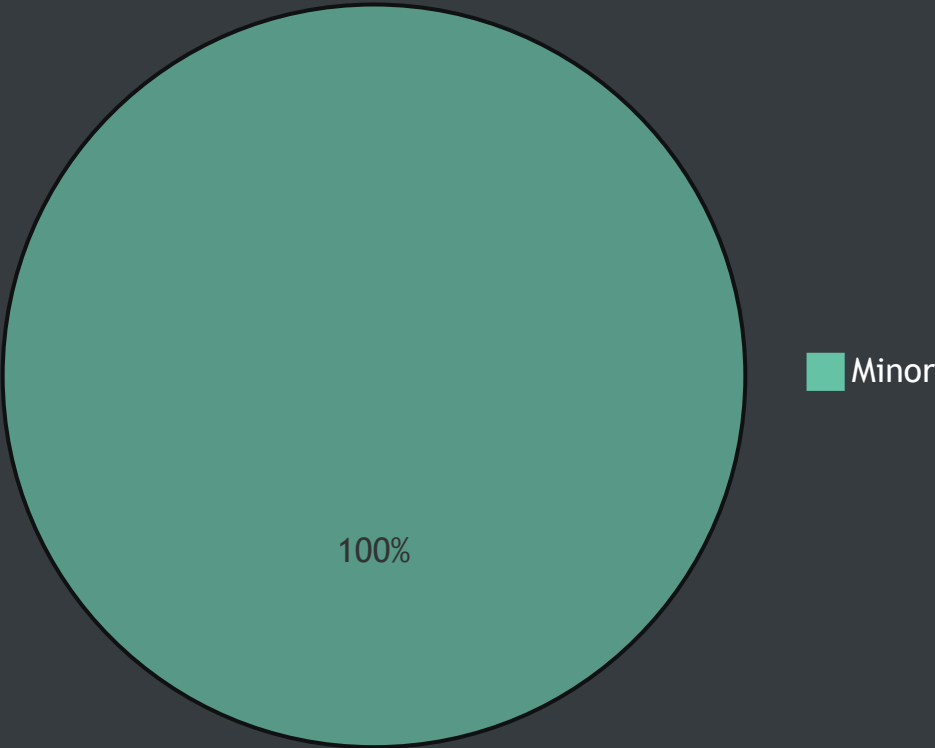
File in Scope

ID	Contract	SHA-256 Checksum
CMF	CheeseMakerFarm.sol	b4e78b47d0bf5223cc18ba85628f63e571d359fe4bc3773e704cbf4cc7b21030



Findings

Pie Chart



ID	Title	Type	Severity	Resolved
CMF-01	Centralized Risk	Logical Issue	● Minor	ⓘ



CMF-01: Centralized Risk

Type	Severity	Location
Logical Issue	● Minor	CheeseMakerFarm.sol

Description:

`owner` is an important role in the contract. The owner address can operate on following functions:

- `mint()`

Recommendation:

We advise the client to carefully manage project's private key and avoid any potential risks of being hacked. We also advise the client to adopt Multisig, Timelock and/or DAO in the project to manage sensitive role accesses.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that 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.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that 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.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an instorage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete` .

Coding Style

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

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.

Icons explanation



: Issue resolved



: Issue not resolved / Acknowledged. The team will be fixing the issues in the own timeframe.



: Issue partially resolved. Not all instances of an issue was resolved.