CERTIK

Security Assessment

ICHI

Apr 1st, 2021

Summary

This report has been prepared for Ichi 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 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.

The security assessment resulted in 8 finding that ranged from major 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	ICHI
Description	ICHI is self-sustaining, community governed infrastructure that enables any cryptocurrency community to create and govern their own in-house, non-custodial oneToken (a stablecoin valued at $1).
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/ichifarm/ichi-farming/blob/main/contracts/ichiFarmV2.sol
Commits	63d6b1ecbf91229f82dcfb352441832bc16c3493

Audit Summary

Delivery Date	Apr 01, 2021
Audit Methodology	Static Analysis, Manual Review, Testnet Deployment
Key Components	

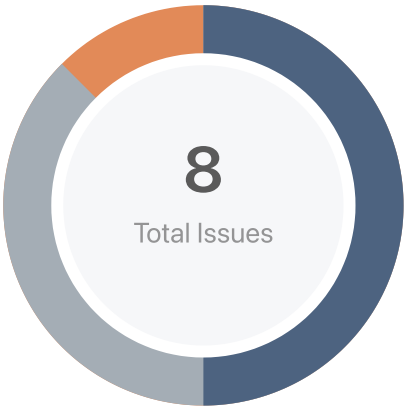
Vulnerability Summary

Total Issues	8
Critical	0
Major	1
Minor	3
Informational	4
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
FV2	ichiFarmV2.sol	e17418c44efeb9927126446acf188286490bd6d6ecef66f4dd07a9f8600d2829

Findings



Critical	0 (0.00%)
Major	1 (12.50%)
Minor	3 (37.50%)
Informational	4 (50.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
FV2-1	add() Function Not Restricted	Volatile Code	Major	Resolved
FV2-2	Variable “nonReentrant” not set Visibility	Volatile Code	Minor	Resolved
FV2-3	Potential Division Overflow	Mathematical Operations	Minor	Resolved
FV2-4	Missing Zero Address Validation	Volatile Code	Minor	Resolved
FV2-5	Multiplication on the Result of a Division	Mathematical Operations	Informational	Resolved
FV2-6	Proper Usage of Public and External	Gas Optimization	Informational	Resolved
FV2-7	Missing Emit Event	Logical Issue	Informational	Resolved
FV2-8	Gas Consumption May Grow Huge	Gas Optimization	Informational	Acknowledged

FV2-1 | add() Function Not Restricted

Category	Severity	Location	Status
Volatile Code	Major	ichiFarmV2.sol: 109	Resolved

Description

The comment in L106, mentioned // DO NOT add the same LP token more than once. Rewards will be messed up if you do.

The total amount of reward ichiReward 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

Using mapping of addresses -> booleans, which can restrict the same address being added twice.

Alleviation

The update has been applied at https://github.com/ichifarm/ichi-farming/commit/661890853d2f555bffd1f2e8d3aa7fb597043d17

FV2-2 | Variable “nonReentrant” not set Visibility

Category	Severity	Location	Status
Volatile Code	Minor	ichiFarmV2.sol: 54	Resolved

Description

No visibility was set for state variable “nonReentrant”

Recommendation

Always set the visibility of state variables explicitly.

Alleviation

The update has been applied at https://github.com/ichifarm/ichi-farming/commit/661890853d2f555bffd1f2e8d3aa7fb597043d17

FV2-3 | Potential Division Overflow

Category	Severity	Location	Status
Mathematical Operations	Minor	ichiFarmV2.sol: 96, 142, 174	Resolved

Description

The calculation of "ichiReward" used Solidity "/" operator, which might cause division overflow when denominator("totalAllocPoint") is zero.

Recommendation

Using function "div()" in BoringMath.sol instead of "/".

Alleviation

The update (by checking if totalAllocPoints > 0) has been applied at https://github.com/ichifarm/ichi-farming/commit/661890853d2f555bffd1f2e8d3aa7fb597043d17

FV2-4 | Missing Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Minor	ichiFarmV2.sol: 254	Resolved

Description

The assigned value to to(address) should be verified as non zero value to prevent being mistakenly assigned as address(0) in emergencyWithdraw() function and safeTransfer().

Recommendation

Check that the address is not zero by adding checks in function emergencyWithdraw(). Please ignore if the team inclines to leverage the same function in a way to renounce the lptoken collections (mimic the token burn in a way).

Alleviation

The update has been applied at https://github.com/ichifarm/ichi-farming/commit/661890853d2f555bffd1f2e8d3aa7fb597043d17

FV2-5 | Multiplication on the Result of a Division

Category	Severity	Location	Status
Mathematical Operations	Informational	ichiFarmV2.sol: 142~143, 174~175	Resolved

Description

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

Recommendation

Consider ordering multiplication before division.

Alleviation

The update has been applied at https://github.com/ichifarm/ichi-farming/commit/661890853d2f555bffd1f2e8d3aa7fb597043d17

FV2-6 | Proper Usage of Public and External

Category	Severity	Location	Status
Gas Optimization	Informational	ichiFarmV2.sol: 75	Resolved

Description

"Public" functions that are never called by the contract could be declared "external" to save gas.

Recommendation

Use the "external" attribute for functions never called from the contract.

Alleviation

The update has been applied at https://github.com/ichifarm/ichi-farming/commit/661890853d2f555bffd1f2e8d3aa7fb597043d17

FV2-7 | Missing Emit Event

Category	Severity	Location	Status
Logical Issue	Informational	ichiFarmV2.sol: 75, 254	Resolved

Description

Functions that affect the status of sensitive variables should be able to emit events as notifications to customers.

Recommendation

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

Alleviation

The update has been applied at https://github.com/ichifarm/ichi-farming/commit/661890853d2f555bffd1f2e8d3aa7fb597043d17

FV2-8 | Gas Consumption May Grow Huge

Category	Severity	Location	Status
Gas Optimization	Informational	ichiFarmV2.sol: 149, 158	Acknowledged

Description

Gas consumption in function massUpdateAllPools() and massUpdatePools() depends on the length of the pool. If the length was too large, the gas required to execute the code may exceed the gas limit, causing a potential denial-of-service condition.

Recommendation

Set an upper limit in add() or update a fixed number of pools every time.

Alleviation

The team confirmed that the function is typically ran by owner when the call setICHIPerBlock is needed. It is a public function but no real value of anyone to call it other than owner after setting the total ichi per block.

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 in storage 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.

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.