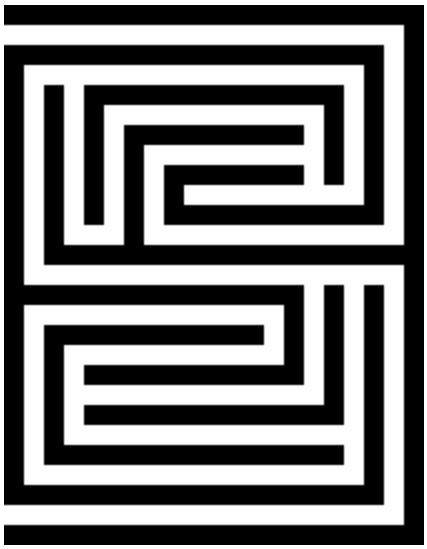

Ichi Farm Public Report

PROJECT: Ichi Farm Review

October 2020

Prepared For:

Masanobu Fukuoka | Ichi Farm

masanobu@ichi.farm

Prepared By:

Jonathan Haas | Bramah Systems, LLC.

jonathan@bramah.systems

Table of Contents

Ichi Farm Security Review

Executive Summary

Scope of Engagement

Bramah Systems, LLC was engaged in October of 2020 to perform a comprehensive security review of multiple Ichi Farm smart contracts (specific contracts denoted below). Our review was conducted over a period of three business days by a member of the Bramah Systems, LLC. executive staff.

Bramah's review pertains to smart contract Solidity code (*.sol) as of commit **2a14fffa14be0cbd0d1d0d0fa84716a585137c40** for the core contracts repository and **d341282be00876c34141c0eae98217788e7a2c5b** for farming & swaps repository, as per request of Ichi Farm. Bramah Systems completed the assessment using manual, static and dynamic analysis techniques.

Timeline

Review Commencement: October 28, 2020

Report Delivery: October 31, 2020

Engagement Goals

The primary scope of the engagement was to evaluate and establish the overall security of the Ichi Farm system, with a specific focus on trading actions. In specific, the engagement sought to answer the following questions:

- Is it possible for an attacker to steal or freeze tokens?
- Does the Solidity code match the specification as provided?
- Is there a way to interfere with the contract mechanisms?

- Are the arithmetic calculations trustworthy?

Protocol Specification

A basic specification document was compiled by the review team based upon review of the Ichi Farm code and discussion with the team.

Overall Assessment

Bramah Systems was engaged to evaluate and identify multiple security concerns in the codebase of the Ichi Farm protocol architecture. During the course of our engagement, Bramah Systems denoted numerous instances wherein the protocol deviated from established best practices and procedures of secure software development. With limited exceptions (as described in this report), these instances were most commonly a result of structural limitations of Solidity and not due to inactions on behalf of the development team. The codebase benefits from detailed code comments throughout, which allowed for Bramah to review the codebase rapidly and without a deeper formal specification. As the code does make heavy usage of third party library code, the Ichi Farm team should stay abreast of any security considerations of these protocols. At the point of our review, there were no known high or medium severity risks presently outstanding with the third party code that had been implemented into the Ichi Farm codebase.

Disclaimer

As of the date of publication, the information provided in this report reflects the presently held, commercially reasonable understanding of Bramah Systems, LLC.'s knowledge of security patterns as they relate to the Ichi Farm Protocol, with the understanding that distributed ledger technologies ("DLT") remain under frequent and continual development, and resultantly carry with them unknown technical risks and flaws. The scope of the review provided herein is limited solely to items denoted within "Scope of Engagement" and contained within "Directory Structure". The report does NOT cover, review, or opine upon security considerations unique to the Solidity compiler, tools used in the development of the protocol, or distributed ledger technologies themselves, or to any other matters not specifically covered in this report.
The contents of this report must NOT be construed as investment advice or advice of any other kind. This report does NOT have any bearing upon the potential economics of the Ichi Farm protocol or any other relevant product, service or asset of Ichi Farm or otherwise. This report is

not and should not be relied upon by Ichi Farm or any reader of this report as any form of financial, tax, legal, regulatory, or other advice.

To the full extent permissible by applicable law, Bramah Systems, LLC. disclaims all warranties, express or implied. The information in this report is provided "as is" without warranty, representation, or guarantee of any kind, including the accuracy of the information provided. Bramah Systems, LLC. makes no warranties, representations, or guarantees about the Ichi Farm Protocol. Use of this report and/or any of the information provided herein is at the users sole risk, and Bramah Systems, LLC. hereby disclaims, and each user of this report hereby waives, releases, and holds Bramah Systems, LLC. harmless from, any and all liability, damage, expense, or harm (actual, threatened, or claimed) from such use.

Timeliness of Content

All content within this report is presented only as of the date published or indicated, to the commercially reasonable knowledge of Bramah Systems, LLC. as of such date, and may be superseded by subsequent events or for other reasons. The content contained within this report is subject to change without notice. Bramah Systems, LLC. does not guarantee or warrant the accuracy or timeliness of any of the content contained within this report, whether accessed through digital means or otherwise.

Bramah Systems, LLC. is not responsible for setting individual browser cache settings nor can it ensure any parties beyond those individuals directly listed within this report are receiving the most recent content as reasonably understood by Bramah Systems, LLC. as of the date this report is provided to such individuals.

General Recommendations

Best Practices & Solidity Development Guidelines

Usage of ABIEncoderV2

A majority of the contracts associated with the protocol make usage of an experimental Solidity version (**pragma experimental ABIEncoderV2**) which enables usage of the new ABI encoder. **ABIEncoderV2** allows for the usage of structs and arbitrarily nested arrays (such as **string[]** and **uint256[][]**) in function arguments and return values.

As no present non experimental version for these constructs exists, one must acknowledge the associated risk in utilizing non release-candidate ("RC") software. It is understood that software in the beta phase will generally have more bugs than completed software as well as speed/performance issues and may cause crashes or data loss.

Resolution: Risk accepted by team.

Lack of consistency in Solidity version

Contracts within the repository lack a singular Solidity version. As structural changes may occur between these versions, we suggest consolidating to a singular cohesive version (ideally as recent as possible).

Versions used: 0.6.12, ^0.6.0, >=0.6.2, >=0.5.0, >=0.5.15

Resolution: Version to be locked to the highest version on file for all contracts written by Ichi.Farm team.

Usage of block.timestamp

Miners can affect block.timestamp for their benefits. Thus, one should not rely on the exact value of block.timestamp. As a result of such, **block.timestamp** and **now** should traditionally only be used within inequalities.

Resolution: Risk accepted by team.

Time considerations

Multiple variables are set relying upon Solidity's inexact time system. As not every year equals 365 days and not every day has 24 hours because of leap seconds, Solidity's one day/week/year values are inexact. As leap seconds cannot be predicted, an exact calendar match would require updating by an external oracle.

Note, the direct comparison of these variables (e.g. comparing two times) within their respective functions poses additional concern, as discussed in "Usage of block.timestamp" above (namely, a proper comparison may not be set). It is worth noting that this has downstream implications on calculations utilising this passage of time.

Resolution: Risk accepted by team.

Supply-chain risk

The code base heavily adopts patterns from pre-existing contract repositories, most notably SushiSwap and Compound Finance. While this allows for some level of expedited review (following validation that there are no material changes), it does pose some level of concern, as known (and unknown) bugs in the forked protocol are likely to also be present in this contract repository. That said, as more visibility is fundamentally on these platforms, one could argue these platforms have also received more scrutiny, with more auditors validating potential security anti-patterns in the codebase.

Resolution: Risk accepted by team.

Functions can be marked external

Functions within a contract that the contract itself does not call should be marked external in order to optimize for gas costs. In our review, most functions should consider this implementation after review of any instances in which the contract may call them. As this largely is a gas optimization, security impact of this change should be relatively immaterial.

Resolution: Functions to be marked external by team.

Multiplication after Division

Functions utilizing multiplication after division may run into issues with precision. Where possible, equations should be rewritten to avoid this common pitfall. This anti-pattern is

common across many of the smart contracts, some of which may not be modified due to existing functionality which cannot be changed. However, during our scope of testing, we did not find any calculations suggesting a truncation of precision had occurred.

Resolution: Team to re-write arithmetic calculations to resolve issues.

Inconsistent SafeMath versions

There are multiple copies of SafeMath (and various derivations of the same concept) throughout the smart contract repository. A singular library where possible should be used to avoid potentially conflicting versions, nomenclature, and usage.

Resolution: Team to delete local SafeMath.sol and use version provided by OpenZeppelin.

Specific Recommendations

Unique to the Ichi Farm Protocol

Lookup table may be more efficient as a string constant

The contract uses a lookup table within **oneFactor.sol** to return factors of an inputted integer. While presently this functionality works by constructing a lookup table, in our testing, a string constant holding the same lookup table allowed for greater gas efficiency. Depending on the deployment of these contracts (main-net, or a chain with less gas concerns) it may make sense to convert this table.

Resolution: Risk accepted by team.

Code comment may lead to confusion

The **Timelock.sol** contract stipulates that “Timelock::constructor: Delay must exceed minimum delay”, however, the delay can be equal to the minimum delay, it does not have to exceed it as presently written.

Resolution: Team to re-write comment.

Addition of new LP could result in reward miscalculation

Similar to the SushiSwap contract it inherits from, **ichiFarm.sol** does not presently have a methodology by which LP’s added twice can be easily removed and properly reset. As the contract explicitly states “DO NOT add the same LP token more than once. Rewards will be messed up if you do”. While mitigations could exist (such as logging which LP have been added so far in a boolean mapping), these are not explicitly necessary if the contract owner is careful to not add any LP twice.

Resolution: Risk accepted by team.

Typographic Errors

INTIAL_SUPPLY (referenced throughout) should be changed to properly stated as “**INITIAL_SUPPLY**”, unless intentionally written as such.

Resolution: Risk accepted by team.

Missing Functionality

The migration JavaScript file **migrations_/2_deploy_contracts.js** does not perform any deployment logic for the smart contract.

Resolution: Risk accepted by team.

lastRewardBlock and lastRewardBonusBlock Initialization

Both reward block values are initially declared as the starting block value, as shown below:

```
lastRewardBlock: _startBlock,
lastRewardBonusBlock:
_startBlock
```

These should be taken into consideration on any actions which are intended to validate how many successful reward blocks have occurred (as one does not actually occur during the first block).

Resolution: Risk accepted by team.

Winners payout handling may result in improper payouts

While the **updateBonusRewards** function explicitly states a control to “keep winners payout to less than the max transaction loop size”, in the event that the transaction size is superseded, the current approach does not adequately handle the reward payout for these individuals. Rather than only escaping the loop, our recommendation is to fail gracefully and instruct the user that their payout is incapable of being processed at this time, and for the team to create some sort of mechanism to control for reducing the payout back to operable levels.

Resolution: This is a known limitation of the protocol and the team has a compensation structure in place for individuals that fall within this category.

Toolset Warnings

Unique to the Ichi Farm Protocol

Overview

In addition to our manual review, our process involves utilizing concolic analysis and dynamic testing in order to perform additional verification of the presence security vulnerabilities. An additional part of this review phase consists of reviewing any automated unit testing frameworks that exist.
The following sections detail warnings generated by the automated tools and confirmation of false positives where applicable, in addition to findings generated through manual inspection.

Compilation Warnings

No warnings were found at time of compilation that presented material concern. Compilation warnings shown regarded varying Solidity versions, a known issue (as per above).

Test Coverage

The contract repository lacks substantial unit test coverage throughout. This testing traditionally would provide a variety of unit tests which encompass the various operational stages of the contract.

Presently, the Ichi Farm protocol (and its relevant components and their respective subcomponents) does not possess tests validating functionality and ensuring that certain behaviors (those relating to erroneous or overflow-prone input) do not see successful execution.

Static Analysis Coverage

The contract repository underwent heavy scrutiny with multiple static analysis agents, including:

- Securify
- MAIAN

- Mythril
- Oyente
- Slither

In each case, the team had either mitigated relevant concerns raised by each of these tools or provided adequate justification for the risk (e.g. inherent risk of using native Ethereum constructs such as **timestamp**).

Certain tools, like Oyenete, do not run on newer versions of Solidity. Where applicable, Bramah manually performed validation checks based upon our understanding of the tool.

Directory Structure

At time of review, the directory structure of the Ichi Farm contract (**./contracts**) appeared as it does below. Our review, at request of Ichi Farm, covers the Solidity code (*.sol) as of commits **2a14fffa14be0cbd0d1d0d0fa84716a585137c40** for the core contracts repository and **d341282be00876c34141c0eae98217788e7a2c5b** for farming & swaps repository.

Core

```
├── Timelock.sol
├── governorAlpha.sol
├── ichiToken.sol
├── lib
│   ├── Math.sol
│   └── SafeMathInt.sol
├── oneETH.sol
├── oneFactor.sol
└── uniswapV2Oracle.sol
```

1 directory, 8 files

Swaps & Farming

```
├── ichiBuyer.sol
├── ichiFarm.sol
├── ichiStake.sol
├── lib
│   ├── HitchensList.sol
│   ├── SafeMathInt.sol
│   └── SortedList.sol
├── readme.md
└── uniswapv2
    ├── LICENSE
    ├── UniswapV2ERC20.sol
```

```
├── UniswapV2Factory.sol
├── UniswapV2Pair.sol
├── UniswapV2Router02.sol
├── interfaces
│   ├── IERC20.sol
│   ├── IUniswapV2Callee.sol
│   ├── IUniswapV2ERC20.sol
│   ├── IUniswapV2Factory.sol
│   ├── IUniswapV2Pair.sol
│   ├── IUniswapV2Router01.sol
│   ├── IUniswapV2Router02.sol
│   └── IWETH.sol
└── libraries
    ├── Math.sol
    ├── SafeMath.sol
    ├── TransferHelper.sol
    ├── UQ112x112.sol
    └── UniswapV2Library.sol

4 directories, 25 files
```