1INCH FARMING SECURITY AUDIT REPORT

June 1, 2022

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of 1inch. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals
- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the cients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal
Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reoprts into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals
- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Customer with an interim report.

5. Bug fixing & re-audit:

- The Customer either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Customer with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Customer with a re-audited report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Other non-essential issues and recommendations reported to/ acknowledged by the team.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

1inch is a DeFi aggregator and a decentralized exchange with smart routing.
The core protocol connects a large number of decentralized and centralized platforms in order to minimize price slippage and find the optimal trade for the users.

Audited smart contracts contain the logic for earning tokens using farming methods.
The user stakes for one token and is rewarded with another token.
There are two types of farming in the scope.
The first type of farming is the `FarmingPool` smart contract, which works on its own. When depositing a `stakingToken`, the user is minted share-tokens in return. In the reverse procedure, the share-tokens are burned and the user gets his tokens back.
The second type of farming is smart contracts `ERC20Farmable` and `Farm` only work in conjunction with each other. The user executes the `join()` and `quit()` procedures. In this case, the user does not give away tokens.

File name	Description
`ERC20Farmable.sol`	Multi-farm contract that works in conjunction with the `Farm` contract. Here the user can start and end his farming.
`Farm.sol`	Separate farm contract that works in conjunction with the `ERC20Farmable` contract. This is where users get rewards.
`FarmingPool.sol`	Separate contract to work with only one farming token. Here the user can start and end his farming.
`FarmAccounting.sol`	Contract farming library that allows you to start farming and calculate the amount of tokens earned.

File name	Description
`UserAccounting.sol`	Library for calculating the balance of earned tokens from users.

1.4 Project Dashboard

Project Summary

Title	Description
Client	1inch
Project name	Farming
Timeline	07.03.2022 - 11.03.2022
Number of Auditors	4

Project Log

Date	Commit Hash	Note
28-02-2022	7a007ec7784cca2899889e99e46cf06d5788a7d9	Initial commit
14-04-2022	2b01fc6afaa43b12c67153f3a631851b2785a22f	Final commit

Project Scope

The audit covered the following files:

File name	Link
ERC20Farmable.sol	ERC20Farmable.sol

File name	Link
Farm.sol	Farm.sol
FarmingPool.sol	FarmingPool.sol
FarmAccounting.sol	FarmAccounting.sol
UserAccounting.sol	UserAccounting.sol

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	3
Medium	3
Low	3

ID	Name	Severity	Status
H-1	Reward tokens may be frozen in `FarmingPool`	High	Fixed
H-2	Actual transferred amount may differ than expected	High	Acknowledged
H-3	Rebasable as reward tokens breaks logic	High	Acknowledged
M-1	Possible arithmetic overflow	Medium	Acknowledged
M-2	Gas overflow during iteration (DoS)	Medium	Fixed

M-3	Zero Token	Medium	Fixed
L-1	Using "magic" numbers	Low	Fixed
L-2	It is possible to block tokens on the balance of the contract	Low	Fixed
L-3	Missed events	Low	Fixed

1.6 Conclusion

During the audit process, 3 HIGH, 3 MEDIUM and 3 LOW severity findings were spotted and acknowledged by the developers. These findings do not affect security of the audited project. After working on the reported findings all of them were acknowledged or fixed by the client.

Final commit identifier with all fixes: 2b01fc6afaa43b12c67153f3a631851b2785a22f

2.FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

H-1	Reward tokens may be frozen in `FarmingPool`
File	FarmAccounting.sol#L19
Severity	High
Status	Fixed in 71233068

Description

FarmAccounting.sol#L19
Farmed tokens amount is calculated when a farm duration > 0, so if the distributor starts farming with duration 0, nobody will farm and, starting the next farming, won't save prev farming rewards because a contract accounts only unfinished prev farming, but in this case the farming with duration 0 finishes immediately.
FarmAccounting.sol#L28

Recommendation

We recommend not to allow farming with 0 duration.

H-2	Actual transferred amount may differ than expected
Files	Farm.sol#L41-L43 FarmingPool.sol#L48-L50
Severity	High
Status	Acknowledged

Description

Some ERC20 tokens, for example, USDT, have fees on transfer. It may affect reward accounting on farming, a farm contract may receive fewer tokens than expected and start farming with an expected amount of rewards, which will lead to insufficiency of liquidity for rewards. Also in the`FarmingPool` contract the `deposit` and `withdraw` functions may work incorrectly.
Farm.sol#L41-L43,
FarmingPool.sol#L48-L50

Recommendation

We recommend starting farming/deposit in `FarmingPool`/withdraw in `FarmingPool` with the actual transferred amount and return an actual deposited/withdrawn/claimed amount.

Client's commentary

Tokens with fees will not be supported.

H-3	Rebasable as reward tokens breaks logic
File	FarmAccounting.sol#L21
Severity	High
Status	Acknowledged

Description

With rebasable reward tokens, the calculation of farmed rewards will be incorrect because it relies on a strict amount of distributed tokens while the underlying reward balance will float.
FarmAccounting.sol#L21

Recommendation

We recommend warning developers not to use rebasable tokens as reward tokens.

Client's commentary

Rebasable tokens will not be supported.

2.3 Medium

M-1	Possible arithmetic overflow
File	UserAccounting.sol#L29
Severity	Medium
Status	Acknowledged

Description

At the line UserAccounting.sol#L29 the number with the type `int256` is converted to the number with the type `uint256`. The number is taken with a minus sign.
But before that, there is no check that the number is less than 0.
If we take a small positive value and apply the transformation `uint256(-amount)` to it, we get a very large value due to arithmetic overflow.
For example, if you take number `1000`, then after conversion you get value `115792089237316195423570985008687907853269984665640564039457584007913129638936`.

Recommendation

Before line 29 you need to check if the value of the variable is not less than 0.
If the value of the variable is positive, then do not do the conversion.

Client's commentary

We believe all `corrections` changes never make it larger than the current user balance multiplied by FPT, that's why subtraction is never negative.

M-2	Gas overflow during iteration (DoS)
File	ERC20Farmable.sol#L70
Severity	Medium
Status	Fixed in 70931257

Description

Each iteration of the cycle requires a gas flow.
A moment may come when more gas is required than it is allocated to record one block. In this case, all iterations of the loop will fail.
Affected lines:

- ERC20Farmable.sol#L70
- ERC20Farmable.sol#L117
- ERC20Farmable.sol#L121
- ERC20Farmable.sol#L137

Recommendation

It is recommended to add a check for the maximum possible number of elements of the arrays.

M-3	Zero Token
File	FarmingPool.sol#L35-L36
Severity	Medium
Status	Fixed in 5b97ae7f

Description

There is no address checking for tokens params in constructor:
FarmingPool.sol#L35-L36

Recommendation

It is recommended to add a check for non-zero address.

2.4 Low

L-1	Using "magic" numbers
Files	FarmAccounting.sol#L21 UserAccounting.sol#L29
Severity	Low
Status	Fixed in d39a2605

Description

The use in the source code of some unknown where taken values impairs its understanding.
At the lines

- FarmAccounting.sol#L21
- FarmAccounting.sol#L29
- UserAccounting.sol#L29
  the value is `1e18`.

Recommendation

It is recommended that you create constants with meaningful names to use numeric values.

L-2	It is possible to block tokens on the balance of the contract
File	FarmingPool.sol#L48
Severity	Low
Status	Fixed in 2b01fc6a

Description

At the line FarmingPool.sol#L48 `rewardsToken` is transferred to the balance of the contract.
But there is no functionality to return tokens in case of an emergency or if not all users call the `claim()` function.

Recommendation

It is necessary to add functionality for the possibility of withdrawing the remaining tokens from the balance of the contract.

L-3	Missed events
Files	FarmingPool.sol#L66 ERC20Farmable.sol#L58
Severity	Low
Status	Fixed in 25e99776

Description

There are missed events for claim, deposit, withdraw, join/quit farming.
FarmingPool.sol#L66
FarmingPool.sol#L72
FarmingPool.sol#L78
ERC20Farmable.sol#L58
ERC20Farmable.sol#L75
ERC20Farmable.sol#L93

Recommendation

We recommend emitting the events above.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://twitter.com/mixbytes