1INCH FEE COLLECTOR SMART CONTRACT AUDIT

September 06, 2021

MixBytes()

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 PROJECT OVERVIEW

A contract that collects user rewards and exchanges it for 1inch tokens through an auction.
The auction has parameters maxValue and minValue, which indicate the maximum and minimum values of the number of 1inch tokens that the contract agrees to receive in exchange for the entire number of certain tokens.

1.3 SECURITY ASSESSMENT METHODOLOGY

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

01 Project architecture review:
> Reviewing project documentation
> General code review
> Reverse research and study of the architecture of the code based on the source code only
> Mockup prototyping

Stage goal:
Building an independent view of the project's architecture and identifying logical flaws in the code.

02 Checking the code against the checklist of known vulnerabilities:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code
> Checking with static analyzers (i.e Slither, Mythril, etc.)

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

03 Checking the code for compliance with the desired security model:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit
> Exploits PoC development using Brownie

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of interim auditor reports into a general one:
> Cross-check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level and provide the client with an interim report.

05 Bug fixing & re-check:
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team 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 project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

The audited scope implements Fee Collector smart contracts that handles governance and referral fees from Liquidity Protocol and Aggregation Protocol.

1.5 PROJECT DASHBOARD

Client	1Inch
Audit name	Fee Collector
Initial version	3c2626763fd829500496f15476d5e98fbdf4f574
Final version	62107c397b8e922afb63dc4c49595fb56db015e8
Date	July 14, 2021 - September 06, 2021
Auditors engaged	2 auditors

FILES LISTING

FeeCollector.sol	https://github.com/1inch/fee-collector/blob/3c2626763fd829500496f15476d5e98fbdf4f574/contracts/FeeCollector.sol
BalanceAccounting.sol	https://github.com/1inch/fee-collector/blob/3c2626763fd829500496f15476d5e98fbdf4f574/contracts/utils/BalanceAccounting.sol
IFeeCollector.sol	https://github.com/1inch/fee-collector/blob/3c2626763fd829500496f15476d5e98fbdf4f574/contracts/interfaces/IFeeCollector.sol
InteractiveMaker.sol	https://github.com/1inch/fee-collector/blob/3c2626763fd829500496f15476d5e98fbdf4f574/contracts/interfaces/InteractiveMaker.sol

FINDINGS SUMMARY

Level	Amount
Critical	1
Major	0
Warning	1
Comment	3

CONCLUSION

Smart contract has been audited and several suspicious places were found. During audit one critical issue was identified. Several issues were marked as warnings and comments. After working on audit report all issues were fixed or acknowledged by the client. Thus, contract is assumed as secure to use according to our security criteria.Final commit identifier with all fixes: `62107c397b8e922afb63dc4c49595fb56db015e8`

2. FINDINGS REPORT

2.1 CRITICAL

CRT-1	Transactions frontrunning
File	FeeCollector.sol
Severity	Critical
Status	Fixed at 62107c39

DESCRIPTION

The code in this line contains an error in calculation
in line: FeeCollector.sol#L360

```
uint256 returnAmount = amount * tokenBalance / value(erc20);
```

According to the formula, it is fixed price for the rest of tokenBalance or whole amount, not the price for unit.

*Example:

Someone(something) makes updateReward for 10 WETH - user1
Let's assume now we have minimal price after a long period of time

Current price is 100 Inch for 10 WETH
user2 and user3 are atackers

User4 wants to make a full trade. He wants to buy 10 WETH for 100 1Inch. He wants make a transaction

attacker puts the next transaction before User4
User2 updateReward with ~101010 WETH
the price is changed
User3 trades after User2 ~101010.mul(price)
User2 executes trade
attacker has a profit equal ~1.4 WETH

User4 got ~8.6 WETH. 1.4 WETH less that expected*

*Example 2

Let's assume, now we have minimal price after a long period of time
Someone(something) makes updateReward for 10 WETH - user1
Current price is 100 Inch for 10 WETH

User2 and User3 want to make partial trade. They want to buy 5 WETH for 50 1Inch. They make a transaction at approximately the same time

User2 trades first and gets 5 WETH and spends ~50 1Inch. OK
User3 trades after User 2.5WETH and spends ~50 Inch. Not correct

User1 never gets his 1Inch because User3 or nobody wants to make this incorrect transaction

In case User3 makes transaction for 5 WETH and ~spends 100 1Inch, User1 will receive ~2 times more 1Inch than excpected*

RECOMMENDATION

Implement correct returnAmount calculation

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	Not Possible to buy whole amount when the price is continuously changing
File	FeeCollector.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

When the price is continuously changing it is not possible to set exact amount of token for buying the whole amount because the price is always decreasing and transation will be always reverted because we will have not enough rewarder tokens. In case you set perspective price, you will have the same situation described in №1 critical case.

RECOMMENDATION

We recommend to add functionality which allows us to set exact `returnAmount` for rewarded token and set restriction for input tokens amount.

CLIENT'S COMMENTARY

It is possible via external contract that will call FeeCollector.value(erc20) beforehand.

2.4 COMMENT

CMT-1	Missing natspec documentation
File	FeeCollector.sol
Severity	Comment
Status	No Issue

DESCRIPTION

Solidity contracts can use a special form of comments to provide rich documentation for functions, return variables and more. This special form is named the Ethereum Natural Language Specification Format (NatSpec).

RECOMMENDATION

It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract.

CMT-2	Possible events are not emitting
File	FeeCollector.sol
Severity	Comment
Status	Fixed at 62107c39

DESCRIPTION

In the `trade()` function the state of smart contract is changed, but events are not emitted:
FeeCollector.sol#L344

RECOMMENDATION

We recommend to add emitting events for this function.

CMT-3	Potentially re-entrancy weak code
File	FeeCollector.sol
Severity	Comment
Status	Fixed at 62107c39

DESCRIPTION

At the lines: FeeCollector.sol#L339-L342
the state changes inside `_updateReward` which happens AFTER:

- `erc20.safeTransferFrom()`
  It makes the method potentially weak for re-entry attack.

RECOMMENDATION

We recommend to add ReentrancyGuard modifier.
or
It is recommended to change the order of calls in a common-way, change-state first, then external-call.

3. ABOUT MIXBYTES

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

BLOCKCHAINS

- Ethereum
- Cosmos
- EOS
- Substrate

TECH STACK

- Python
- Solidity
- Rust
- C++

CONTACTS

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://t.me/MixBytes

https://twitter.com/mixbytes