1INCH LIMIT ORDER PROTOCOL SMART CONTRACT AUDIT

November 01, 2021

MixBytes()

CONTENTS

1.INTRODUCTION 2
DISCLAIMER 2
SECURITY ASSESSMENT METHODOLOGY 3
PROJECT OVERVIEW 5
PROJECT DASHBOARD 5
2.FINDINGS REPORT 7
2.1.CRITICAL 7
2.2.MAJOR 7
MJR-1 Increase in the amount of tokens due to arithmetic overflow of a variable 7
2.3.WARNING 8
WRN-1 Use a call to existing functions from another contract 8
WRN-2 Add a condition to save gas 9
WRN-3 There is no processing of the value returned by the function 10
WRN-4 Wrong number of params 11
2.4.COMMENT 12
CMT-1 Hardcoded values 12
3.ABOUT MIXBYTES 13

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 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.

1.3 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.The audited scope implements a protocol of limit orders with two types:

- `OrderRFQMixin` is a simple limit order with gas optimized option
- `OrderMixin` is a limit order has complex option with significant configuration variability

with supported tokens ERC20, EC721, ERC1155 with some new adds like: taker permit.
The code is written in a very gas-efficient manner for cheap usage by end-users.
List of changes from version 1:

- added the ability to specify the recipient of tokens (`fillOrderTo()` and `fillOrderRFQTo()`)
- RFQ fills now return `makingAmount` and `takingAmount`
- added fill with taker permit
- the logic for `fillOrder()` has changed and now it continues to work if the `remainingMakerAmount` is not enough
- added support for `DAI-like` permits
- added `Cancel` event

1.4 PROJECT DASHBOARD

Client	1Inch
Audit name	Limit Order Protocol
Initial version	9d118307df7acc3bcef73407f3964acd6aa0f35c
Final version	c2f71472c55f50dc583ef2165180127c4b8d95a3
Date	October 04, 2021 - November 01, 2021
Auditors engaged	3 auditors

FILES LISTING

LimitOrderProtocol.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/LimitOrderProtocol.sol

OrderMixin.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/OrderMixin.sol
OrderRFQMixin.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/OrderRFQMixin.sol
AmountCalculator.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/helpers/AmountCalculator.sol
ChainlinkCalculator.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/helpers/ChainlinkCalculator.sol
NonceManager.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/helpers/NonceManager.sol
PredicateHelper.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/helpers/PredicateHelper.sol
AggregatorInterface.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/interfaces/AggregatorInterface.sol
IDaiLikePermit.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/interfaces/IDaiLikePermit.sol
InteractiveNotificationReceiver.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/interfaces/InteractiveNotificationReceiver.sol
ArgumentsDecoder.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/libraries/ArgumentsDecoder.sol
Permitable.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/libraries/Permitable.sol
RevertReasonParser.sol	https://github.com/1inch/limit-order-protocol/blob/9d118307df7acc3bcef73407f3964acd6aa0f35c/contracts/libraries/RevertReasonParser.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	1
Warning	4
Comment	1

CONCLUSION

Smart contracts have been audited and several suspicious places have been detected. During the audit no critical issues were found, one major, four warnings and one comment were spotted. After working on the reported findings all of them were fixed or acknowledged by the client. So, the contracts are assumed as secure to use according to our security criteria.Final commit identifier with all fixes: `c2f71472c55f50dc583ef2165180127c4b8d95a3`

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Increase in the amount of tokens due to arithmetic overflow of a variable
File	ChainlinkCalculator.sol
Severity	Major
Status	Fixed at 328d3674

DESCRIPTION

At the lines:

- ChainlinkCalculator.sol#L26
- ChainlinkCalculator.sol#L28
- ChainlinkCalculator.sol#L40
  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.
  This is demonstrated by the following example https://gist.github.com/mixbytes-audit/b471cc82105f856d1546ba638de20f4e.
  For example, if you take the number `1000`, then after conversion you get the value `115792089237316195423570985008687907853269984665640564039457584007913129638936`.
  We see an increase in the initial value of the variable.

RECOMMENDATION

Before lines 26, 28 and 40, you need to check the value of the variable for being less than 0.
If the value of the variable is positive, then do not do the conversion.

2.3 WARNING

WRN-1	Use a call to existing functions from another contract
File	OrderRFQMixin.sol AmountCalculator.sol
Severity	Warning
Status	Fixed at 2e921e4f

DESCRIPTION

- Instead of the line OrderRFQMixin.sol#L107, you can use the code from here AmountCalculator.sol#L19-L21.

- Instead of the line OrderRFQMixin.sol#L111, you can use the code from here AmountCalculator.sol#L13-L15.

RECOMMENDATION

In the `OrderRFQMixin` contract, you must use the functions from the `AmountCalculator` contract.

WRN-2	Add a condition to save gas
File	OrderMixin.sol
Severity	Warning
Status	Fixed at 7cc252b2

DESCRIPTION

If the order is cancelled, the `fillOrder()` function for the lines OrderMixin.sol#L165-L259 will be executed anyway.
Correct option is when the transaction will be reverted.
Before line 177, you can add a condition:

```
    require(remainingMakerAmount != 1, "LOP: the order has already been canceled");
```

RECOMMENDATION

Add a condition that will save gas.

WRN-3	There is no processing of the value returned by the function
File	OrderMixin.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

For the lines
OrderMixin.sol#L227-L236, the `_makeCall()` function is called to transfer the `transferFrom()` tokens.
At the lines
OrderMixin.sol#L247-L256 there is a similar case.
According to the ERC-20 standard, in case of a successful token transfer, it always returns `true`.
The `_makeCall()` function on the
OrderMixin.sol#L284-L287 does not handle the situation when nothing is transferred.

RECOMMENDATION

It is necessary to correct the logic of the `_makeCall()` function.

CLIENT'S COMMENTARY

Not an issue

WRN-4	Wrong number of params
File	OrderMixin.sol
Severity	Warning
Status	Fixed at **c2f71472**

DESCRIPTION

At `OrderMixin` contract in `fillOrderTo()` function at the lines:
OrderMixin.sol#L234
OrderMixin.sol#L254
used extra params because `transferFrom()` takes 3 params only.
The lines 229-235 should look like this

```
            abi.encodePacked(
                IERC20.transferFrom.selector,
                uint256(uint160(msg.sender)),
                uint256(uint160(order.receiver == address(0) ? order.maker : order.receiver)
                takingAmount
            )
```

and at the lines 249-255 should look like this

```
            abi.encodePacked(
                IERC20.transferFrom.selector,
                uint256(uint160(order.maker)),
                uint256(uint160(target)),
                makingAmount
            )
```

RECOMMENDATION

It is necessary to remove extra params.

2.4 COMMENT

CMT-1	Hardcoded values
File	RevertReasonParser.sol
Severity	Comment
Status	**Fixed** at **b2d305b7**

DESCRIPTION

At the lines:

- RevertReasonParser.sol#L16
- RevertReasonParser.sol#L34
  have hardcoded value to selector.

RECOMMENDATION

It is recommended to change these values to constants and use these constants instead of hardcoded values

```
    bytes4 constant public LIMIT_ORDER_PANIC_SELECTOR = bytes4(keccak256("Panic(uint256)"));
    bytes4 constant public LIMIT_ORDER_ERROR_SELECTOR = bytes4(keccak256("Error(string)"));
```

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
- EOS
- Cosmos
- Substrate

TECH STACK

- Python
- Rust
- Solidity
- C++

CONTACTS

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://t.me/MixBytes

https://twitter.com/mixbytes