ONEINCH EXCHANGE SMART CONTRACT AUDIT

March 12, 2021

MixBytes()

CONTENTS

1.INTRODUCTION.... 1

DISCLAIMER.... 1

PROJECT OVERVIEW.... 1

SECURITY ASSESSMENT METHODOLOGY.... 2

EXECUTIVE SUMMARY.... 4

PROJECT DASHBOARD.... 4

2.FINDINGS REPORT.... 6

2.1.CRITICAL.... 6

2.2.MAJOR.... 6

2.3.WARNING.... 6

WRN-1 1/span>`_permit` doesn't keep invariant `amount` <= `allowance`.... 6

WRN-2 Unpausable `unoswap` in pausable `OneInchExchange`.... 7

2.4.COMMENTS.... 8

CMT-1 Transfer is not the last statement.... 8

CMT-2 Lack of docs in ASM.... 9

CMT-3 Lack of re-entrancy guard.... 10

CMT-4 Use ASM case for gas efficiency.... 11

CMT-5 Unclear bytes values.... 12

CMT-6 Unclear flags.... 13

3.ABOUT MIXBYTES.... 14

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 OneInch. 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

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.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 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 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> 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

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

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> 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

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> 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
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 part of the exchange aggregation protocol. Such protocol allows users to do swap selecting the best exchanges. The code is written in a very gas-efficient manner for cheap usage by end-users.

1.5 PROJECT DASHBOARD

Client	OneInch
Audit name	Exchange
Initial version	d3def083b875d3e04faf2caee758a1c4aaf43b7d
Final version	d3def083b875d3e04faf2caee758a1c4aaf43b7d
SLOC	366
Date	2021-02-24 - 2021-03-12
Auditors engaged	2 auditors

FILES LISTING

OneInchExchange.sol	OneInchExchange.sol
OneInchUnoswap.sol	OneInchUnoswap.sol
Permitable.sol	Permitable.sol
UniERC20.sol	UniERC20.sol
RevertReasonParser.sol	RevertReasonParser.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	2
Comment	6

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit no critical or major issues were spotted, but several warnings and comments were found and discussed with the client. After working on the reported findings all of them were acknowledged (as the problem was not critical). So, the contracts are assumed as secure to use according to our security criteria.Final commit identifier with all fixes: `d3def083b875d3e04faf2caee758a1c4aaf43b7d`

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	1/span>`_permit` doesn't keep invariant `amount` <= `allowance`
File	Permitable.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

According to `_permit` function logic defined at Permitable.sol#L21 function should revert if allowance less than required `amount`, however after successfull result of ERC-20 `permit` call that invariant never checked. `permit` call doesn't guarantee that resulting allowance is greater than that amount because calldata for permit call is forwarded from external call and contract never check that calldata contains right approval amount.

RECOMMENDATION

We recommend to check invariant not only for unsuccessfull result

CLIENT'S COMMENTARY

It's impossible to use an additional allowance check after successful permit due to gas expenses. If permit was wrongly formed, the transaction will fall on the next step, when transferFrom lacks allowances.

WRN-2	Unpausable `unoswap` in pausable `OneInchExchange`
File	OneInchExchange.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

`OneInchExchange` contract defined at OneInchExchange.sol#L15 is pausable, but have unpausable `unoswap` method since contract derived from `OneInchUnoswap` which doesn't have pause checker.

RECOMMENDATION

We recommend to add pause checked to `OneInchUnoswap`

CLIENT'S COMMENTARY

Decided to emit pausable functionality to save the gas.

2.4 COMMENTS

CMT-1	Transfer is not the last statement
File	OneInchExchange.sol
Severity	Comment
Status	No issue

DESCRIPTION

There is no bugs now, but it is potentialy dangerous for the re-entrancy as some state change maybe added.

- OneInchExchange.sol#L109 (30 lines before the end!)

RECOMMENDATION

It is recommended to put transfers to the end of methods.

CLIENT'S COMMENTARY

Impossible to do so, as the most of the exchangers require the initial token before the call to proceed the swap.

CMT-2	Lack of docs in ASM
File	OneInchUnoswap.sol
Severity	Comment
Status	**Acknowledged**

DESCRIPTION

- OneInchUnoswap.sol#L46
- OneInchUnoswap.sol#L71
- OneInchUnoswap.sol#L58
  It's difficult to understand the idea of the code and the purpose of methods, so it makes it difficult to verify and increases a chance to miss a bug.

RECOMMENDATION

It is recommended to add detailed explanations and docstrings for ASM code.

CMT-3	Lack of re-entrancy guard
File	OneInchExchange.sol UniERC20.sol
Severity	Comment
Status	**No issue**

DESCRIPTION

At the lines:

- OneInchExchange.sol#L52
- OneInchExchange.sol#L89
- UniERC20.sol#L27

And in all places where transfers are called.
There is no bug here but it's better to explicitly add re-entrancy guard to make code robust.

RECOMMENDATION

It is recommended to add re-entrancy guard.

CLIENT'S COMMENTARY

We don't believe re-entrancy is an issue, as tokens are always taken from msg.sender.

CMT-4	Use ASM case for gas efficiency
File	OneInchUnoswap.sol
Severity	Comment
Status	No issue

DESCRIPTION

At the line

- OneInchUnoswap.sol#L66 probably, using case for init vars in the proper way is cheaper than initiate them wrongly and then swap.

Here is a link to an example implementation:
https://gist.github.com/vsmelov/7f1b8b3eb50714998c66d49eafbce3c1

RECOMMENDATION

It is recommended to use `case/default`.

CLIENT'S COMMENTARY

In our code reserve0 and reserve1 are declared through let, and if `case/default` used, there is 0 put into reserve0 and reserve1 and then in one of the cases something valuable being put. This causes gas overhead.

CMT-5	Unclear bytes values
File	OneInchUnoswap.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the lines:

- OneInchUnoswap.sol#L9
- OneInchUnoswap.sol#L52 looks like random numbers

RECOMMENDATION

It is recommended to add comments.

CLIENT'S COMMENTARY

bytes32 which are aligned with selectors have clear values. We can add to revertWithReason.

CMT-6	Unclear flags
File	OneInchExchange.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line

- OneInchExchange.sol#L20 unclear flags.

RECOMMENDATION

It is recommended to write:

```
0x10 === 0b10000 or 1 << 4
```

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