1INCH TOKEN SMART CONTRACT AUDIT

December 23, 2020

1inch EXCHANGE

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 (name of Client). 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. 1inch platform provides a variety of features in addition to swaps. Users can trade via limit orders, deposit funds into lending protocols, move coins between different liquidity pools, and this list expands constantly.

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 contract is a standard ERC-20 token with an additional `permit` method that implements EIP-712. This improvement allows users to sign allowance for token spending without a transaction. In other words, a spender can issue a signed allowance and provide it by an off-chain to another spender or any other user, then this allowance can be applied without the participation of the original sender and it will not spend any gas from his account.

1.5 PROJECT DASHBOARD

Client	1Inch
Audit name	1Inch Token
Initial version	99fd056f91005ca521a02a005f7bcd8f77e06afc
Final version	5332caa9b91403a022e74b49c9d0fc9c6d5419f4
SLOC	48
Date	2020-12-22 - 2020-12-23
Auditors engaged	2 auditors

FILES LISTING

ERC20Permit.sol	ERC20Permit.sol
OneInch.sol	OneInch.sol
ECDSA.sol	ECDSA.sol
EIP712.sol	EIP712.sol
IERC20Permit.sol	IERC20Permit.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	1
Comment	1

CONCLUSION

The smart contract has been audited and no critical and major issues have been found. The contract massively uses libraries from a widely known OpenZeppelin set of contracts which are safe and of high quality. All spotted issues are minor and the contract itself assumed as safe to use according to our security criteria.

For the following files from the scope only consistency with their original copies from openzeppelin repository is checked (used commit: `ecc66719bd7681ed4eb8bf406f89a7408569ba9b`):

- https://github.com/1inch-exchange/1inch-token/blob/99fd056f91005ca521a02a005f7bcd8f77e06afc/contracts/ECDSA.sol
- https://github.com/1inch-exchange/1inch-token/blob/99fd056f91005ca521a02a005f7bcd8f77e06afc/contracts/EIP712.sol
- https://github.com/1inch-exchange/1inch-token/blob/99fd056f91005ca521a02a005f7bcd8f77e06afc/contracts/IERC20Permit.sol

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	Potential front running attack or losing of allowance
File	ERC20Permit.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the line ERC20Permit.sol#L53 `_approve` method replaces the allowance, so there are two potential problems here:

1. If a signer wants to increase the allowance from `A` to `B`, a receiver may withdraw `A+B` using the front-running attack.
2. If a signer wants to send `A` and `B`, but a receiver forgot to withdraw `A`, the receiver will lose ability to withdraw `A`.

RECOMMENDATION

We suggest to add `permitIncrease`, `permitDecrease` methods and use it instead of the `permit`.

2.4 COMMENTS

CMT-1	Mismatch of argument name in `_PERMIT_TYPEHASH` and `permit` function
File	ERC20Permit.sol
Severity	Comment
Status	**Fixed** at **5332caa9**

DESCRIPTION

At the line ERC20Permit.sol#L27 the `value` argument is used but at the line ERC20Permit.sol#L32 the argument name is `amount`.

RECOMMENDATION

We suggest to rename `amount` to `value` in the `permit` function.

CLIENT'S COMMENTARY

Created PR at OpenZeppelin's repo: PR-2445

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