1INCH FIXED FEE SWAP SMART CONTRACT AUDIT

July 09, 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

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 the simple contract which stores the balances of two tokens and allows exchanges between them at a rate of 1: 1 with a commission dynamically depends on balances.

1.5 PROJECT DASHBOARD

Client	1Inch
Audit name	Fixed Fee Swap
Initial version	ac5dbd4a5e46f501e0a4a728f1725095b11f3fbd 60a36947261bfe8e2914684f74c1ca72060cf3e3
Final version	60a36947261bfe8e2914684f74c1ca72060cf3e3
SLOC	111
Date	2021-06-30 - 2021-07-09
Auditors engaged	2 auditors

FILES LISTING

FixedFeeSwap.sol	FixedFeeSwap.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	2
Comment	2

CONCLUSION

Smart contract has been audited and several suspicious places were found. During audit no critical and major issues were identified. Several issues were marked as warnings and comments. After working on audit report all issues were fixed or acknowledged by client. Thus, contract is assumed as secure to use according to our security criteria.Final commit identifier with all fixes: `60a36947261bfe8e2914684f74c1ca72060cf3e3`

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	User can swap without FEE on small amounts
File	FixedFeeSwap.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

FixedFeeSwap has a method `_swap()` for exchanging tokens (FixedFeeSwap.sol#L110-L115). `_swap()` takes a comission (FixedFeeSwap.sol#L46-L48).

But there is another way to exchange tokens via `FFS` tokens. Flow:

```
// FixedFeeSwap has 1USDC, 1USDT

// 1. User deposits 1USDT token
await this.fixedFeeSwap.deposit(ether('0'), ether('1'), { from: userAddress });

// 2. User takes 1FFS tokens
// 3. User withdraw 1FFS
await this.fixedFeeSwap.withdraw(ether('1'), { from: userAddress });

// 4. User swapped part of USDT to USDC
// USDT: -0.666666666666666
// USDC: +0.333333333333334
```

Thus, the user is able to exchange tokens without commission.

RECOMMENDATION

If you think that these losses are significant for your business logic, we recommend rewriting the logic of `FixedFeeSwap`.

CLIENT'S COMMENTARY

The `deposit()` function was block for users.

WRN-2	Avoid transfer fee
File	FixedFeeSwap.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

Potentially, a token (for examle: USDT) can have a transfer fee. For example, there is ERC20 of USDT in `transfer` / `transferFrom` :

```
balances[_to] = balances[_to].add(sendAmount);
balances[owner] = balances[owner].add(fee);
balances[_from] = balances[_from].sub(_value);
```

Nowadays fee is zero for USDT. But this can change at any moment.

The `FixedFeeSwap` ignores this fee. For example, there is a next flow:

1. User call `deposit` and pass 1USDT. `safeTransferFrom` 'll be called by `FixedFeeSwap` (FixedFeeSwap.sol#L66).
2. The contract'll get <1USDT (due to fee) but the user'll get 1FFS
3. 1FFS is more than transferred to contract

RECOMMENDATION

We recommend that you carefully approach the selection of tokens for `FixedFeeSwap` .

2.4 COMMENTS

CMT-1	Constants not used
File	FixedFeeSwap.sol
Severity	Comment
Status	**Fixed** at **FixedRateSwap.sol**

DESCRIPTION

At lines:
FixedFeeSwap.sol#L19-L20
constants `_DIRECTION_MASK` and `_AMOUNT_MASK` are not used in the logic of this contract.

RECOMMENDATION

We recommend removing them.

CMT-2	Check address is not `FixedFeeSwap`
File	FixedFeeSwap.sol
Severity	Comment
Status	**Fixed** at **FixedRateSwap.sol**

DESCRIPTION

In `withdrawFor` and `_swap` there aren't checks for `to` variable:

1. FixedFeeSwap.sol#L110
2. FixedFeeSwap.sol#L78

If the user mistakenly specifies the address of the contract, then the funds will be lost forever in the contract.

RECOMMENDATION

We recommend to add the next check in `withdrawFor()`, `_swap()`:

```
require(to != address(this), "to address cannot be FixedFeeSwap");
```

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