1INCH AGGREGATION ROUTER V4 SMART CONTRACT AUDIT

October 01, 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 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 smart contracts reviewed in this audit are designed to create a universal exchange for tokens. Now it includes functionality for the following routers: `Uniswap`, `Uniswap V3` and `Clipper`. All external functions of the smart contracts `UnoswapRouter`, `UnoswapV3Router` and `ClipperRouter` will be available after the deployment of the smart contract `AggregationRouterV4`. The `swap()` function is also in the `AggregationRouterV4` contract itself. This fact is both good and bad. Good for advanced functionality and dangerous for different routers to interact with each other.Below is the description of the purpose of the studied smart contracts:

- `ClipperRouter` is intended to interact with the Clipper exchanger. Clipper is the decentralized exchange built to give the self-made crypto trader the best possible prices on small trades (< $10k).

- `LimitOrderProtocolRFQ` is designed to work with Orders. Functions often use the `OrderRFQ` structure. This structure stores order data.

- `UnoswapRouter` is designed to interact with the Uniswap exchange version less than 3.

- `UnoswapV3Router` is designed to interact with the Uniswap version 3 exchange.

- `AggregationRouterV4` is the inheritor from all previous smart contracts and includes all their functionality.

- `ArgumentsDecoder` is a low-level library for converting `uint256` and `address` variables from a `byte` variable.

- `EthReceiver` is an abstract contract for verifying that `msg.sender != tx.origin`.

- `Permitable` is an auxiliary contract with one internal function `_permit()`. It is needed to issue permissions for managing tokens.

- `UniERC20` is a library to facilitate the work with tokens and ETH.

1.4 PROJECT DASHBOARD

Client	1Inch
Audit name	Aggregation Router V4
Initial version	93868c483180cf74fc2551568f0396938b3eeaa8
Final version	0c89b19e78af194c3a85f74de5954cfc72fbe7b1
Date	September 15, 2021 - October 01, 2021
Auditors engaged	3 auditors

FILES LISTING

AggregationRouterV4.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/AggregationRouterV4.sol
ClipperRouter.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/ClipperRouter.sol
LimitOrderProtocolRFQ.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/LimitOrderProtocolRFQ.sol
UnoswapRouter.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/UnoswapRouter.sol
UnoswapV3Router.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/UnoswapV3Router.sol
ArgumentsDecoder.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/helpers/ArgumentsDecoder.sol
EthReceiver.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/helpers/EthReceiver.sol
Permitable.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/helpers/Permitable.sol

UniERC20.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/helpers/UniERC20.sol
IAggregationExecutorExtended.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/interfaces/IAggregationExecutorExtended.sol
IChi.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/interfaces/IChi.sol
IClipperExchangeInterface.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/interfaces/IClipperExchangeInterface.sol
IDaiLikePermit.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/interfaces/IDaiLikePermit.sol
IERC1271.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/interfaces/IERC1271.sol
IUniswapV3Pool.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/interfaces/IUniswapV3Pool.sol
IUniswapV3SwapCallback.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/interfaces/IUniswapV3SwapCallback.sol
IWETH.sol	https://github.com/1inch/1inch-contract/blob/93868c483180cf74fc2551568f0396938b3eeaa8/contracts/interfaces/IWETH.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	3
Warning	3
Comment	3

CONCLUSION

Smart contracts have been audited and several suspicious places have been detected. During the audit no critical issues were found, several majors, warnings and comments were spotted. After working on the reported findings all of them were fixed by the client or acknowledged (if the problem was not critical).Final commit identifier with all fixes: `0c89b19e78af194c3a85f74de5954cfc72fbe7b1`

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Decrease in the amount of tokens during exchange due to arithmetic overflow of a variable
File	UnoswapV3Router.sol
Severity	Major
Status	Fixed at 0c89b19e

DESCRIPTION

At the lines:

- UnoswapV3Router.sol#L166 и
- UnoswapV3Router.sol#L175
  a number with the type `uint256` is converted to a number with the type `int256`. This number is passed in the function parameter and, after conversion, is sent to the `UniswapV3Pool` contract.
  But, if the value of the number is greater than the maximum value for the type `int256`, an arithmetic overflow will occur.
  This is demonstrated by the following example: https://gist.github.com/mixbytes-audit/b471cc82105f856d1546ba638de20f4e.
  For example, if you take the number `57896044618658097711785492504343953926634992332820282019728792003956564819970`, then after the conversion you get the value-`57896044618658097711785492504343953926634992332820282066728791990 39`
  We see a decrease in modulus of the initial value of the variable.

RECOMMENDATION

Before lines 166 and 175, you need to check that the value of the number is less than the maximum value for the type `int256`.

MJR-2	Increase in the amount of tokens during exchange due to arithmetic overflow of a variable
File	UnoswapV3Router.sol
Severity	Major
Status	Fixed at 0c89b19e

DESCRIPTION

At the lines:

- UnoswapV3Router.sol#L170 и
- UnoswapV3Router.sol#L179
  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 166 and 175, 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.

MJR-3	No validation of the value of the variable `msg.value`
File	ClipperRouter.sol
Severity	Major
Status	Fixed at c40696f8

DESCRIPTION

At the lines ClipperRouter.sol#L57-L68
processing of the input data is done before the exchange procedure.
`ETH` is not required to work with `WETH` and regular tokens. But the user can inadvertently transfer it.
In this case, the user will lose these `ETH`.
To prevent this from happening, you need to add checks before lines 60 and 67:

```
require(msg.value == 0, "CL1IN: wrong msg.value");
```

RECOMMENDATION

Additional checks need to be added.

2.3 WARNING

WRN-1	There is no processing of the value returned by the function
File	ClipperRouter.sol AggregationRouterV4.sol RouteWrapperExtension.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

When working with tokens according to the ERC-20 standard, it is necessary to check the values that return functions.
In some places it is not done. This can be seen on the following lines:

- ClipperRouter.sol#L58
- ClipperRouter.sol#L76
- AggregationRouterV4.sol#L152
- RouteWrapperExtension.sol#L36

RECOMMENDATION

It is recommended to add processing of the value returned by the function.

CLIENT'S COMMENTARY

- WETH always returns true
- uniTransfer already has those checks in place

WRN-2	Accessing an interface using the `uint256` type
File	UnoswapV3Router.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the lines:

- UnoswapV3Router.sol#L163 и
- UnoswapV3Router.sol#L172
  the `IUniswapV3Pool` interface is called to call the `swap()` function.
  To work through the interface, the value of the `pool` variable must be of type `address`.
  But on the line UnoswapV3Router.sol#L160 this variable is of type `uint256`.
  Other types must be cast to type `address`.

RECOMMENDATION

The source code needs to be corrected.

CLIENT'S COMMENTARY

Type is uint256 as we used tight packing to pack additional flags to the same slot with pool address, like _ONE_FOR_ZERO, _WETH_WRAP and _WETH_UNWRAP.
And uint256 -> address casts are still allowed in solidity 0.7.6

WRN-3	Zero-address checking
File	ClipperRouter.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

The `_clipperPool` are assignments in constructor, but not checks to zero-address before: ClipperRouter.sol#L25

RECOMMENDATION

It is recommended to check

```
    require(clipperExchange.theExchange() != address(0));
```

CLIENT'S COMMENTARY

We'll skip this as it is unlikely that we'll pass some address with valid ClipperExchangeInterface ABI that will return invalid pool.

2.4 COMMENT

CMT-1	Invalid function parameter
File	Permitable.sol
Severity	Comment
Status	Fixed at 993fe111

DESCRIPTION

At the lines Permitable.sol#L16-L39 lines have a function for granting permissions to work with tokens from another address.
This function has an `amount` parameter, which is needed to transfer the number of tokens.
But on lines 22 and 25, permission is approved without using this parameter.
The number of tokens is also passed in the `permit` variable.
This logic can lead to errors when working with this function.

RECOMMENDATION

It is recommended to refactor your source code.

CLIENT'S COMMENTARY

Amount was used to continue execution in case of permit failure but when there is enough allowance. But we decided to remove that optimisation in PR-81

CMT-2	The likelihood that anyone can use another approval
File	ClipperRouter.sol UnoswapV3Router.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

All external functions of the smart contracts `UnoswapRouter`, `UnoswapV3Router` and `ClipperRouter` will be available after the deployment of the smart contract `AggregationRouterV4`.
There are functions that exchange tokens. Before this operation it is required that the user gives an approval to the address of the contract for the disposal of his tokens.
For example, we see it in this line:
ClipperRouter.sol#L67.
But there is some time between the `approve()` operation and the `swap()` operation. They are not done at the same time.
In between these events, an attacker can run the external function `uniswapV3SwapCallback()` located here:
UnoswapV3Router.sol#L83-L158.
As a parameter `calldata`, he can specify the necessary data for transferring tokens to his wallet on lines
UnoswapV3Router.sol#L148 and
UnoswapV3Router.sol#L155.
Here `payer` will be the wallet address of the user who gave the `approve()`, but has not yet performed the data exchange.

RECOMMENDATION

It is recommended to restrict the call of this function only from the Uniswap V3 pool.

CLIENT'S COMMENTARY

It is already restricted. See UnoswapV3Router.sol#L136

CMT-3	The likelihood that anyone can withdraw tokens from the balance of the contract
File	UnoswapV3Router.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

All external functions of the smart contracts `UnoswapRouter`, `UnoswapV3Router` and `ClipperRouter` will be available after the deployment of the smart contract `AggregationRouterV4`.
At the lines UnoswapV3Router.sol#L151-L153 have a `rescueFunds()` function. It is needed for the owner of the contract so that he can withdraw any tokens from the balance of the contract.
Thus, it is assumed that there are tokens on the contract balance.
An attacker could run the external function `uniswapV3SwapCallback()` located here: UnoswapV3Router.sol#L83-L158.
As a parameter `calldata`, he can specify the necessary data for transferring tokens to his wallet on lines
UnoswapV3Router.sol#L146 and
UnoswapV3Router.sol#L153.

RECOMMENDATION

It is recommended to restrict the call of this function only from the Uniswap V3 pool.

CLIENT'S COMMENTARY

Same as above
Also it is not expected that tokens will appear on Router balance. So that `rescueFunds` is only here to rescue accidentally sent tokens.

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