1inch

SMART CONTRACT AUDIT

ZOKYO.

August 5th 2022 | v. 2.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

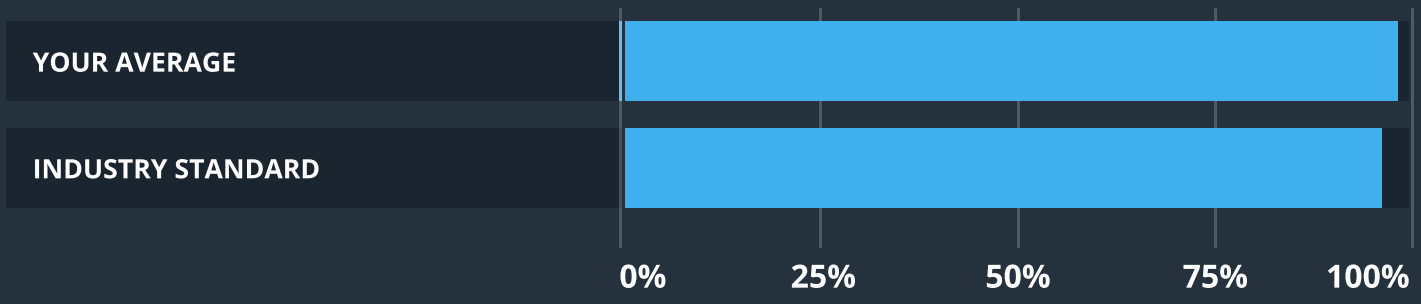
TECHNICAL SUMMARY

This document outlines the overall security of the 1inch smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the 1inch smart contract codebase for quality, security, and correctness.

Contract Status

Testable Code



The testable code is 99%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the 1inch team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the 1inch repository.
https://github.com/1inch/1inch-contract

1st audit revision
https://github.com/1inch/1inch-contract/releases/tag/v5-audit
Last audited commit: 274d414497f57d42db928b5380d88ca66b239811, master branch

2nd audit revision
https://github.com/1inch/1inch-contract/tree/v5-audit-pre-release
8aa5ec4b4871b1d63bb045ddb78aaf7c5dc84dfa, branch v5-audit-pre-release
Last audited commit: 47f1bc6b5d715efc2d9a8af2d20987ed71722d02

Additional tools in the list of dependancies
https://github.com/1inch/limit-order-protocol
d8437885744543e3f057e84e1b0a05c4c211c553, master branch

https://github.com/1inch/solidity-utils
eec6b523860af5215a8dd196fe3aff3a4d252fc9, master branch
Last audited at d43b4afe9a6ea5089dd5181f059dbe4e9909a112
Within the scope of this audit Zokyo auditors have reviewed the following contract(s):

- AggregationRouterV5.sol
- ClipperRouter.sol
- Errors.sol
- GenericRouter.sol
- UnoswapRouter.sol
- UnoswapV3Router.sol
- OrderLib.sol
- OrderMixin.sol
- ECDSA.sol
- RevertReasonForwarder.sol
- SafeERC20.sol
- StringUtil.sol
- UniERC20.sol
- LimitOrderProtocol
- OrderRFQLib.sol
- OrderRFQMixin.sol
- AmountCalculator.sol
- NonceManager.sol
- PredicateHelper.sol
- ArgumentsDecoder.sol
- Callib.sol
- EthReceiver.sol

Throughout the review process, care was taken to ensure that the contract:

- Implements and adheres to existing standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of resources, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of 1inch smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

During the first iteration, the Zokyo security team audited the 1inch contracts set for limit orders over Uniswap and UniswapV3. Auditors have met the high-quality code, which follows Solidity best practices and provides all possible gas optimizations. The team has performed in-deep review of the code with a full analysis of the business logic of smart contracts. There was performed exploratory testing of the protocol to detect any suspicious issues or unclear functionality. The team has carefully checked contracts logic to detect probable misordering of commands or loopholes, as well as the assembly code in _unoswap() function, which appears to be in the system's core.

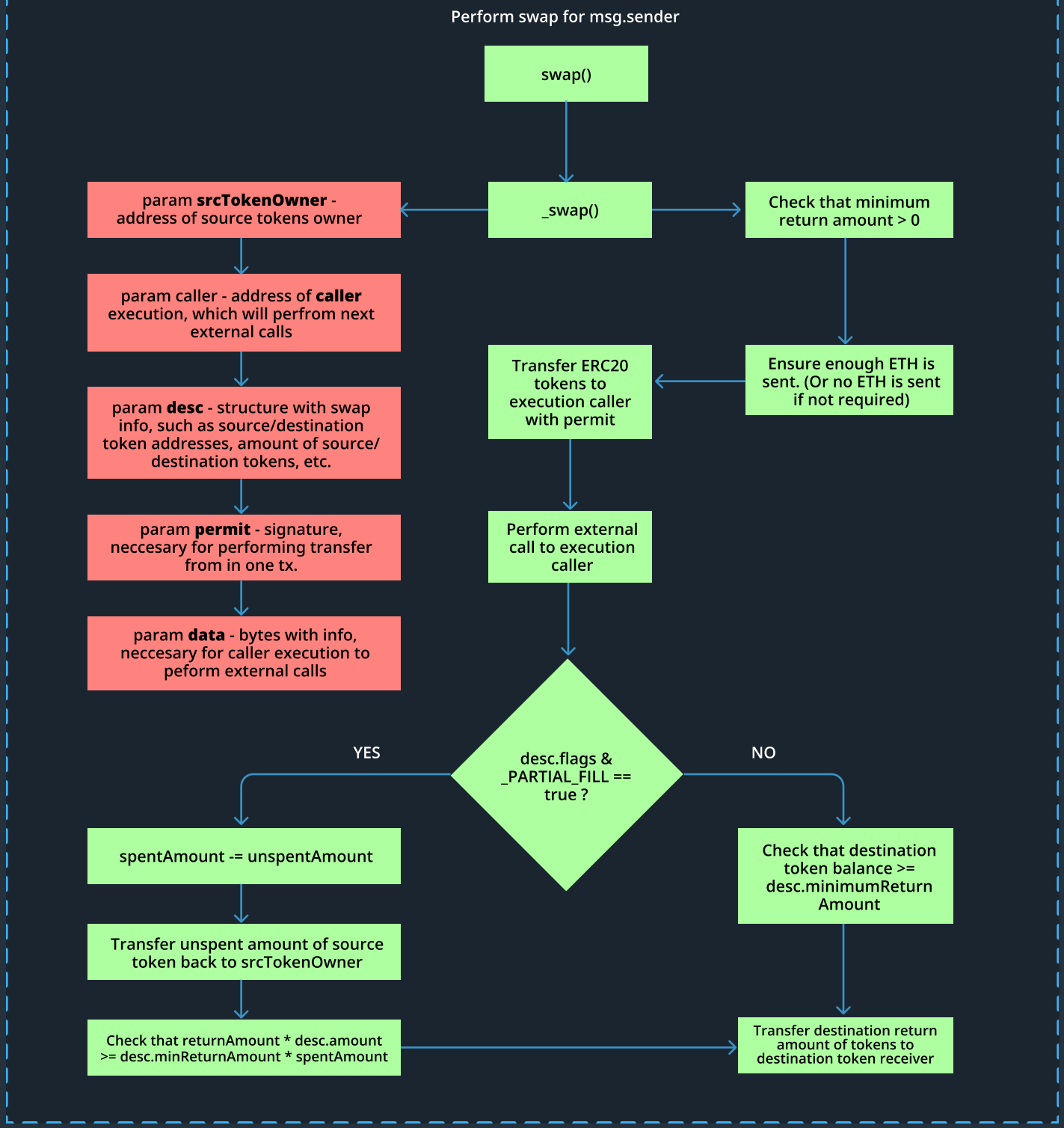
There were no critical issues found. The only issues were connected to the standard usage of ETH transfer via call() method and the possibility to block smart-contracts performance via GenericRouter direct usage.

Also, during the audit, the Security team compared gas spending between Uniswap V2/V3 Routers and 1Inch Aggregation V5 Router. On average, 1Inch Router consumes 15% less gas, than Uniswap Routers. For example, swapping tokens with 3 assets in the route through Uniswap V2 Router consumes around 185000 Gas Units, whereas 1Inch Router consumes around 15000 Gas Units. With the current Gas and ETH price, savings with using 1Inch Router are 1-1.10$ in average.

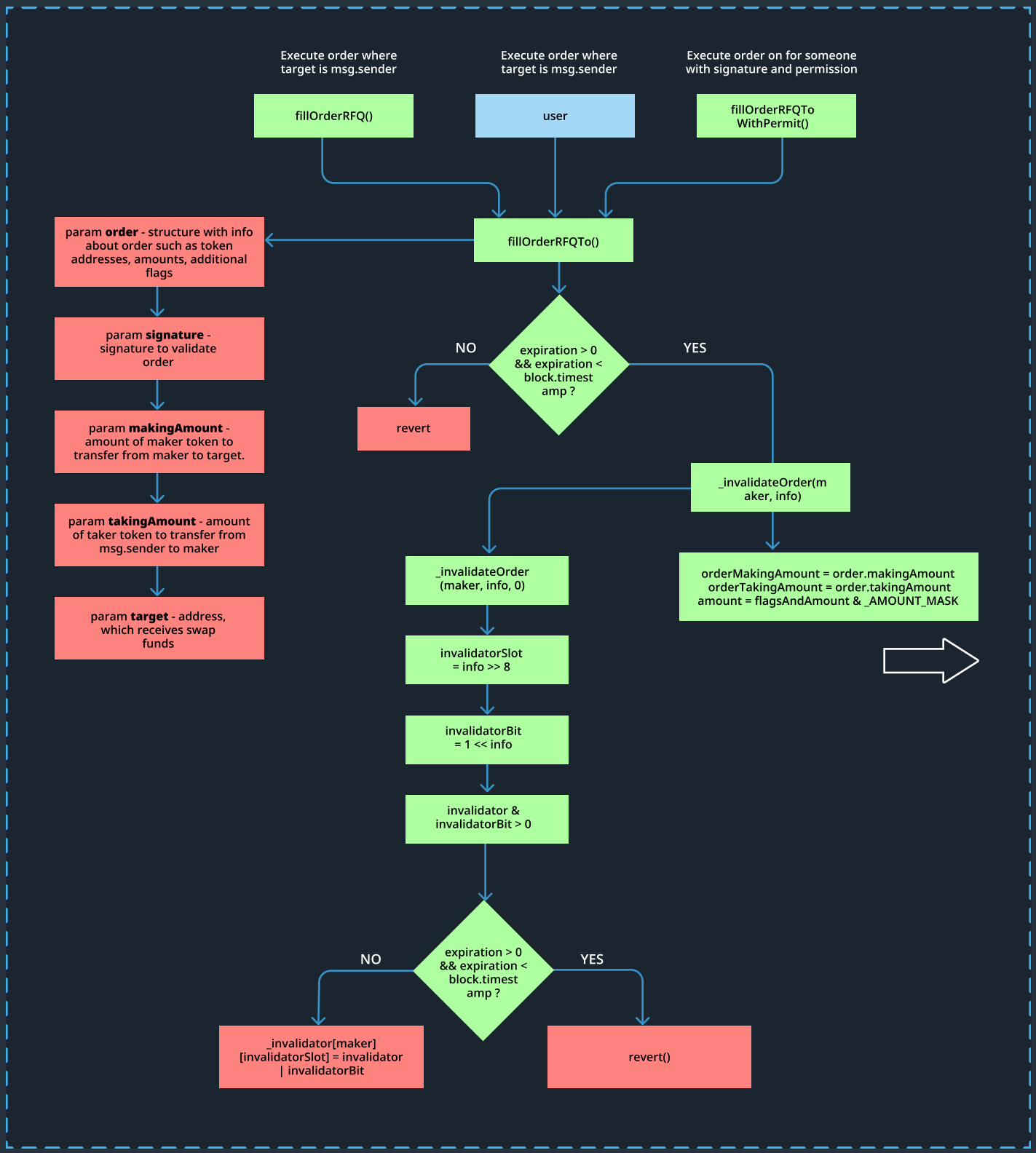
During the second audit iteration, auditors have carefully checked the updated set of contracts for the AggregtionRouterV5 and all dependencies (including updated routers and limit order functionality). 1inch team has diversified the hierarchy of contracts, updated connections between contracts, and added a lot of documentation and new tests.

There were no new crucial issues found except problems with standard ETH transfer functionality. However, Zokyo team has added its own set of tests to verify the updated functionality, despite excellent native coverage.

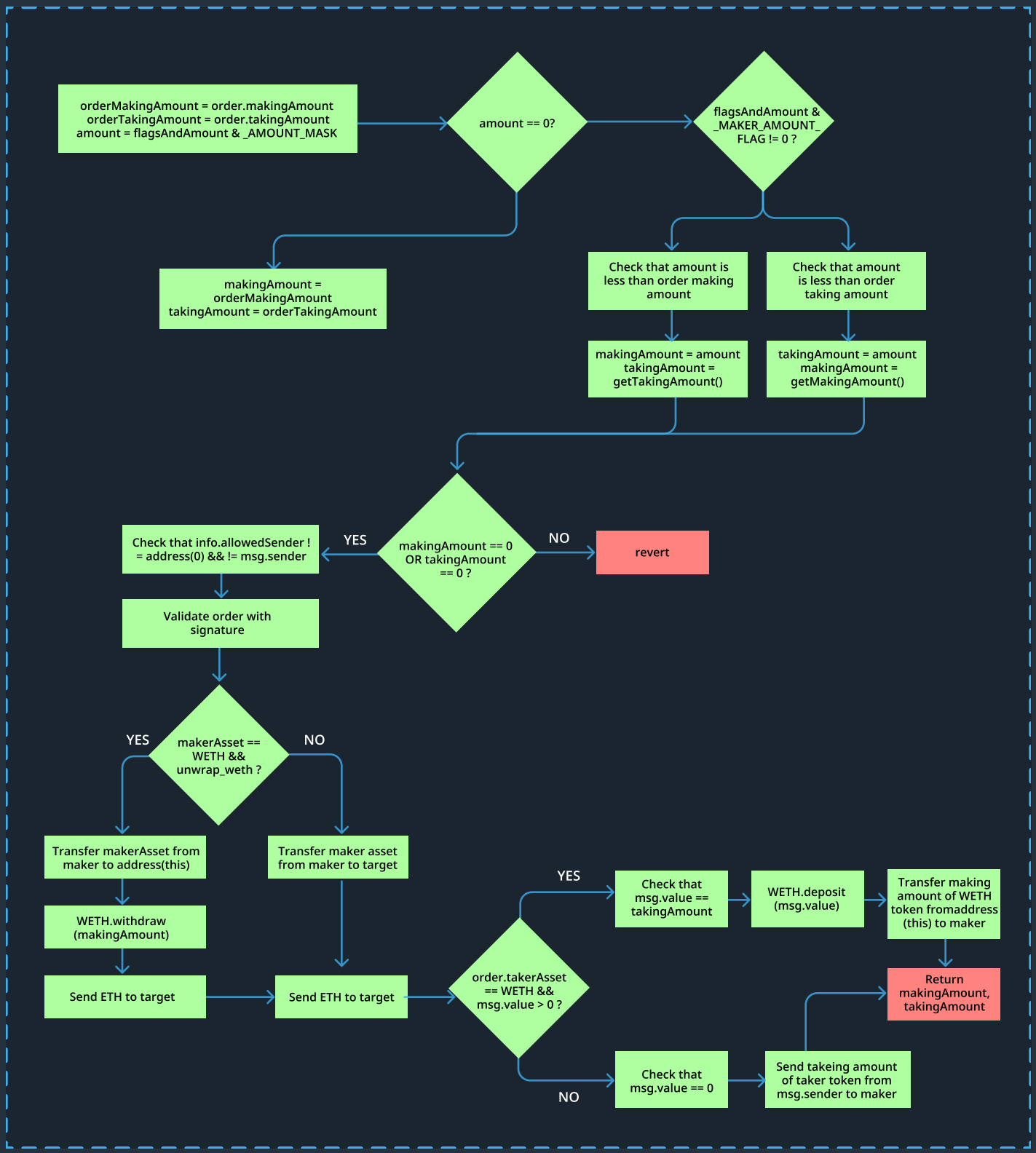
PROTOCOL OVERVIEW
1. GENERICROUTER.SOL



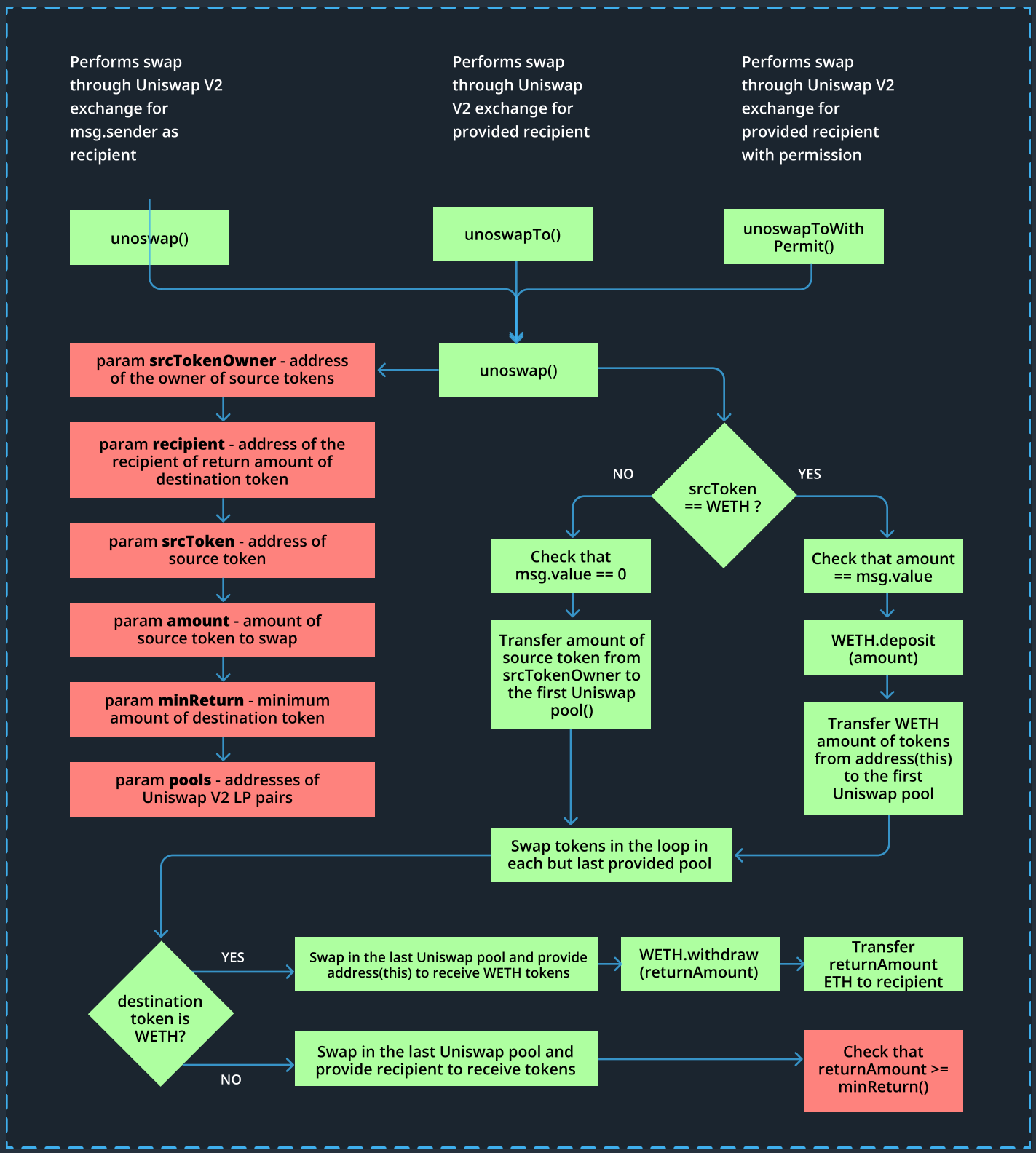
2. LIMITORDERPROTOCOLRFQ.SOL



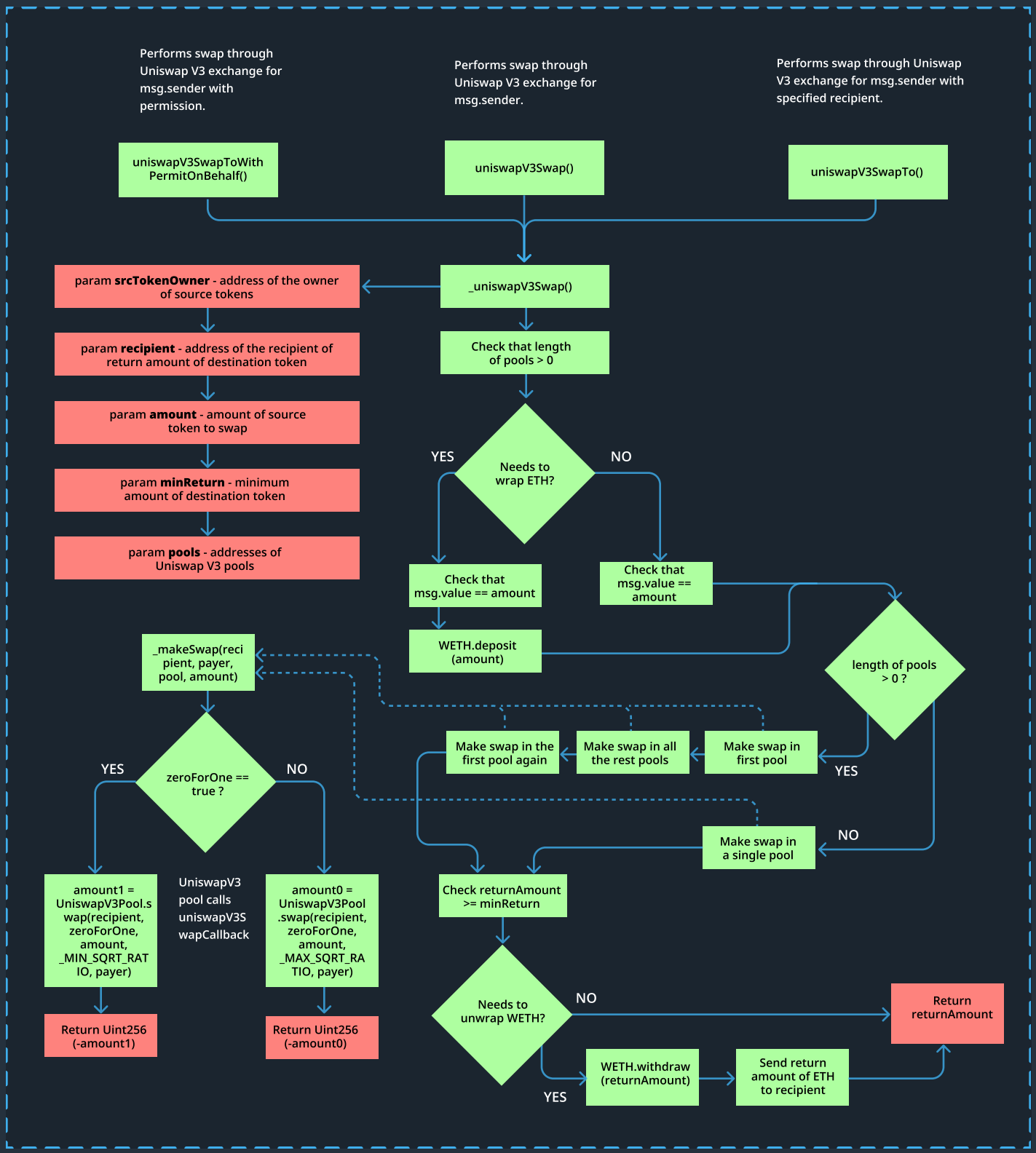
2. LIMITORDERPROTOCOLRFQ.SOL



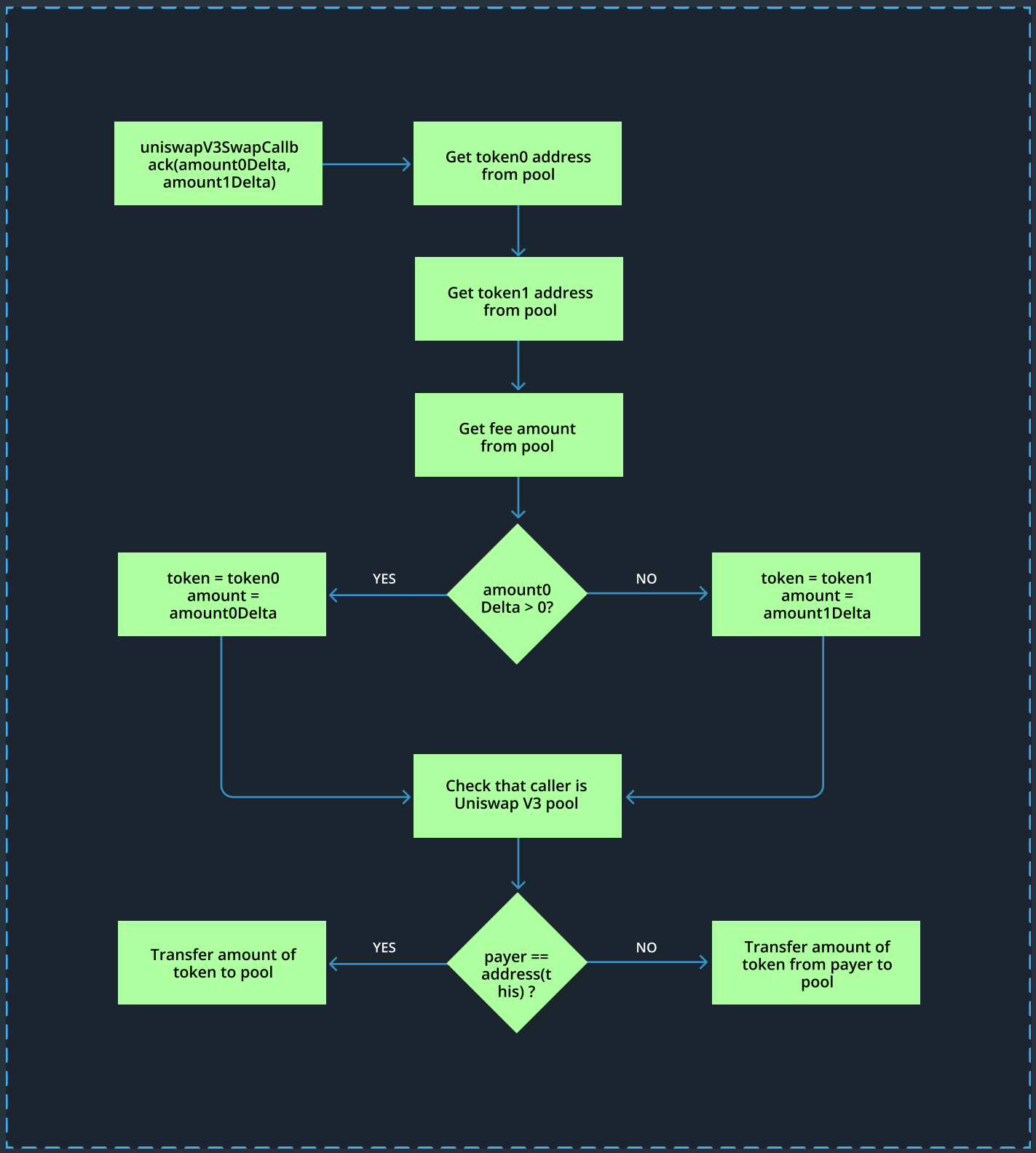
3. UNOSWAPROUTER.SOL



4. UNOSWAPV3ROUTER.SOL



4. UNOSWAPV3ROUTER.SOL (CALLBACK)



ORDERMIXIN.SOL

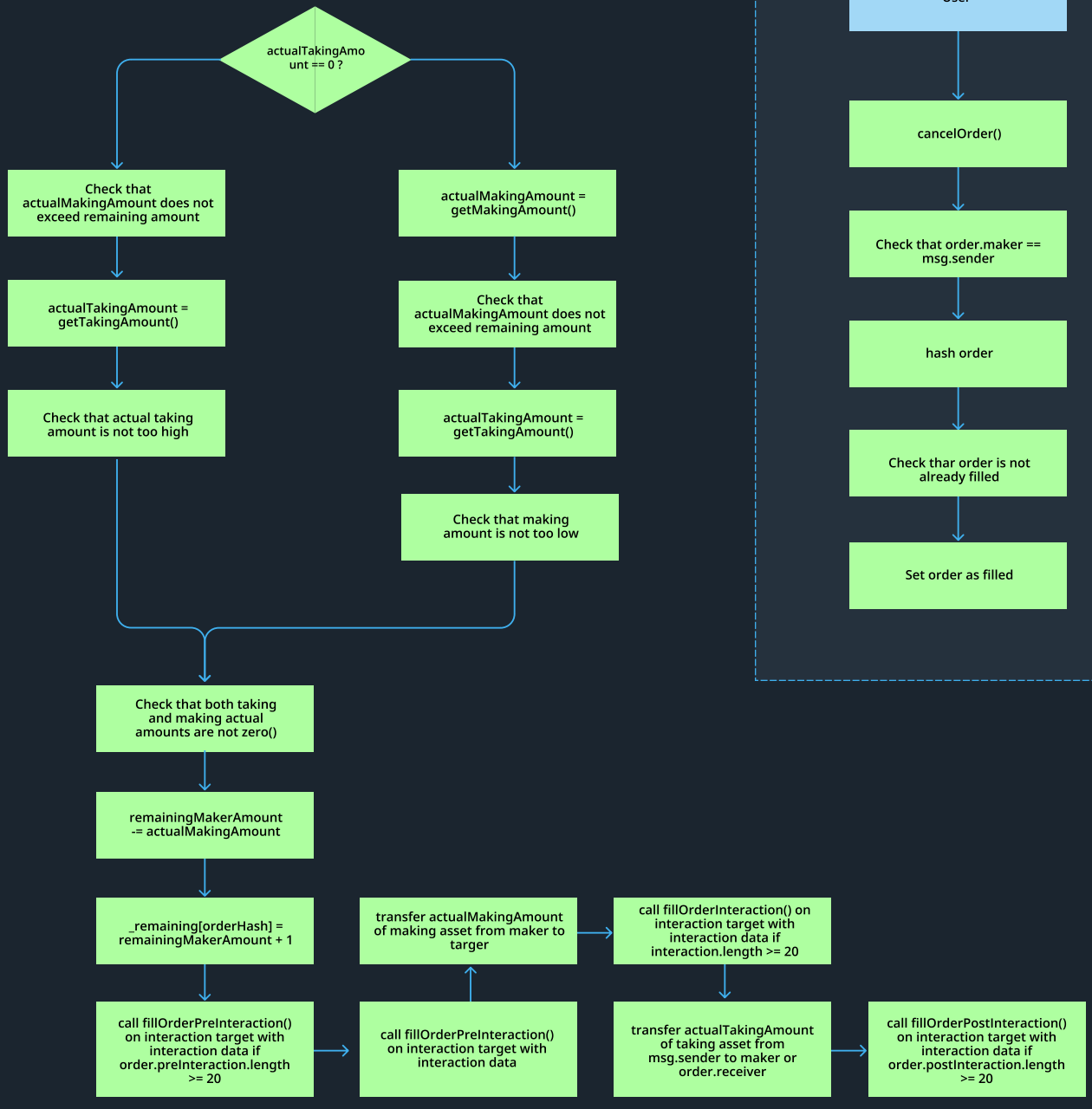
Allows to create order for maker and fill it fully or partially, or fill already existing order. Amount of maker asset is transferred to taker, and taker asset is transferred to maker.

ORDERMIXIN.SOL

Allows to create order for maker and fill it fully or partially, or fill already existing order. Amount of maker asset is transferred to taker, and taker asset is transferred to maker.

ORDERMIXIN.SOL

Allows to create order for maker and fill it fully or partially, or fill already existing order. Amount of maker asset is transferred to taker, and taker asset is transferred to maker.



CLIPPERROUTER.SOL

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Issues tagged “Verified” contain unclear or suspicious functionality that either needs explanation from the Customer’s side or it is an issue that the Customer disregards as an issue. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

HIGH | RESOLVED

Deprecated ETH transfer.

LimitOrderProtocolRFQ.sol: function fillOrderRFQTo(), line 165.
Due to the Istanbul update there were several changes provided to the EVM, which made .transfer() and .send() methods deprecated for the ETH transfer. Thus it is highly recommended to use .call() functionality with mandatory result check, or the built-in functionality of the Address contract from OpenZeppelin library.

Recommendation:
Correct ETH sending functionality.

Post-audit
Fixed after the 1st audit iteration

HIGH | RESOLVED

Deprecated ETH transfer.

UniERc20.sol: function uniTransfer(), line 38.
UniERc20.sol: function uniTransferFrom(), line 53.
Due to the Istanbul update there were several changes provided to the EVM, which made .transfer() and .send() methods deprecated for the ETH transfer. Thus it is highly recommended to use .call() functionality with mandatory result check, or the built-in functionality of the Address contract from OpenZeppelin library.

Recommendation:
Correct ETH sending functionality.

MEDIUM | UNRESOLVED

Performance of swap can be blocked.

GenericRouter.sol: function swap(), line 84.
In case, there are unspent source tokens after swap, unspent amount is subtracted from spentAmount. In case a malicious actor transfers some amount of source tokens to contract, which would be greater than spentAmount of swap, subtraction will underflow, reverting transaction.

Recommendation:
Since a contract is not supposed to hold any tokens, consider adding a function to sweep any provided tokens in order for the protocol not to get blocked.

Post-audit:
1inch team is acknowledged about the issue.

INFORMATIONAL | VERIFIED

Commented code.

OrderLib.sol: struct Order, lines 18-25.
ClipperRouter.sol: function clipperSwapTo(), lines 99-100, 130.
Preproduction code should not contain unfinished logic or commented code.

Recommendation:
Remove commented code.

Post-audit:
For OrderLib these comments are left to explain the structure of comressed bytes data. For ClipperRouter they are left to explain what assembly code is supposed to do. We'll keep them as we think that they provide some information for the reader.

	EthReceiver.sol	ECDSA.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	NonceManager.sol	Callib.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	ArgumentsDecoder.sol	PredicateHelper.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	LimitOrderProtocol.sol	AmountCalculator.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	OrderRFQMixin.sol	OrderRFQLib.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	OrderMixin.sol	OrderLib.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	UnoswapV3Router.sol	GenericRouter.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	ClipperRouter.sol	AggregationRouterV5.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Secured team

As part of our work assisting 1inch in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the 1inch contract requirements for details about issuance amounts and how the system handles these.

The first audit iteration tests.

Contract: AggregationRouterV5
- ✓ Function: .rescueFunds() (409ms)
- ✓ Function: .destroy()

Contract: GenericRouter
- ✓ Should revert if transaction requires extra ETH and not enough ETH sent (64ms)
- ✓ Should revert if extra ETH sent (64ms)
- ✓ Should revert if partial fill and not enough return amount (1587ms)
- ✓ Should revert if not enough return amount (564ms)

Contract: LimitOrderProtocolRFQ

Permiting
- ✓ Permit: makingAmount = 0, takingAmount = 0 (248ms
- ✓ Permit: makingAmount = 0, takingAmount > makingAmount (108ms)
- ✓ Permit: makingAmount = 0, takingAmount > makingAmount (119ms)
- ✓ Permit: makingAmount > makingAmount, takingAmount = 0 (110ms)
- ✓ Permit: makingAmount = makingAmount (103ms)
- ✓ Permit: makingAmount < takingAmount (281ms)
- ✓ Permit: makingAmount > takingAmount, takingAmount = 0 (172ms)
- ✓ Permit: makingAmount > takingAmount (102ms)
- ✓ Permit: makingAmount = takingAmount (102ms)

Contract: NonceManager
- ✓ Get nonce
- ✓ Get advance nonce

Contract: Unoswap

Permit
- ✓ USDC => DAI with big minReturn (2127ms)

18 passing (17s)

The second audit iteration tests.

```
Contract: AggregationRouterV5
    ✓ Normal deploy (630ms)
    ✓ Can`t deploy if weth is zero address (681ms)
    ✓ Should rescue funds (122ms)
    ✓ Normal destroy (310ms)
Contract: UnoswapV3Router
    ✓ Should revert if length of pools is 0
    ✓ Should revert if eth should be wrapped and msg.value != amount
Contract: UnoswapRouter
    ✓ Should execute unoswapTo
    ✓ Should revert if output amount is less than minimum output amount
Contract: GenericRouter
    ✓ Should revert if transaction requires extra ETH and not enough ETH sent
    ✓ Should revert if extra ETH sent
    ✓ Should revert if partial fill and not enough return amount
    ✓ Should revert if not enough return amount
    ✓ Should revert if minimum return amount is 0
Contract: GenericRouter
    ✓ Should perform clipper swap with permit
    ✓ Normal deploy (63ms)
    ✓ Normal clipper swap (1181ms)
    ✓ Should revert if source is ETH and msg.value != input amount (739ms)
    ✓ Should revert if source is WETH and msg.value != 0
    ✓ Should revert if source is ERC20 and msg.value != 0
  8 passing
```

FILE	% STMTS	% BRANCH	% FUNCS
AggregationRouterV5.sol	100	100	100
Errors.sol	100	100	100
ClipperRouter.sol	100	100	100
GenericRouter.sol	94.12	85.71	100
UnoswapRouter.sol	100	100	100
UnoswapV3Router.sol	100	100	100
All files	**99**	**97.6**	**100**

Contract: OrderMixin

- ✓ Should revert getting remainings if order not exists
- ✓ Should return raw remainings
- ✓ Should not cancel filled order
- ✓ Should revert fill order with permit is permit too short
- ✓ Should revert fill order to if target is zero address
- ✓ Should execute permit during filling the order
- ✓ Should fulfill order in two transactions
- ✓ Should revert if transfer from maker failed
- ✓ Should revert if transfer from taker failed
- ✓ Should simulate call

Contract: OrderRFQMixin

- ✓ Should revert if target is zero address
- ✓ Should revert if private order is executed not by allowed sender
- ✓ Should revert if making amount exceed order making amount
- ✓ Should revert if taking amount exceed order taking amount
- ✓ Should revert if signed by wrong user
- ✓ Should verify signature if signer is contract
- ✓ Should cancel order

17 passing (1s)

FILE	% STMTS	% BRANCH	% FUNCS
OrderLib.sol	100	100	100
OrderMixin.sol	97.83	92.86	100
OrderRFQLib.sol	100	100	100
OrderRFQMixin.sol	98.36	84.78	100
AmountCalculator.sol	100	100	100
NonceManager.sol	100	100	100
PredicateHelper.sol	93.88	83.33	90.91
ArgumentsDecoder.sol	100	100	100
Callib.sol	100	100	100
Errors.sol	100	100	100
All files	**99**	**96**	**99**

Library: **UniERC20**

✓ Should revert increasing allowance if value is too big

Library: **StringUtil**

✓ Should hexify address

Library: **UniERC20**

✓ Should return true if token is ETH
✓ Should return false if token is not ETH
✓ Should return ETH balance
✓ Should return ERC20 balance
✓ Should transfer ETH
✓ Should transfer ERC20 tokens
✓ Should not transfer zero amount
✓ Should transfer from ETH
✓ Should revert transferring from ETH if msg.value < amount
✓ Should revert transferring from ETH if from != msg.sender
✓ Should revert transferring from ETH if to != contract address
✓ Should return exceeded ETH when transferring from
✓ Should transfer from ERC20
✓ Should not transfer from 0
✓ Should return symbol
✓ Should return name
✓ Should return ETH name if provided token is ETH

✓ Should approve ERC20 tokens
✓ Should revert approving if provided token is ETH

Contract: **EthReceiver**

✓ Should not receive ETH directly
✓ Should receive ETH not directly

23 passing (2s)

FILE	% STMTS	% BRANCH	% FUNCS
EthReceiver.sol	100	100	100
ECDSA.sol	100	100	100
RevertReasonForwarder.sol	100	100	100
SafeERC20.sol	100	94.44	100
StringUtil.sol	100	100	100
UniERC20.sol	80	78.13	100
All files	**90.28**	**86.67**	**100**

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by 1inch team

As part of our work assisting 1inch team in verifying the correctness of their contract code, our team has checked the complete set of unit tests prepared by the 1inch team.

It needs to be mentioned, that the original code has a significant original coverage with testing scenarios provided by the 1inch team. All of them were also carefully checked by the auditors' team.

```
Contract: AggregationRouterV5
  Protection
    ✓ protection should pass (2098ms)
    ✓ protection should fail (368ms)
  Main
  eth -> dai
    ✓ simple mooniswap eth -> dai (1811ms)
    ✓ mooniswap eth -> dai with patched balance (377ms)
    ✓ simple uniswapv2 eth -> dai (450ms)
    ✓ direct uniswapv2 eth -> dai (387ms)
    ✓ simple uniswapv1 eth -> dai (576ms)
    ✓ uniswapv1 eth -> dai with patched balance (392ms)
    ✓ uniswapv1 + mooniswap eth -> dai (857ms)
  dai -> eth
    ✓ simple mooniswap dai -> eth (848ms)
    ✓ simple uniswapv1 dai -> eth (426ms)
    ✓ simple kyber dmm dai -> eth (1239ms)
    ✓ simple uniswapv2 dai -> eth (367ms)
    ✓ direct uniswapv2 dai -> weth (522ms)
Contract: Clipper-v2
    ✓ usdc -> eth (2539ms)
    ✓ eth -> usdc (2058ms)
    ✓ usdc -> dai (1587ms)
Contract: Dodo
  usdc -> weth
    ✓ patch by balance (5849ms)
    ✓ simple patch (1145ms)
Contract: Leftovers
  eth -> dai
```

```
    ✓ no referral (852ms)
    ✓ simple referral (818ms)
    ✓ bad rate (629ms)
    ✓ priority referral (757ms)
  dai -> eth
    ✓ simple referral (696ms)
Contract: RouteWrapper
  eth -> dai
    ✓ test wrapper (3802ms)
Contract: Unoswap
  Permit
    ✓ USDC => DAI (2728ms)
    ✓ rejects reused signature (444ms)
    ✓ rejects other signature (193ms)
    ✓ rejects expired permit (167ms)
Contract: Main
  ETH => DAI
    ✓ uniswap router eth => dai (785ms)
    ✓ 0x router ETH => DAI (1005ms)
    ✓ 1inch unirouter ETH => DAI (764ms)
    ✓ 1inch exchange ETH => DAI (1873ms)
  ETH => USDC => DAI
    ✓ uniswap router ETH => USDC => DAI (3148ms)
    ✓ 0x router ETH => USDC => DAI (1472ms)
    ✓ 1inch unirouter ETH => USDC => DAI (1230ms)
    ✓ 1inch exchange ETH => USDC => DAI (2242ms)
  DAI => ETH
    ✓ Uniswap router DAI => ETH (1346ms)
    ✓ 0x router DAI => ETH (763ms)
    ✓ 1inch unirouter DAI => ETH (675ms)
    ✓ 1inch exchange DAI => ETH (1746ms)
  DAI => WETH
    ✓ Uniswap router DAI => WETH (970ms)
    ✓ 0x router DAI => WETH (860ms)
    ✓ 1inch unirouter DAI => WETH (789ms)
    ✓ 1inch exchange DAI => WETH (1357ms)
  DAI => WETH => USDC
    ✓ Uniswap router DAI => WETH => USDC (2133ms)
    ✓ 0x router DAI => WETH => USDC (1550ms)
    ✓ 1inch unirouter DAI => WETH => USDC (1527ms)
    ✓ 1inch exchange DAI => WETH => USDC (3986ms)
  DAI => WETH => USDC => USDT257710
```

```
    ✓ Uniswap router DAI => WETH => USDC => USDT (3874ms)
    ✓ 0x router DAI => WETH => USDC => USDT (1738ms)226696
    ✓ 1inch unirouter DAI => WETH => USDC => USDT (1556ms)
    ✓ 1inch exchange DAI => WETH => USDC => USDT (3311ms)
Contract: UnoswapV3
  Permit
    ✓ USDC => DAI (1401ms)
    ✓ rejects reused signature (518ms)
    ✓ rejects other signature (209ms)
    ✓ rejects expired permit (175ms)
  Main
  MinReturn
    ✓ WETH => DAI (975ms)
  Tokens
    ✓ WETH => DAI (1732ms)
    ✓ WETH => DAI => USDC (2894ms)
    ✓ WETH => USDC => DAI (2980ms)
    ✓ WETH => DAI => WETH (2737ms)
    ✓ WETH => DAI => WETH => USDC (3308ms)
    ✓ WETH => DAI => WETH => USDC => USDT (5723ms)
  Native
    ✓ ETH => DAI (1349ms)
    ✓ DAI => ETH (1076ms)
    ✓ ETH => USDC => DAI (2646ms)
    ✓ DAI => USDC => ETH (1741ms)
```

FILE	% STMTS	% BRANCH	% FUNCS
AggregationRouterV5.sol	33.33	50	33.33
Errors.sol	100	100	100
ClipperRouter.sol	61.29	62.5	75
GenericRouter.sol	92.31	54.55	100
UnoswapRouter.sol	80	50	75
UnoswapV3Router.sol	100	85.71	100
All files	**77.82**	**67.12**	**80.55**

```
Contract: ChainLinkExample
    ✓ eth -> dai chainlink+eps order (87ms)
    ✓ dai -> 1inch stop loss order (72ms)
    ✓ dai -> 1inch stop loss order predicate is invalid (84ms)
    ✓ eth -> dai stop loss order (69ms)
Contract: ContractRFQ
    ✓ should fill contract-signed RFQ order (63ms)
Contract: LimitOrderProtocol
    ✓ domain separator
Contract: Interactions
  whitelist
    ✓ should fill and unwrap token (60ms)
    ✓ should check whitelist and fill and unwrap token (66ms)
    ✓ should revert transaction when address is not allowed by whitelist (63ms)
  recursive swap
    ✓ opposite direction recursive swap (88ms)
    ✓ unidirectional recursive swap (98ms)
    ✓ triple recursive swap (116ms)
  check hash
    ✓ should check hash and fill (49ms)
    ✓ should revert transaction when orderHash not equal target
Contract: LimitOrderProtocol
  wip
    ✓ transferFrom
    ✓ should not swap with bad signature
    ✓ should not fill (1,1)
    ✓ should not fill above threshold
    ✓ should not fill below threshold
    ✓ should fail when both amounts are zero
    ✓ should swap fully based on signature (364ms)
    ✓ should swap half based on signature (301ms)
    ✓ should floor maker amount (39ms)
    ✓ should fail on floor maker amount = 0
    ✓ should ceil taker amount (42ms)
    ✓ ERC721Proxy should work (53ms)
  Permit
  fillOrderToWithPermit
    ✓ DAI => WETH (59ms)
    ✓ rejects reused signature (63ms)
    ✓ rejects other signature
    ✓ rejects expired permit
  Amount Calculator
```

```
    ✓ empty getTakingAmount should work on full fill (43ms)
    ✓ empty getTakingAmount should not work on partial fill
    ✓ empty getMakingAmount should work on full fill (41ms)
    ✓ empty getMakingAmount should not work on partial fill
  Order Cancelation
    ✓ should cancel own order
    ✓ should not cancel foreign order
    ✓ should not fill cancelled order
  Private Orders
    ✓ should fill with correct taker (40ms)
    ✓ should not fill with incorrect taker
  Predicate
    ✓ benchmark gas
    ✓ benchmark gas real case
    ✓ benchmark gas real case (optimized)
    ✓ `or` should pass (47ms)
    ✓ `or` should fail
    ✓ `and` should pass (48ms)
    ✓ nonce + ts example (49ms)
    ✓ advance nonce
    ✓ `and` should fail
  Expiration
    ✓ should fill when not expired (44ms)
    ✓ should not fill when expired
    ✓ should fill partially if not enough coins (taker) (40ms)
    ✓ should fill partially if not enough coins (maker) (42ms)
  ETH fill
    ✓ should fill with ETH (41ms)
    ✓ should reverted with takerAsset WETH and incorrect msg.value (72ms)
    ✓ should reverted with takerAsset non-WETH and msg.value greater than 0 (59ms)
Contract: NonceManager
    ✓ Get nonce - should return zero by default
    ✓ Advance nonce - should add to nonce specified amount
    ✓ Increase nonce - should add to nonce only 1
    ✓ Nonce equals - should return false when nonce does not match
    ✓ Nonce equals - should return true when nonce matches
Contract: RFQ Orders in LimitOrderProtocol
  wip
    ✓ should swap fully based on RFQ signature (329ms)
    ✓ should swap fully based on RFQ signature (compact) (278ms)
  Permit
  fillOrderRFQToWithPermit
```

✓ DAI => WETH (69ms)
✓ rejects reused signature (66ms)
✓ rejects other signature (47ms)
✓ rejects expired permit (41ms)
OrderRFQ Cancelation
✓ should cancel own order
✓ should cancel own order with huge number
✓ should not fill cancelled order
✓ should not fill cancelled order (compact)
Expiration
✓ should fill RFQ order when not expired
✓ should fill RFQ order when not expired (compact)
✓ should partial fill RFQ order (38ms)
✓ should partial fill RFQ order (compact)
✓ should fully fill RFQ order
✓ should fully fill RFQ order wih (compact)
✓ should not partial fill RFQ order when 0
✓ should not partial fill RFQ order when 0 (compact)
✓ should not fill RFQ order when expired
✓ should not fill RFQ order when expired (compact)
ETH fill
✓ should fill with ETH
✓ should receive ETH after fill (41ms)
✓ should reverted with takerAsset WETH and incorrect msg.value (54ms)
✓ should reverted with takerAsset non-WETH and msg.value greater than 0 (45ms)

Contract: SeriesNonceManager
✓ Get nonce - should return zero by default
✓ Advance nonce - should add to nonce specified amount
✓ Increase nonce - should add to nonce only 1
✓ Nonce equals - should return false when nonce does not match
✓ Nonce equals - should return true when nonce matches

Contract: SolidityTests
ArgumentsDecoderTest
testDecodeUint256 with offset
✓ should decode while data length and offset correct (38ms)
testDecodeSelector
✓ should decode
✓ should not decode with incorrect data length
testDecodeTailCalldata
✓ should decode
✓ should not decode with incorrect data length
testDecodeTargetAndCalldata

✓ should decode
✓ should not decode with incorrect data length
96 passing (11s)

FILE	% STMTS	% BRANCH	% FUNCS
OrderLib.sol	100	100	100
OrderMixin.sol	86.96	80	80
OrderRFQLib.sol	100	100	100
OrderRFQMixin.sol	90.16	58.7	90
AmountCalculator.sol	100	100	100
NonceManager.sol	100	100	100
PredicateHelper.sol	93.88	83.33	90.91
ArgumentsDecoder.sol	100	100	100
Callib.sol	100	100	100
Errors.sol	100	100	100
All files	**97.1**	**92.2**	**96**

Contract: assertRoughlyEqualValues
✓ should be work with expected = actual, any relativeDiff
✓ should be work with expected = actual, relativeDiff = 0
✓ should be work with diff between expected and actual less than relativeDiff * expected
✓ should be work with negative diff between expected and actual less than relativeDiff * expected
✓ should be work with diff between expected and actual equals to relativeDiff
✓ works on negative
✓ different signs throws
✓ should be thrown with expected != actual with relativeDiff = 0
✓ should be thrown with expected > actual more than relativeDiff * expected
✓ should be thrown with expected < actual more than relativeDiff * expected
Contract: AddressArray
length
✓ should be calculate length 0
✓ should be calculate length 1

at
- ✓ should be get from empty data
- ✓ should be get from data with 1 element
- ✓ should be get from data with several elements

get
- ✓ should be get empty data
- ✓ should be get from data with 1 element
- ✓ should be get from data with several elements

push
- ✓ should be push to empty data
- ✓ should be push to data with 1 element
- ✓ should be get push to data with several elements

pop
- ✓ should be thrown when data is empty
- ✓ should be pop in data with 1 element
- ✓ should be pop in data with several elements
- ✓ should be several pops
- ✓ should be thrown when pops more than elements

set
- ✓ should be thrown when set index less than data length
- ✓ should be set to index 0 to data with 1 element
- ✓ should be set to index 0 to data with several elements
- ✓ should be set to index non-0 to data with several elements

Contract: AddressSet

length
- ✓ should be calculate length 0
- ✓ should be calculate length 1

at
- ✓ should be get from empty data
- ✓ should be get from data with 1 element
- ✓ should be get from data with several elements

contains
- ✓ should be not contains in empty data
- ✓ should be contains address
- ✓ should be contains addresses

add
- ✓ should be add to empty data
- ✓ should not be add a double element without another elements in data
- ✓ should be add to data with 1 element
- ✓ should not be add a double element with another elements in data

remove
- ✓ should not be remove from empty data

✓ should be remove from data
✓ should not be remove element which is not in data
✓ should be remove from data and keep the remainder

Contract: ECDSA

recover
with invalid signature
✓ with short signature
✓ with long signature
with valid signature
using web3.eth.sign
✓ returns signer address with correct signature
✓ returns signer address with correct signature for arbitrary length message
✓ returns a different address
✓ returns zero address with invalid signature
with v0 signature
✓ returns zero address with 00 as version value
✓ works with 27 as version value
✓ returns zero address when wrong version
✓ works with short EIP2098 format
with v1 signature
✓ returns zero address with 01 as version value
✓ works with 28 as version value
✓ returns zero address when wrong version
✓ works with short EIP2098 format
isValidSignature
with invalid signature
✓ with short signature
✓ with long signature
with valid signature
using web3.eth.sign
✓ returns true with correct signature and only correct signer
✓ returns true with correct signature and only correct signer for arbitrary length message
✓ returns false with invalid signature
with v0 signature
✓ returns false with 00 as version value
✓ returns true with 27 as version value, and only for signer
✓ returns false when wrong version
✓ returns true with short EIP2098 format, and only for signer
with v1 signature
✓ returns false with 01 as version value
✓ returns true with 28 as version value, and only for signer (127ms)
✓ returns false when wrong version (61ms)

✓ returns true with short EIP2098 format, and only for signer
isValidSignature65
✓ with matching signer and signature
✓ with invalid signer
✓ with invalid signature
recoverOrIsValidSignature
with invalid signature
✓ with short signature
✓ with long signature
with valid signature
using web3.eth.sign
✓ returns true with correct signature and only correct signer
✓ returns true with correct signature and only correct signer for arbitrary length message
✓ returns false with invalid signature
with v0 signature
✓ returns false with 00 as version value
✓ returns true with 27 as version value, and only for signer (44ms)
✓ returns false when wrong version
✓ returns true with short EIP2098 format, and only for signer
with v1 signature
✓ returns false with 01 as version value
✓ returns true with 28 as version value, and only for signer (39ms)
✓ returns false when wrong version
✓ returns true with short EIP2098 format, and only for signer
recoverOrIsValidSignature65
✓ with matching signer and signature
✓ with invalid signer
✓ with invalid signature
toEthSignedMessageHash
✓ correct hash
toTypedDataHash
✓ correct hash
gas price
recover
✓ with signature
✓ with v0 signature
✓ with v1 signature
recoverOrIsValidSignature
✓ with signature
✓ with v0 signature (39ms)
✓ with v1 signature (39ms)
✓ recoverOrIsValidSignature65

isValidSignature
✓ with signature
✓ with v0 signature
✓ with v1 signature
✓ isValidSignature65
Additional methods
✓ toEthSignedMessageHash
✓ toTypedDataHash
Contract: RevertReasonParser
parse
✓ should be parsed as Unknown (Invalid revert reason)
✓ should be parsed as empty Error
✓ should be parsed as Error
✓ should be parsed as Unknown
✓ should be parsed as Panic
✓ should be parsed as Error with long string
✓ should be reverted in _test()
Contract: SafeERC20
with address that has no contract code
✓ reverts on transfer
✓ reverts on transferFrom
✓ reverts on approve
✓ reverts on increaseAllowance
✓ reverts on decreaseAllowance
with token that returns false on all calls
✓ reverts on transfer
✓ reverts on transferFrom
✓ reverts on approve
✓ reverts on increaseAllowance
✓ reverts on decreaseAllowance
with token that returns true on all calls
✓ doesn't revert on transfer
✓ doesn't revert on transferFrom
approvals
with zero allowance
✓ doesn't revert when approving a non-zero allowance
✓ doesn't revert when approving a zero allowance
✓ doesn't revert when increasing the allowance
✓ reverts when decreasing the allowance
with non-zero allowance
✓ doesn't revert when approving a non-zero allowance
✓ doesn't revert when approving a zero allowance

```
    ✓ doesn't revert when increasing the allowance
    ✓ doesn't revert when decreasing the allowance to a positive value
    ✓ reverts when decreasing the allowance to a negative value
  with token that returns no boolean values
    ✓ doesn't revert on transfer
    ✓ doesn't revert on transferFrom
  approvals
  with zero allowance
    ✓ doesn't revert when approving a non-zero allowance
    ✓ doesn't revert when approving a zero allowance
    ✓ doesn't revert when increasing the allowance
    ✓ reverts when decreasing the allowance
  with non-zero allowance
    ✓ doesn't revert when approving a non-zero allowance
    ✓ doesn't revert when approving a zero allowance
    ✓ doesn't revert when increasing the allowance
    ✓ doesn't revert when decreasing the allowance to a positive value
    ✓ reverts when decreasing the allowance to a negative value
  with token that doesn't revert on invalid permit
    ✓ accepts owner signature
    ✓ revert on reused signature (43ms)
    ✓ revert on invalid signature
StringUtil
  Validity
    ✓ Uint 256
    ✓ Uint 128
    ✓ Very long byte array
    ✓ Extremely long byte array (362ms)
    ✓ Empty bytes. Skipped until resolved: https://github.com/ChainSafe/web3.js/issues/4512
    ✓ Single byte
Methods
    ✓ should be trimmed
    ✓ should not be changed
    ✓ should correctly build data for permit
    ✓ should correctly build data for dai-like permit
    ✓ should concat target with prefixed data
    ✓ should concat target with raw data
Contract: Permitable
    ✓ should be permitted for IERC20Permit
    ✓ should not be permitted for IERC20Permit
    ✓ should be permitted for IDaiLikePermit
    ✓ should not be permitted for IDaiLikePermit
```

```
    ✓ should be wrong permit length
Contract:
  profileEVM
    ✓ should be counted ERC20 Transfer
    ✓ should be counted ERC20 Approve
  gasspectEVM
    ✓ should be counted ERC20 Transfer
    ✓ should be counted ERC20 Approve
    ✓ should be counted ERC20 Transfer with minOpGasCost = 2000
    ✓ should be counted ERC20 Transfer with args
    ✓ should be counted ERC20 Transfer with res
timeIncreaseTo
    ✓ should be increased on 1000 sec (424ms)
    ✓ should be increased on 2000 sec (1000ms)
    ✓ should be increased on 1000000 sec (995ms)
    ✓ should be thrown with increase time to a moment in the past (995ms)
fixSignature
    ✓ should not be fixed geth sign
    ✓ should be fixed ganache sign
Contract:
  signMessage
    ✓ should be signed test1
    ✓ should be signed test2
    ✓ should be signed test3
  trackReceivedTokenAndTx
    ✓ should be tracked ERC20 Transfer
    ✓ should be tracked ERC20 Approve
  trackReceivedToken
    ✓ should be tracked ERC20 Transfer
    ✓ should be tracked ERC20 Approve
  countInstructions
    ✓ should be counted ERC20 Transfer
    ✓ should be counted ERC20 Approve
  182 passing (7s)
  6 pending
```

FILE	% STMTS	% BRANCH	% FUNCS
EthReceiver.sol	100	0	100
ECDSA.sol	100	100	100
RevertReasonForwarder.sol	100	100	100
SafeERC20.sol	100	88.89	100
StringUtil.sol	50	100	66.67
UniERC20.sol	0	0	0
All files	**50**	**40**	**72.73**

Zokyo Secured team has carefully checked the whole set of unit tests and checked the original coverage of those tests. The audit tests were verified for the correctness, wholesomeness, sufficient coverage, meaning of the scenarios, overall structure, and the correct implementation (despite the covered functionality).

As a result - all tests are verified, and the coverage is evaluated as conforming to security requirements. Original functionality has necessary coverage, Zokyo Secured auditors confirmed standard and sub-standard functionality in a separate set of exploratory testing scenarios.

We are grateful to have been given the opportunity to work with the 1inch team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the 1inch team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.