1INCH

SMART CONTRACT AUDIT

zokyo

June, 27th 2022 | v. 1.0

Security Audit Score

PASS

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

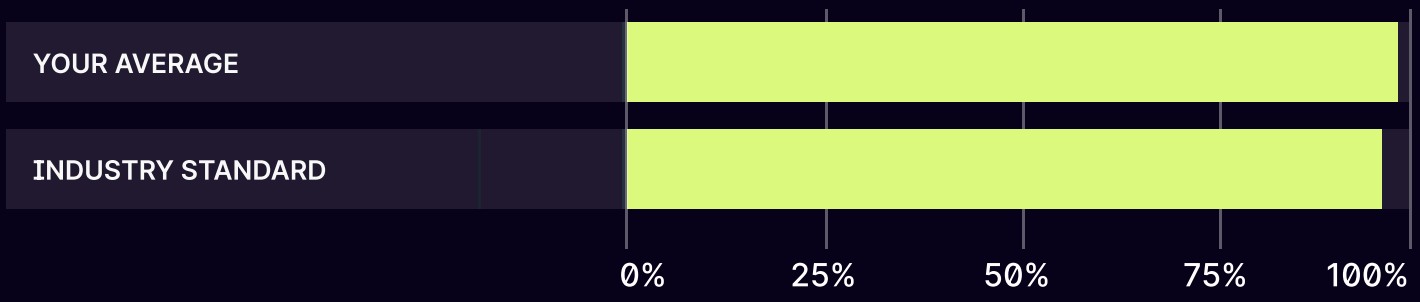
TECHNICAL SUMMARY

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 98%, 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

Last audited commit: 274d414497f57d42db928b5380d88ca66b239811, master branch

Within the scope of this audit Zokyo auditors have reviewed the following contract(s):

- AggregationRouterV5.sol
- GenericRouter.sol
- LimitOrderProtocolRFQ.sol
- UnoswapRouterBase.sol
- UnoswapRouter.sol
- UnoswapV3Router.sol
- helpers/EIP712Instance.sol
- helpers/Errors.sol
- helpers/NonceManager.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:

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

Executive Summary

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

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, Security team has compared gas spending between Uniswap V2/V3 Routers and 1Inch Aggregation V5 Router. In 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 current Gas and ETH price, savings with using 1Inch Router are 1-1.10$ in average.

PROTOCOL OVERVIEW

AggregationRouterV5 is contract for performing swaps via different protocols. Contract is able to perform next types of swaps :

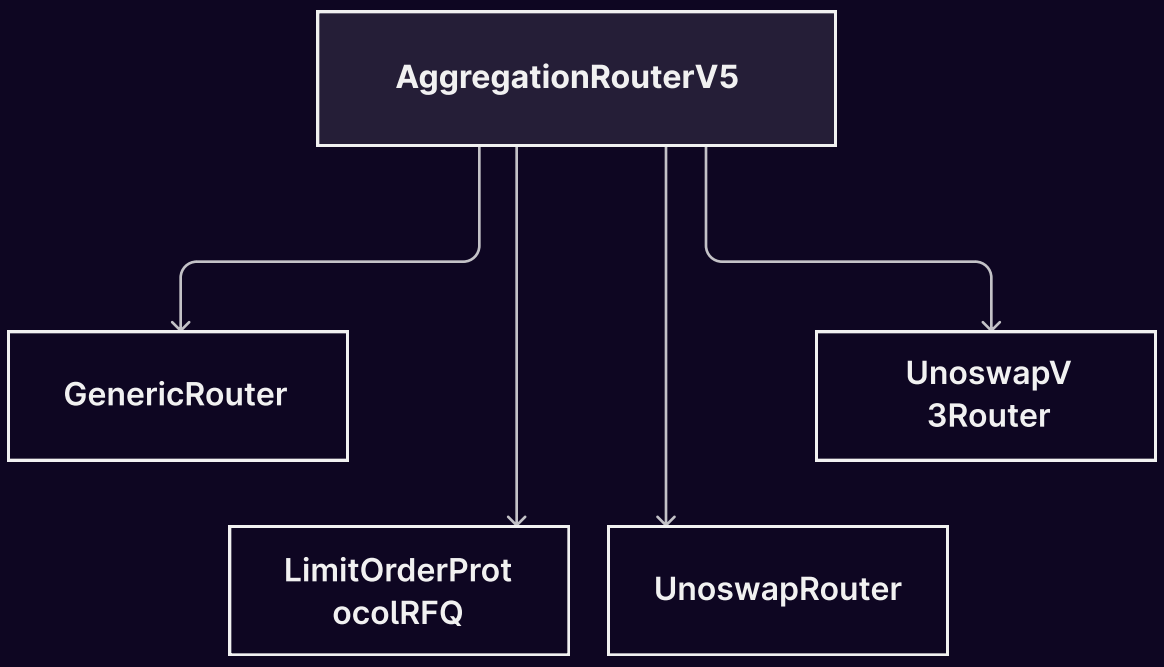
1) through Aggregator executor, which would perform any calls to any protocol.

2) Limit order swap, which would fulfill an order between two users.

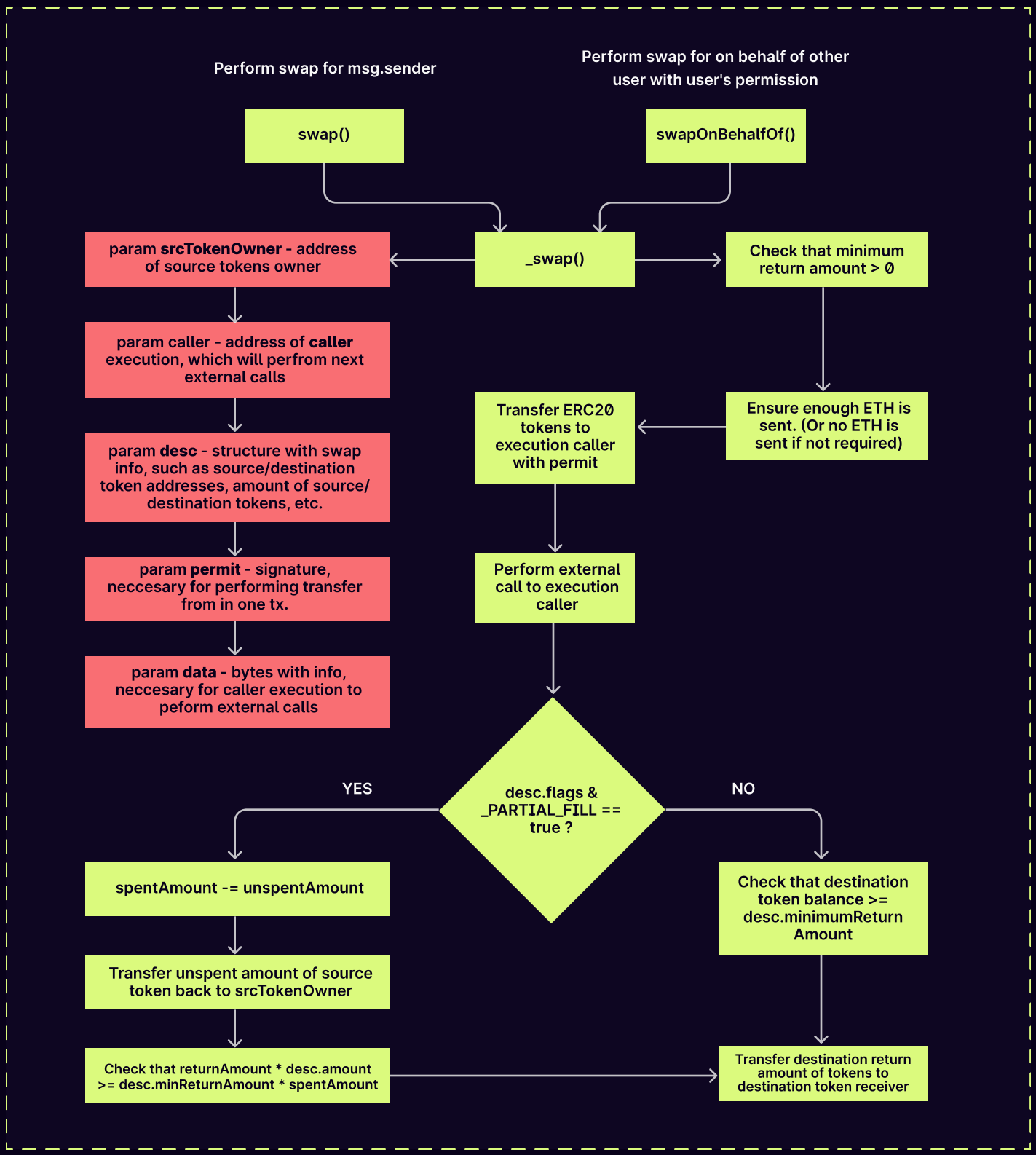
3) through Uniswap V2 protocol.

4) through Uniswap V3 protocol.

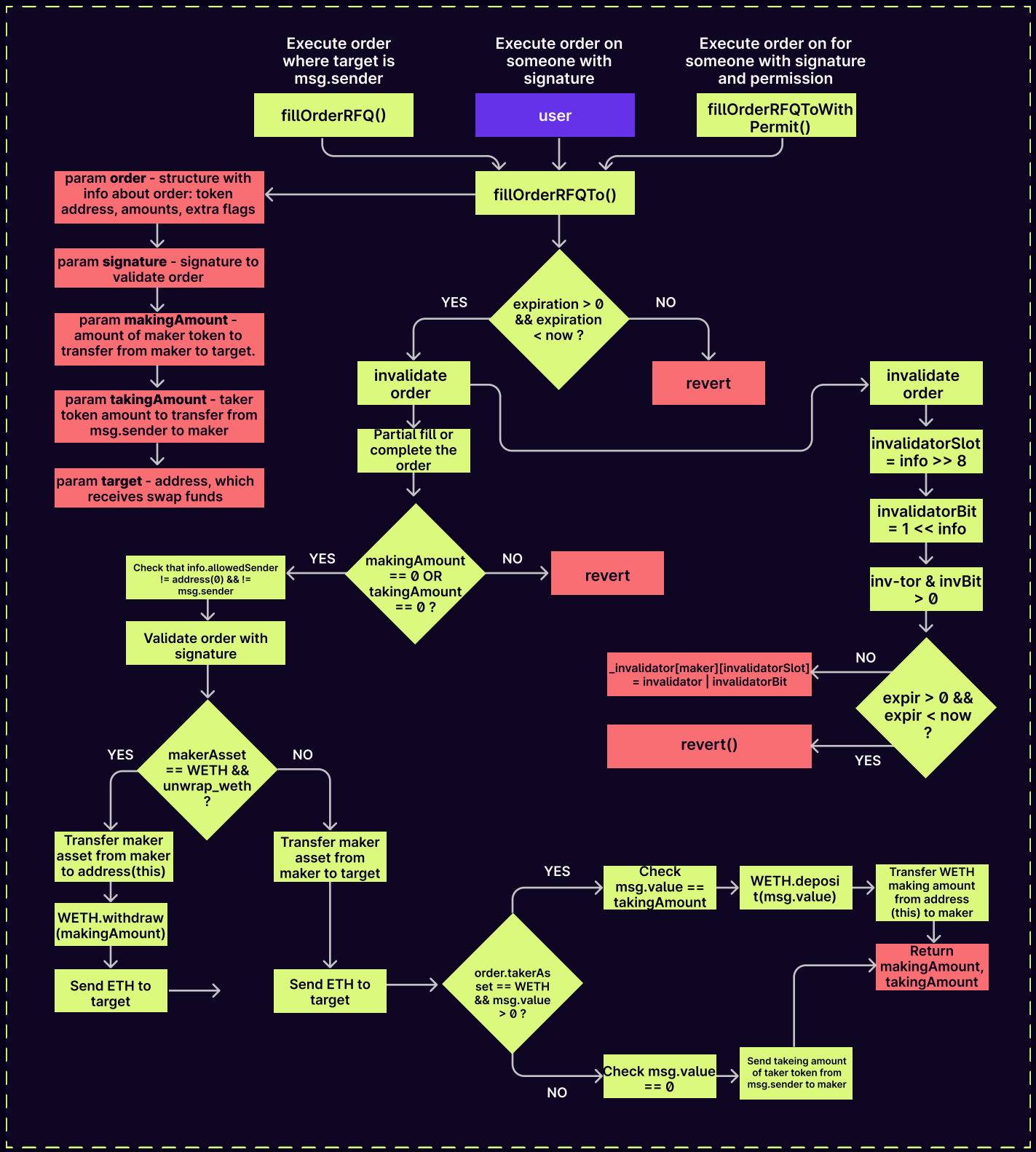
Contract is gas-optimized, uses assemble to decrease gas spendings. Protocol is capable to swap token on behalf of other users with signatures, swap token with permission instead of approval, swap tokens with approval in advance.



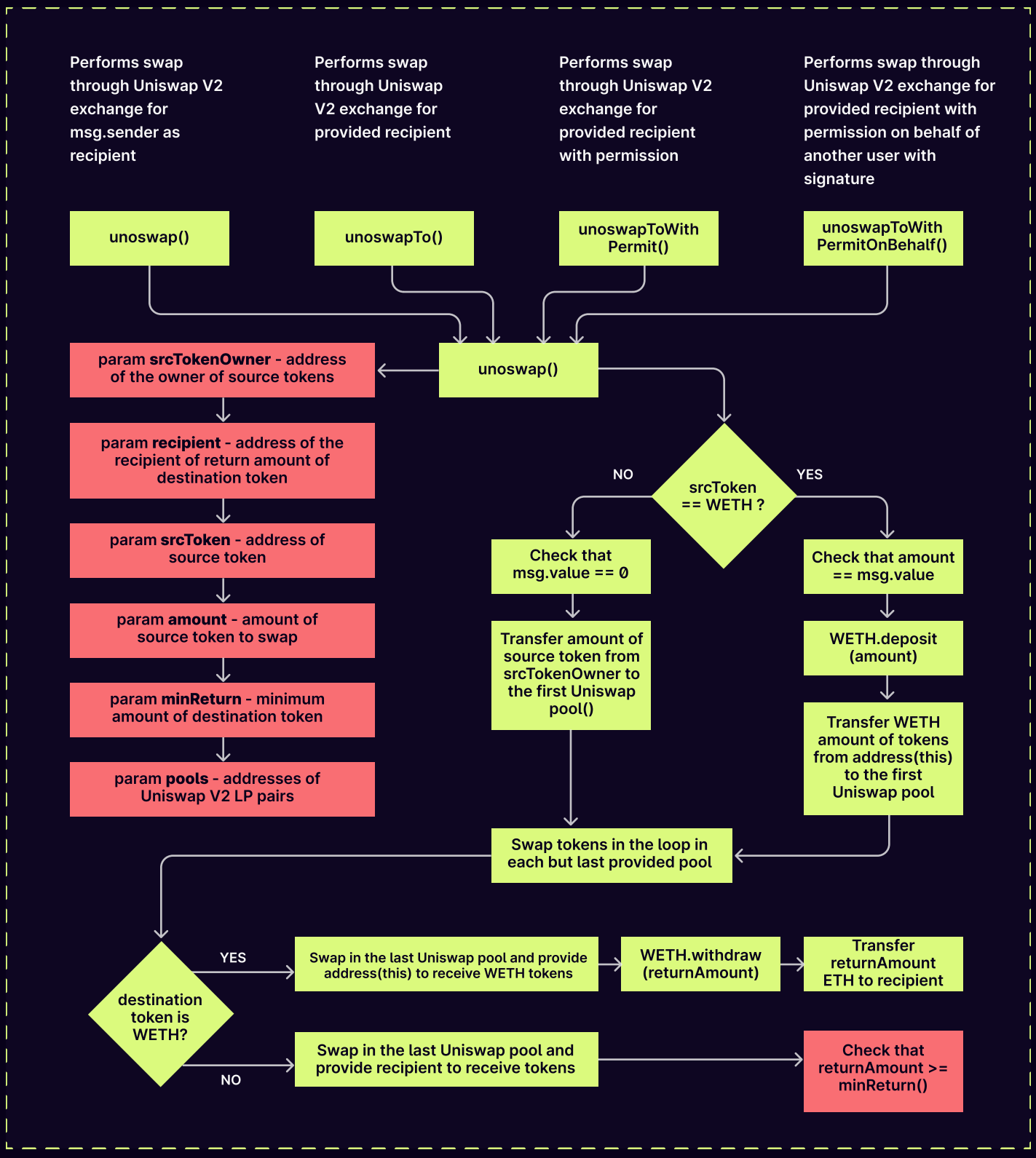
1. GENERICROUTER.SOL



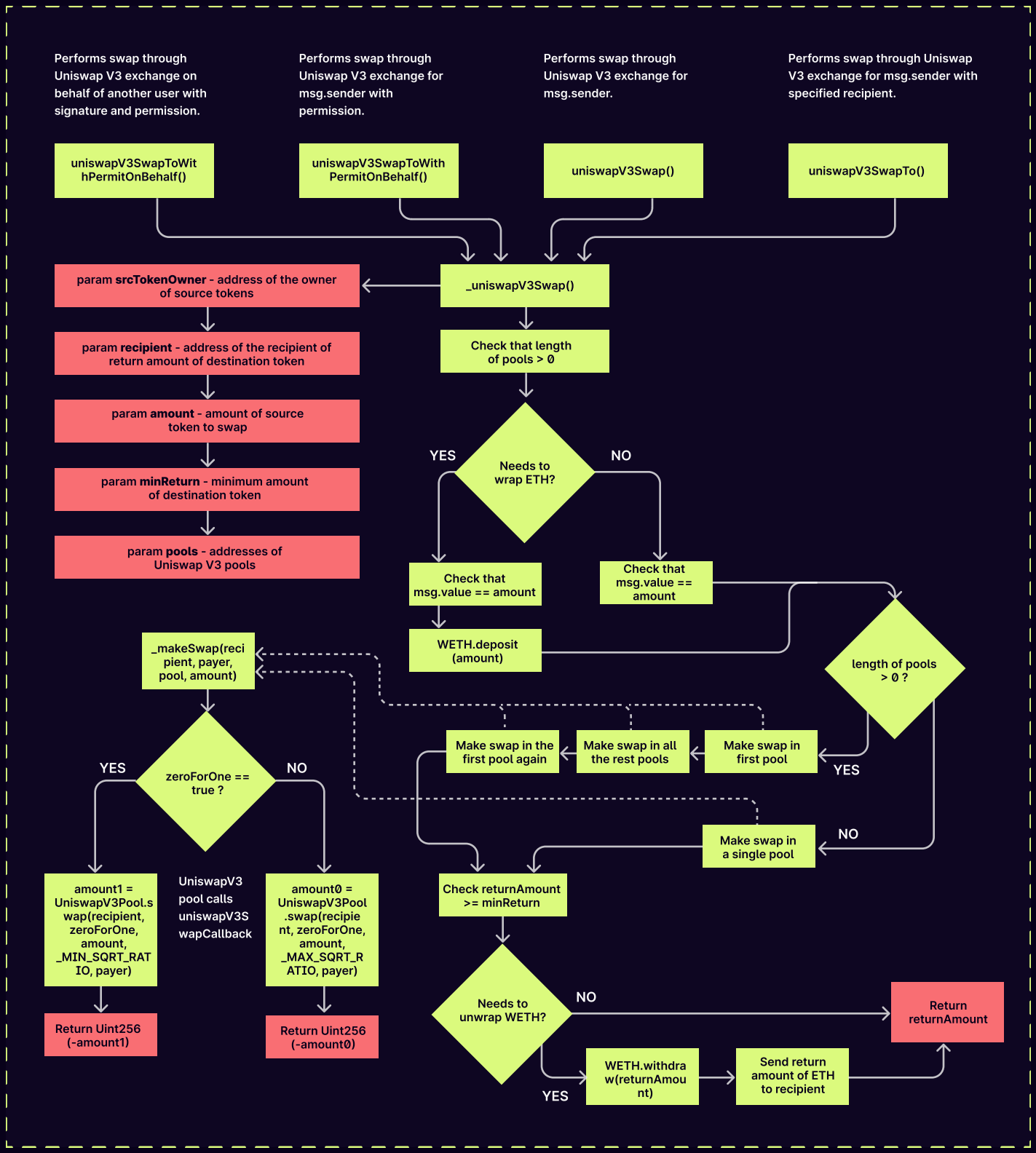
2. LIMITORDERPROTOCOLRFQ.SOL



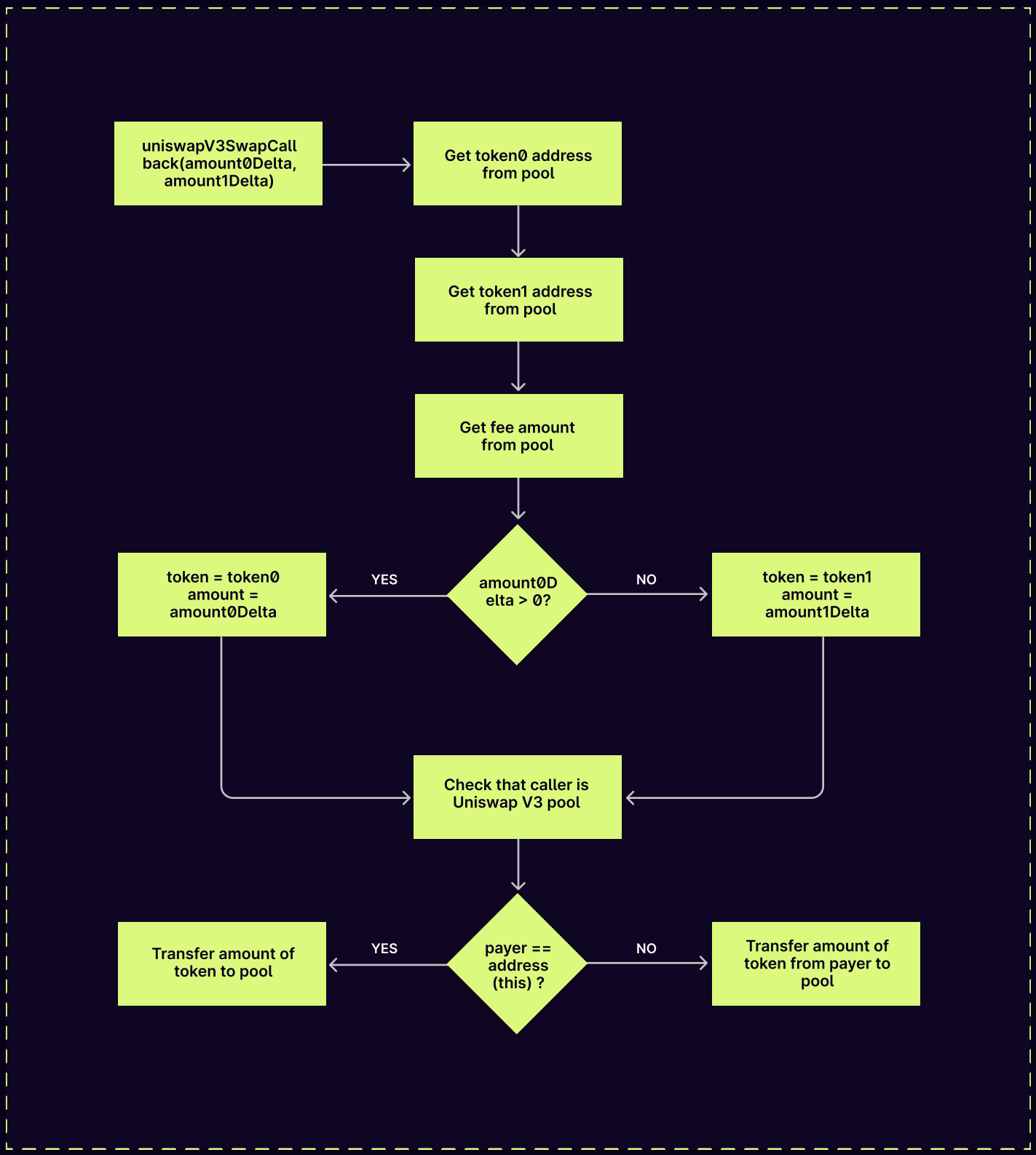
3. UNOSWAPROUTER.SOL



4. UNOSWAPV3ROUTER.SOL



4. UNOSWAPV3ROUTER.SOL (CALLBACK)



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.

MEDIUM | UNRESOLVED

Performance of swap can be blocked.

GenericRouter.sol: function _swap(), line 165.
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:
Correct ETH sending functionality.

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

	AggregationRouterV5.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

	LimitOrderProtocolRFQ.sol	UnoswapRouterBase.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

	UnoswapRouter.sol	UnoswapV3Router.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 are 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.

AggregationRouterV5
Protection
✓ protection should pass (2174ms)
✓ protection should fail (165ms)
Main
eth → dai
✓ simple mooniswap eth → dai (2139ms)
✓ mooniswap eth → dai with patched balance (127ms)
✓ simple uniswapv2 eth → dai (303ms)
✓ direct uniswapv2 eth → dai (226ms)
✓ simple uniswapv1 eth → dai (505ms)
✓ uniswapv1 eth → dai with patched balance (102ms)
✓ uniswapv1 + mooniswap eth → dai (135ms)
dai → eth
✓ simple mooniswap dai → eth (406ms)
✓ simple uniswapv1 dai → eth (239ms)
✓ simple kyber dmm dai → eth (1924ms)
✓ simple uniswapv2 dai → eth (119ms)
✓ direct uniswapv2 dai → weth (381ms)
Main: swapOnBehalf
✓ should revert on attempt to swap native token (ETH)
swap: dai → eth
✓ simple swapOnBehalf addr1 by addr2 using uniswapv2 dai → eth (1886ms)
✓ shouldn't allow to swap when singed not by owner (66ms)
✓ shouldn't allow to swap with expired deadline
✓ shouldn't allow to replay swap (185ms)
✓ shouldn't allow execute canceled swap (54ms)
✓ should allow to executed second swap once increased nonce signed (266ms)
Dodo

usdc → weth
- ✓ patch by balance (5349ms)
- ✓ simple patch (251ms)

NonceManager
- ✓ Get nonce - should return zero by default
- ✓ Advance nonce - should add to nonce specified amount

Leftovers

eth → dai
- ✓ no referral (599ms)
- ✓ simple referral (462ms)
- ✓ bad rate (148ms)
- ✓ priority referral (180ms)

dai → eth
- ✓ simple referral (383ms)

LimitOrderProtocolRFQ

ETH

maker weth
- ✓ should unwrap weth (171ms)
- ✓ should not unwrap weth (47ms)
- ✓ should revert if pass eth on token swap (65ms)

taker weth
- ✓ should accept weth (167ms)
- ✓ should wrap eth (216ms)
- ✓ should revert if not enough eth (44ms)
- ✓ should revert if too much eth (48ms)

Permit
- ✓ COIN ⇒ TOCK (70ms)
- ✓ rejects reused signature (86ms)
- ✓ rejects other signature (45ms)
- ✓ rejects expired permit

Main
- ✓ should swap fully based on RFQ signature (1630ms)

OrderRFQ Cancelation
- ✓ should cancel own order
- ✓ should cancel own order with huge number
- ✓ should not fill cancelled order (40ms)

Expiration
- ✓ should fill RFQ order when not expired (68ms)
- ✓ should partial fill RFQ order (59ms)
- ✓ should fully fill RFQ order (69ms)
- ✓ should not partial fill RFQ order when 0
- ✓ should not fill RFQ order when expired

Unoswap

Permit

- ✓ USDC ⇒ DAI (1943ms)
- ✓ rejects reused signature (82ms)
- ✓ rejects other signature (38ms)
- ✓ rejects expired permit

Main

ETH ⇒ DAI

- ✓ uniswap router eth ⇒ dai (207ms)
- ✓ 0x router ETH ⇒ DAI (477ms)
- ✓ 1inch unirouter ETH ⇒ DAI (293ms)
- ✓ 1inch exchange ETH ⇒ DAI (895ms)

ETH ⇒ USDC ⇒ DAI

- ✓ uniswap router ETH ⇒ USDC ⇒ DAI (1698ms)
- ✓ 0x router ETH ⇒ USDC ⇒ DAI (368ms)
- ✓ 1inch unirouter ETH ⇒ USDC ⇒ DAI (359ms)
- ✓ 1inch exchange ETH ⇒ USDC ⇒ DAI (1272ms)

DAI ⇒ ETH

- ✓ Uniswap router DAI ⇒ ETH (317ms)
- ✓ 0x router DAI ⇒ ETH (184ms)
- ✓ 1inch unirouter DAI ⇒ ETH (198ms)
- ✓ 1inch exchange DAI ⇒ ETH (820ms)

DAI ⇒ WETH

- ✓ Uniswap router DAI ⇒ WETH (223ms)
- ✓ 0x router DAI ⇒ WETH (166ms)
- ✓ 1inch unirouter DAI ⇒ WETH (181ms)
- ✓ 1inch exchange DAI ⇒ WETH (517ms)

DAI ⇒ WETH ⇒ USDC

- ✓ Uniswap router DAI ⇒ WETH ⇒ USDC (551ms)
- ✓ 0x router DAI ⇒ WETH ⇒ USDC (412ms)
- ✓ 1inch unirouter DAI ⇒ WETH ⇒ USDC (283ms)
- ✓ 1inch exchange DAI ⇒ WETH ⇒ USDC (8773ms)

DAI ⇒ WETH ⇒ USDC ⇒ USDT

- ✓ Uniswap router DAI ⇒ WETH ⇒ USDC ⇒ USDT (2649ms)
- ✓ 0x router DAI ⇒ WETH ⇒ USDC ⇒ USDT (418ms)
- ✓ 1inch unirouter DAI ⇒ WETH ⇒ USDC ⇒ USDT (361ms)
- ✓ 1inch exchange DAI ⇒ WETH ⇒ USDC ⇒ USDT (1411ms)

Main: unoswapToWithPermitOnBehalf

- ✓ Doesn't allow to swap ETH OnBehalf

DAI ⇒ ETH addr1 OnBehalf of addr2

- ✓ Swap from addr1 OnBehalf of addr2 with valid signature (227ms)
- ✓ Pays fee to feeReceiver in source token (210ms)

- ✓ Perform swat with permit USDC ⇒ DAI (383ms)
- ✓ Shouldn't allow to swap with expired deadline
- ✓ Shouldn't allow to swap with nonce that already used (81ms)
- ✓ Shouldn't allow to execute canceled swap (45ms)
- ✓ Shouldn't allow to swap with someone's else signature (38ms)

UnoswapV3

Permit

- ✓ USDC ⇒ DAI (1700ms)
- ✓ rejects reused signature (93ms)
- ✓ rejects other signature
- ✓ rejects expired permit

Main

MinReturn

- ✓ WETH ⇒ DAI (1428ms)

Tokens

- ✓ WETH ⇒ DAI (397ms)
- ✓ WETH ⇒ DAI ⇒ USDC (1060ms)
- ✓ WETH ⇒ USDC ⇒ DAI (1368ms)
- ✓ WETH ⇒ DAI ⇒ WETH (691ms)
- ✓ WETH ⇒ DAI ⇒ WETH ⇒ USDC (967ms)
- ✓ WETH ⇒ DAI ⇒ WETH ⇒ USDC ⇒ USDT (2760ms)

Tokens OnBehalf

- ✓ WETH ⇒ DAI from addr1 by addr2 (66ms)
- ✓ Allow to execute with permit (277ms)
- ✓ Pays fee to feeReceiver in source token (339ms)
- ✓ Doesn't allow to swap with expired deadline
- ✓ Shouldn't allow to swap with nonce that already used (95ms)
- ✓ Shouldn't allow to execute canceled swap
- ✓ Shouldn't allow to swap with someone's else signature

Native

- ✓ ETH ⇒ DAI (257ms)
- ✓ DAI ⇒ ETH (327ms)
- ✓ ETH ⇒ USDC ⇒ DAI (511ms)
- ✓ DAI ⇒ USDC ⇒ ETH (467ms)

108 passing (1m)

FILE	% STMTS	% BRANCH	% FUNCS
AggregationRouterV5.sol	0	100	33
GenericRouter.sol	82.35	60.71	100
LimitOrderProtocolRFQ.sol	97.92	85.71	87.5
UnoswapRouterBase.sol	100	100	100
UnoswapRouter.sol	90	75	80
UnoswapV3Router.sol	100	100	100
helpers/EIP712Instance.sol	100	100	100
helpers/Errors.sol	100	100	100
helpers/NonceManager.sol	100	100	100
All files	**84.29**	**96.43**	**89.13**

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

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

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests are 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.

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)

FILE	% STMTS	% BRANCH	% FUNCS
AggregationRouterV5.sol	100	100	100
GenericRouter.sol	94.12	85.71	100
LimitOrderProtocolRFQ.sol	97.92	96.43	87.5
UnoswapRouterBase.sol	100	100	100
UnoswapRouter.sol	90	100	80
UnoswapV3Router.sol	100	100	100
helpers/EIP712Instance.sol	100	100	100
helpers/Errors.sol	100	100	100
helpers/NonceManager.sol	100	100	100
All files	**98.57**	**100**	**97.5**

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