# 1inch

# Limit Order Protocol v2

## SMART CONTRACT AUDIT

**16.10.2021**

**Made in Germany by Chainsulting.de**

# Table of contents

# 1. Disclaimer

The audit makes no statements or warrantees 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 Exchange. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

| Major Versions / Date | Description |
|---|---|
| 0.1 (12.10.2021) | Layout |
| 0.5 (13.10.2021) | Manual & Automated Security Testing |
| 0.6 (14.10.2021) | Testing SWC Checks |
| 0.7 (15.10.2021) | Verify Claims |
| 0.9 (16.10.2021) | Summary and Recommendation |
| 1.0 (16.10.2021) | Final document |
| 1.1 (TBA) | Added deployed contract addresses |

## 2. About the Project and Company

**Company address:**

1Inch Limited
Quijano Chambers, P.O. Box 3159, Road Town
Tortola, British Virgin Islands

Sergej Kunz Co-Founder & Chief Executive Officer
Anton Bukov Co-Founder & Chief Technology Officer

**Discord:** **https://discord.gg/FZADkCZ**

**Blog:** **https://blog.1inch.io**

**Medium:** **https://medium.com/@1inch.exchange**

**Website:** **https://app.1inch.io**

**Twitter:** **https://twitter.com/1inchExchange**

**Reddit:** **https://www.reddit.com/r/1inch_exchange**

**Telegram:** **https://t.me/OneInchExchange**

**Forum:** **https://gov.1inch.io**

## 2.1 Project Overview

The 1inch Network unites decentralized protocols whose synergy enables the most lucrative, fastest and protected operations in the DeFi space. The initial protocol of the 1inch Network is a DEX aggregator solution that searches deals across multiple liquidity sources, offering users better rates than any individual exchange.

This protocol incorporates the Pathfinder algorithm which finds the best paths among different markets over 50+ liquidity sources on Ethereum, 20+ liquidity sources on Binance Smart Chain and 8+ liquidity sources on Polygon. In just two years the 1inch DEX aggregator surpassed $50B in overall volume on the Ethereum network alone. The 1inch Aggregation Protocol facilitates cost-efficient and secure swap transactions across multiple liquidity sources.

The 1inch Liquidity Protocol is a next-generation automated market maker that protects users from front-running attacks and offers attractive opportunities to liquidity providers. The 1inch Limit Order Protocol facilitates the most innovative and flexible limit order swap opportunities in DeFi. The protocol’s features, such as dynamic pricing, conditional orders and extra RFQ support, power various implementations, including stop-loss and trailing stop orders, as well as auctions.

1inch limit order protocol is a set of smart contracts, that can work on any EVM based blockchains (Ethereum, Binance Smart Chain, Polygon, etc.). Key features of the protocol is extreme flexibility and high gas efficiency that achieved by using two different order types - regular Limit Order and RFQ Order. Smart Contract allows users to place limit orders and RFQ Orders, that later could be filled on-chain. Both type of orders is a data structure created off-chain and signed according to EIP-712.

## 3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

| Level | Value | Vulnerability | Risk (Required Action) |
|---|---|---|---|
| Critical | 9 – 10 | A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken. | Immediate action to reduce risk level. |
| High | 7 – 8.9 | A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way. | Implementation of corrective actions as soon as possible. |
| Medium | 4 – 6.9 | A vulnerability that could affect the desired outcome of executing the contract in a specific scenario. | Implementation of corrective actions in a certain period. |
| Low | 2 – 3.9 | A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective. | Implementation of certain corrective actions or accepting the risk. |
| Informational | 0 – 1.9 | A vulnerability that have informational character but is not effecting any of the code. | An observation that does not determine a level of risk |

# 4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

## 4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

## 4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

| Dependency / Import Path | Source |
|---|---|
| @openzeppelin/contracts/token/ERC1155/IERC1155.sol | https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.3.2/contracts/token/ERC1155/IERC1155.sol |
| @openzeppelin/contracts/token/ERC20/IERC20.sol | https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.3.2/contracts/token/ERC20/IERC20.sol |
| @openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol | https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.3.2/contracts/token/ERC20/extensions/draft-IERC20Permit.sol |
| @openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol | https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.3.2/contracts/token/ERC20/utils/SafeERC20.sol |
| @openzeppelin/contracts/token/ERC721/IERC721.sol | https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.3.2/contracts/token/ERC721/IERC721.sol |
| @openzeppelin/contracts/utils/Address.sol | https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.3.2/contracts/utils/Address.sol |
| @openzeppelin/contracts/utils/cryptography/SignatureChecker.sol | https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.3.2/contracts/utils/cryptography/SignatureChecker.sol |
| @openzeppelin/contracts/utils/cryptography/draft-EIP712.sol | https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.3.2/contracts/utils/cryptography/draft-EIP712.sol |

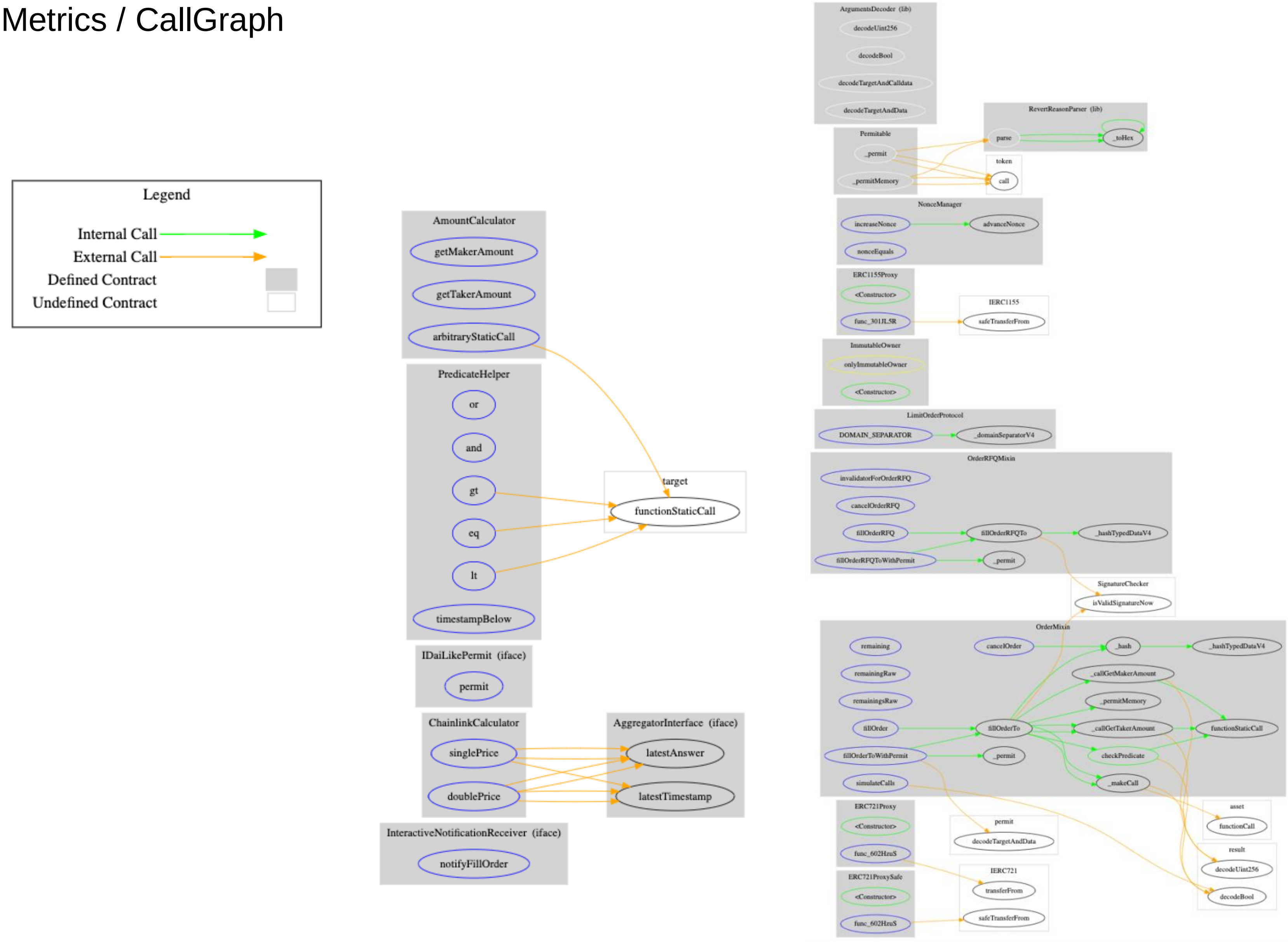
## 4.3 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

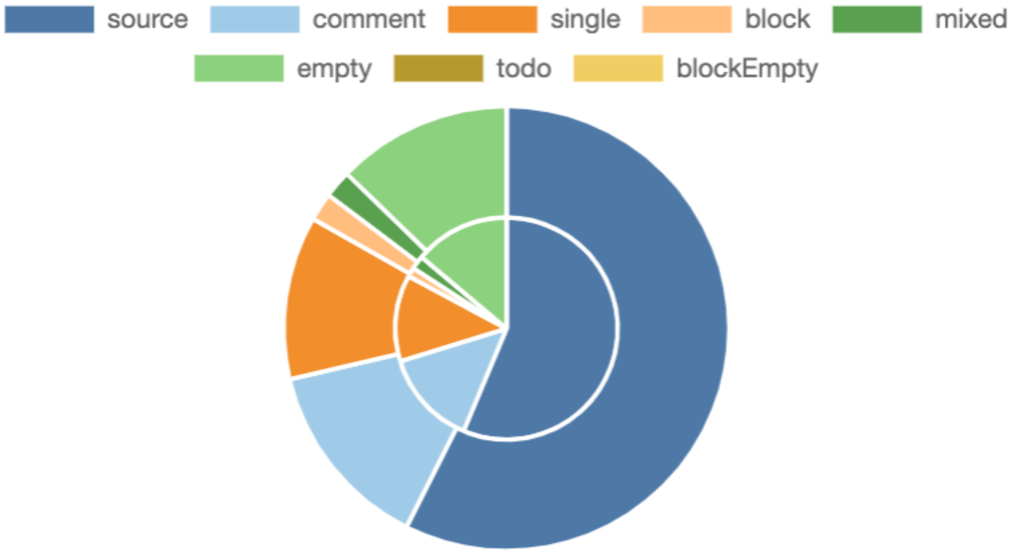
| File | Fingerprint (MD5) |
|---|---|
| LimitOrderProtocol.sol | 773875475cf314c7a56bdc546cac02d0 |
| OrderMixin.sol | 8a251bf265c4b3042d0fad3b11beb922 |
| OrderRFQMixin.sol | a6a8b2c7d3afe4c2c64759f53c86c551 |
| ./helpers/AmountCalculator.sol | 35c1214179db933fa1719d3cc33bd742 |
| ./helpers/ERC721Proxy.sol | f23e26a88f6acdf4a948c68384a20b15 |
| ./helpers/PredicateHelper.sol | 75b8c3c5ce5ec6e37ae0101113e5fad2 |
| ./helpers/ChainlinkCalculator.sol | 6ab30cc0cc4eeecc011cf1ab923240c9 |
| ./helpers/ImmutableOwner.sol | 5eccf3977e9c2018b94a085f5e8fcfcc |
| ./helpers/ERC721ProxySafe.sol | fabc4a8db094dd528d8832148dd3d12c |
| ./helpers/ERC1155Proxy.sol | fe973bdba9b251d9e366c833f5de14d1 |
| ./helpers/NonceManager.sol | 3428130d5b1e370ef7bd309d19b11499 |
| ./libraries/Permitable.sol | dcc7f03730b22d7dd55762bcafb9ba5c |
| ./libraries/ArgumentsDecoder.sol | aa87cdf8aea0a80d278ff0ab67dafc39 |
| ./libraries/RevertReasonParser.sol | 1b7f06f88c57f514c9a851ce5471ce9a |
| ./interfaces/IDaiLikePermit.sol | de64a23241710e682a5851e47a23fc4d |
| ./interfaces/AggregatorInterface.sol | f6187143d64146f2af9836ea2002deb7 |
| ./interfaces/InteractiveNotificationReceiver.sol | ff9a940e4220e4a76042f331980716e4 |

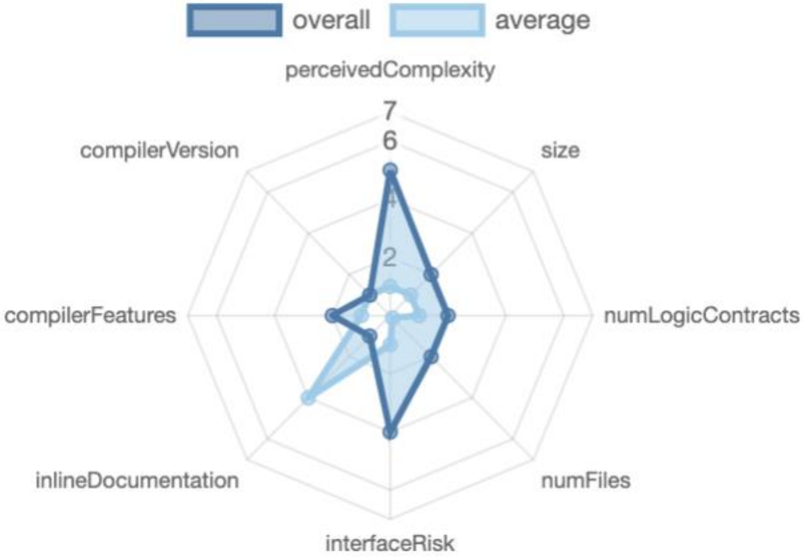
Commit: https://github.com/1inch/limit-order-protocol/commit/9d118307df7acc3bcef73407f3964acd6aa0f35c

## 4.4 Metrics / CallGraph



## 4.5 Metrics / Source Lines & Risk





## 4.6 Metrics / Capabilities

| Solidity Versions observed | ⍰ Experimental Features | ⍰ Can Receive Funds | ⍰ Uses Assembly | ⍰ Has Destroyable Contracts |
|---|---|---|---|---|
| `^0.8.0` | | | `yes`<br>(8 asm blocks) | |

| ⍰ Transfers ETH | ⍰ Low-Level Calls | ⍰ DelegateCall | ⍰ Uses Hash Functions | ⍰ ECRecover | ⍰ New/Create/Create2 |
|---|---|---|---|---|---|
| | | | `yes` | | |

*Exposed Functions*

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

| ⍰Public | ⍰Payable |
|---|---|
| 36 | 0 |

| External | Internal | Private | Pure | View |
|---|---|---|---|---|
| 32 | 30 | 6 | 9 | 21 |

*StateVariables*

| Total | ⍰Public |
|---|---|
| 9 | 4 |

## 4.7 Metrics / Source Unites in Scope

| Type | File | Logic Contracts | Interfaces | Lines | nLines | nSLOC | Comment Lines | Complex. Score | Capabilities |
|---|---|---|---|---|---|---|---|---|---|
| ⍰ | contracts/interfaces/InteractiveNotificationReceiver.sol | | 1 | 15 | 7 | 3 | 1 | 3 | |
| ⍰ | contracts/interfaces/AggregatorInterface.sol | | 1 | 8 | 6 | 3 | 1 | 5 | ⍰ |
| ⍰ | contracts/interfaces/IDaiLikePermit.sol | | 1 | 8 | 7 | 3 | 1 | 3 | |
| ⍰ | contracts/helpers/AmountCalculator.sol | 1 | | 29 | 26 | 12 | 8 | 9 | |
| ⍰ | contracts/helpers/ERC721Proxy.sol | 1 | | 24 | 24 | 12 | 4 | 11 | |
| ⍰ | contracts/helpers/PredicateHelper.sol | 1 | | 72 | 45 | 29 | 18 | 42 | |
| ⍰ | contracts/helpers/ChainlinkCalculator.sol | 1 | | 42 | 42 | 22 | 14 | 23 | |
| ⍰ | contracts/OrderRFQMixin.sol | 1 | | 132 | 114 | 79 | 23 | 47 | ⍰ |
| ⍰ | contracts/OrderMixin.sol | 1 | | 311 | 269 | 213 | 38 | 165 | ⍰⍰ |
| ⍰ | contracts/LimitOrderProtocol.sol | 1 | | 19 | 19 | 13 | 3 | 10 | |
| ⍰ | contracts/helpers/ImmutableOwner.sol | 1 | | 17 | 17 | 11 | 2 | 3 | |
| ⍰ | contracts/helpers/ERC721ProxySafe.sol | 1 | | 24 | 24 | 12 | 4 | 11 | |
| ⍰ | contracts/helpers/ERC1155Proxy.sol | 1 | | 24 | 24 | 12 | 4 | 11 | |

| Type | File | Logic Contracts | Interfaces | Lines | nLines | nSLOC | Comment Lines | Complex. Score | Capabilities |
|---|---|---|---|---|---|---|---|---|---|
| ⍰ | contracts/helpers/NonceManager.sol | 1 | | 23 | 23 | 14 | 3 | 10 | |
| ⍰ | contracts/libraries/Permitable.sol | 1 | | 47 | 47 | 38 | 5 | 23 | |
| ⍰ | contracts/libraries/ArgumentsDecoder.sol | 1 | | 33 | 33 | 26 | 5 | 45 | ⍰ |
| ⍰ | contracts/libraries/RevertReasonParser.sol | 1 | | 62 | 62 | 39 | 17 | 56 | ⍰ |
| ⍰⍰⍰⍰ | **Totals** | **14** | **3** | **890** | **789** | **541** | **151** | **477** | ⍰⍰⍰ |

Legend: [+]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

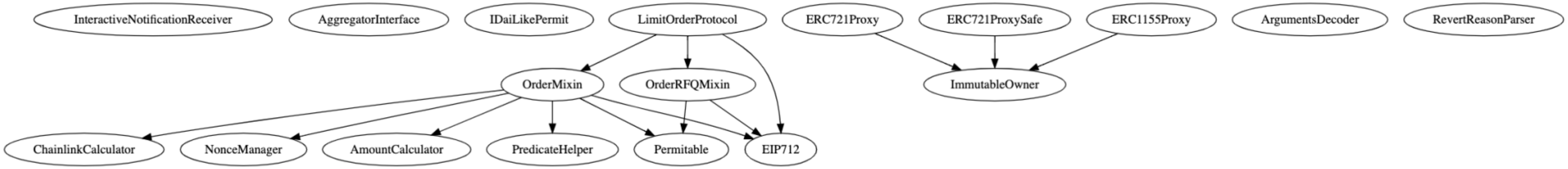
## 5. Scope of Work

The 1inch Team provided us with the files that needs to be tested. The scope of the audit is the limit order protocol v2 (latest commit 9d11830) contracts.

The team put forward the following assumptions regarding the security, usage of the contracts:

- The smart contract is coded according to the newest standards and in a secure way
- The changes since the last audit didn't effected the codebase https://github.com/chainsulting/Smart-Contract-Security-Audits/blob/master/1inch_Exchange/02_Smart%20Contract%20Audit_1inch_limit_order_protocol_solc07.pdf

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



## 5.1 Manual and Automated Vulnerability Test

### CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

## HIGH ISSUES

During the audit, Chainsulting’s experts found **no High issues** in the code of the smart contract.

## MEDIUM ISSUES

During the audit, Chainsulting’s experts found **no Medium issues** in the code of the smart contract

## LOW ISSUES

During the audit, Chainsulting’s experts found **no Low issues** in the code of the smart contract

## INFORMATIONAL ISSUES

During the audit, Chainsulting’s experts found **no Informational issues** in the code of the smart contract

## 5.2. SWC Attacks

| ID | Title | Relationships | Test Result |
|---|---|---|---|
| SWC-131 | Presence of unused variables | CWE-1164: Irrelevant Code | ☑ |
| SWC-130 | Right-To-Left-Override control character (U+202E) | CWE-451: User Interface (UI) Misrepresentation of Critical Information | ☑ |
| SWC-129 | Typographical Error | CWE-480: Use of Incorrect Operator | ☑ |
| SWC-128 | DoS With Block Gas Limit | CWE-400: Uncontrolled Resource Consumption | ☑ |
| SWC-127 | Arbitrary Jump with Function Type Variable | CWE-695: Use of Low-Level Functionality | ☑ |
| SWC-125 | Incorrect Inheritance Order | CWE-696: Incorrect Behavior Order | ☑ |
| SWC-124 | Write to Arbitrary Storage Location | CWE-123: Write-what-where Condition | ☑ |
| SWC-123 | Requirement Violation | CWE-573: Improper Following of Specification by Caller | ☑ |

| ID | Title | Relationships | Test Result |
|---|---|---|---|
| SWC-122 | Lack of Proper Signature Verification | CWE-345: Insufficient Verification of Data Authenticity | ✅ |
| SWC-121 | Missing Protection against Signature Replay Attacks | CWE-347: Improper Verification of Cryptographic Signature | ✅ |
| SWC-120 | Weak Sources of Randomness from Chain Attributes | CWE-330: Use of Insufficiently Random Values | ✅ |
| SWC-119 | Shadowing State Variables | CWE-710: Improper Adherence to Coding Standards | ✅ |
| SWC-118 | Incorrect Constructor Name | CWE-665: Improper Initialization | ✅ |
| SWC-117 | Signature Malleability | CWE-347: Improper Verification of Cryptographic Signature | ✅ |
| SWC-116 | Timestamp Dependence | CWE-829: Inclusion of Functionality from Untrusted Control Sphere | ✅ |
| SWC-115 | Authorization through tx.origin | CWE-477: Use of Obsolete Function | ✅ |
| SWC-114 | Transaction Order Dependence | CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') | ✅ |

| ID | Title | Relationships | Test Result |
|---|---|---|---|
| SWC-113 | DoS with Failed Call | CWE-703: Improper Check or Handling of Exceptional Conditions | ✓ |
| SWC-112 | Delegatecall to Untrusted Callee | CWE-829: Inclusion of Functionality from Untrusted Control Sphere | ✓ |
| SWC-111 | Use of Deprecated Solidity Functions | CWE-477: Use of Obsolete Function | ✓ |
| SWC-110 | Assert Violation | CWE-670: Always-Incorrect Control Flow Implementation | ✓ |
| SWC-109 | Uninitialized Storage Pointer | CWE-824: Access of Uninitialized Pointer | ✓ |
| SWC-108 | State Variable Default Visibility | CWE-710: Improper Adherence to Coding Standards | ✓ |
| SWC-107 | Reentrancy | CWE-841: Improper Enforcement of Behavioral Workflow | ✓ |
| SWC-106 | Unprotected SELFDESTRUCT Instruction | CWE-284: Improper Access Control | ✓ |
| SWC-105 | Unprotected Ether Withdrawal | CWE-284: Improper Access Control | ✓ |
| SWC-104 | Unchecked Call Return Value | CWE-252: Unchecked Return Value | ✓ |

| ID | Title | Relationships | Test Result |
|---|---|---|---|
| SWC-103 | Floating Pragma | CWE-664: Improper Control of a Resource Through its Lifetime | X |
| SWC-102 | Outdated Compiler Version | CWE-937: Using Components with Known Vulnerabilities | ✅ |
| SWC-101 | Integer Overflow and Underflow | CWE-682: Incorrect Calculation | ✅ |
| SWC-100 | Function Default Visibility | CWE-710: Improper Adherence to Coding Standards | ✅ |

## 5.3. Verify Claims

**5.3.1 The smart contract is coded according to the newest standards and in a secure way**
**Status:** tested and verified ✓

**5.3.2 The changes since the last audit didn’t affected the codebase**
**Status:** tested and verified ✓

## 6. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debrief took place on the October 16, 2021. The main goal of the audit was to verify the claims regarding the security of the smart contract and regards the changes that has been made since the last audit.

During the audit, no critical issues were found after the manual and automated security testing and the claims been successfully verified.