1Inch Aggregator, LimitOrder Protocol and Utils Audit

Cumulative Update

September 2022

By CoinFabrik

Introduction

CoinFabrik was asked to audit the contracts for the 1Inch project. First we will provide a summary of our discoveries, and then we will show the details of our findings.

Scope

The audited files are from the git repositories and commits, detailed in the following table:

1inch-contract (https://github.com/1inch/1inch-contract)	`3461c75d00481d111f0323d7e1cb6e56b0dc7ec0`
limit-order-protocol (https://github.com/1inch/limit-order-protocol)	`403be0f583b863b3b13373c23e8f2652f5ef720d`
solidity-utils (https://github.com/1inch/solidity-utils)	`32b04c767db00d4ddc2428534fd6e90abbacb12e`

For 1inch-contracts:

- `contracts/AggregationRouterV5.sol:` Facilitates trading across multiple DEX.
- `contracts/routers/ClipperRouter.sol:` Facilitates swapping assets using the Clipper DEX.
- `contracts/routers/GenericRouter.sol:` Base swapping contract.
- `contracts/routers/UnoswapRouter.sol:` Facilitates swapping assets using Uniswap.
- `contracts/routers/UnoswapV3Router.sol:` Facilitates swapping assets using Uniswap V3.
- `contracts/interfaces/IClipperExchangeInterface.sol:` Interface.
- `contracts/interfaces/IAggregationExecutor.sol:` Interface.
- `contracts/interfaces/IUniswapV3Pool.sol:` Interface.
- `contracts/interfaces/IUniswapV3SwapCallback.sol:` Interface.
- `contracts/helpers/Errors.sol:` Error definition.

For limit-order-protocol:

- `contracts/OrderMixin.sol:` Function implementations for filling orders from the OrderLib library.

- `contracts/OrderRFQMixin.sol:` Function implementations for filling orders from the `OrderRFQLib` library.
- `contracts/OrderLib.sol:` Library which defines the Order struct and its methods.
- `contracts/OrderRFQLib.sol:` Library which defines the `OrderRFQ` struct and its methods.
- `contracts/helpers/AmountCalculator.sol:` Helper contract to calculate swap taker/maker amount.
- `contracts/helpers/NonceManager.sol:` Helper for managing nonce of each `tx-sender`.
- `contracts/helpers/PredicateHelper.sol:` Helper contract for executing boolean functions on arbitrary target call results.
- `contracts/interfaces/IOrderMixin.sol:` Order Base interface.
- `contracts/interfaces/NotificationReceiver.sol:` Notification interface.
- `contracts/libraries/ArgumentsDecoder.sol:` Argument helper contract.
- `contracts/libraries/Errors.sol:` Error definition.

For solidity-utils:

- `contracts/EthReceiver.sol:` Support lib.
- `contracts/OnlyWethReceiver.sol:` Support lib.
- `contracts/libraries/StringUtil.sol:` String manipulation lib.
- `contracts/libraries/UniERC20.sol:` Uniswap compatibility lib.
- `contracts/libraries/SafeERC20.sol:` Safe token functions lib.
- `contracts/libraries/ECDSA.sol:` Signature library.
- `contracts/libraries/RevertReasonForwarder.sol:` Error handling
- `contracts/interfaces/IWETH.sol:` WETH Interface.
- `contracts/interfaces/IDaiLikePermit.sol:` Permit Interface

The scope of the audit is limited to those files. No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Analyses

Without being limited to them, the audit process included the following analyses:

- Arithmetic errors
- Outdated version of Solidity compiler
- Race conditions

- Reentrancy attacks
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

Summary of Findings

We found one minor issue, no medium issues and no critical issues.

Security Issues

ID	Title	Severity	Status
MI-01	Use of Preliminary Version Contracts	Minor	Unresolved

Security Issues Found

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of, but can be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit can have four distinct statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code, but is a result of an intentional decision.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk

Critical Severity Issues

No issues found.

Medium Severity Issues

No issues found.

Minor Severity Issues

MI-01 Use of Preliminary Version Contracts

Location:

- `contracts/AggregationRouterV5.sol:9`

`AggregationRouterV5` uses a preliminary version of EIP712:

```
import { EIP712 } from
"@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol";
```

OpenZeppelin 3.4 as of 8-11-2022 contains the final version of EIP712. The draft is deprecated. Consider using the final version of contracts that might contain improvements and bug-fixes.

Recommendation

Update the contract to the non-draft version, or include documentation about features that need the specific draft version.

Status

Unresolved

Enhancements

No enhancements are suggested in this audit.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest for other stakeholders of the project, including users of the audited contracts, owners or project investors.

Centralization

The contract owner can retire any funds that the `AggregationRouterV5` contracts may have. As this contract implements the `Ownable` OpenZeppelin contract, it can renounce ownership in case it is required.

Upgrades

There is no provision in any contract for upgrades, and making contracts upgradables will require refactoring of the code, due to the use of constructors.

Privileged Roles

These are the privileged roles that we identified on each of the audited contracts.

AggregationRouterV5

Owner

The owner of the contract can rescue funds and delete the contract.

Arbitrary token transfers

The contract `executors/AggregationExecutorSimple.sol` has three externally callable functions: `execute()`, `func_70hHgmC()` and `func_43kDPns()`. All three allow any address to craft arbitrary `IERC20.transfer()` and `IERC20.approve()` calls from any Token with any amount. The dev team acknowledged this as a design decision in previous audits, as the contract will only hold balance temporarily.

Changelog

- 2022-09-17 – Initial report based on commits `3461c75d00481d111f0323d7e1cb6e56b0dc7ec0` (1inch), `403be0f583b863b3b13373c23e8f2652f5ef720d` (limit-order) and `32b04c767db00d4ddc2428534fd6e90abbacb12e` (solidity-utils)

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the 1Inch project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.