1inch Aggregation Router V6 Security Analysis

by Pessimistic

This report is public

March 27, 2024

Abstract

In this report, we consider the security of smart contracts of 1inch Aggregation Router V6 project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

Summary

In this report, we considered the security of 1inch Aggregation Router V6 smart contracts. We described the audit process in the section below.

The initial audit showed only two issues of low severity. All issues were fixed, and audit notes were commented on.

The overall code quality is high.

After the initial audit, we reviewed several new versions of the code. The scope of all codebase updates is described in the Project overview section. Each stage of the audit is detailed in the Audit process section.

General recommendations

We do not have any recommendations.

Project overview

Project description

For the audit, we were provided with the following repositories:

1. Aggregation Router private GitHub repository, commit b301c15f8c58d51b371cfd10baf445461d09d1b7;
2. Limit Order public GitHub repository, commit 38eb988e9f496432e8eb12ef47ee134ceba89a40;
3. Solidity Utils public GitHub repository, commit 42615efa5dca3ed43be565097ccb82a9c3e87273;

The scope of the audit included:

1. Aggregation Router:
    - **AggregationRouterv6.sol**;
    - **helpers/RouterErrors.sol**;
    - **libs/ProtocolLib.sol**;
    - **routers/ClipperRouter.sol** (difference between commit numbers bc00c75f2c99d62e1a206aa2f81c408caba4b370 and b301c15f8c58d51b371cfd10baf445461d09d1b7);
    - **routers/GenericRouter.sol** (difference between commit numbers bc00c75f2c99d62e1a206aa2f81c408caba4b370 and b301c15f8c58d51b371cfd10baf445461d09d1b7);
    - **routers/UnoswapRouter.sol**;
    - dependencies.

2. Solidity Utils:
    - **libraries/SafeERC20.sol**.

3. Limit Order:
    - **OrderLib.sol**;
    - **OrderMixin.sol**;
    - **helpers/AmountCalculator.sol**;
    - **helpers/PredicateHelper.sol**;
    - **helpers/SeriesEpochManager.sol**;
    - **libraries/BitInvalidatorLib.sol**;
    - **libraries/ConstraintsLib.sol**;
    - **libraries/Errors.sol**;
    - **libraries/ExtensionLib.sol**;
    - **libraries/LimitsLib.sol**;
    - **libraries/OffsetsLib.sol**;
    - **libraries/RemainingInvalidatorLib.sol**;
    - dependencies.

The documentation for the project included:

- Online documentation;
- NatSpec comments.

Tests results:

1. **Aggregation Router**: 111 tests out of 112 pass successfully, one pending. Passing tests cover 59.44% of the codebase.

2. **Limit Order**: 107 tests out of 109 pass successfully, two pending. Passing tests cover 86.61% of the codebase.

3. **Solidity Utils**: All 259 tests pass successfully. The code coverage is 98.96%.

The total LOC of audited sources is 1767.

Codebase update #1

After the initial audit, the developers provided us with the following commits:

1. Aggregation Router private GitHub repository, commit 230fe7e53ab10744eb1afa38dac5ddc888ca7b99;
2. Limit Order public GitHub repository, commit da3e187f46885dfd234d9f630bd0afdf5c2ccfc0;

In the updated code, the developers updated the documentation, simplified the flow of the Limit Order protocol, changed the way the code works with the user permits, and made several optimizations. In addition to this, they added support for curve callbacks. We reviewed the changes and found one low severity issue (L03).

Tests results:

1. **Aggregation Router**: 143 tests out of 145 pass successfully, two pending. Passing tests cover 71.22% of the codebase.

2. **Limit Order**: 135 tests out of 140 pass successfully, five pending. Passing tests cover 85.09% of the codebase.

Codebase update #2

After the first recheck, the codebase was updated again.

1. Aggregation Router private GitHub repository, commit cb472e7d0919918195ff3f14d463743a2738a1eb;
2. Limit Order public GitHub repository, commit 1a32e059f78ddcf1fe6294baed6cafb73a04b685;

With this update the developers introduced pausable functionality, several minor features, and made several optimizations. The codebase is heavily optimized already and includes numerous edge cases that are not clearly commented or described. However, the developers promptly provided the required information. Besides, the code is thoroughly tested and reviewed. As a result, the recheck discovered just two notes (N05, N06).

Tests results:

1. **Aggregation Router**: 151 tests out of 156 pass successfully, with four unit tests pending and one failing. Passing tests cover 69.14% of the codebase.

2. **Limit Order**: 145 tests out of 150 pass successfully, five pending. Passing tests cover 94.14% of the codebase.

Codebase update #3

For the third codebase update, we received a new commit in the Limit Order GitHub repository: 562fa69c3848ac90dcdf5a5f0c474cd4b59490bf.

In this update, the developers implemented the **FeeTaker** contract and added several new tests.

149 tests out of 154 pass successfully, with five pending. The passing tests cover 94.04% of the codebase.

Audit process

We started the audit on March 17, 2023, and finished on April 4, 2023.

We inspected the materials provided for the audit. Then, we contacted the developers for an introduction to the project. After a discussion, we performed preliminary research.

During the work, we stayed in touch with the developers and discussed confusing or suspicious parts of the code.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Among other, we verified the following properties of the contracts:

- The usage of memory in the assembly parts of the code. This includes the parts that are not memory-safe, despite being annotated as such;
- The usage of introduced permit functionality;
- The safety of user funds in the router, based on the security assumptions;
- Whether the orders could be maliciously invalidated (see N03 note);
- The possibility of reentrancy attacks;
- The safety of Uniswap V3 callback.

We ran tests and calculated the code coverage.

We combined in the report all the verified issues we found during the manual audit and reflected all the fixes and comments from the developers.

On October 6, 2023, the developers provided a new code version. This update includes revisions to the project documentation, adjustments to the system flow, and simplifications of the Limit Order protocol. Additionally, it addresses several issues that were not identified during the initial audit.

We reviewed the updated codebase and identified one issue of the low severity (see L03). Finally, we updated the report.

On February 1, 2024, we received an updated version of the codebase. The introduction of Ownable and Pausable features caused most of the changes.

We checked all changes since the previous iteration. Some of them might affect security third-party code (see N05 and N06). No other issues were discovered during the review. We updated the report to reflect these findings.

On March 12, 2024, we received an updated version of the code. The changes were limited to the Limit Order protocol repository. The developers added a new **FeeTaker** contract and introduced new tests.

We examined the new contract and found no issues. We updated the report accordingly.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

The audit showed no issues of medium severity.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. Missing check (fixed)

In the **SafeERC20** contract, `safeTransferFromUniversal` function of the solidity-utils repository, consider adding a check for the value of the `amount` variable. In case when the `amount` is greater than the maximum value of the `uint160` type, line 33 would not work properly.

This issue has been fixed at commit 4ecd7340e016da7fcdad69ca811f1b201c5c84fa

L02. Magic number (fixed)

In the **UnoswapRouter** contract of the Aggregation Router repository, consider using the `_ADDRESS_MASK` constant instead of `0xffffffffffffffffffffffffffffffffffffffff` at lines 219, 280, and 390.

This issue has been fixed at commit d63421fafccddbad814eda3144c184b6be4bebc2

L03. Missing check (fixed)

ExtensionLib.sol file contains functions for extracting data from the extension. In the previous version, all of the fields were obtained via the `_get` function. However, in a new version of the code, the developers implemented a `customData` field that is extracted with the use of `customData` function. That function lacks the check on the extension length, that is present in the `_get` function. We recommend adding the mentioned check to the `customData` function to align it with the existing check in the `_get` function.

Note that the `customData` function is not currently used in the project, but we still recommend adding the aforementioned check to it.

This issue has been fixed at commit df84580da9f2b888de318d29a5c014a12072f614

Notes

N01. Accidentally sent funds (commented)

The **UnoswapRouter** contract has two functions to retrieve funds that were accidentally sent to the contract. The `rescueFunds` function allows only authorized roles to perform the transaction. However, the `uniswapV3SwapCallback` function allows anyone to transfer funds from the contract to any address by calling the function with the `payer` argument set to the router address.

> *Comment from the developers: Won't fix. The router is not expected to hold any funds, so the rescueFunds is there only for convenience in case of MEV bots not caring about the stuck funds.*

N02. Security assumptions (commented)

In the **UnoswapRouter** contract, the `minReturn` check is performed by pools only. Thus, the user must check the address of the last pool by themselves.

> *Comment from the developers: Won't fix. This is done on purpose to save on gas.*

N03. Invalidation of the partially filled orders (commented)

Limit orders have two options for the invalidation of the order. Bit invalidation invalidates the order after the first exchange. This allows malicious users to invalidate orders with partial fills enabled by exchanging a small amount of tokens.

> *Comment from the developers: Won't fix. This is the expected behaviour.*

N04. Raw call gas limit (commented)

Limit orders project provides 5000 gas for the raw calls. We recommend clarifying the selected value in the documentation. This amount allows the reentrancy; however, the gas amount might not be enough for some `fallback` functions (if they are expected and allowed).

> *Comment from the developers: Won't fix. We will improve the documentation.*

N05. Renaming public variables

Public constant `immutableOwner` of **ImmutableOwner** contract was renamed to `IMMUTABLE_OWNER`. This change might break third-party (both contracts and off-chain) that integrates with the system.

N06. Order cancelation event

After the update, an order can be cancelled without emitting an `OrderCancelled` event. The `cancelOrder` function of the **OrderMixin** contract implements two approaches to order invalidation, and they emit different events. Any off-chain code that monitors orders should process both event types.

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Pavel Kondratenkov, Senior Security Engineer
Yhtyyar Sahatov, Security Engineer
Irina Vikhareva, Project Manager

March 27, 2024