OpenZeppelin | security

Aggregation Protocol Diff Audit

1inch NETWORK

December 15, 2023

Table of Contents

Table of Contents 2

Summary 3

Scope 4

Overview of the Changes 4

Non-Issue Observations 5

Low Severity 6

L-01 Restricted External Calls Can Be Made on Behalf of AggregationRouter 6

L-02 Assembly Block Diverges from Solidity's Memory Model 6

Notes & Additional Information 7

N-01 Code Clarity Suggestions 7

N-02 Reliance on External Slippage Protection 7

N-03 Lack of Security Contact 7

N-04 Unused Errors 8

Conclusion 9

Summary

Type	DeFi
Timeline	From 2023-11-16 To 2023-11-22
Languages	Solidity, Yul

Total Issues	6 (5 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	2 (2 resolved)
Notes & Additional Information	4 (3 resolved)

Scope

We audited the changes made to the 1inch/1inch-contract repository between the `6.0.0-prerelease-2` tag and the `master` branch at that time. The corresponding commits were db5dfdf and 83db150.

In scope were the following contracts:

```
contracts
├── AggregationRouterV6.sol
└── routers
    ├── ClipperRouter.sol
    ├── GenericRouter.sol
    └── UnoswapRouter.sol
```

Overview of the Changes

The 1inch team incorporated the following changes into the protocol:

- Added support for Curve pools with callbacks.
- Fixed two bugs and refactored one component of the `ClipperRouter` contract:
  - pull request #243 resolves the dirty memory issue by storing the length prefix and the tag as separate values.
  - pull request #240 changes the `expiryWithFlags` parameter, which previously contained both the permit2 flag and Clipper's `goodUntil` parameter, to the `goodUntil` parameter, which now only contains the Clipper's `goodUntil` parameter. The permit2 flag is now stored in the `srcToken` parameter.
  - pull request #236 introduces a code refactor and abstracts WETH assembly blocks into a library.
- Users can now use the `permitAndCall` convenience function inherited from `OrderMixin` to execute a permit and perform a swap in one function call. At the same time, the permit parameter has been removed from the swap functions. Supported permits include ERC-2612 permits, DAI-like permits, and Permit2.

Non-Issue Observations

Presented below are observations regarding the protocol's behavior that might be concerning. However, we have not found a particular way in which they may be used maliciously.

- The `curveSwapCallback` function does not have access control even though the `rescueFunds` function with similar functionality does. Through the former function, it is possible to sweep 1 wei from the protocol's balance of each token. This is because the protocol always owns some amount of both tokens for gas optimization reasons. However, it is worth noting that the cost of sweeping is high enough to disincentivize such behavior.

Low Severity

L-01 Restricted External Calls Can Be Made on Behalf of `AggregationRouter`

The Aggregation protocol is supposed to integrate with any Curve pool. Given the variety of the pool types within the protocol, this provides users with a significant degree of control. For example, one of the external calls made on behalf of the protocol takes most of its parameters directly from untrusted user input. This includes the function selector, target address and parameters. This might become problematic because the protocol holds user token approvals. However, the two most significant parameters are restricted to just one byte (values 0-255) which limits the potential impact.

Consider further restricting the parameters of this external call.

Update: *Resolved in pull request #268 at commits 8458b1e and 184ad46. The 1inch team removed the ability to freely specify any 4-byte selector within the* `dex` *parameter. Instead, users can now choose one of the 18 preconfigured selectors by specifying a 1-byte index from 0 to 17. This choice heavily reduces the ability to perform arbitrary external calls.*

L-02 Assembly Block Diverges from Solidity's Memory Model

One of the assembly blocks marked as memory safe contains operations that might be memory unsafe according to the Solidity documentation. The reason is that the return data size might be greater than the scratch space for some WETH implementations.

Consider using the free memory pointer to retrieve an unused memory location as implemented in the `safeWithdraw` function.

Update: *Resolved in pull request #104 at commit ac7d637.*

Notes & Additional Information

N-01 Code Clarity Suggestions

The `tokenBalanceOf` function returns the balance of a specific token for a specific address, minus 1. However, the function name suggests that it simply returns the token balance of the provided address. This might be confusing for some readers.

Consider either moving the subtraction out of the function as implemented in the previous case within the same switch statement. Alternatively, consider renaming the function.

Update: *Resolved in pull request #276 at commit 6c22250.*

N-02 Reliance on External Slippage Protection

For swaps via Curve pools, the Aggregation protocol relies on slippage protection being implemented by the pools. While we were not able to bypass the slippage protection, the Aggregation protocol nonetheless allows for reentrancy.

Consider adding slippage protection at the protocol level in a way similar to how it is done for Uniswap pools.

Update: *Resolved in pull request #270 at commit 9bd2950.*

N-03 Lack of Security Contact

Providing a specific security contact (such as an email or ENS name) within a smart contract significantly simplifies the process for individuals to communicate if they identify a vulnerability in the code. This practice proves beneficial as it permits the code owners to dictate the communication channel for vulnerability disclosure, eliminating the risk of miscommunication or failure to report due to a lack of knowledge on how to do so. Additionally, if the contract incorporates third-party libraries and a bug surfaces in these, it becomes easier for the maintainers of those libraries to make contact with the appropriate person about the problem and provide mitigation instructions.

Throughout the codebase, there are contracts that do not have a security contact:

- The `AggregationRouterV6` contract
- The `ClipperRouter` contract
- The `GenericRouter` contract
- The `UnoswapRouter` contract

Consider adding a NatSpec comment containing a security contact above the contract definition. Using the `@custom:security-contact` convention is recommended as it has been adopted by the OpenZeppelin Wizard and the ethereum-lists.

Update: *Acknowledged, will resolve. The 1inch team stated that they will follow this practice in the future.*

N-04 Unused Errors

Throughout the codebase, several unused errors were identified:

- The `ZeroReturnAmount` error in `GenericRouter.sol`
- The `SwapAmountTooLarge` error in `UnoswapRouter.sol`

To improve the overall clarity, intentionality, and readability of the codebase, consider either using or removing any unused errors.

Update: *Resolved in pull request #276 at commits d6fac8e and 1a7ff67.*

Conclusion

The 1inch Aggregation Protocol provides a common interface to perform atomic swaps against a wide variety of pools (such as Clipper, Curve, Uniswap-like V2 and V3 pools) and even more generic swaps via custom executors. This is so that users can benefit from the best prices available across a myriad of different decentralized exchanges. We found the codebase to be very well-written, thoroughly documented and optimized for gas consumption.

No major issues were found during the audit, although some best practice recommendations were made in order to make the 1inch codebase even more robust. We focused on ensuring that the fixes and enhancements implemented on this diff audit work as expected and maintain high security standards.

The team was very responsive throughout the engagement, providing us with the necessary insight to expedite our understanding of the changes implemented and how they affect the codebase.