ABDK CONSULTING

SMART CONTRACT AUDIT

1inch

Aggregation Router V5

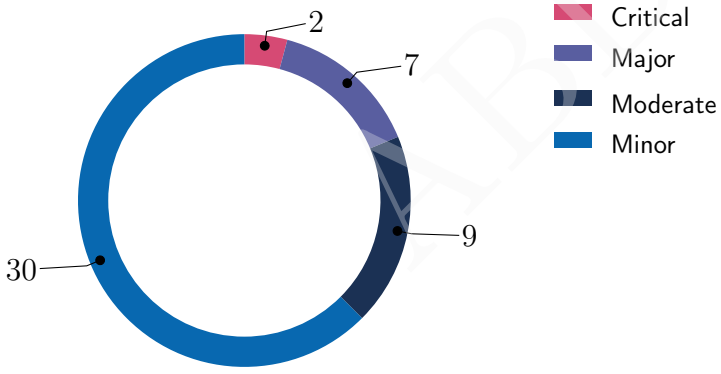
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
16th September 2022

We've been asked to review updates to 30 files in a GitHub repo. We found 2 critical, 7 major, and a few less important issues. All identified critical have been fixed. All major issues have been fixed or otherwise addressed in collaboration with the client.



Findings

ID	Severity	Category	Status
CVF-1	Moderate	Documentation	Fixed
CVF-2	Moderate	Suboptimal	Info
CVF-3	Moderate	Unclear behavior	Info
CVF-4	Minor	Suboptimal	Info
CVF-5	Moderate	Flaw	Fixed
CVF-6	Critical	Flaw	Fixed
CVF-7	Critical	Flaw	Fixed
CVF-8	Minor	Documentation	Fixed
CVF-9	Major	Flaw	Fixed
CVF-10	Minor	Bad datatype	Info
CVF-11	Minor	Suboptimal	Info
CVF-12	Minor	Suboptimal	Fixed
CVF-13	Minor	Readability	Info
CVF-14	Major	Unclear behavior	Fixed
CVF-15	Minor	Readability	Info
CVF-16	Minor	Bad datatype	Info
CVF-17	Minor	Suboptimal	Fixed
CVF-18	Major	Suboptimal	Fixed
CVF-19	Minor	Readability	Info
CVF-20	Minor	Suboptimal	Fixed
CVF-21	Minor	Suboptimal	Info
CVF-22	Major	Suboptimal	Fixed
CVF-23	Moderate	Flaw	Fixed
CVF-24	Moderate	Suboptimal	Fixed
CVF-25	Minor	Suboptimal	Info
CVF-26	Minor	Suboptimal	Info
CVF-27	Minor	Suboptimal	Fixed

ID	Severity	Category	Status
CVF-28	Minor	Suboptimal	Fixed
CVF-29	Minor	Suboptimal	Info
CVF-30	Minor	Suboptimal	Info
CVF-31	Moderate	Suboptimal	Info
CVF-32	Minor	Suboptimal	Info
CVF-33	Major	Flaw	Fixed
CVF-34	Moderate	Suboptimal	Fixed
CVF-35	Minor	Suboptimal	Fixed
CVF-36	Major	Overflow/Underflow	Info
CVF-37	Moderate	Flaw	Info
CVF-38	Minor	Documentation	Info
CVF-39	Minor	Procedural	Info
CVF-40	Major	Overflow/Underflow	Info
CVF-41	Minor	Documentation	Info
CVF-42	Minor	Procedural	Info
CVF-43	Minor	Documentation	Info
CVF-44	Minor	Suboptimal	Info
CVF-45	Minor	Suboptimal	Fixed
CVF-46	Minor	Procedural	Info
CVF-47	Minor	Documentation	Info
CVF-48	Minor	Documentation	Fixed

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	September 15, 2022	A. Zveryanskaya	Initial Draft
0.2	September 15, 2022	A. Zveryanskaya	Minor revision
1.0	September 16, 2022	A. Zveryanskaya	Release

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.
We have reviewed the contracts at repository:

- 1inchcontr/AggregationRouterV5.sol
- 1inchcontr/helpers/Errors.sol
- 1inchcontr/interfaces/IAggregationExecutor.sol
- 1inchcontr/interfaces/IClipperExchangeInterface.sol
- 1inchcontr/interfaces/IUniswapV3Pool.sol
- 1inchcontr/interfaces/IUniswapV3SwapCallback.sol
- 1inchcontr/routers/ClipperRouter.sol
- 1inchcontr/routers/GenericRouter.sol
- 1inchcontr/routers/UnoswapRouter.sol
- 1inchcontr/routers/UnoswapV3Router.sol
- limit-order/helpers/AmountCalculator.sol
- limit-order/helpers/NonceManager.sol
- limit-order/helpers/PredicateHelper.sol
- limit-order/interfaces/IOrderMixin.sol
- limit-order/interfaces/NotificationReceiver.sol
- limit-order/libraries/ArgumentsDecoder.sol
- limit-order/libraries/Callib.sol
- limit-order/libraries/Errors.sol
- limit-order/OrderLib.sol
- limit-order/OrderMixin.sol
- limit-order/OrderRFQLib.sol
- limit-order/OrderRFQMixin.sol
- solidity-utils/EthReceiver.sol
- solidity-utils/interfaces/IDaiLikePermit.sol

- solidity-utils/interfaces/IWETH.sol
- solidity-utils/libraries/ECDSA.sol
- solidity-utils/libraries/RevertReasonForwarder.sol
- solidity-utils/libraries/SafeERC20.sol
- solidity-utils/libraries/StringUtil.sol
- solidity-utils/libraries/UniERC20.sol

The fixes were provided in a new commit.

2.1 About ABDK

ABDK Consulting, established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function. The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

2.2 Disclaimer

Note that the performed audit represents current best practices and smart contract standards which are relevant at the date of publication. After fixing the indicated issues the smart contracts should be re-audited.

2.3 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places and that their visibility scopes and access levels are relevant. At this phase we understand overall system architecture and how different parts of the code are related to each other.

- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and is done properly. At this phase we understand user roles and permissions, as well as what assets the system ought to protect.

- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

3 Detailed Results

3.1 CVF-1

- **Severity** Moderate
- **Status** Fixed
- **Category** Documentation
- **Source** IAggregationExecutor.sol

Description This comment is not relevant anymore as no data is now specified.
Recommendation Consider fixing the comment.

Listing 1:

```
9 /// @notice Make calls on `msgSender` with specified data
```

3.2 CVF-2

- **Severity** Moderate
- **Status** Info
- **Category** Suboptimal
- **Source** SafeERC20.sol

Recommendation This function could be simplified by using "_makeCalldataCall" line this: _makeCalldataCall (token, token.transferFrom.selector, msg.data [0x24, 0x84])
Client Comment We have no assumptions about contents of msg.data in this library.

Listing 2:

```
20 function safeTransferFrom(IERC20 token, address from, address to
   ↪ , uint256 amount) internal {
```

3.3 CVF-3

- **Severity** Moderate
- **Status** Info
- **Category** Unclear behavior
- **Source** SafeERC20.sol

Description The returned data from the inner failed call is lost.
Recommendation Consider including it into the error.
Client Comment This is a valid point, but we decided to make the required changes in the next release. For now it is won't fix.

Listing 3:

```
35 revert SafeTransferFromFailed();

52     revert ForceApproveFailed();

82 RevertReasonForwarder.reRevert();
```

3.4 CVF-4

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** SafeERC20.sol

Recommendation This function is redundant, as "_makeCalldataCall" may be used instead like this: _makeCalldataCall(token, token.approve.selector, msg.data[0x24, 0x64]) or _makeCalldataCall(token, token.approve.selector, msg.data[0x24, 0x44]) to pass zero value.
Client Comment We have no assumptions about contents of msg.data in this library.

Listing 4:

```
86 function _makeCall(IERC20 token, bytes4 selector, address to,
   ↪ uint256 amount) private returns(bool done) {
```

3.5 CVF-5

- **Severity** Moderate
- **Status** Fixed
- **Category** Flaw
- **Source** ECDSA.sol

Description The ECDSA recover precompiler signals error by returning empty output, rather than by reverting, so this check doesn't make much sense. Even if the precompile would revert, this function doesn't properly handle failed call and just silently return zero address, thus anybody would be able to "sign" anything on behalf of the zero address.
Recommendation Consider reverting on failed call.
Client Comment We do not want to revert in library but instead want to mimick solidity's behavior by returning zero address. precompile intercation was fixed.

Listing 5:

```
18 if staticcall(gas(), 0x1, ptr, 0x80, 0, 0x20) {

33 if staticcall(gas(), 0x1, ptr, 0x80, 0, 0x20) {

71         if staticcall(gas(), 0x1, ptr, 0x80, 0, 0x20) {
```

3.6 CVF-6

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** ECDSA.sol

Description In case of invalid signature, the precompile call will return nothing (zero return data size0, this the return buffer would remain untouched, i.e. could contain arbitrary jnk value, and this junk will be returned as the recovered address. Assuming that a user may manipulate the contents of the zero memory slot, this allows a user to "sign" anything on behalf of an arbitrary address.
Recommendation Consider checking the return data size.

Listing 6:

```
19 signer := mload(0)

34 signer := mload(0)

72         signer := mload(0)
```

3.7 CVF-7

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** ECDSA.sol

Description In case the "isValidSignature" call returned nothing, for example from a contract that has a fallback function but don't have the "isValidSignature" function, or it an EOA, the return buffer will remain untouched, i.e. would contain an arbitrary junk value. Assuming that a user may manipulate the content of the zero memory slot, this would allow a user to "sign" anything on behalf of such smart contract or on behalf of an EOA.
Recommendation Consider checking the return data size.
Client Comment We store zero in zero memory slot to avoid checking return data size. If function returns anything apart from valid selector isValidSignature will return false.

Listing 7:

```
123 success := eq(selector, mload(0))

144 success := eq(selector, mload(0))

166 success := eq(selector, mload(0))

189 success := eq(selector, mload(0))
```

3.8 CVF-8

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** OrderRFQMixin.sol

Description Actually, this function is closer to "fillOrderRFQTo" rather than to "fillOrderRFQ", as it allows specifying where to send swap funds.
Recommendation Consider changing the comment.

Listing 8:

```
133 * @notice Fills Same as `fillOrderRFQ` but calls permit first.
```

3.9 CVF-9

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** OrderRFQMixin.sol

Description When several orders are being invalidated, this check will revert in case any of these orders is already invalidated, thus no orders will be invalidated.
Recommendation Consider reverting only when all the orders are already invalidated, i.e. when invalidator & invalidatorBits == invalidatorBits.

Listing 9:

```
254 if (invalidator & invalidatorBits != 0) revert InvalidatedOrder
    ↪ ();
```

3.10 CVF-10

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** OrderRFQLib.sol

Recommendation The type of these fields should be "IERC20".
Client Comment Won't fix due to probability miscalculation of _LIMIT_ORDER_RFQ_TYPEHASH.

Listing 10:

```
10 address makerAsset;
   address takerAsset;
```

3.11 CVF-11

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** OrderMixin.sol

Description These errors might contain additional information as parameters, such as problematic order ID, amount, account etc.
Recommendation won't fix

Listing 11:

```
28 error UnknownOrder();
   error AccessDenied();
30 error AlreadyFilled();
   error PermitLengthTooLow();
   error ZeroTargetIsForbidden();
   error RemainingAmountIsZero();
   error PrivateOrder();
   error BadSignature();
   error ReentrancyDetected();
   error PredicateIsNotTrue();
   error OnlyOneAmountShouldBeZero();
   error TakingAmountTooHigh();
40 error MakingAmountTooLow();
   error SwapWithZeroAmount();
   error TransferFromMakerToTakerFailed();
   error TransferFromTakerToMakerFailed();
   error WrongAmount();
   error WrongGetter();
   error GetAmountCallFailed();
```

3.12 CVF-12

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** OrderMixin.sol

Recommendation This could be simplified as: unchecked { return amount - 1; }

Listing 12:

```
80 unchecked { amount -= 1; }
   return amount;
```

3.13 CVF-13

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** OrderMixin.sol

Recommendation Should be ‘else‘.
Client Comment Won’t fix.

Listing 13:

```
122 emit OrderCanceled(msg.sender, orderHash, orderRemaining);
```

3.14 CVF-14

- **Severity** Major
- **Status** Fixed
- **Category** Unclear behavior
- **Source** OrderMixin.sol

Description As the actual taking amount depends on many factors, it could be hard for the taker to precisely estimate it.
Recommendation Consider allowing more ether to be sent, and returning the unused ether back to “msg.sender”.

Listing 14:

```
270 if (msg.value != actualTakingAmount) revert InvalidMsgValue();
```

3.15 CVF-15

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** OrderMixin.sol

Recommendation Should be “else return”.
Client Comment Won’t fix

Listing 15:

```
330 return _callGetter(getter, orderHash, orderExpectedAmount,
    ↪ amount, orderResultAmount, remainingAmount);

338 return _callGetter(getter, orderHash, orderExpectedAmount,
    ↪ amount, orderResultAmount, remainingAmount);
```

3.16 CVF-16

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** OrderLib.sol

Recommendation The type of these fields should be "IERC20".
Client Comment Won't fix due to probability of miscalculation of _LIMIT_ORDER_TYPEHASH.

Listing 16:

```
10 address makerAsset;
   address takerAsset;
```

3.17 CVF-17

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** OrderLib.sol

Recommendation This code is redundant, as the code below would have the same effect in case "field" is zero.

Listing 17:

```
60 if (uint256(field) == 0) {
       return order.interactions[0:uint32(order.offsets)];
   }
```

3.18 CVF-18

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** PredicateHelper.sol

Description This code doesn't look efficient and could probably consume all the gas saved in the constant time dispatcher.
Recommendation Consider using a hardcoded binary tree dispatcher that has logarithmic complexity.

Listing 18:

```
112 if (selector == [this.or, this.and, this.eq, this.lt, this.gt][
    ↪ index].selector) {

114     return (true, [or, and, eq, lt, gt][index](arg, param) ? 1 :
        ↪  0);
```

3.19 CVF-19

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** PredicateHelper.sol

Recommendation Should be "} else" for.
Client Comment Won't fix.

Listing 19:

```
115 }

120 }

123 }

128 if (selector == this.arbitraryStaticCall.selector) {

131 }
```

3.20 CVF-20

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** UniERC20.sol

Description Using "transfer" is discouraged.
Recommendation Consider using "call" instead.

Listing 20:

```
38 to.transfer(amount);

53     unchecked { from.transfer(msg.value - amount); }
```

3.21 CVF-21

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** UniERC20.sol

Description It is actually possible to support "to" other than "this" by transferring ether from "this" to "to".
Recommendation Consider implementing this logic for completeness.
Client Comment Won't fix.

Listing 21:

```
50 if (to != address(this)) revert ToIsNotThis();
```

3.22 CVF-22

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** UniERC20.sol

Recommendation Here a selector is derived from a string signature at run time, while it would be more efficient to precompute selectors and pass then as arguments instead of signatures.

Listing 22:

```
81 abi.encodeWithSignature(lowerCaseSignature)

85     abi.encodeWithSignature(upperCaseSignature)
```

3.23 CVF-23

- **Severity** Moderate
- **Status** Fixed
- **Category** Flaw
- **Source** UniERC20.sol

Description This condition doesn't guarantee that abi.decode will succeed.
Recommendation Consider adding the following check: ... && data.length == len + 0x40

Listing 23:

```
91 if (offset == 0x20 && len > 0 && len <= 256) {
```

3.24 CVF-24

- **Severity** Moderate
- **Status** Fixed
- **Category** Suboptimal
- **Source** UniERC20.sol

Description Apart from decoding an ABI-encoded message, this call spends gas validating the message structure, but the message structure was already validated above.
Recommendation Consider simplifying like this: string memory result; assembly { result := add (data, 0x20) } return result;

Listing 24:

```
92 return abi.decode(data, (string));
```

3.25 CVF-25

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** UniERC20.sol

Description Here, the value is truncated at the first non-printable or non-ASCII character, while token symbols may contain non-ASCII characters.
Recommendation Consider stopping at a zero character only.
Client Comment This is to support weird tokens that use bytes32 type for their token or name. Sometimes they are weird enough to just put some unprintable characters in those bytes32 so we'll keep it as it is for sanity reasons.

Listing 25:

```
98 while (len < data.length && data[len] >= 0x20 && data[len] <= 0
   ↪ x7E) {
```

3.26 CVF-26

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** UniERC20.sol

Description A linear loop here is inefficient.
Recommendation Consider using binary search like this: https://gist.github.com/3sGgpQ8H/aa51bbc6535ce3fef5d2773716df3a93
Client Comment Won't fix.

Listing 26:

```
98 while (len < data.length && data[len] >= 0x20 && data[len] <= 0
   ↪ x7E) {
```

3.27 CVF-27

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** StringUtil.sol

Recommendation Shift would be more efficient than multiplication.

Listing 27:

```
54 let resultLength := add(mul(length, 2), 2)
```

3.28 CVF-28

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** StringUtil.sol

Recommendation The bytecode size could be reduced as: mstore(add (result, 2), 0x3078) mstore(result, resultLength)

Listing 28:

```
57 mstore(result, resultLength)
   mstore(toPtr, 0
      ↪ x3078000000000000000000000000000000000000000000000000000000000000
      ↪ ) // set 0x as first two bytes
```

3.29 CVF-29

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** StringUtil.sol

Description This logic is performed even if we are already behind the end of the data.
Recommendation Consider performing one, rather than two, _toHex16 calls per loop iteration and advancing "fromPtr" by 0x10 rather than 0x20.
Client Comment Won't fix. I tried and in most cases compyting both _toHex16 for each 32 byte slot reduces gas cost.

Listing 29:

```
71 hexData := _toHex16(shl(128, rawData))
   mstore(toPtr, hexData)
   toPtr := add(toPtr, 0x20)
```

3.30 CVF-30

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** RevertReasonForwarder.sol

Description This function returns the original return data as is, so there is no way to tell whether the return data from a failed transaction was forwarded from an inner call, or was produced by the main call.
Recommendation Consider wrapping the returned data into a named error.
Client Comment Won't fix. For this we have RevertReasonParser.

Listing 30:

```
5 function reRevert() internal pure {
```

3.31 CVF-31

- **Severity** Moderate
- **Status** Info
- **Category** Suboptimal
- **Source** RevertReasonForwarder.sol

Description Using the free memory pointer here is redundant, as this function reverts, so memory won't be used afterwards.
Recommendation Consider just copying the return data starting at zero offset.
Client Comment Won't fix.

Listing 31:

```
9 let ptr := mload(0x40)
```

3.32 CVF-32

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Errors.sol

Recommendation This error might include the actual msg.value as a parameter.
Client Comment Won't fix.

Listing 32:

```
5 error InvalidMsgValue();
```

3.33 CVF-33

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** Callib.sol

Description In case the return dat size is less than 32, a non-deterministic junk value could be assigned to "res".
Recommendation Consider deterministically assigning zero in such a case.

Listing 33:

```
17 res := mload(0)
```

3.34 CVF-34

- **Severity** Moderate
- **Status** Fixed
- **Category** Suboptimal
- **Source** NonceManager.sol

Description An event is emitted when amount is zero.
Recommendation Consider reverting for zero amount.

Listing 34:

```
18 function advanceNonce(uint8 amount) public {
```

3.35 CVF-35

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** AmountCalculator.sol

Recommendation This contract could be turned into a library.

Listing 35:

```
9 contract AmountCalculator {
```

3.36 CVF-36

- **Severity** Major
- **Status** Info
- **Category** Overflow/Underflow
- **Source** AmountCalculator.sol

Description Phantom overflow is possible here.
Recommendation Consider using the "muldiv" function.
Client Comment Won't fix.

Listing 36:

```
15 return swapTakerAmount * orderMakerAmount / orderTakerAmount;

21 return (swapMakerAmount * orderTakerAmount + orderMakerAmount -
   ↪ 1) / orderMakerAmount;
```

3.37 CVF-37

- **Severity** Moderate
- **Status** Info
- **Category** Flaw
- **Source** AggregationRouterV5.sol

Description Implementing an ability to destroy a smart contract usually causes more harm than good. Destructing a contract effectively burns all the tokens at the contract's balance and also any ether sent to the contract after the destruction. This includes ether transfer initiated before the destruction but executed after it.
Recommendation Consider removing this function.
Client Comment Won't fix.

Listing 37:

```
44 function destroy() external onlyOwner {
```

3.38 CVF-38

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** GenericRouter.sol

Description The reason for returning the amount of gas left is unclear.
Recommendation Consider explaining or not returning the amount of gas left.

Listing 38:

```
49 +uint256 gasLeft
```

3.39 CVF-39

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** GenericRouter.sol

Description The value "srcETH ? desc.amount : 0" is used in both branches.
Recommendation Consider calculating before the conditional statement.
Client Comment Won't fix.

Listing 39:

```
59 +if (msg.value <= (srcETH ? desc.amount : 0)) revert Errors.
    ↪ InvalidMsgValue();

61 +if (msg.value != (srcETH ? desc.amount : 0)) revert Errors.
    ↪ InvalidMsgValue();
```

3.40 CVF-40

- **Severity** Major
- **Status** Info
- **Category** Overflow/Underflow
- **Source** GenericRouter.sol

Description Phantom overflow is possible here.
Recommendation Consider using full multiplication and compare 512-bit values.
Client Comment Won't fix.

Listing 40:

```
87 +if (returnAmount * desc.amount < desc.minReturnAmount *
   ↪ spentAmount) revert Errors.ReturnAmountIsNotEnough();
```

3.41 CVF-41

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** GenericRouter.sol

Description This doesn't properly ABI-encode the data, but the callee could still decode it.
Recommendation Consider documenting how the data is actually passed, as there is no separate argument for data in the "IAggregationExecutor.execute" function.

Listing 41:

```
112 +calldatacopy(add(ptr, 0x44), data.offset, data.length)
```

3.42 CVF-42

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Errors.sol

Description These errors might contain additional information such as actual an desired amounts and return data from the failed inner call.
Client Comment Won't fix.

Listing 42:

```
13 +error ReturnAmountIsNotEnough();
   +error InvalidMsgValue();
   +error ERC20TransferFailed();
```

3.43 CVF-43

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** ClipperRouter.sol

Description The returned values are not documented.
Recommendation Consider documenting.

Listing 43:

```
48 +) external returns(uint256 returnAmount) {
69 +) external payable returns(uint256 returnAmount) {
92 +) public payable returns(uint256 returnAmount) {
```

3.44 CVF-44

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ClipperRouter.sol

Recommendation This check could be moved out of the conditional statement to avoid code duplication like this: if (srcETH) {...} else { if (msg.value != 0) revert Errors.invalidMsgValue(); if (srcToken == _WETH) {...} else {...} }
Client Comment Won't fix.

Listing 44:

```
98 +if (msg.value != 0) revert Errors.InvalidMsgValue();
125 +if (msg.value != 0) revert Errors.InvalidMsgValue();
```

3.45 CVF-45

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** ClipperRouter.sol

Description These two branches have mych common code.
Recommendation Consider merging them to avoid code duplication.

Listing 45:

```
154 +} else if (dstToken == _ETH) {
179 +} else if (dstToken == _WETH) {
```

3.46 CVF-46

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** NotificationReceiver.sol

Recommendation These interfaces should be moved to separate files named after the interfaces.
Client Comment Won't fix.

Listing 46:

```
 9 interface PreInteractionNotificationReceiver {

21 interface PostInteractionNotificationReceiver {

35 interface InteractionNotificationReceiver {
```

3.47 CVF-47

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** IOrderMixin.sol

Description The comment misses an important fact, that this function always reverts and returned the simulation result in revert data.
Recommendation Consider mentioning this in the comment.

Listing 47:

```
44 * @notice Delegates execution to custom implementation. Could be
   ↪ used to validate if `transferFrom` works properly
```

3.48 CVF-48

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** IOrderMixin.sol

Description The function is actually closer to "fillOrderTo" than to "fillOrder", as it allows specifying the funds destination other than "msg.sender"..
Recommendation Consider changing the comment.

Listing 48:

```
80 * @notice Same as `fillOrder` but calls permit first,
```