Composable Conditional orders audit / May-Jul 2023

Files in scope

All solidity files in https://github.com/rndlabs/composable-cow/tree/6edd5dfa78f34fd0a6babc35d610cde33b5d3a7c/src and https://github.com/rndlabs/safe-contracts/tree/5b059aec2c788a81b855d61cd9677b8ca4733c6b/contracts/handler/extensible; ExtensibleFallbackHandler contract https://github.com/rndlabs/safe-contracts/blob/5b059aec2c788a81b855d61cd9677b8ca4733c6b/contracts/handler/ExtensibleFallbackHandler.sol

Differences of the above-mentioned scope with final versions (StopLoss contract was added to the scope): https://github.com/rndlabs/composable-cow/compare/6edd5df..cd893fa and https://github.com/rndlabs/safe-contracts/compare/5b059ae..e53ffea

Current status

All discovered issues have been fixed or addressed.

Issues

L-01. In the `fallback` of `FallbackHandler` contract it should be enforced that `msd.data.length` is >= 4

severity: low

The `fallback` of `FallbackHandler` can have `msg.data` that is equal to `(d || sender)`, where `len(d)` is smaller than 4. In this case handler will receive `d` as `data`, which in most cases means that it will be handled by the fallback. But `_getContextAndHandler` uses `msg.sig`, which actually means `bytes4(calldataload(0))`, so the missing bytes of `msg.sig` (the least significant `(4-len(d))` bytes) in such a scenario will be filled in by the high bits of the sender's address. Can be used to force the contract to use a handler that is not supposed to handle such case.

status - fixed

I-01. Incompatibility of the `getTradeableOrderWithSignature` function with handlers that supports EIP-165

severity: informational

In `getTradeableOrderWithSignature` in the `try ExtensibleFallbackHandler(owner).supportsInterface(type(ISignatureVerifierMuxer).interfaceId)` logic it is not possible to use `EIP-1271 Forwarder` flow (catch branch) in case `owner` supports EIP-165.

status - acknowledged

I-02. Collisions between different orders

severity: informational

In the `GoodAfterTime`, `PerpetualStableSwap`, `TradeAboveThreshold` and `StopLoss` contracts collision between different orders are possible, so there will be no chance to execute both of them.

status - acknowledged

I-03. Griefings on order executions

severity: informational

In the `PerpetualStableSwap` and `TradeAboveThreshold` contracts griefings are possible – it is possible to send small amount of tokens to the `owner` to invalidate the swap order.

status - acknowledged

I-04. Missing check for ERC1271 interface in the `ExtensibleFallbackHandler::_supportsInterface` function

severity: informational

In `ExtensibleFallbackHandler::_supportsInterface` it is missed that `ERC1271` interface is also implemented.

status - fixed

I-05. Short calldata in `_msgSender` function

severity: informational

From the design perspective it will be good to revert the `_msgSender` function execution in case `calldatasize()` is smaller than 20.

status - fixed

I-06. Wrong flow in the `SignatureVerifierMuxer::isValidSignature` function

severity: informational

In the `SignatureVerifierMuxer::isValidSignature` function there is a chance that it was expected to use `defaultIsValidSignature` flow, but the `(sigSelector == SAFE_SIGNATURE_MAGIC_VALUE)`-related checks were successful.

status - partially fixed

I-07. Wrong implementation of `MarshalLib::encodeWithSelector` function

severity: informational

In the `MarshalLib::encodeWithSelector` function main formula should contain `(isStatic ? 0 : (1 << 248))` instead of `(isStatic ? 0 << 248 : 1)`.

status - fixed

I-08. Missing `removeSupportedInterfaceBatch` functionality

severity: informational

It would be useful to have functionality inverse to the setSupportedInterfaceBatch logic.

status - fixed

I-09. Missing check on `_domainSeparator` value in the `ComposableCoW::isValidSafeSignature` function

severity: informational

It is safer from design perspective to check `_domainSeparator` against `domainSeparator` in the `ComposableCoW::isValidSafeSignature`, even through it is checked in `SignatureVerifierMuxer`.

status - acknowledged

I-10. Non-standartly packed calldata in `SignatureVerifierMuxer::isValidSignature`

severity: informational

In `SignatureVerifierMuxer::isValidSignature` `calldataload` and `calldatacopy` are used, but it is not guaranteed that the calldata will be packed in the expected “standard” way. While the calldata encoding can be different (but with the same actual value of the input parameters), the `isValidSignature` will handle it in a wrong way. In the same time it is possible that the real user will be not aware of such substitution, as his UI will show only value of input parameters, not the actual enconding of the calldata.

status - fixed

I-11. Limitations of the `StopLoss` contract functionality

severity: informational

The practical functionality of the `StopLoss` is very limited in case the prices have comparable values. The worst scenario to see this problem is something like “I want to sell my 1 GNO for ETH in case price of GNO/ETH drops under 0.05, I am providing GNO/USD and ETH/USD oracles” – as long as GNO/USD price is smaller than ETH/USD price the `latestSellPrice/latestBuyPrice` will be equal to zero, so there is no way to distinguish 0.05 price from 0.10 price. Of course this will depend on the number of decimal places for answers from the oracles, but the problem generally remains unresolved.

status - acknowledged