Core

Core contracts of CoW Protocol are those that are necessary for the protocol to function.

Architecture

The goal of the protocol is to bundle multiple user orders together to minimize fees deriving from using external liquidity sources and stave off maximal extractable value.

User orders are signed off-chain and sent to the protocolSolvers compete in an auction to settle the orders. Orders without a direct match are settled by the solver using on-chain liquidity sources through the use ofinteractions .

A settlement is a list of orders traded together, their prices, and the on-chain interactions necessary to retrieve external liquidity.

To achieve this, the protocol needs to be able to:

- Collect and verify user orders, interact with on-chain liquidity, store information on the state of the orders
- Determine which addresses are solvers
- Firewall user funds from solvers

Guarantees and Invariants

1. User funds areNOT
2. accessibleexcept
3. for settling an order authorized by the user
4. A tradeMUST
5. be executed at thelimit price or better
6. as specified in the order
7. Once an order is fulfilled, itCANNOT
8. be traded again

Deployments

tip CoW Protocol contracts are deployed using deterministic addresses. This means that the same contract code will always be deployed to the same address. This is done by using theCREATE2 opcode. caution Take care whensigning , ensuring that theEIP-712 domain separator is specified correctly. Contracts Address / Chains GPv2Settlement

Upgradeable: No ☒GitHub 0x9008D19f58AAbD9eD0D60971565AA8510560ab41

-Ethereum mainnet

-Gnosis chain

-Sepolia GPv2AllowListAuthentication

Upgradeable: Yes ⚠ GitHub 0x2c4c28DDBdAc9C5E7055b4C863b72eA0149D8aFE

-Ethereum mainnet

-Gnosis chain

-Sepolia GPv2VaultRelayer

Upgradeable: No ☒GitHub 0xC92E8bdf79f0507f65a392b0ab4667716BFE0110

-Ethereum mainnet

-Gnosis chain

-Sepolia Upgradeable allow-list The allow-list is upgradeable to facilitate future decentralization of the protocol. Reasons to upgrade the allow-list include if CoWDAO decides to:

- add/remove solvers
- change the logic for who is a solver (such as automating a stake requirement)

The security considerations of upgradeability are discussed in theGPv2AllowlistAuthentication section.

Security / Known Issues

Partially fillable orders with zero amounts

If a user creates an order with:

- sellAmount = 0
- buyAmount = 0
- partiallyFillable = true

Then this order could be executed an arbitrary amount of times instead of just a single time. This means that any solver could drain the fee amount from the user until not enough funds are available anymore.

tip Never sign orders withsellAmount = 0 and/orbuyAmount = 0 . If developing smart contracts that create orders, make sure at a contract level that such orders cannot be created.

ERC-1271

Replayability

The security ofERC-1271 signatures depend on the developers' implementation of thesigning smart contract . Different smart contract wallets sign and verify messages very differently from each other.

There's a known issue in some smart-contract wallet implementations that should be known before usingERC-1271 signatures on CoW Protocol.

Signature Replayability

On some popular smart-contract wallets, the signature of an owner is replayable: this means that if an address is an owner of two smart-contract wallets then a signature for the first wallet may also be a valid signature for the second wallet.

CoW Protocol only checks if the signature is valid; it doesn't know which wallet the signature was intended for. This means that an order signed by a wallet with replayable signatures could be executed by another wallet owned by the same owner.

As a general recommendation, you should never sign any message withERC-1271 on wallets whose signature are replayable. CoW Protocol can't protect your order from this attack at a protocol level.

tip Signatures from a Safe wallet aren't replayable and don't require any special handling.

Mitigation

You are still able to createERC-1271 orders from any wallet.

The CoW Protocol API allows you to specify some signedmetadata in the order. This metadata can be sent along when creating an order on the CoW Protocol API.

The signer property in themetadata is used by the CoW Protocol API to check that the signature is meant for the specified wallet. If this doesn't match the order origin, the order is rejected. A minimal example of app data is the following:

{

"metadata" :

{

"signer" :

"0x291cf356e5639fb4c19ba5bfb4e2f8f82b371573"

}

} where0x29..73 is the address of the smart-contract wallet that usesERC-1271 to sign the order.

However, the CoW Protocol API must know thefull metadata in advance for rejecting unknown orders, and by default you don't need to reveal the entire metadata when you submit the order, only its hash (the app data). Once your order is created, the metadata will be permanently stored in the database and linked to the app data; no one will be able to post a replayed order on the CoW Protocol API for a different wallet.

If your signature is valid before your order is submitted and you want to make sure that the no one uses it before your order is submitted on the API, you can send the metadata to the API in advance. Then, any incoming order with the same app

data hash will be rejected if the smart contract wallet doesn't match.

The metadata can be sent in advance to theAPI using thePUT method/app_data/{app_data_hash} .

caution This process only prevents the order from being created in the CoW Protocol API. A malicious solver would still be able to replay a signature when submitting a transaction on-chain. However, as long as the metadata is known, this abuse will be detected upon inclusion of the transaction in a block and the solver will be penalized by the amount of the misused order.

EIP-712 cached domain separator

To maximise gas efficiency thedomain separator for theGPv2Settlement contract is initialized in the constructor and cached for all subsequent invocations. Therefore, any signatures for a chain on which theGPv2Settlement contract has been deployed are replayableon any subsequent fork of that chain (such as ETHPOW forking Ethereum mainnet). Edit this page Previous Solver API Next GPv2Settlement