Compound Open Oracle Audit

FEBRUARY 11, 2020 | IN SECURITY AUDITS | BY OPENZEPPELIN SECURITY

Compound's Open Oracle System is a protocol aiming to provide an on-chain price oracle based on trusted reporting sources. The oracle allows reporters to post prices of assets and calculates the median price out of all reported samples, then providing the median price as the official reference price for the asset.

Audit history and current scope

In this audit we looked into all Solidity contracts inside the Compound Open Oracle System repository. The audited commit is `e7a928334e5e454a88eec38e4ee1be5ee3b13f08`. The exact files included in this scope are:

- contracts/DelFiPrice.sol
- contracts/OpenOracleData.sol
- contracts/OpenOraclePriceData.sol
- contracts/OpenOracleView.sol

All off-chain systems related to the Open Oracle were left out of scope. Moreover, we did not audit the oracle's integration with Compound nor any other project in the ecosystem.

We have previously audited several phases of the Compound Protocol with details below. Yet note that this is our first audit of the Open Oracle System.

1. A subset of Compound's contracts in commit `f385d71983ae5c5799faae9b2dfea43e5cf75262` of Compound's public repository. Read the report's summary.
2. A patch that introduced a time delay for critical admin functions and the ability to pause others. That patch is reflected in

commit `681833a557a282fba5441b7d49edb05153bb28ec` of Compound's public repository. Read the report.

3. A refactor of the core `CToken` contract with the purpose of accommodating underlying tokens that may extract a fee when transferring tokens (e.g., USDT). This refactor is presented in commit `2535734126c7c26e9bc452f27f45c5408acff71f` of Compound's private repository. The report is not available to the public yet.
4. The difference between the code at commit `2535734126c7c26e9bc452f27f45c5408acff71f` of Compound's private repository and commit `bcf0bc7b00e289f9b661a0ae934626e018188040` of their public repository. These changes introduced the ability to handle underlying ERC20 tokens whose implementations can be upgraded (e.g., DAI). The report is not available to the public yet.
5. The difference between commit `bcf0bc7b00e289f9b661a0ae934626e018188040` and commit `9ea64ddd166a78b264ba8006f688880085eeed13` in Compound's public repository. The audit included changes to the `JumpRateModel` contract and the two newly added files `CDaiDelegate.sol` and `DAIInterestRateModel.sol`. Read the report.
6. An alpha version of Compound's Governance System including the Compound Governance Token (COMP) together with the core governance contracts in initial commit `6858417c91921208c0b3ff342b11065c09665b1b`, as well as a follow-up commit `f5976a8a1dcf4e14e435e5581bade8ef6b5d38ea`. The report is not available to the public yet.

High-level overview

Compound's Open Oracle revolves around a consortium of reporters publishing signed prices for a set of assets. This is achieved by signing messages containing information such as asset prices and timestamp. The oracle validates the submitted data and saves it to a storage mapping which holds a struct containing both the asset's price and timestamp. The official Reference Price for each asset is essentially the median of the prices from each trusted reporter, intended to be computed by the oracle whenever a price change from these trusted reporters occurs.

The individual prices reported by each source are always available and verifiable. Prices are attached to a timestamp at the time of signing, and updates are only ever stored if they represent a later time according to their source. Anyone can post prices to the oracle, however only those coming from trusted sources are used to calculate the median price of an asset.

Following we present our findings, in order of importance.

Critical severity

None.

High severity

[H01] Inconsistencies in stored data may lead to incorrect median price

The `DelFiPrice` contract has a public `prices` mapping intended to store the median price for each asset. Following the specification, an asset is referenced by its symbol (a string in uppercase letters) and the corresponding median price is stored as a `uint64` value. Anyone can post prices on chain, but only those signed by trusted sources are taken into account when calculating the median price. There are two ways to post an asset's price on chain.

The first method is via the external `put` function of the `OpenOraclePriceData` contract. Given a signature and the associated message containing the price and the asset's symbol (among other things), the function will first recover the source from the message and signature, and then update the asset's price for this particular source. Price data for each source and symbol is kept in the contract's storage in private `data` mapping. Note that prices updated via this method *do not* trigger a recalculation of the official (median) price hold in the public `prices` mapping of the `DelFiPrice` contract. As a consequence, if at this point a user queries an asset's price reading the `prices` mapping, the returned median price will be incorrect.

Furthermore, if the user queries the asset's price using the `medianPrice` function, the returned median price may be inaccurate as well, since it will not be compared against the anchor price to validate it.

The other way to submit prices is via the external `postPrices` function of the `DelFiPrice`, which allows to report a batch of prices (with associated signatures) of one or more assets in a single transaction. Internally, `postPrices` calls the `put` function. In this case, the `postPrices` function also correctly recalculates the median price of each asset passed in the `symbols` parameter, updating the corresponding entries in the `prices` mapping if the median is within safe bounds. However, if there is a mismatch between the assets referenced in the `symbols` array and the assets which prices were updated, then some median prices will not be recalculated.

According to the specification, *"Whenever a post happens, the official (median) price is recalculated"*. This is not strictly followed by the current implementation, leaving the door open for potential data inconsistencies in the registered prices and their medians. Taking into account that reporting accurate median prices is the fundamental purpose of the system, we understand this to be a High severity issue. Consider applying the necessary modifications to strictly follow the specification. In particular, there should be a single reliable way to post prices on chain, and a single reliable way to query the median price for an asset.

Medium severity

[M01] Potentially inaccurate median price during sudden, extreme, price variations

The "official" price reported by the Open Oracle consists of the median price calculated over a set of individual prices reported by trusted sources. As it is known, the actual prices of most cryptocurrency-related assets are highly volatile, prone to suffer sudden, extreme and unpredictable fluctuations in short periods of time.

Since the prices reported by the Open Oracle are based on the median of a set of individual prices, the oracle needs at least 50% of the individual prices to be accurate for the median to be accurate as well. In an event of a sudden, extreme, asset price variation, the median price reported by the oracle would only reflect the new price once 50% of the sources report on the new price. As a consequence, the Open Oracle may not reflect sudden, extreme, variations in a timely manner. This behavior becomes more problematic at times when the Ethereum network is highly congested, since transactions reporting newer prices will take longer than usual to be mined. Projects relying on the Open Oracle feed must be aware of the associated risks during times of high price volatility, and implement the necessary logic to adapt to potential delays in the prices reported by the oracle.

To better reflect price variations, consider not only calculating the median of the individual prices reported for a given asset, but also computing their standard deviation. This should favor the oracle's transparency, and help projects querying the Open Oracle better understand the actual distribution of the prices, thus allowing them to take more informed decisions.

[M02] Lack of minimum sample size requirement for official price calculation

To obtain the official price for an asset, Compound's Open Oracle takes all reported price data from trusted sources, calculates a median price using the `medianPrice` function of the `DelFiPrice` contract, and finally supplies this median price as the asset's price.

However, this process does not take into account the sample size of the prices when calculating the median price of an asset, which may render the oracle vunerable to a price manipulation attack when the sample size is low (see **[N09] Considerations on rogue sources** note for a more detailed description).

Consider either implementing a minimum sample size requirement or returning the sample size together with the oracle's official price, so as to raise awareness on the potential risks when the sample size is dangerously low.

[M03] Function put allows storing invalid signed data

The `put` function of the `OpenOraclePriceData` is used to submit signed price data. It first recovers the signing address from the given signature and message by means of the inherited `source` function, and then uses the recovered address (i.e., the source) to update the price and timestamp under the corresponding source address and asset symbol.

The inherited `source` function uses the `ecrecover` precompile to recover the signing address. If there is an error during the recovering process, `ecrecover` will return the zero address (as stated in its documentation). Since the returned address is never validated in the `put` function, this could lead to submitting invalid data with a successful transaction. While this does not pose a security risk, since data would be written in an entry corresponding to the zero address, the function should fail earlier to favor a more predictable behavior.

Consider ensuring the address returned by the `source` function is not the zero address before continuing execution in the `put` function.

[M04] Oracle might fail if anchor price is not updated fast enough

The `DelFiPrice` contract implements a safety mechanism to disregard reports that, while coming from trusted sources, may lead to a drastic variation of the official median price reported by the oracle. The mechanism is based on an anchor price (reported by a trusted `anchor` address), together with upper and lower bounds that calculated median prices are checked against. Should the ratio between a new median price and the anchor price be out-of-bounds, the official median price of the asset would not be updated. Both the address that reports the anchor price and the tolerance allowed between the anchor and the median are set during construction.

In an scenario of high price volatility, a trusted source may report a sharp variation of an asset's price. As a consequence, the newly calculated median price may easily fall out of the acceptable range defined by the anchor price. Unless the anchor value is rapidly updated by the trusted `anchor` address, the oracle will never accept the new median price, thus causing the oracle to fail.

The address in charge of reporting the anchor price plays a fundamental role in the Open Oracle system, particularly in times of steep price variations. While the way this address is managed is out of this audit's scope, we strongly advise closely monitoring price fluctuations and trends, so as to be able to quickly update the anchor price record and ensure the oracle prices can always be correctly updated.

Low severity

[L01] Commented out code

The `OpenOracleData` contract includes commented out lines of code without giving developers enough context on why those lines have been discarded, thus apparently providing them with little to no value at all.

As the purpose of these lines is unclear and may confuse future developers and external contributors, consider removing them from the code base. If they are to provide some sort of documentation of an interface, consider extracting them to a separate document where a deeper and more thorough explanation could be included.

[L02] Lack of indexed parameters in events

None of the parameters in the events defined in the `DelFiPrice` contract are indexed. Consider indexing event parameters to avoid hindering the task of off-chain services searching and filtering for specific events.

[L03] Incomplete docstrings

- Docstrings for the `postPrices` function of the `DelFiPrice` contract are missing documentation for the `symbols` parameter. Note that they should state the format in which symbols are expected to be submitted (i.e., in uppercases according to the specification).
- Docstrings in the `OpenOracleView` contract should include a link to the "Open Oracle standard specification" mentioned.
- The `put` function of the `OpenOraclePriceData` contract expect the `message` parameter to be formatted in a specific way that is not fully documented in the parameter's docstrings. In particular, it is never mentioned that `message` must contain the string "prices" to be valid.

[L04] Use of magic constants

There are several occurrences of magic constants in the `DelFiPrice` contract. See for example lines 38, 39, 40 and 65, where the value `100e16` is used.

Consider defining a constant state variable for every magic constant, giving it a clear and self-explanatory name. For complex values, consider also adding an inline comment explaining how they were calculated or why they were chosen. All of this will allow for added readability, easing the code's maintenance.

Notes & Additional Information

[N01] Inconsistent coding style for function parameters

There is an inconsistent use of underscores in the names of function parameters. While some parameters include an underscore and the end of their names, others do not. To favor readability throughout the code base, consider always following a consistent coding style.

[N02] Redundant assignment of sorted price array

The `medianPrice` function of the `DelFiPrice` contract calls the private `sort` function to sort the `postedPrices` in-memory array, then assigning the result of the operation to a new `sortedPrices` array. In Solidity, arrays are reference types, so the `postedPrices` array will be sorted after calling the `sort` function. Therefore, for a more gas-efficient execution, consider removing both the return statement from the `sort` function and the unnecessary assignment to the `sortedPrices` array.

[N03] Undocumented system design for untrusted price sources

The Open Oracle is designed to allow anyone to submit asset prices by calling the `postPrices` function of the `DelFiPrice` contract or the `put` function of the `OpenOraclePriceData` contract. However, only prices reported from trusted sources, defined in the `sources` array, are taken into account to calculate the official median price of an asset. This system design renders prices reported from untrusted reporters pointless, though at the same time allows them to submit prices. While this seems to be inline with the system's intended behavior, a clearer explanation addressing this (either in code or external documentation) might be beneficial for users, developers and auditors alike.

[N04] Lack of explicit visibility in state variables

State variables `anchor`, `upperBoundAnchorRatio` and `lowerBoundAnchorRatio` of the `DelFiPrice` contract are implicitly using the default visibility. To favor readability, consider explicitly declaring the visibility of all state variables.

[N05] Naming

- The `data` mapping of the `OpenOraclePriceData` contract should be renamed with a more self-explanatory name that states what kind of data the mapping stores. Similarly, the `get` and `put` functions can also benefit from more self-explanatory naming.
- The `data` address of the `OpenOracleView` contract should be renamed to clearly denote it refers to a contract address.
- The `source` function of the `OpenOracleData` contract should be renamed to `getSource`, `getSourceFromMessage` or similar.
- The `PriceUpdated` event of the `DelFiPrice` contract should be renamed to `MedianPriceUpdated`.
- The `prices` mapping of the `DelFiPrice` contract should be renamed to `medianPrices`.

[N06] Declare uint as uint256

The project's code often uses `uint` as variable type. See for example lines 36, 53 and 58 of the `DelFiPrice.sol` contract.

To favor explicitness, all instances of `uint` should be declared as `uint256`.

[N07] Unnecessary casting operations

In line 54 of `DelFiPrice.sol`, the `data` state variable is first casted to `address` and then to the `OpenOraclePriceData` type. However, `data` is already stored as an `OpenOraclePriceData` type. Therefore, consider removing the unnecessary casting operations.

[N08] Unused named return variable

The `medianPrice` function of the `DelFiPrice` contract declares a named return variable called `median`. Given that it is not used, consider removing this variable.

[N09] Considerations on rogue sources

According to its specification, Compound's Open Oracle relies on a consortium of trusted sources that make asset prices available on known https endpoints. Anyone can query these public web endpoints, obtaining the signed data that is later expected to be posted on Ethereum smart contracts. Anyone can post data on chain, but only data signed by the trusted sources will be considered when calculating the median price for an asset.

Each source will have an associated Ethereum address under which their prices will be registered on chain. The number of sources is not enforced in any way at the smart contract level. More interestingly, the array holding all trusted sources' addresses is set at deployment and cannot be later modified.

The oracle's "official" price for an asset is the median of a set of prices reported by the trusted sources. Therefore, it only takes 50% of the sources to go rogue in order to manipulate the oracle's price at will. The cartelized sources will have total control of the oracle's official price, even if the remaining 50% of the sources keep reporting the real prices of an asset. It is unclear how many sources the Open Oracle will trust, but the lower the number of sources, the easier it would be to conduct this attack. While a conglomerate of malicious sources may sound unlikely at first, one should take into account that the incentives for a set of sources to take control over the oracle increase with the number of projects fully relying on the oracle's prices. If initially trusted sources happen to go rogue and manipulate prices before the Compound team notices, all on-chain finance systems trusting the Open Oracle may suffer severe financial losses.

The Compound team is aware of the potential risk of rogue sources, and according to the specification *"The consortium operates on a trust-but-verify principle. If reporters use timestamps too far in the future or report prices which deviate too far from the norm, they will likely be voted out of the Compound Reference Price view"*. At the time of the audit there is no governance mechanism in place to "vote out" a trusted source. As a consequence, the only available way to replace sources is to re-deploy the `DelFiPrice` contract with a new set of sources.

So as to rapidly detect and mitigate any malicious behavior, we strongly advise closely monitoring the prices and timestamps posted on chain, tracking the `Write` event triggered after every price update. Furthermore, consider outlining a response and recovery plan so as to act quickly in case an attempt to compromise the Open Oracle is detected.

Conclusions

No critical and one high severity issue were found. Some changes were proposed to follow best practices and reduce potential attack surface.

Security Audits

- If you are interested in smart contract security, you can continue the discussion in our forum, or even better, join the team 🚀
- If you are building a project of your own and would like to request a security audit, please do so here.

RELATED POSTS

SECURITY AUDITS

Compound Finance