

OEV Auction House: API3

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	OEV auctions	Documentation quality	Medium	<div><div></div></div>
Timeline	2023-12-08 through 2023-12-20	Test quality	High	<div><div></div></div>
Language	Solidity	Total Findings	19	<div><div></div><div>Fixed: 6 Acknowledged: 10 Mitigated: 3</div></div>
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review	High severity findings ⓘ	1	<div><div></div><div>Acknowledged: 1</div></div>
Specification	README file OEV litepaper ↗	Medium severity findings ⓘ	3	<div><div></div><div>Fixed: 1 Acknowledged: 1 Mitigated: 1</div></div>
Source Code	<ul style="list-style-type: none">api3dao/oev-auction-house ↗#803aa3a ↗	Low severity findings ⓘ	8	<div><div></div><div>Fixed: 2 Acknowledged: 4 Mitigated: 2</div></div>
Auditors	<ul style="list-style-type: none">Valerian Callens Senior Auditing EngineerRoman Rohleder Senior Auditing EngineerAdrian Koegl Auditing Engineer	Undetermined severity findings ⓘ	1	<div><div></div><div>Acknowledged: 1</div></div>
		Informational findings ⓘ	6	<div><div></div><div>Fixed: 3 Acknowledged: 3</div></div>

Summary of Findings

This audit focused on the OEV (oracle extractable value) Auction House platform where OEV searchers can bid on data feed updates that satisfy specific conditions, and report that they have fulfilled these updates. OEV is a subset of MEV where oracles have exclusive priority of extraction. API3 holds OEV auctions for its data feed services and forwards the proceeds to the respective user dApps.

Most of the issues identified by the audit team involve risks that could occur depending on how the codebase in scope interacts with other system components. The contracts in scope were well-written, and we exhibited no critical issues. Also, we consider that clearer guidelines should be provided to actors interacting with the system. Finally, the API3 team enumerated self-identified issues in their documentation, which illustrates their risk awareness.

It should be highlighted that searchers must fully rely on the honesty and reactivity of the auctioneers, who select the bids to award based on arbitrary rules. These off-chain components were out-of-scope for this audit.

Regarding the project quality, the test suite's quality is high, with a test coverage of 100% and a mutation testing score (SuMo) of 87%, which is above the industry standards. The contracts are well-documented, but more contextual and integration details would help external observers better understand how the codebase is supposed to integrate with the whole system.

Our team frequently interacted with the API3 team to clarify the code, expected behavior, and intended interactions with other system components. Their active engagement in answering our questions was crucial and greatly assisted in completing the audit.

Fix Review Update

The API3 team has either fixed, mitigated, or acknowledged the issues in this report. Also, it correctly pointed out that some of the issues identified are not in the scope of this audit. The test suite was updated accordingly, still resulting in a test coverage of 100%. The documentation was improved with integration details and clarifications about how the system should be used.

ID	DESCRIPTION	SEVERITY	STATUS
OEVA-1	Risk of Delayed Oracle Updates Impacting Liquidation Mechanisms	• High ⓘ	Acknowledged
OEVA-2	Amount to Lock Is Not Known when Placing the Bid	• Medium ⓘ	Fixed
OEVA-3	Centralization, Privileged Roles and Off-Chain Dependency	• Medium ⓘ	Mitigated
OEVA-4	Lack of Auctioneer Incentive Compatibility	• Medium ⓘ	Acknowledged
OEVA-5	Current Two-Step Mechanism for Withdrawals Can Be Used to Disrupt the System	• Low ⓘ	Acknowledged
OEVA-6	Protocol Fees Are Not Charged when a Bid Is Contradicted	• Low ⓘ	Acknowledged
OEVA-7	Unsafe Casts May Lead to Overflows and Incorrect Collateral and Protocol Fee Calculations	• Low ⓘ	Fixed
OEVA-8	Risks Related to Using Price Feed Oracles	• Low ⓘ	Mitigated
OEVA-9	Missing Input Validation	• Low ⓘ	Mitigated
OEVA-10	Complexity in Risk Management and Potential Strategic Bidding Vulnerabilities	• Low ⓘ	Acknowledged
OEVA-11	Integration Requirements for Smart Contracts Are Not Documented	• Low ⓘ	Fixed
OEVA-12	Slashing Mechanism Can Be Bypassed Through Backstage Collusion	• Low ⓘ	Acknowledged
OEVA-13	Risks Related to Off-Chain Actions Triggered by On-Chain Events	• Informational ⓘ	Fixed
OEVA-14	Risks Related to Reorganizations	• Informational ⓘ	Fixed
OEVA-15	OEV-Impacted Parties May Themselves Place Bids and Not Fulfill Them to Buy Time	• Informational ⓘ	Acknowledged
OEVA-16	Awarded Oracle Updates May Still Be Front-Run via Classical MEV	• Informational ⓘ	Fixed
OEVA-17	Any Auctioneer Can Update the State of Any Bid	• Informational ⓘ	Acknowledged
OEVA-18	Hash Collisions Are Possible for the Bid IDs	• Informational ⓘ	Acknowledged
OEVA-19	The Granularity Level of Application of collateralInBasisPoints May Not Be Optimal	• Undetermined ⓘ	Acknowledged

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

i Disclaimer

Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
 1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
 2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
 1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

The scope of this audited was limited to the following contracts at commit `803aa3addb6ef4c711018374577769cf5cad2943` :

- `interfaces/IOevAuctionHouse.sol`
- `OevAuctionHouse.sol`

Findings

OEVA-1

Risk of Delayed Oracle Updates Impacting Liquidation Mechanisms

• High ⓘ Acknowledged

i Update

We agree that this issue does not directly relate to the audited scope.

While we understand that it is a design trade-off to capture as much OEV as possible in case of a significant change to a data feed value, we recommend running simulations of extreme market conditions to assess if the protocols could be severely impacted (i.e., insolvency) as a result of the delay incurred through oracle update and/or auctioning. If yes, the client should evaluate the detailed factors in which this delay could negatively impact the protocol and reduce the impact during such market conditions.

Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

This issue does not refer to the contract implementation, but rather README.md in the oev-auction-house repo, which reads: "To ensure effective OEV extraction, API providers should apply a slight delay (e.g., 15 seconds) to their (non-OEV) updates..." The rest of the paragraph explains why doing so has no practical downside in theory, assuming OEV updates will be available and used as necessary. This issue points out that if we disregard OEV updates, the delay may have downsides, which does not contradict our argument. The recommended strategy is to forgo the OEV capture opportunity whenever the data feed value changes significantly. However, a significant change to a data feed value is typically what causes an OEV capture opportunity, and thus forgoing such cases would result in not being able to capture the majority of the total OEV. It is safe to assume that any foregone OEV opportunity will be capitalized on by an MEV exploit, which causes the user DeFi protocol to lose a well-defined and typically significant amount of funds in practice. Therefore, the recommendation here is to subject the user DeFi protocol to constant exploits in an effort to eliminate a type of risk whose relevance is inherently ambiguous because it depends on supposed circumstantial factors. For protocols, smart contract security is ultimately a means to prevent loss of funds. Any design choice that forgoes OEV capture causes loss of funds in practice, and capturing as much OEV as possible is a security objective as much as a financial objective. Our target users are DeFi protocols that leak a significant amount of OEV, for which opting for the risk–return tradeoff of maximal OEV capture is the rational choice.

File(s) affected: `OevAuctionHouse.sol`

Description: A key feature of this system is the intentional delay (e.g., 15 seconds) in API provider updates of type 1 to give priority to updates won in the auction (type 2). While this delay is designed to optimize OEV extraction, it introduces a potential risk in the context of rapidly changing market conditions.

This delay can have a critical impact on, for example, DeFi protocols, especially those that involve collateralized loans or require real-time liquidation mechanisms. In scenarios of high market volatility, the oracle's delayed price feed might not reflect accurately the real-time market prices. Consequently, if a sharp market downturn occurs, the liquidation mechanisms may not trigger in time, potentially leading to under-collateralized loans and significant financial losses for the protocol.

This issue is further compounded if multiple protocols rely on the same oracle system, potentially leading to systemic risks in the blockchain ecosystem.

Recommendation: Develop a fallback mechanism that can be triggered in the event of significant price discrepancies. This could include updates of type 1 being available in real-time without any delay if a certain ratio of price drop occurs.

OEVA-2 Amount to Lock Is Not Known when Placing the Bid • Medium ⓘ Fixed

Update

Bidders can now use two additional function parameters to express what maximum amount they tolerate to be locked for their bid (`maxCollateralAmount` and `maxProtocolFeeAmount`). Now bidders' funds are protected since only an accepted amount can be locked.

Update

Marked as "Fixed" by the client. Addressed in: `4c2aba12c476cbd614813a62fa171f54ba334e9e` . The client provided the following explanation:

We extended the recommended solution by requiring the bidder to specify `maxCollateralAmount` and `maxProtocolFeeAmount` separately, as these values have different implications.

File(s) affected: `OevAuctionHouse.sol`

Description: OEV searchers can place bids for a given `bidAmount` via the function `placeBidWithExpiration()` and `placeBid()` . When these functions are executed, a `Bid` object is recorded on-chain with two values representing the amount to be locked:

- `bid.collateralAmount` , proportional to the current value of the system parameter `collateralInBasisPoints` ;
- `bid.protocolFeeAmount` , proportional to the current value of the system parameter `protocolFeeInBasisPoints` ;

Even if the value of `protocolFeeInBasisPoints` cannot be higher than the hardcoded value `MAXIMUM_PROTOCOL_FEE_IN_BASIS_POINTS` , there is no limit for the maximum value of `collateralInBasisPoints` . As a

result, a searcher does not know the underlying amount to be locked when the bid is awarded when placing the bid, and the scenario described below is possible.

However, the likelihood of this scenario is unlikely because only the manager can change `collateralInBasisPoints` and `protocolFeeInBasisPoints`.

This issue could occur if incorrect rates are provided by oracles when the function `getCurrentCollateralAndProtocolFeeAmounts()` is executed.

Exploit Scenario: The initial system can be described as follows:

- `collateralInBasisPoints == 10000` ;
 - `protocolFeeInBasisPoints == 0` ;
 - Searcher has a deposit of `500` ;
1. Searcher places a bid of `100` and expects `100%` of this amount to be locked, meaning `100` ;
 2. Within the same block, a transaction with a higher priority is executed to update `collateralInBasisPoints` to `50000` ;
 3. The block is executed and the bid is placed with an amount to lock of `500` ;
 4. If that bid is awarded by an auctioneer, an amount of `500` is locked;

Recommendation: Consider adding to the functions `placeBidWithExpiration()` and `placeBid()` a parameter `maxAcceptableAmountLocked` to let the bidder control the maximum amount to be locked for that bid.

OEVA-3

Centralization, Privileged Roles and Off-Chain Dependency

• Medium ⓘ Mitigated

i Update

As fixes, the API3 team added to the readme file a description of the roles and the associated privileges in the system.

i Update

Marked as "Acknowledged" by the client. Addressed in: `6276c0db2606ebe50126b27a52267b73042385b9`. The client provided the following explanation:

As a general note, we add the requested documentation in README.md because we do not have any other documentation platform yet.

i Update

Prior to the audit, the API3 team enumerated in their documentation (README file) the following risks and security considerations:

- A bidder may want to deny service to other searchers by winning a bid but not execute the underlying update on the target chain. For that, a slashing mechanism has been implemented.
- Actors of the system could misbehave. For instance, an auctioneer not awarding the winning bid to the best bid. For that, the system expects on-chain monitoring by external actors to identify improper behavior and update the reputation of such actors.
- If a searcher wins a bid but the underlying OEV opportunity is no longer valid, the searcher will have to decide between executing the update and paying the bid amount (plus the transaction fees), or skipping the update and being slashed by the collateral amount.
- An auctioneer bot may award an update to multiple bids of the same bidder because the update matches all bids. In this case, the associated logic defining for instance the amount of collateral to be locked may be defined through pre-communicated rules.
- Locking up bidder funds at bid-time was not an option preferred, because it would reduce their capital efficiency greatly.
- The auctioneer bot is trusted to facilitate the auction honestly. This enables the following unwanted scenarios:
 - It can deny service (selectively or to everyone) by not awarding bids and not confirming fulfillment;
 - It can contradict fulfillments that have been correctly reported;
 - It can award bids that should have been beaten by competitors;
 - It can provide award details that are not valid;

Based on the fact that these scenarios are possible, the team listed the following mitigations measures:

1. any misbehavior should be provable because operations are recorded on-chains.
2. collateral unjustly slashed is available to the manager multi-sig, and this issue is intended to be resolved retrospectively by the multi-sig based on the on-chain records through an off-chain dispute resolution mechanism.

File(s) affected: `OevAuctionHouse.sol`

Description: The system expects to work by granting privileged roles to different addresses. Such roles may pose a risk to end-users. The following lists these roles:

The root role, only owned by the immutable address `manager` , can:

- Grant `adminRole` and `auctioneerRole` to any address;
- Revoke `adminRole` and `auctioneerRole` from any address;

The immutable address `manager` , defined when the contract is deployed, can:

- Update the collateral ratio requirement for bids to any `uint256` value via the function `setCollateralInBasisPoints()` ;
- Update the protocol fees ratio for bids to any value within the range `[0-1000]` (corresponding to `[0%-10%]`) via the function `setProtocolFeeInBasisPoints()` ;
- Update the address of the `collateralRateProxy` via the function `setCollateralRateProxy()` . This contract is used to obtain the price of the native currency of the chain where the contract is deployed;
- Update the address of the `chainIdToNativeCurrencyRateProxy` for a given `chainId` via the function `setChainNativeCurrencyRateProxy()` . This contract is used to obtain the price of the native currency of the chain identified by `chainId` where OEV should be extracted by executing an update;
- Withdraw slashed collateral accumulated by the contract via the function `withdrawAccumulatedSlashedCollateral()` ;
- Withdraw protocol fees accumulated by the contract via the function `withdrawAccumulatedProtocolFees()` ;

The `auctioneer` role, granted by the manager, can:

- Renounce the role (**and thereby prevent any future calls to the following listed functions, until re-appointed by the `manager`**) via the function `renounceRole()` ;
- Award a pending bid and lock up corresponding protocol fees and collateral via the function `awardBid()` ;
- Confirm an awarded bid fulfillment and release locked collateral via the function `confirmFulfillment()` ;
- Deny an awarded bid fulfillment and slash corresponding collateral via the function `contradictFulfillment()` ;
- Extract OEV opportunities for itself;

Note: Core operations of this protocol like bid award selection are controlled off-chain and centrally guarded by auctioneers according to arbitrary rules defined and shared off-chain with bidders. These off-chain parts were out-of-scope for this audit but users should be aware of above mentioned risks.

Recommendation: These roles and the associated privileges should be documented to users. Also, key management should follow the latest best practices in security.

OEVA-4 Lack of Auctioneer Incentive Compatibility

• Medium ⓘ

Acknowledged

i

Update

Marked as "Acknowledged" by the client. Addressed in: `6276c0db2606ebe50126b27a52267b73042385b9` . The client provided the following explanation:

The auction service is an essential component of the protocol, which implies that the protocol fee includes the auctioneer incentive. The contract allows multiple auctioneer accounts, but that is for practical reasons (e.g., to be able to use multiple EOAs to act as auctioneers) and shouldn't be interpreted as multiple entities acting as auctioneers independently from the protocol (which is represented by the manager). This is documented in README.md as a part of the fix to [OEVA-3](#).

File(s) affected: `OevAuctionHouse.sol`

Description: In the current mechanism design, the auctioneer has no incentive to behave honestly other than incentive by reputation. On the contrary, the auctioneer has to pay gas fees for any transaction and is, therefore, disincentivized to behave honestly. Furthermore, as mentioned in a separate issue, the auctioneer is incentivized to extract OEV opportunities they can identify instead of allowing the searchers to execute an awarded update. Nevertheless, the auctioneer is not incentivized to behave dishonestly by unjustly slashing bidders through `contradictFulfillment()` because the rewards go to the manager multi-sig account. This mechanism design inevitably leads to the fact that the auctioneer must be operated by a centralized entity and cannot be decentralized.

Recommendation: We recommend re-designing the mechanism such that the auctioneer is incentivized to behave honestly, e.g., by receiving some fixed fee from the bidder. Furthermore, they should be disincentivized to behave maliciously, e.g., through the possibility of being slashed. However, such a disincentive may be hard to implement because a counterparty would have to prove dishonest behavior.

OEVA-5

Current Two-Step Mechanism for Withdrawals Can Be Used to Disrupt the System

• Low ⓘ

Acknowledged

Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

It is already documented that the auctioneer should not attempt to award bids placed by bidders that have an active withdrawal (in case the award transaction does not get confirmed before the bidder withdraws), which we believe to be an adequate measure by itself. To go over the listed cases that are pointed out to be potentially problematic:

1. We see no reason to prevent a withdrawing bidder from depositing or being deposited for.
2. We see no reason to prevent a withdrawing bidder from placing more bids, which will not be awarded, only to activate them later by canceling the withdrawal. Furthermore, even if we disallowed this, one would be able to multi-call to cancel the withdrawal, place a bid and reinitialize the withdrawal to reach a similar effect.
3. Canceling the withdrawal is implemented as a feature to allow the bidder to revert their withdrawal initiation, which is intended to make their bids eligible to be awarded again. To address the second point, if `awardBid()` required the bidder to not have initiated a withdrawal, bidders would have been able to grief awarding transactions by front-running them with a withdrawal initiation, which defeats the purpose of the two-step withdrawal. Therefore, it should be allowed to award bids placed by bidders that have initiated a withdrawal, as in the original implementation.

File(s) affected: `OevAuctionHouse.sol`

Description: According to the documentation: "a two-step withdrawal process is implemented to prevent the bidders from frontrunning bid awarding transactions with withdrawals to deny service".

However, in practice, a searcher can initiate a withdrawal to pause all his active bids (since auctioneers will not award them once a withdrawal is initiated). Also, he will still be able to perform the following operations:

1. deposit more funds via the functions `deposit()` and `depositForBidder()`;
2. place more bids via the functions `placeBidWithExpiration()` and `placeBid()`;
3. cancel this withdrawal via the function `cancelWithdraw()` to make his pending bids valid again, which can be seen as a way to use it as a bid-awarding temporary locker;
4. execute a withdrawal via the function `withdraw()`;

Finally, the fact that auctioneers should only call `awardBid()` if the searcher has no ongoing initiated withdrawal is not enforced on-chain.

Recommendation: Consider adding on-chain restrictions to limit the capabilities of a searcher to use the withdrawal process as a way to disrupt the system. Also, consider giving guidelines to auctioneers about how to treat the bids of searchers with an initiated withdrawal.

OEVA-6

Protocol Fees Are Not Charged when a Bid Is Contradicted

• Low ⓘ

Acknowledged

Update

Marked as "Acknowledged" by the client. Addressed in: `52af57c8b5fa10dbfc8c0607931b90bd3eef744a`. The client provided the following explanation:

The objective of the protocol should be to maximize OEV capture to align its incentives with the ones of its users. The fulfillment being contradicted is an admission of the fact that OEV capture was unsuccessful, in which case it would be more appropriate to waive the protocol fee. This is intended to incentivize the protocol to improve itself by implementing better/more granular collateral requirements, and possibly a reputation system. While not addressing this issue, we implemented a relevant optimization. Since the only possible outcomes are to slash the collateral or charge the protocol fee, we do not need to lock the sum of them, and we can only lock the larger of the two instead. This results in locking less funds per awarded bid, improving searcher capital efficiency.

File(s) affected: `OevAuctionHouse.sol`

Description: When a bid is awarded, auctioneers can either:

- confirm its fulfillment. In that case, the collateral is released but the protocol fees are charged;

- contradict its fulfillment. In that case, the collateral is slashed but the protocol fees are released; If we consider a situation where `collateralInBasisPoints < protocolFeeInBasisPoints`, contradicting a fulfillment will lead to charging less than the protocol fees, even if the protocol was used by the searcher because he had to consider this bid and pay transaction fees to award it and contradict it.

Recommendation: Consider adding a check in the function `contradictFulfillment()` to always charge at least the protocol fees corresponding to the bid amount.

OEVA-7

Unsafe Casts May Lead to Overflows and Incorrect Collateral and Protocol Fee Calculations

• Low ⓘ Fixed



Update

`SafeCast` is now used as suggested.



Update

Marked as "Fixed" by the client. Addressed in: `732337af254c207fcc7734e400955e4b3d3d267d`.

File(s) affected: `OevAuctionHouse.sol`

Description: The function `getCurrentCollateralAndProtocolFeeAmounts()` computes the required collateral and protocol fee amounts, given current oracle values and preset basis points percentages. To do so, it performs several unsafe cast operations:

```
collateralAmount = uint104(
    (bidAmount *
      uint256(int256(nativeCurrencyRateValue)) *
      collateralInBasisPoints) /
      uint256(int256(collateralRateValue)) /
      HUNDRED_PERCENT_IN_BASIS_POINTS
);
```

While the inner cast operations are safe (`uint256(int256(...))`) due to previous checks on `nativeCurrencyRateValue` and `collateralRateValue` being positive values, the outer downcasts using `uint104()` may silently truncate the resulting computation, returning lower collateral and protocol fee amounts. The same applies to the computation of `protocolFeeAmount`.

Recommendation: Consider replacing the unsafe cast operation `uint104(...)` with, e.g., [OpenZeppelins SafeCast library](#) to prevent any such potential truncations.

OEVA-8 Risks Related to Using Price Feed Oracles

• Low ⓘ Mitigated



Update

The risk of an incorrect price negatively impacting Bidders is now prevented by the fix of [OEVA-2](#), and additional guidelines were added to the readme file. However, two risks remain:

- an incorrect price positively impacting Bidders;
- a data feed that reverts would make it impossible to place bids;



Update

Marked as "Fixed" by the client. Addressed in: `30879df3d0ba7738e31ea39cbad524455ee8afd6`. The client provided the following explanation:

Searchers should get their price data from a trusted source and only send a bid placing transaction if the `getCurrentCollateralAndProtocolFeeAmounts()` response is consistent with their expectation. Combined with the fix to [OEVA-2](#), this eliminates all risk related to data feed usage.

File(s) affected: `OevAuctionHouse.sol`

Description: The system uses two price feeds in the function `getCurrentCollateralAndProtocolFeeAmounts()` to calculate the amount of collateral (denominated in the native currency of the current chain) required for a bid amount (denominated in the

native currency of the target chain). This leads to several risks:

1. Extreme prices can be provided due to an error, a price manipulation attack, or extreme market conditions;
2. The number of decimals used can differ between both price feeds;
3. A stale price can be provided if the internal parameters to trigger a price feed update are not adapted to its use case (i.e. maximum deviation or heartbeat thresholds' value too high);
4. The price feed can revert;

These scenarios would prevent searchers from placing bids or let them place bids with incorrect values for the protocol fees and the associated collateral.

Note that two additional mitigation measures are enforced and make sure that:

- the price returned is not lower and equal to 0 ;
- the last update of the price returned was recorded less than MAXIMUM_RATE_AGE seconds ago (1 day).

Recommendation: Consider adding measures to reduce the likelihood of these scenarios, or failure with a dedicated custom error if they occur.

OEVA-9 Missing Input Validation

• Low ⓘ Mitigated

Update

Item 1 was acknowledged and underlying risks were documented. Item 2 was fixed. Item 3 was acknowledged.

Update

Marked as "Fixed" by the client. Addressed in: 8394f18dd3320e1e5331efd588dfc0c3f604015a . The client provided the following explanation:

1. bidTopic and bidDetails are determined using an arbitrary convention communicated off-chain between auctioneers and searchers. Furthermore, if the chainId of a bid to be placed is congruent with the bidTopic and bidDetails may depend on off-chain information. Therefore, OevAuctionHouse cannot confirm that bidTopic , chainId and bidDetails are valid. We find it acceptable for the funds to be locked when the contract is not used in the intended way. We documented in README.md that such bids may be slashed.
2. The example bidDetails in our unit tests is 305 bytes long. The awardDetails , which is an Api3ServerV1.updateOevProxyDataFeedWithSignedData() calldata for a Beacon set of 21 Beacons, would be 6336 bytes long. The fulfillmentDetails , which is a transaction hash, would be 32 bytes long. Keeping in mind that these are examples and are not definitive, we enforced two limits, one for the bidders (1024 bytes for bidDetails and fulfillmentDetails) and one for the auctioneers (8192 bytes for awardDetails).
3. It is also possible to bid significant amounts for unrealistic conditions (e.g., ETH/USD to be less than 1), which increases the number of bids that have to be considered with no benefit to the auctioneer. We intend for the transaction fees (for bid placement) and the maximum bid lifetime to mitigate this issue, in which case we wouldn't need a separate spam protection mechanism. In the case that the bid amount does not justify the cost of awarding the bid, the auctioneer can simply not award the bid. Such a minimum bid limit can be documented off-chain, it does not need to be enforced on-chain.

File(s) affected: OevAuctionHouse.sol

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error:

1. No check in the function placeBidWithExpiration() checks that the values of bidTopic and chainId match an existing bidTopic . If the bid is awarded, it could result in funds locked temporarily or infinitely, depending if an auctioneer accepts to confirm or contradict its fulfillment.
2. No check in the function placeBidWithExpiration() checks that the size of bidDetails is too long. The same remark applies to the functions awardBid() with the size of awardDetails and reportFulfillment() with the size of fulfillmentDetails .
3. No check in the function placeBidWithExpiration() checks that the bidAmount is high enough. It makes sure that it is a non-zero amount, but bids of 1 wei can be placed for instance, increasing the number of bids to be considered by auctioneers.

Recommendation: We recommend adding the relevant checks.

OEVA-10

Complexity in Risk Management and Potential Strategic Bidding Vulnerabilities

• Low ⓘ Acknowledged



Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

Our goal is for the auctioneer bot to operate at near real-time, which should allow the searchers to not be required to take its behavior specifics into consideration. We will document any relevant idiosyncrasies as its implementation matures.

File(s) affected: `OevAuctionHouse.sol`

Description: As the protocol developers have pointed out, the bidders should incorporate the risk of losing the OEV opportunity into their bidding strategy. However, the protocol presents a level of complexity and potential vulnerability in its bidding process due to the dynamic and non-transparent nature of auction timing.

The core of the issue lies in the absence of a well-defined, transparent framework for the timing of bid evaluations and awards by the auctioneer bot. This lack of clarity can significantly impact the ability of bidders (searchers) to effectively incorporate the risk of losing an OEV opportunity into their bidding strategy.

Recommendation: Consider documenting more transparent guidelines or communication regarding the auctioneer bot's operational cycle and the factors influencing bid evaluation timing. This could involve providing statistical averages, typical scenarios, or more explicit operational details.

OEVA-11

Integration Requirements for Smart Contracts Are Not Documented

• Low ⓘ

Fixed



Update

Integration requirements were documented in the readme file.



Update

Marked as "Fixed" by the client. Addressed in: `584872b213fdb55e2d9396fa3d7b30331ac0422a` .

File(s) affected: `OevAuctionHouse.sol`

Description: When an address makes a direct or delegated deposit to the system, native tokens are transferred to the contract, and its internal balance is increased accordingly. Then, it may want to:

- place a bid by executing the functions `placeBid()` or `placeBidWithExpiration()` ;
- report a fulfillment by executing the functions `reportFulfillment()` ;
- withdraw these tokens by executing the functions `initiateWithdrawal()` and `withdraw()` ;

If the address is an EOA, it can trigger all these functions. However, if it is a smart contract, it should be able to interact with these functions and receive native tokens. Right now, that aspect is not documented.

Recommendation: Consider documenting integration requirements for smart contracts.

OEVA-12

Slashing Mechanism Can Be Bypassed Through Backstage Collusion

• Low ⓘ

Acknowledged



Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

`accumulatedProtocolFees` and `accumulatedSlashedCollateral` are accounted for separately. `accumulatedProtocolFees` is paid to the protocol for fulfilling its promise of OEV extraction, which will be distributed to stakeholders (to API3 for the operation, and to API providers for the data). In contrast, `accumulatedSlashedCollateral` would ideally be zero, but in the case that it is not, it will be used for (protocol) public good. Separately from that, the OEV beneficiary at the target chain receives the bid amount in the case that the update is executed. It would be safe to assume that all recipients will be adequately incentivized to monitor for misbehavior that will upset the balance between these three pools of funds. Furthermore, under-penalizing searchers would incentivize them to be less conservative about placing bids that they may not execute, which would reduce the total OEV captured, affecting all parties (except the colluding

ones) adversely, which is another incentive for all stakeholders to monitor for such misbehavior. We acknowledge the need for building tools that will enable the auctioneer behavior to be audited retrospectively.

File(s) affected: OevAuctionHouse.sol

Description: A bid with the status `Awarded` can reach the states:

- 1. `FulfillmentReported` if the bidder calls the function `reportFulfillment()` ;
- 2. `FulfillmentConfirmed` if the auctioneer calls the function `confirmFulfillment()` ;
- 3. `FulfillmentContradicted` if the auctioneer calls the function `contradictFulfillment()` ;

Only the first option requires to submit fulfillment details on-chain. Only the second option results in unlocking collateral. As a result, we can imagine a scenario where the Searcher with a bid `Awarded` but an underlying OEV opportunity no longer valid can offer a part of the locked collateral to the Auctioneer as a reward to get its bid confirmed and get back its locked collateral.

The fact that an auctioneer performs collusion actions could only be detected if an external observer monitors for each fulfillment confirmation the target chain to make sure that it correctly matches a valid update. However, an actor playing this role of observer does not explicitly exist in the current system and no actor is incentivized to do so, leaving these collusion events rarely identified.

Exploit Scenario: The initial system can be described as follows:

- Searcher placed a bid `B` on the `bidTopic` `BT` which is now `Awarded` ;
 - The locked collateral for `B` is `500` ;
 - The locked protocol fee amount for `B` is `50` ;
 - The underlying OEV opportunity for `B` is no longer valid;
- 1. Searcher contacts the auctioneer in charge of `BT` and offers to pay `200` to get `B` confirmed instead of contradicted;
 - 2. Auctioneer:
 - accepts the offer;
 - receives `200` in an external transfer;
 - calls the function `confirmFulfillment()` ;
 - Searcher gets back `500` when the collateral is unlocked;
 - 3. At the end:
 - Searcher gets back `300` (`500` - `200`) instead of `50` for a bid `Awarded` with an underlying OEV opportunity no longer valid;

Recommendation: Consider updating the flow, the roles, and/or the incentivization model to mitigate the likelihood of collusion described above, or making it detectable more easily and more quickly. This could be done by introducing an observer role on-chain or off-chain in charge of notifying the manager if it detects that a fulfillment was incorrectly confirmed.

OEVA-13

Risks Related to Off-Chain Actions Triggered by On-Chain Events

• Informational ⓘ Fixed

i Update

Observation requirements were documented in the readme file.

✓ Update

Marked as "Fixed" by the client. Addressed in: e2f200398d5adf2b91a1c492d5e21057e9ae2bdd .

File(s) affected: OevAuctionHouse.sol

Description: Auctioneers use on-chain events emitted by the contract like `PlacedBid` and `ReportedFulfillment` as triggers for further actions, such as awarding a bid or confirming/contradicting a fulfillment. If the way to detect events is not implemented correctly, off-chain observers could be tricked by malicious users and consider invalid events as valid ones. This could happen for instance if the event to be emitted by contract `C` is emitted by another contract than `C` , within a transaction also involving `C` or not.

Recommendation: Consider documenting the fact that auctioneers should make sure that only specific events emitted by specific contracts should be considered valid by their observation script.

OEVA-14 Risks Related to Reorganizations

• Informational ⓘ Fixed



Update

Risks and guidelines were documented in the readme file.



Update

Marked as "Fixed" by the client. Addressed in: `baa13064c6f0012ca25dd678124a53ea8ac0fb53`.

File(s) affected: `OevAuctionHouse.sol`

Description: When a bid is awarded on the chain where the contract `OevAuctionHouse` is deployed, the associated searcher gets the right to submit an update on a target chain.

If a reorg happens on the chain where the contract `OevAuctionHouse` is deployed, the following scenario could happen:

1. The auctioneer awards the bid to Bob.
2. A reorg happens, transactions are reordered somehow, the outcome of the auction becomes different and Bob is no longer the winner of that auction. However, he already got the details to perform the update on the target chain.

If a reorg happens on the target chain where the update should be executed, the following scenario could happen:

1. Searcher performs the update.
2. Searcher reports the fulfillment and `fulfillmentDetails` should contain the block number and/or the block hash and/or the transaction hash.
3. A reorg happens, transactions are reordered somehow, and the details associated with `fulfillmentDetails` are no longer valid.
4. Searcher cannot update the value of `fulfillmentDetails` because the bid has now the state `Awarded`.
5. Auctioneer will incorrectly consider that the fulfillment should be contradicted.

Recommendation: Consider:

- documenting that the process could be negatively impacted by reorgs on the chain where the contract `OevAuctionHouse` is deployed;
- recommending searchers to wait for the block finality before reporting a fulfillment;

OEVA-15

OEV-Impacted Parties May Themselves Place Bids and Not Fulfill Them to Buy Time

• Informational ⓘ

Acknowledged ⓘ



Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

As pointed out, the ideal collateral requirement would be specific to the searcher, the dApp-data feed pair and the bid condition. However, working at this level of granularity is intractable, as it would require us to identify the pseudonymous searcher and predict the OEV opportunity that they plan to capture (or deny service to being captured). We, instead, use a global collateral requirement, which is at the other end of the granularity spectrum. We will observe if this causes a practical problem, and address it as necessary in future iterations.

File(s) affected: `OevAuctionHouse.sol`

Description: OEV-impacted parties, i.e. users who would be subject to liquidation, could themselves place bids, get awarded but not execute the update. They could be incentivized to do so if a delayed liquidation is more profitable than the slashed collateral. However, the likelihood is low, as such scenarios would affect the reputation of the bidder and only work for a short period since the update will be performed 15 seconds later by the API provider. Anyway, isolated cases may exist.

Recommendation: We are currently not aware of a systematic and general solution to prevent these scenarios.

OEVA-16

Awarded Oracle Updates May Still Be Front-Run via Classical MEV

• Informational ⓘ

Fixed



Update

Guidelines were documented in the readme file.



Update

Marked as "Fixed" by the client. Addressed in: `e2649c8a39e94be4b36c36a2009b4c77b6929640`.

File(s) affected: `OevAuctionHouse.sol`

Description: While the solution offered by this protocol mitigates the OEV subset of MEV, classical MEV, like transaction re-ordering still applies. Even though certain oracle updates would be shielded, the OEV-impacted party in case of a liquidation could still observe the mempool and front-run it by preventing the liquidation, by adding more collateral to his position. This would make the OEV opportunity invalid.

Recommendation: Consider recommending bidders to avoid submitting publicly the oracle update.

OEVA-17

Any Auctioneer Can Update the State of Any Bid

• Informational ⓘ

Acknowledged



Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

Considering that all auctioneers will be operated by the same entity, an auctioneer bot operating on a bid (topic) that it shouldn't would be the result of a bug. Protecting against an arbitrary subset of potential bugs in the contract at the cost of additional complexity and gas cost is not attractive. Furthermore, specific auctioneers being in charge of a specific bid or all bids with a bid topic for their entire lifetime is more of an explanation aid than a hard requirement. Awarding bids and confirming/contradicting fulfillments have very different requirements, and two separate sets of workers doing each would be a very natural design, which would not have been possible with the recommended fix.

File(s) affected: `OevAuctionHouse.sol`

Description: The current system treats all addresses owning the role of Auctioneer as sharing the same interests. According to the code, any auctioneer can award any bids and can confirm and/or contradict fulfillments of any bid. Here, we can imagine a scenario of incorrect configuration, leading to two distinct auctioneers using contradicting off-chain rules being in charge of the same bid. Since state transitions cannot be rolled back, the update would be immutable. Another scenario would be a compromised auctioneer, which could temporarily affect the whole system as he would be able to affect all active bids. According to the documentation, measures will be taken if one auctioneer behaves incorrectly but this would be a reactive measure in response to the bid state incorrectly updated, instead of a preventive measure.

Recommendation: Consider limiting the privileges of auctioneers to only a subset of bid topics.

OEVA-18

Hash Collisions Are Possible for the Bid IDs

• Informational ⓘ

Acknowledged



Update

Marked as "Acknowledged" by the client. Addressed in: `da3583f7cc14c8102ff94d2352b7b155b6b0d951`. The client provided the following explanation:

There are two fundamental solutions to this issue. We can keep an on-chain, incrementing nonce for the searchers to generate their bid IDs from, which would ensure that their IDs are always unique. This is cumbersome in practice, as the searcher bot implementation now needs to keep track of its on-chain nonce. This can be abstracted away by using the bid expiration timestamp as a unique nonce (this would not allow multiple identical bids, but this may not be seen as a significant issue). However, this suffers from the same issue as the explicit on-chain nonce, where the searcher bot doesn't know what the bid ID will be once they call `placeBid()`, as it depends on the chain state at the time when the transaction is confirmed. The alternative solution is to let the searcher maintain their nonces off-chain, only requiring them to specify a unique one per bid. This is easy enough to do by generating a random nonce on demand, which allows the searcher bot to precompute bid IDs across multiple asynchronous threads. The searcher will be instructed to add a nonce to their bid details so that a collision is not possible. A potential collision only harms the respective searcher (in that it prevents them from placing the bid they want), which is why we are not concerned about the user failing to do so.

File(s) affected: OevAuctionHouse.sol

Description: Bids are identified in the system by a bytes32 identifier called bidId . They are calculated by the expression: keccak256(abi.encodePacked(bidder, bidDetailsHash)) , with bidder being the address that placed the bid and bidDetailsHash being the hash of the parameter bytes bidDetails having a format that should be defined in the documentation that the auctioneer provides for the bidders for how these parameters should be set.

Depending on the guidelines defining how to build bidDetails , the following scenario could lead to a hash collision: the same address uses twice the same bidDetails that does not depend on the bidTopic and/or the bid expiration date (for instance, using the same constraint), which will lead to the same bidId and only the first submission attempt will be accepted by the system.

Recommendation: Consider giving guidelines to auctioneers to make sure that the two scenarios above cannot happen in practice.

OEVA-19

The Granularity Level of Application of collateralInBasisPoints May Not Be Optimal

• Undetermined ⓘ Acknowledged

i Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

We understand this as a version of OEVA-14 with the difference being that a more moderate and achievable level of granularity is being recommended. We will observe if the global collateral requirement causes a problem in practice, and address it as necessary in future iterations.

File(s) affected: OevAuctionHouse.sol

Description: The system parameter collateralInBasisPoints is used to determine the amount of collateral that should be associated with a bid amount. According to the documentation: "The collateral requirement can range from 0% to values larger than 100% . This is because one can hypothesize cases where denying service by being slashed by the full bid amount is still profitable, in which case a collateral requirement that is larger than 100% would be justifiable." However, if setting this value to more than 100% would be justifiable for only some hypothesized cases, it would have the side effect of increasing the initial capital requirements for any other cases. As a result, if the value of collateralInBasisPoints should depend on the use case, using a single value for the whole system may not be the best solution.

Recommendation: Consider assessing if only one value of collateralInBasisPoints should be defined at the system level, or if its value should vary for instance based on the bidTopic or the chainId .

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Code Documentation

In OevAuctionHouse.sol :



1. **Fixed** In the function `contradictFulfillment()`, the name of the custom error `BidIsNotAwaitingFulfillmentConfirmation` does not match the context when it is triggered.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- [Slither](#) 
- [SuMo](#)  (customized internal version)

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

1. `npm install @morenabarboni/sumo`
2. `npx/yarn sumo pretest`
3. `npx/yarn sumo test`

Automated Analysis

Slither

Slither was used to get a static analysis of the repository. Only relevant issues were considered for this report.

SuMo

SuMo generated 528 mutants and resulted in a mutation score of 87.16%. We checked the live mutants and nothing severe was identified. In details:

- 61 live;
- 414 killed;
- 51 stillborn;
- 0 equivalent;
- 0 redundant;
- 2 timed-out;

Test Suite Results

Run the commands:

- `yarn`
- `yarn test`

```
OevAuctionHouse
  constructor
    ✓ constructs (5969ms)
  setCollateralInBasisPoints
    Sender is the manager
      ✓ sets collateral requirement in basis points (208ms)
    Sender is not the manager
      ✓ reverts (233ms)
  setProtocolFeeInBasisPoints
    Sender is the manager
      Protocol fee is not larger than the maximum
        ✓ sets collateral requirement in basis points (143ms)
      Protocol fee is larger than the maximum
        ✓ reverts (98ms)
    Sender is not the manager
      ✓ reverts (87ms)
  setCollateralRateProxy
    Sender is the manager
      Collateral rate proxy address is not zero
```

```
    ✓ sets collateral rate proxy (176ms)
Collateral rate proxy address is zero
    ✓ reverts (59ms)
Sender is not the manager
    ✓ reverts (66ms)
setChainNativeCurrencyRateProxy
Sender is the manager
Chain ID is not zero
Collateral rate proxy address is not zero
    ✓ sets collateral rate proxy (164ms)
Collateral rate proxy address is zero
    ✓ reverts (125ms)
Chain ID is zero
    ✓ reverts (67ms)
Sender is not the manager
    ✓ reverts (82ms)
withdrawAccumulatedSlashedCollateral
Sender is the manager
Recipient address is not zero
Amount is not zero
Amount does not exceed balance
Transfer is successful
    ✓ withdraws accumulated slashed collateral (3920ms)
Transfer is not successful
    ✓ reverts (173ms)
Amount exceeds balance
    ✓ reverts (158ms)
Amount is not zero
    ✓ reverts (80ms)
Recipient address is zero
    ✓ reverts (105ms)
Sender is not the manager
    ✓ reverts (107ms)
withdrawAccumulatedProtocolFees
Sender is the manager
Recipient address is not zero
Amount is not zero
Amount does not exceed balance
Transfer is successful
    ✓ withdraws accumulated protocol fees (2822ms)
Transfer is not successful
    ✓ reverts (110ms)
Amount exceeds balance
    ✓ reverts (66ms)
Amount is not zero
    ✓ reverts (55ms)
Recipient address is zero
    ✓ reverts (82ms)
Sender is not the manager
    ✓ reverts (71ms)
depositForBidder
Bidder address is not zero
Deposit amount is not zero
    ✓ deposits for bidder (254ms)
Deposit amount is zero
    ✓ reverts (61ms)
Bidder address is zero
    ✓ reverts (76ms)
deposit
Deposit amount is not zero
    ✓ deposits (292ms)
Deposit amount is zero
    ✓ reverts (76ms)
initiateWithdrawal
Bidder does not have an initiated withdrawal
    ✓ initiates withdrawal (177ms)
Bidder has an initiated withdrawal
    ✓ reverts (579ms)
```

```
withdraw
  Recipient address is not zero
  Amount is not zero
  Amount does not exceed balance
  Sender has an initiated withdrawal
    It is time for the bidder to be able to withdraw
      Transfer is successful
        ✓ withdraws (564ms)
      Transfer is not successful
        ✓ reverts (290ms)
    It is not time yet for the bidder to be able to withdraw
      ✓ reverts (201ms)
  Sender does not have an initiated withdrawal
    ✓ reverts (106ms)
  Amount exceeds balance
    ✓ reverts (177ms)
  Amount is zero
    ✓ reverts (187ms)
  Recipient address is zero
    ✓ reverts (129ms)
cancelWithdrawal
  Sender has an initiated withdrawal
    ✓ cancels the withdrawal (103ms)
  Sender does not have an initiated withdrawal
    ✓ reverts (53ms)
placeBidWithExpiration
  Chain ID is not zero
  Bid amount is not zero
  Bid details are not empty
    Bid lifetime is not larger than maximum
    Bid lifetime is not shorter than minimum
    Bid is not already placed
      Collateral amount can be calculated
        ✓ places bid with expiration (2077ms)
      Collateral amount cannot be calculated
        ✓ reverts (271ms)
    Bid is already placed
      ✓ reverts (246ms)
    Bid lifetime is shorter than minimum
      ✓ reverts (167ms)
    Bid lifetime is larger than maximum
      ✓ reverts (112ms)
  Bid details are empty
    ✓ reverts (61ms)
  Bid amount is zero
    ✓ reverts (102ms)
  Chain ID is zero
    ✓ reverts (513ms)
placeBid
  ✓ places bid (410ms)
expediteBidExpiration
  Bid is awaiting award
  Bid has not expired
    Timestamp expedites bid expiration
      Resulting bid lifetime is not shorter than minimum
        ✓ expedites bid expiration (324ms)
      Resulting bid lifetime is shorter than minimum
        ✓ reverts (878ms)
    Timestamp does not expedite bid expiration
      ✓ reverts (301ms)
  Bid has expired
    ✓ reverts (259ms)
  Bid is not awaiting award
    ✓ reverts (97ms)
expediteBidExpirationMaximally
  ✓ expedites bid expiration maximally (262ms)
awardBid
  Sender is an auctioneer
```

```

Award details are not empty
  Bid is awaiting award
    Bid has not expired
      Bidder balance is not lower than the sum of collateral and protocol fee amounts
        ✓ awards the bid (1947ms)
      Bidder balance is lower than the sum of collateral and protocol fee amounts
        ✓ reverts (141ms)
    Bid has expired
      ✓ reverts (112ms)
  Bid is not awaiting award
    ✓ reverts (234ms)
Award details are empty
  ✓ reverts (106ms)
Sender is not an auctioneer
  ✓ reverts (98ms)
reportFulfillment
  Fulfillment details are not empty
    Bid is awaiting fulfillment report
      Bid has not expired
        ✓ reports fulfillment (1810ms)
      Bid has expired
        ✓ reverts (79ms)
    Bid is not awaiting fulfillment report
      ✓ reverts (100ms)
  Fulfillment details are not empty
    ✓ reverts (45ms)
confirmFulfillment
  Sender is an auctioneer
    Bid is awaiting fulfillment confirmation
      ✓ confirms fulfillment (2127ms)
    Bid is not awaiting fulfillment confirmation
      ✓ reverts (136ms)
  Sender is not an auctioneer
    ✓ reverts (61ms)
contradictFulfillment
  Sender is an auctioneer
    Bid is awaiting fulfillment confirmation
      ✓ contradicts fulfillment (264ms)
    Bid is not awaiting fulfillment confirmation
      ✓ reverts (144ms)
  Sender is not an auctioneer
    ✓ reverts (77ms)
getCurrentCollateralAndProtocolFeeAmounts
  Collateral rate proxy is valid
    Collateral rate is positive
      Collateral rate is not stale
        Native currency rate proxy is valid
          Native currency rate is positive
            Native currency rate is not stale
              Collateral and protocol fee amounts are small enough to be typecasted to
uint104
                ✓ gets current collateral and protocol fee amounts (55ms)
              Collateral and protocol fee amounts are not small enough to be typecasted to
uint104
                ✓ revert (107ms)
            Native currency rate is stale
              ✓ reverts (113ms)
          Native currency rate is not positive
            ✓ reverts (88ms)
        Native currency rate proxy is not valid
          ✓ reverts (53ms)
      Collateral rate is stale
        ✓ reverts (119ms)
    Collateral rate is not positive
      ✓ reverts (101ms)
  Collateral rate proxy is not valid
    ✓ reverts (64ms)

```


80 passing (39s)

----- Fix Review Update -----

OevAuctionHouse

constructor

✓ constructs (5834ms)

setCollateralInBasisPoints

Sender is the manager

✓ sets collateral requirement in basis points (235ms)

Sender is not the manager

✓ reverts (235ms)

setProtocolFeeInBasisPoints

Sender is the manager

Protocol fee is not larger than the maximum

✓ sets collateral requirement in basis points (152ms)

Protocol fee is larger than the maximum

✓ reverts (76ms)

Sender is not the manager

✓ reverts (79ms)

setCollateralRateProxy

Sender is the manager

Collateral rate proxy address is not zero

✓ sets collateral rate proxy (136ms)

Collateral rate proxy address is zero

✓ reverts (65ms)

Sender is not the manager

✓ reverts (60ms)

setChainNativeCurrencyRateProxy

Sender is the manager

Chain ID is not zero

Collateral rate proxy address is not zero

✓ sets collateral rate proxy (695ms)

Collateral rate proxy address is zero

✓ reverts (155ms)

Chain ID is zero

✓ reverts (77ms)

Sender is not the manager

✓ reverts (80ms)

withdrawAccumulatedSlashedCollateral

Sender is the manager

Recipient address is not zero

Amount is not zero

Amount does not exceed balance

Transfer is successful

✓ withdraws accumulated slashed collateral (2854ms)

Transfer is not successful

✓ reverts (168ms)

Amount exceeds balance

✓ reverts (94ms)

Amount is not zero

✓ reverts (70ms)

Recipient address is zero

✓ reverts (100ms)

Sender is not the manager

✓ reverts (82ms)

withdrawAccumulatedProtocolFees

Sender is the manager

Recipient address is not zero

Amount is not zero

Amount does not exceed balance

Transfer is successful

✓ withdraws accumulated protocol fees (2515ms)

Transfer is not successful

✓ reverts (112ms)

```
    Amount exceeds balance
      ✓ reverts (85ms)
    Amount is not zero
      ✓ reverts (59ms)
    Recipient address is zero
      ✓ reverts (356ms)
    Sender is not the manager
      ✓ reverts (182ms)
depositForBidder
  Bidder address is not zero
    Deposit amount is not zero
      ✓ deposits for bidder (369ms)
    Deposit amount is zero
      ✓ reverts (72ms)
  Bidder address is zero
    ✓ reverts (67ms)
deposit
  Deposit amount is not zero
    ✓ deposits (212ms)
  Deposit amount is zero
    ✓ reverts (47ms)
initiateWithdrawal
  Bidder does not have an initiated withdrawal
    ✓ initiates withdrawal (160ms)
  Bidder has an initiated withdrawal
    ✓ reverts (113ms)
withdraw
  Recipient address is not zero
    Amount is not zero
      Amount does not exceed balance
        Sender has an initiated withdrawal
          It is time for the bidder to be able to withdraw
            Transfer is successful
              ✓ withdraws (374ms)
            Transfer is not successful
              ✓ reverts (180ms)
          It is not time yet for the bidder to be able to withdraw
            ✓ reverts (172ms)
        Sender does not have an initiated withdrawal
          ✓ reverts (101ms)
      Amount exceeds balance
        ✓ reverts (150ms)
    Amount is zero
      ✓ reverts (193ms)
  Recipient address is zero
    ✓ reverts (134ms)
cancelWithdrawal
  Sender has an initiated withdrawal
    ✓ cancels the withdrawal (105ms)
  Sender does not have an initiated withdrawal
    ✓ reverts (43ms)
placeBidWithExpiration
  Chain ID is not zero
    Bid amount is not zero
      Bid details length does not exceed the maximum
        Bid details are not empty
          Bid lifetime is not larger than maximum
            Bid lifetime is not shorter than minimum
              Bid is not already placed
                Collateral amount can be calculated
                  Maximum collateral amount is not exceeded
                    Maximum protocol fee amount is not exceeded
                      ✓ places bid with expiration (2495ms)
                    Maximum protocol fee amount is exceeded
                      ✓ reverts (221ms)
                  Maximum collateral amount is exceeded
                    ✓ reverts (189ms)
                Collateral amount cannot be calculated
```

```
        ✓ reverts (228ms)
        Bid is already placed
        ✓ reverts (257ms)
        Bid lifetime is shorter than minimum
        ✓ reverts (587ms)
        Bid lifetime is larger than maximum
        ✓ reverts (218ms)
        Bid details are empty
        ✓ reverts (82ms)
        Bid details length exceeds the maximum
        ✓ reverts (63ms)
        Bid amount is zero
        ✓ reverts (106ms)
        Chain ID is zero
        ✓ reverts (106ms)
    placeBid
        ✓ places bid (312ms)
    expediteBidExpiration
        Bid is awaiting award
        Bid has not expired
        Timestamp expedites bid expiration
        Resulting bid lifetime is not shorter than minimum
        ✓ expedites bid expiration (282ms)
        Resulting bid lifetime is shorter than minimum
        ✓ reverts (266ms)
        Timestamp does not expedite bid expiration
        ✓ reverts (263ms)
        Bid has expired
        ✓ reverts (266ms)
        Bid is not awaiting award
        ✓ reverts (131ms)
    expediteBidExpirationMaximally
        ✓ expedites bid expiration maximally (302ms)
    awardBid
        Sender is an auctioneer
        Award details length does not exceed the maximum
        Award details are not empty
        Bid is awaiting award
        Bid has not expired
        Bidder balance is not lower than the larger of collateral and protocol fee
amounts
        Collateral amount is larger than protocol fee amount
        ✓ awards the bid (1737ms)
        Collateral amount is not larger than protocol fee amount
        ✓ awards the bid (1952ms)
        Bidder balance is lower than the larger of collateral and protocol fee amounts
        ✓ reverts (130ms)
        Bid has expired
        ✓ reverts (123ms)
        Bid is not awaiting award
        ✓ reverts (188ms)
        Award details are empty
        ✓ reverts (97ms)
        Award details length exceeds the maximum
        ✓ reverts (151ms)
        Sender is not an auctioneer
        ✓ reverts (130ms)
    reportFulfillment
        Fulfillment details length does not exceed the maximum
        Fulfillment details are not empty
        Bid is awaiting fulfillment report
        Bid has not expired
        ✓ reports fulfillment (1666ms)
        Bid has expired
        ✓ reverts (81ms)
        Bid is not awaiting fulfillment report
        ✓ reverts (117ms)
        Fulfillment details are not empty
```

```

    ✓ reverts (65ms)
  Fulfillment details length exceeds the maximum
    ✓ reverts (80ms)
confirmFulfillment
  Sender is an auctioneer
    Bid is awaiting fulfillment confirmation
      Collateral amount is larger than protocol fee amount
        ✓ confirms fulfillment (2108ms)
      Collateral amount is not larger than protocol fee amount
        ✓ confirms fulfillment (2655ms)
    Bid is not awaiting fulfillment confirmation
      ✓ reverts (169ms)
  Sender is not an auctioneer
    ✓ reverts (66ms)
contradictFulfillment
  Sender is an auctioneer
    Bid is awaiting fulfillment confirmation
      Collateral amount is larger than protocol fee amount
        ✓ contradicts fulfillment (320ms)
      Collateral amount is not larger than protocol fee amount
        ✓ contradicts fulfillment (2073ms)
    Bid is not awaiting fulfillment confirmation
      ✓ reverts (177ms)
  Sender is not an auctioneer
    ✓ reverts (75ms)
getCurrentCollateralAndProtocolFeeAmounts
  Collateral rate proxy is valid
  Collateral rate is positive
  Collateral rate is not stale
  Native currency rate proxy is valid
  Native currency rate is positive
  Native currency rate is not stale
    Collateral and protocol fee amounts are small enough to be typecasted to
uint104
    ✓ gets current collateral and protocol fee amounts (63ms)
    Collateral and protocol fee amounts are not small enough to be typecasted to
uint104
    ✓ revert (195ms)
  Native currency rate is stale
    ✓ reverts (85ms)
  Native currency rate is not positive
    ✓ reverts (86ms)
  Native currency rate proxy is not valid
    ✓ reverts (58ms)
  Collateral rate is stale
    ✓ reverts (80ms)
  Collateral rate is not positive
    ✓ reverts (81ms)
  Collateral rate proxy is not valid
    ✓ reverts (45ms)

88 passing (45s)

```

Code Coverage

Run the commands:

- yarn
- yarn test:coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
OevAuctionHouse.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
IOevAuctionHouse.sol	100	100	100	100	
contracts/test/	100	100	100	100	
AccessControlRegistry.sol	100	100	100	100	
MockProxy.sol	100	100	100	100	
All files	100	100	100	100	

Fix Review Update

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
OevAuctionHouse.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
IOevAuctionHouse.sol	100	100	100	100	
contracts/test/	100	100	100	100	
AccessControlRegistry.sol	100	100	100	100	
MockProxy.sol	100	100	100	100	
All files	100	100	100	100	

Changelog

- 2023-12-21 - Initial report
- 2024-01-10 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp’s mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp’s team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over \$200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that

Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.



Quantstamp

© 2024 – Quantstamp, Inc.

OEV Auction House: API3