Quantstamp DRAFT

API3Market

Executive Summary

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

Type	dAPI Market
Timeline	2024-02-07 through 2024-02-20
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Doc Files Whitepaper Website
Source Code	• api3dao/contracts #0056a44
Auditors	• Adrian Koegl Auditing Engineer • Jonathan Mevs Auditing Engineer • Michael Boyle Auditing Engineer

Documentation quality	Medium
Test quality	High
Total Findings	10 Acknowledged: 10
High severity findings	0
Medium severity findings	3 Acknowledged: 3
Low severity findings	5 Acknowledged: 5
Undetermined severity findings	1 Acknowledged: 1
Informational findings	1 Acknowledged: 1

Summary of Findings

Update: The client has acknowledged all of the 11 findings. None of them pose severe security issues as long as the processes are correctly executed off-chain. However, user trust in off-chain entities is required.

The contracts in scope constitute the dAPI market of API3:

- Through the `API3Market` contract, users can purchase subscriptions for data feed updates with specific configurations.
- The `AirseekerRegistry` keeps track of the currently active subscriptions and the constraints under which Airseekers are supposed to update the data feeds.
- The `HashRegistry` contains the Merkle roots of all allowed configurations managed by a set of signers.

The code in scope is generally well-written and follows best practices. We have not found any significant security vulnerabilities. However, the code and its security heavily depend on correct and honest executing off-chain components, such as the `owner`, different sets of `signers`, and Airseekers. Furthermore, the out-of-scope `API3ServerV1` contract maintaining the data feeds interacts with the contracts in scope at several points. The security of that `API3ServerV1` contract, its interaction with the contracts in scope, and the off-chain entities were not assessed by Quantstamp.

ID	DESCRIPTION	SEVERITY	STATUS
A3M-1	Attacker Can DoS Queue with Low-Quality Subscriptions	Medium	Acknowledged
A3M-2	Sponsor Wallet Owner Needs to Be Trusted	Medium	Acknowledged
A3M-3	Reliance on Correctness of Merkle Tree Construction	Medium	Acknowledged
A3M-4	Greedy Subscription Purchase	Low	Acknowledged
A3M-5	Privileged Roles and Ownership	Low	Acknowledged
A3M-6	Critical Role Transfer Not Following Two-Step Pattern	Low	Acknowledged

ID	DESCRIPTION	SEVERITY	STATUS
A3M-7	Owner Can Override Signers	Low	Acknowledged
A3M-8	All Chains Must Be in Sync with Same State	Low	Acknowledged
A3M-9	Ownership Can Be Renounced	Informational	Acknowledged
A3M-10	Out of Scope Contracts Being Called	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.

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 code revisions are excluded from consideration in this report. Specifically, the `API3ServerV1` contract is not in scope, as well as any off-chain components that interact with the smart contracts.

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

Files Included

`AirseekerRegistry.sol` `API3Market.sol` `HashRegistry.sol`

Files Excluded

`API3ServerV1.sol`

Findings

A3M-1
Attacker Can DoS Queue with Low-Quality Subscriptions

Medium | Acknowledged

Update

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

> The current implementation iterates through the entire subscription queue while buying a new subscription. We enforce a hard limit on the size of the queue to ensure that this is viable. Allowing this limit to be overridden (recommendation 1) would only be safe if it was overly restrictive to begin with, in which case relaxing the limit (e.g., 10 instead of 5) would be preferable due to its simplicity. Purging a subscription that has been paid for (recommendation 2) does not appear to be a valid option to us. In short, we will not address this issue for the time being, yet we may slightly increase the limit later to alleviate it.

File(s) affected: `API3Market.sol`

Description: The subscription queues have a maximum size of 5 subscriptions. As long as the new subscription is equal to or better than at least one subscription in the queue, it will added. If it is equal, it must have an `endTimestamp` further in the future. However, if the queue is full and someone wants to queue a subscription that is better than all of them but has a lower `endTimestamp`, that subscription will fail to be added with error "Subscription queue full". Therefore, an attacker could fill the queue with the lowest-quality subscriptions and prevent better subscriptions from being added. However, this requires that more unique update parameters exist in the Merkle tree for a given dAPI name than the maximum allowed in the queue. The API3 team has confirmed that more than five possible update parameter configurations should not exist in the Merkle tree. Since the number of approved update parameter configurations, however, cannot be verified on-chain, this issue remains possible.

Recommendation: We have two different suggestions to resolve this issue:

1. Consider making the length of the queue modifiable by the owner in case more update parameter configurations are desired in the future.
2. Consider a mechanism to purge the queue from the worst subscription. However, it should be noted, that this would allow an attacker to queue a subscription with the best conditions but extremely low timestamp. This could be prevented by enforcing a minimum timestamp.

A3M-2 Sponsor Wallet Owner Needs to Be Trusted

Medium | Acknowledged

Update

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

> It is not a requirement for the sponsor wallet addresses specified in the dAPI management Merkle tree to belong to EOAs that send data feed update transactions. Without needing to make any changes to the Api3Market contract, these addresses can also belong to contracts that drip-feed funds to the respective EOAs, or contracts that implement an account abstraction scheme. However, we have used the former method (of funding EOAs directly) in production with good success in the context of “self-funded feeds”, and will continue to do so for the time being.

File(s) affected: `API3Market.sol`

Description: Purchasing a subscription will result in a payment to sponsor wallets that can be quite substantial (> 5 ETH). These funds should be used by Airseekers to pay for the gas fees of updates. However, the owner of the sponsor wallet needs to be trusted not to mistreat the funds. The only incentive that keeps them from draining those funds is reputation. Furthermore, compromised sponsor wallets will also prevent the continued operation of already purchased subscriptions unless the API3 team provides funds.

Recommendation: Consider implementing a mechanism such that those funds are not immediately and entirely transferred to the sponsor's wallet. Instead, they should be awarded only some funds regularly in advance to pay for the interactions.

A3M-3 Reliance on Correctness of Merkle Tree Construction

Medium | Acknowledged

Update

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

> Considering that these checks can be offloaded to the respective hash signers, we do not find the added complexity to be worthwhile. As a note, we do not agree with the statement “A dAPI name in the management Merkle tree should not be re-pointed to a different data feed during an active subscription. This might violate the purchasing conditions of the subscriber.” A dAPI is understood to be a managed service, and API3 maintaining the configuration of the dAPI over the lifetime of the subscription to uphold the expectations of the user as described by the dAPI name (e.g., “ETH/USD”) and update parameters is an important aspect of the service. For example, in the case that an API provider discontinues service,

> the dAPI name would be re-pointed to a Beacon set that replaces said API provider, and the contracts are designed to be able to handle this.

Description: The smart contracts in scope rely on the correct off-chain construction of Merkle trees. The following constraints are not enforced on-chain upon Merkle root submission and, if ever accidentally or intentionally violated, could lead to security issues and/or inconsistencies:

1. The Management Merkle Tree cannot contain a specific dAPI name more than once. Otherwise, anyone can change the pointer of the dAPI name to the data feed ID.
2. A dAPI name in the management Merkle tree should not be re-pointed to a different data feed during an active subscription. This might violate the purchasing conditions of the subscriber.
3. All entries of the same dAPI name in the management Merkle tree must point to the same sponsor wallet.

Recommendation: The correctness of merkle trees is hard to verify on-chain. A potential solution would be to pass a zk proof, such that a smart contract can verify that some constraints hold for the submitted merkle root.

A3M-4 Greedy Subscription Purchase

Low | Acknowledged

Update

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

> We opted against refunding the change due to there not being a practical way of doing so. We do not want to push payments to the user's account, and we do not find it practical to capture dust in Api3Market that the user can get back by sending another transaction. Furthermore, it is an intractable task to tell if an amount is to be considered negligible, so we also do not find an adaptive approach to be viable. Compared to this, we find letting the extra payment move to the sponsor wallet to be an elegant solution, considering that the user is likely to renew the subscription, in which case they will be practically refunded without any of the issues above.

File(s) affected: `API3Market.sol`

Description: Users can deposit more funds than necessary to purchase subscriptions through `buySubscription()`. Although the API3 Front-end likely mitigates this, it should not be possible for a user to deposit exceptionally more than needed when buying a subscription. Particularly, if the sponsor wallet already has (almost) sufficient funds to cover subscriptions, the function shouldn't accept these excess funds.

Recommendation: Consider refunding any amounts that would result in the sponsor wallet being a configurable threshold amount beyond the computed expected wallet balance following the subscription purchase.

A3M-5 Privileged Roles and Ownership

Low | Acknowledged

Update

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

> We have already made an effort to document all privileged roles in the repo docs and will carry these over to end user-facing docs once they are deployed. Our Merkle tree-based approach is suitable for DAO governance due to requiring a minimal number of transactions/proposals to signal decisions. The current proliferation of rollups is also encouraging due to them being able to bridge data from their L1 in a trust-minimized way. We foresee a future where the API3 DAO can feasibly curate dAPI configurations across many chains.

File(s) affected: `Api3Market.sol`, `HashRegistry.sol`, `AirseekerRegistry.sol`

Description: The following are the smart contracts and their associated roles. Many of the smart contracts contain roles that are necessary for the correct operation of the system. Therefore, roles, such as the default admin role, should be held by a multi-sig wallet.

1. `Api3Market`
    1. Has an `owner` but all privileged functions exist in the inherited `HashRegistry`.
2. `HashRegistry`
    1. Is inherited by `Api3Market` and shares an `owner`.
    2. The `owner` can set signers for specific hash types including removing or adding signers.
    3. The `owner` can set hashes for hash types, including overriding hashes set by the signers.
    4. The signers can use m-of-m signatures to set the hash for a hash type they are allowed to set.
    5. The signers can also delegate their signature to any address including addresses that do not have an associated private key. This will effectively prevent the hash from being changed until the delegation expires or the owner removes them. This is equivalent to a signer choosing not to sign.
3. `AirseekerRegistry`
    1. All of its state-changing functions can only be called by the owner except for `registerDataFeed()`.
    2. It may be owned by `Api3Market` or exist as a standalone smart contract with a different owner.

The `owner` privileges on `HashRegistry.sol` implies the following rights:

1. The `owner` can change the data feed ID(s) the dAPI name points to.
2. The sponsor wallet that will receive funds for a (dAPI name, data feed ID) tuple.
3. The accepted `updateParameters`, `duration`, and `price` of a dAPI name.

Recommendation: Consider using a multi-sig wallet for the owner of `Api3Market` and only allow trusted addresses to be signers. Make users aware of the privileged roles via documentation. If appropriate, set a roadmap to reduce those privileged rights or give more power to the DAO.

A3M-6 Critical Role Transfer Not Following Two-Step Pattern

Low Acknowledged

Update

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

> We will opt for Ownable for the time being for consistency across codebases.

File(s) affected: `HashRegistry.sol`

Description: The owner of the contracts can call `transferOwnership()` to transfer the ownership to a new address. If an uncontrollable address is accidentally provided as the new owner address then the contract will no longer have an active owner, and functions with the `onlyOwner` modifier can no longer be executed. This is only relevant if `HashRegistry.sol` is used as a standalone contract since ownership transfer is disallowed in `API3Market`.

Recommendation: Consider using OpenZeppelin's `Ownable2Step` contract to adopt a two-step ownership pattern in which the new owner must accept their position before the transfer is complete.

A3M-7 Owner Can Override Signers

Low Acknowledged

Update

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

> As noted, the owner can already override signers by setting a new set of signers and having them sign a new hash. Therefore, we do not see this to be an escalation of privilege. About the signers potentially being more trustworthy than the owner, this is only circumstantial and not inherent to the contract, but it should also not be surprising. It is the owner's responsibility to come up with a trustworthy configuration that it can delegate to, and this trustworthiness is not necessarily limited by the trustworthiness of the owner, which can be expected to result in an overshoot. That being said, we are intending to improve the said 4-of-8 threshold now that the multisig will not need to send day-to-day transactions to conduct dAPI operations.

File(s) affected: `API3Market.sol`, `HashRegistry.sol`, `AirseekerRegistry.sol`

Description: In `HashRegistry`, the owner can specify which signers are required to set the hash for a specific hash type. However, the owner can also set the hash for any hash type. This leads to a situation in which the owner's address has a disproportionate control when compared to the signers. For example, consider a scenario where the owner's address operates under a 4-of-8 multisig scheme, and there exists a specific hash type that requires authorization from 4 signers. In this case, the 4-of-8 multisig configuration is, in essence, less secure compared to a setup that directly uses a 4-of-4 multisig structure, which would be entirely made up of these signers. Additionally, the separation of roles may be violated in a case where the addresses responsible for updating a specific hash type are overruled by the owner address. Of course, the owner can also set the signers, so it needs to be the most secure.

Recommendation: Consider only allowing the signers to set the hash.

A3M-8 All Chains Must Be in Sync with Same State

Low Acknowledged

Update

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

> Requiring chain-many signatures has practical shortcomings that worsen with more chains, and thus this was a subjective tradeoff based on our requirements.

Description: The Merkle roots of management Merkle trees can be replayed across all chains. This is intended to reduce the signing overhead. However, if all chains are not synchronized in terms of data feed IDs, this could potentially result in invalid Merkle trees being activated and, consequently, incorrect data feed IDs being activated or transactions reverting because the data feed ID is not registered anymore. Furthermore, identical data feed IDs must be used across all chains.

Recommendation: In its current design, this limitation can only be overcome by requiring signers to sign of all chains individually, disallowing signature replays.

A3M-9 Ownership Can Be Renounced

Informational | Acknowledged

> **Update**
> Marked as "Acknowledged" by the client. The client provided the following explanation:
>
> > HashRegistry allows its ownership to be renounced for the sake of flexibility (e.g., the owner unsets signers and renounces ownership to make the currently registered hashes immutable). The respective function can be overridden by the inheriting contract if this is not desired.

File(s) affected: `HashRegistry.sol`

Description: If the `owner` renounces their ownership, all ownable contracts will be left without an owner. Consequently, any function guarded by the `onlyOwner` modifier will no longer be able to be executed. This is only relevant in case `HashRegistry.sol` is used as a standalone contract.

Recommendation: Confirm that this is the intended behavior. If not, override and disable the `renounceOwnership()` function in the affected contracts.

A3M-10 Out of Scope Contracts Being Called

Undetermined | Acknowledged

> **Update**
> Marked as "Acknowledged" by the client. The client provided the following explanation:
>
> > Versions of Api3ServerV1 (including its current state) have been audited three times, and we are planning to have it audited again in the near future.

Description: In several instances, the `API3Market` contract calls functions in the `API3ServerV1` contract. However, the target contract is out of scope for this audit. While we checked the interdependencies, we trust that the `API3ServerV1` correctly handles all function calls. Particularly, we trust that necessary checks for external functions in `API3Market` are made in `API3ServerV1` to prevent unauthorized state changes, such as `Api3Market.updateBeaconWithSignedData()`.

Recommendation: Consider auditing the `API3ServerV1` contract and the interdependencies with `API3Market`.

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).

Automated Analysis

N/A

Test Suite Results

We ran the tests using the following commands:

```
pnpm install
yarn test
```

All 153 tests successfully passed.

```
AirseekerRegistry
    constructor
      Owner address is not zero
        Api3ServerV1 address is not zero
          ✔ constructs (174ms)
        Api3ServerV1 address is zero
          ✔ reverts
      Owner address is zero
        ✔ reverts
    renounceOwnership
      ✔ reverts
    transferOwnership
      ✔ reverts
    setDataFeedIdToBeActivated
      Sender is the owner
        Data feed ID is not zero
          Data feed ID is not activated
            ✔ activates the data feed ID
          Data feed ID is already activated
            ✔ does nothing
        Data feed ID is zero
          ✔ reverts
      Sender is not the owner
        ✔ reverts
    setDapiNameToBeActivated
      Sender is the owner
        dAPI name is not zero
          dAPI name is not activated
            ✔ activates the dAPI name
          dAPI name is already activated
            ✔ does nothing
        dAPI name is zero
          ✔ reverts
      Sender is not the owner
        ✔ reverts
    setDataFeedIdToBeDeactivated
      Sender is the owner
        Data feed ID is not zero
          Data feed ID is activated
            ✔ activates the data feed ID
          Data feed ID is not activated
            ✔ does nothing
        Data feed ID is zero
          ✔ reverts
      Sender is not the owner
        ✔ reverts
    setDapiNameToBeDeactivated
      Sender is the owner
        dAPI name is not zero
          dAPI name is activated
            ✔ activates the dAPI name
          dAPI name is not activated
            ✔ does nothing
        dAPI name is zero
          ✔ reverts
      Sender is not the owner
        ✔ reverts
    setDataFeedIdUpdateParameters
      Sender is the owner
        Data feed ID is not zero
```

```
      Update parameters length does not exceed the maximum
        Values update update parameters
          Values have not been used before
            ✔ updates update parameters
          Values have been used before
            ✔ updates update parameters
        Values do not update update parameters
          ✔ does nothing
      Update parameters length exceeds the maximum
        ✔ reverts
    Data feed ID is zero
      ✔ reverts
  Sender is not the owner
    ✔ reverts
setDapiNameUpdateParameters
  Sender is the owner
    dAPI name is not zero
      Update parameters length does not exceed the maximum
        Values update update parameters
          Values have not been used before
            ✔ updates update parameters
          Values have been used before
            ✔ updates update parameters
        Values do not update update parameters
          ✔ does nothing
      Update parameters length exceeds the maximum
        ✔ reverts
    dAPI name is zero
      ✔ reverts
  Sender is not the owner
    ✔ reverts
setSignedApiUrl
  Sender is the owner
    Airnode address is not zero
      Signed API URL is not too long
        Value updates signed API URL
          ✔ updates signed API URL
        Value does not update signed API URL
          ✔ does nothing
      Signed API URL is too long
        ✔ reverts
    Airnode address is zero
      ✔ reverts
  Sender is not the owner
    ✔ reverts
registerDataFeed
  Data feed details are long enough to specify a single Beacon
    Airnode address is not zero
      Data feed is not registered
        ✔ registers data feed
      Data feed is already registered
        ✔ does nothing
    Airnode address is zero
      ✔ reverts
  Data feed details are at least long enough to specify a Beacon set composed of two Beacons
    Data feed details length does not exceed specifications for a Beacon set composed of the maximum
number of Beacons
      Data feed details data does not trail
        Data feed detail parameter lengths match
          None of the Airnode addresses is zero
            Data feed is not registered
              ✔ registers data feed
            Data feed is already registered
              ✔ does nothing
          Some of the Airnode addresses are zero
            ✔ reverts
        Data feed detail parameter lengths do not match
          ✔ reverts
      Data feed details data trail
        ✔ reverts
    Data feed details length exceeds specifications for a Beacon set composed of the maximum number
of Beacons
```

```
          ✔ reverts
      Data feed details neither long enough to specify a single Beacon or at least long enough to specify
a Beacon set composed of two Beacons
        ✔ reverts
    activeDataFeed
      The index belongs to an active data feed ID
        Data feed ID update parameters have been set
          Data feed details have been set
            Data feed is a Beacon set
              ✔ returns data feed ID, details, reading, Beacon readings, update parameters and respective
signed API URLs (50ms)
            Data feed is a Beacon
              ✔ returns data feed ID, details, reading, Beacon reading, update parameters and the
respective signed API URL
          Data feed details have not been set
            ✔ returns data feed ID, reading and update parameters
        Data feed ID update parameters have not been set
          Data feed details have been set
            Data feed is a Beacon set
              ✔ returns data feed ID, details, reading, Beacon readings and respective signed API URLs
(41ms)
            Data feed is a Beacon
              ✔ returns data feed ID, details, reading, Beacon reading and the respective signed API URL
          Data feed details have not been set
            ✔ returns data feed ID and reading
      The index belongs to an active dAPI name
        dAPI name has been set at Api3ServerV1
          dAPI name update parameters have been set
            Data feed details have been set
              Data feed is a Beacon set
                ✔ returns data feed ID, dAPI name, details, reading, Beacon readings, update parameters
and respective signed API URLs (47ms)
              Data feed is a Beacon
                ✔ returns data feed ID, dAPI name, details, reading, Beacon reading, update parameters
and the respective signed API URL
            Data feed details have not been set
              ✔ returns data feed ID, dAPI name, reading and update parameters
          dAPI name update parameters have not been set
            Data feed details have been set
              Data feed is a Beacon set
                ✔ returns data feed ID, dAPI name, details, reading, Beacon readings and respective
signed API URLs (45ms)
              Data feed is a Beacon
                ✔ returns data feed ID, dAPI name, details, reading, Beacon reading and the respective
signed API URL
            Data feed details have not been set
              ✔ returns data feed ID, dAPI name, details, reading and respective signed API URLs
        dAPI name has not been set at Api3ServerV1
          dAPI name update parameters have been set
            ✔ returns dAPI name and update parameters
          dAPI name update parameters have not been set
            ✔ returns dAPI name
      The index does not belong to an active data feed ID or dAPI name
        ✔ returns nothing

  HashRegistry
    constructor
      Owner address is not zero
        ✔ constructs (42ms)
      Owner address is zero
        ✔ reverts
    renounceOwnership
      ✔ renounces ownership
    transferOwnership
      ✔ transfers ownership
    setSigners
      Sender is the owner
        Hash type is not zero
          Signers are not empty
            First signer address is not zero
              Signer addresses are in ascending order
                ✔ sets signers
```

```
          Signer addresses are not in ascending order
            ✔ reverts
        First signer address is zero
          ✔ reverts
      Signers are empty
        ✔ reverts
    Hash type is zero
      ✔ reverts
  Sender is not the owner
    ✔ reverts
setHash
  Sender is the owner
    ✔ sets hash (60ms)
  Sender is not the owner
    ✔ reverts
registerHash
  Hash value is not zero
    Timestamp is not from the future
      Timestamp is more recent than the previous one
        Signers are set for the hash type
          No delegation signature is used
            All signatures match
              ✔ registers hash
            Not all signatures match
              ✔ reverts
          Delegation signatures are used
            All signatures have a valid length
              None of the delegation signatures have expired
                All delegate hash signatures are valid
                  All delegation signatures are valid
                    ✔ registers hash
                  Not all delegation signatures are valid
                    ✔ reverts
                Not all delegate hash signatures are valid
                  ✔ reverts
              Some of the delegation signatures have expired
                ✔ reverts
            Not all signatures have a valid length
              ✔ reverts
      Signers are not set for the hash type
        ✔ reverts
    Timestamp is not more recent than the previous one
      ✔ reverts (60ms)
  Timestamp is from the future
    ✔ reverts
Hash value is not zero
  ✔ reverts

Api3Market
  constructor
    ProxyFactory address belongs to a contract with the expected interface
      ✔ constructs (7578ms)
    ProxyFactory address belongs to a contract without the expected interface
      ✔ reverts (55ms)
    ProxyFactory address does not belong to a contract
      ✔ reverts (47ms)
  renounceOwnership
    ✔ reverts
  transferOwnership
    ✔ reverts
  buySubscription
    Arguments are valid
      New subscription can be added to the queue
        Payment is enough to get the sponsor wallet balance over the expected amount
          Payment amount is not zero
            Payment transfer succeeds
              Subscription is added to the start of the queue
                dAPI name needs to be updated
                  ✔ updates dAPI name and buys subscription (118ms)
                dAPI name does not need to be updated
                  ✔ buys subscription (190ms)
              Subscription is not added to the start of the queue
```

```
                Current subscription ID does not need to be updated
                  dAPI name needs to be updated
                    ✔ updates dAPI name and buys subscription (199ms)
                  dAPI name does not need to be updated
                    ✔ buys subscription (134ms)
                Current subscription ID needs to be updated
                  dAPI name needs to be updated
                    ✔ updates current subscription ID, updates dAPI name and buys subscription (219ms)
                  dAPI name does not need to be updated
                    ✔ updates current subscription ID and buys subscription (209ms)
            Payment transfer fails
              ✔ reverts (97ms)
          Payment amount is zero
            ✔ buys subscription (99ms)
        Payment is not enough to get the sponsor wallet balance over the expected amount
          ✔ reverts (44ms)
      New subscription cannot be added to the queue...
        ...because its deviation reference differs from the subscriptions in the queue
          ✔ reverts (67ms)
        ...because its deviation threshold and heartbeat interval are not comparable to a subscription
in the queue
          ✔ reverts (65ms)
        ...because new subscription does not upgrade the queue
          ✔ reverts (61ms)
        ...because the queue is full
          ✔ reverts (241ms)
        ...because doing so will result in a dAPI name to be set to a stale data feed
          ✔ reverts (43ms)
        ...because doing so will result in a dAPI name to be set to an unregistered data feed
          ✔ reverts
        ...because doing so requires Api3Market to set a dAPI name and Api3Market does not have the
respective Api3ServerV1 role
          ✔ reverts (49ms)
    Arguments are not valid
      Data feed ID is zero
        ✔ reverts
      Sponsor wallet address is zero
        ✔ reverts
      dAPI management Merkle proof verification is not successful...
        ...because dAPI name is zero
          ✔ reverts
        ...because dAPI management Merkle data cannot be decoded
          ✔ reverts
        ...because dAPI management Merkle root is not registered
          ✔ reverts
        ... dAPI management Merkle proof is not valid
          ✔ reverts
      dAPI pricing Merkle proof verification is not successful...
        ...because update parameters length is invalid
          ✔ reverts
        ...because duration is zero
          ✔ reverts
        ...because price is zero
          ✔ reverts
        ...because dAPI pricing Merkle data cannot be decoded
          ✔ reverts
        ...because dAPI pricing Merkle root is not registered
          ✔ reverts
        ... dAPI pricing Merkle proof is not valid
          ✔ reverts
  updateCurrentSubscriptionId
    dAPI subscription queue is not empty
      Current subscription ID needs to be updated
        Queue will be empty after the current subscription ID is updated
          ✔ updates the current subscription ID and deactivates the dAPI (150ms)
        Queue will not be empty after the current subscription ID is updated
          ✔ updates the subscription ID and updates the update parameters (152ms)
      Current subscription ID does not need to be updated
        ✔ reverts (49ms)
    dAPI subscription queue is empty
      ✔ reverts
  updateDapiName
```

```
    Arguments are valid
      Data feed ID is different than what the dAPI name is currently set to
        Sets the dAPI name to a non-zero data feed ID
          Data feed is ready
            ✔ updates dAPI name
          Data feed is not ready
            Data feed is stale
              ✔ reverts
            Data feed has not been registered
              ✔ reverts
        Sets the dAPI name to zero data feed ID
          ✔ updates dAPI name
      Data feed ID is not different than what the dAPI name is currently set to
        ✔ reverts
    Arguments are not valid
      Sponsor wallet address is zero while data feed ID is not
        ✔ reverts
      Data feed ID is zero while sponsor wallet address is not
        ✔ reverts
      dAPI management Merkle proof verification is not successful...
        ...because dAPI name is zero
          ✔ reverts
        ...because dAPI management Merkle data cannot be decoded
          ✔ reverts
        ...because dAPI management Merkle root is not registered
          ✔ reverts
        ... dAPI management Merkle proof is not valid
          ✔ reverts
    updateSignedApiUrl
      Signed API URL Merkle proof verification is successful
        Signed API URL is different than that the signed API URL is currently set to
          ✔ updates signed API URL
        Signed API URL is not different than that the signed API URL is currently set to
          ✔ reverts
      Signed API URL Merkle proof verification is not successful...
        ...because signed API URL Merkle data cannot be decoded
          ✔ reverts
        ...because signed API URL Merkle root is not registered
          ✔ reverts
        ... signed API URL Merkle proof is not valid
          ✔ reverts
    updateBeaconWithSignedData
      ✔ updates Beacon with signed data
    updateBeaconSetWithBeacons
      ✔ updates Beacon set with Beacons
    deployDapiProxy
      ✔ deploys DapiProxy
    deployDapiProxyWithOev
      ✔ deploys DapiProxyWithOev
    registerDataFeed
      ✔ registers data feed
    computeExpectedSponsorWalletBalance
      ✔ computes expected sponsor wallet balance (122ms)
    computeExpectedSponsorWalletBalanceAfterSubscriptionIsAdded
      Update parameters length is valid
        ✔ computes expected sponsor wallet balance after subscription is added (89ms)
      Update parameters length is invalid
        ✔ reverts (82ms)
    getDapiData
      dAPI name is set to a Beacon set
        ✔ gets dAPI data (148ms)
      dAPI name is set to a Beacon
        ✔ gets dAPI data (132ms)
    getDataFeedData
      Data feed ID belongs to a Beacon set
        ✔ gets data feed data
      Data feed ID belongs to a Beacon
        ✔ gets data feed data
    subscriptionIdToUpdateParameters
      Subscription exists
        ✔ returns the update parameters of the subscription (48ms)
      Subscription does not exist
```

```
      ✔ returns empty bytes string


  153 passing (13s)
```

Code Coverage

The test suite achieves perfect branch coverage of the contracts in scope.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
AirseekerRegistry.sol	100	100	100	100	
Api3Market.sol	100	100	100	100	
HashRegistry.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
IAirseekerRegistry.sol	100	100	100	100	
IApi3Market.sol	100	100	100	100	
IHashRegistry.sol	100	100	100	100	
IOwnable.sol	100	100	100	100	
All files	100	100	100	100	

Changelog

- 2024-02-20 - Initial report
- 2024-02-27 - 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&aspo; 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