SHERLOCK

Security Review For Midas

Collaborative Audit Prepared For: **Midas**
Lead Security Expert(s): **pkqs90**
Date Audited: **July 31 - August 2, 2025**

Introduction

Midas focuses on creating secure, transparent, and efficient structures that allow investors to access the performance of select reference strategies through tokenized formats. All Midas products are issued with careful consideration of applicable regulatory frameworks in Europe and globally.

Scope

Repository: midas-apps/contracts

Audited Commit: c215b378026e24d8c1b9bc72078c7d53415336cf

Files:

- contracts/feeds/CustomAggregatorV3CompatibleFeedGrowth.sol
- contracts/interfaces/IAggregatorV3CompatibleFeedGrowth.sol

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	1	1

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue M-1: Using `updatedAt` instead of `startedAt` for staleness checks may misrepresent data freshness

Source: https://github.com/sherlock-audit/2025-07-midas-july-31st-2025/issues/3

This issue has been acknowledged by the team but won't be fixed at this time.

Summary

`DataFeed.sol` uses the `updatedAt` timestamp to indicate data freshness for `CustomAggregatorV3CompatibleFeed`, but this may not accurately reflect the true age of the price data, potentially causing staleness checks to pass when the underlying data is actually outdated.

Vulnerability Detail

In `CustomAggregatorV3CompatibleFeedGrowth`, when a new round is created in `setRoundData()`, the `updatedAt` field is set to `block.timestamp`, while the `startedAt` field records the actual initial timestamp of the price data:

https://github.com/sherlock-audit/2025-07-midas-july-31st-2025/blob/main/contracts/contracts/feeds/CustomAggregatorV3CompatibleFeedGrowth.sol#L229

```
_roundData[roundId] = RoundDataWithGrowth({
    roundId: roundId,
    answer: _data,
    startedAt: _dataTimestamp,
    updatedAt: block.timestamp,
    answeredInRound: roundId,
    growthApr: _growthApr
});
```

Downstream contracts, such as `DataFeed`, may use `updatedAt` to check if the feed is stale:

https://github.com/sherlock-audit/2025-07-midas-july-31st-2025/blob/main/contracts/contracts/feeds/DataFeed.sol#L161

```
require(
    block.timestamp - updatedAt <= healthyDiff &&
        _answer >= minExpectedAnswer &&
        _answer <= maxExpectedAnswer,
    "DF: feed is unhealthy"
);
```

However, in `CustomAggregatorV3CompatibleFeedGrowth`, the initial timestamp of the data is actually `startedAt`, since that is the beginning of the linear price function. `updatedAt` is

only the time the price is recorded onchain. So it makes more sense to check staleness against `startedAt`.

Impact

- Staleness checks may be bypassed, allowing outdated price data to be treated as fresh.

Recommendation

For DataFeed.sol, also use the `startedAt` timestamp for staleness checks.

Discussion

pkqs90

Team acknowledged the issue.

”The key idea here is that updatedAt reflects the actual timestamp when the price was aligned within the team and propagated onchain, and startedAt is just something precomputed and carefully selected value. In any way, we have safety checks that wont allow to propagate the price that is very different from the current one (unsafe method wont be used and wont be even whitelisted in the multisig that we using, so its designed only for the emergency cases)”.

Issue L-1: `getRoundData()` and `latestRoundData()` does not revert if uninitialized.

Source: https://github.com/sherlock-audit/2025-07-midas-july-31st-2025/issues/5

This issue has been acknowledged by the team but won't be fixed at this time.

Summary

The `getRoundData()` and `latestRoundData()` function does not revert when queried for a round that has not been initialized, which may lead to consumers receiving invalid or default data.

Vulnerability Detail

The `CustomAggregatorV3CompatibleFeedGrowth` contract implements a Chainlink AggregatorV3-compatible interface, including the `getRoundData()` function, which is expected to revert if the requested round is not initialized. However, the current implementation simply returns the default struct values for uninitialized rounds:

https://github.com/sherlock-audit/2025-07-midas-july-31st-2025/blob/main/contracts/contracts/feeds/CustomAggregatorV3CompatibleFeedGrowth.sol#L337

```
function getRoundData(uint80 _roundId)
    public
    view
    returns (
        uint80 roundId,
        int256 answer,
        uint256 startedAt,
        uint256 updatedAt,
        uint80 answeredInRound
    )
{
    RoundDataWithGrowth memory roundData = _roundData[_roundId];
    bool isLatestRound = _roundId == latestRound;

    return (
        roundData.roundId,
        applyGrowth(
            roundData.answer,
            roundData.growthApr,
            roundData.startedAt,
            isLatestRound
                ? block.timestamp
                : _roundData[_roundId + 1].updatedAt
        ),
```

```
        roundData.startedAt,
        roundData.updatedAt,
        roundData.answeredInRound
    );
}
```

According to the Chainlink AggregatorV3Interface specification, this function should revert if the round is not initialized. Returning default values may cause downstream consumers to process invalid data.

```
interface AggregatorV3Interface {
  // getRoundData and latestRoundData should both raise "No data present"
  // if they do not have data to report, instead of returning unset values
  // which could be misinterpreted as actual reported values.
  function getRoundData(uint80 _roundId)
    external
    view
    returns (
      uint80 roundId,
      int256 answer,
      uint256 startedAt,
      uint256 updatedAt,
      uint80 answeredInRound
    );
  function latestRoundData()
    external
    view
    returns (
      uint80 roundId,
      int256 answer,
      uint256 startedAt,
      uint256 updatedAt,
      uint80 answeredInRound
    );

}
```

Impact

There is no impact for the Midas system, because the only use case for Aggregator is DataFeed, which has a check for `answer > 0`. However, it is still recommended to make this check to prevent future integration issues.

Recommendation

Add a check if `updatedAt` is zero, revert.

Discussion

pkqs90

Team acknowledged the issue.

”It's implemented in the same way in the original custom aggregator, so we would keep current impl for the consistency”.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.