PUBLIC

# Code Assessment

## of the Capped Oracle Feed Smart Contracts

March 12, 2026

Produced for

Sky

by

CHAINSECURITY

# Contents


1 Executive Summary 3
2 Assessment Overview 5
3 Limitations and use of report 7
4 Terminology 8
5 Open Findings 9

# 1 Executive Summary

Dear all,

Thank you for trusting us to help Sky with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of Capped Oracle Feed according to Scope to support you in forming an opinion on their security risks.

Sky implements an immutable Chainlink-compatible oracle wrapper that caps the reported price of an underlying feed by an immutable maximum.

The subjects covered in our audit are functional correctness, trust assumptions on external dependencies, and unit testing. Security regarding all the aforementioned subjects is high.

In summary, we find that the codebase provides a high level of security.

It is important to note that security audits are time-boxed and cannot uncover all vulnerabilities. They complement but don't replace other vital measures to secure a project.

The following sections will give an overview of the system, our methodology, the issues uncovered, and how they have been addressed. We are happy to receive questions and feedback to improve our service.

Sincerely yours,

ChainSecurity

## 1.1 Overview of the Findings

Below we provide a brief numerical overview of the findings and how they have been addressed.

| | |
|---|---|
| Critical-Severity Findings | 0 |
| High-Severity Findings | 0 |
| Medium-Severity Findings | 0 |
| Low-Severity Findings | 0 |

# 2 Assessment Overview

In this section, we briefly describe the overall structure and scope of the engagement, including the code commit which is referenced throughout this report.

## 2.1 Scope

The following file is in scope of this review:

```
src/
    interfaces/MinAggregatorV3Interface.sol
    CappedOracleFeed.sol
```

The assessment was performed on the source code files inside the Capped Oracle Feed repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

| V | Date | Commit Hash | Note |
|---|---|---|---|
| 1 | 04 Mar 2026 | 7a07072f00a5712ca33dc4fad269c816ed752aae | Initial Version |

For the solidity smart contracts, the compiler version `0.8.28` was chosen and evm version is set to `prague`.

### 2.1.1 Excluded from scope

Any other files are out of scope.

## 2.2 System Overview

This system overview describes the initially received version (Version 1) of the contracts as defined in the Assessment Overview.

Furthermore, in the findings section, we have added a version icon to each of the findings to increase the readability of the report.

Sky implements an oracle wrapper that caps the price of an underlying Chainlink-like feed by a maximum price.

The purpose of `CappedOracleFeed` is to prevent upwards depegs for stablecoin prices on Morpho (see morpho-blue-oracles). Since Morpho only requires `latestRoundData()` and `decimals()`, only these two functions are implemented. `latestRoundData()` returns the data of the immutable `source` oracle but with `answer` capped by the immutable `maxPrice`. `decimals()` returns 8 (same as `source`).

## 2.3 Trust Model

The contract has no privileged roles, no access control, and no mutable state. It is not upgradeable.

- `source` (trusted): The contract delegates to the source feed and trusts it to return well-formed round data, and that `decimals()` remains `8` (validated at deployment, never rechecked). Staleness and validity checks (e.g. `answer > 0`) are intentionally omitted and consumers are expected to apply their own.

• Other addresses (untrusted): No actions besides querying `view` functions.

# 3 Limitations and use of report

Security assessments cannot uncover all existing vulnerabilities; even an assessment in which no vulnerabilities are found is not a guarantee of a secure system. However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it. Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code assessment aimed at determining all locations that need to be fixed. Within the customer-determined time frame, ChainSecurity has performed an assessment in order to discover as many vulnerabilities as possible.

The focus of our assessment was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself. Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.

# 4 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- *Likelihood* represents the likelihood of a finding to be triggered or exploited in practice
- *Impact* specifies the technical and business-related consequences of a finding
- *Severity* is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

| Likelihood | Impact | | |
|---|---|---|---|
| | High | Medium | Low |
| High | Critical | High | Medium |
| Medium | High | Medium | Low |
| Low | Medium | Low | Low |

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

# 5 Open Findings

In this section, we describe our findings. The findings are split into these different categories:

Below we provide a numerical overview of the identified findings, split up by their severity.

| | |
|---|---|
| Critical-Severity Findings | 0 |
| High-Severity Findings | 0 |
| Medium-Severity Findings | 0 |
| Low-Severity Findings | 0 |