

# Code Assessment of the Core & Oracles V3.10 Smart Contracts

March 31, 2025

Produced for



by



# Contents

<b>1</b>	<b>Executive Summary</b>	<b>3</b>
<b>2</b>	<b>Assessment Overview</b>	<b>5</b>
<b>3</b>	<b>Limitations and use of report</b>	<b>9</b>
<b>4</b>	<b>Terminology</b>	<b>10</b>
<b>5</b>	<b>Open Findings</b>	<b>11</b>
<b>6</b>	<b>Resolved Findings</b>	<b>12</b>
<b>7</b>	<b>Informational</b>	<b>14</b>

# 1 Executive Summary

Dear Gearbox Team,

Thank you for trusting us to help Gearbox with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of Core & Oracles V3.10 according to [Scope](#) to support you in forming an opinion on their security risks.

Gearbox implements V3.10 of the Core protocol and the oracles. The new version aims to make the system compatible with the new governance module. Moreover, it adds partial liquidations.

The most critical subjects covered in our audit are the correctness and potential regressions of the refactored code, the correctness of partial liquidations and the new Tumbler rate keeper. Security regarding all the aforementioned subjects is high.

The general subjects covered are gas efficiency, testing, documentation and specification. 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.

<b>Critical</b> -Severity Findings	0
<b>High</b> -Severity Findings	0
<b>Medium</b> -Severity Findings	1
• <b>Code Corrected</b>	1
<b>Low</b> -Severity Findings	3
• <b>Code Corrected</b>	3

## 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 assessment was performed on the source code files inside the Core & Oracles V3.10 repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

#### Core

V	Date	Commit Hash	Note
1	8 Jan 2025	<a href="#">1ba776314057e3dbcb05d0fb65e4d86547e830d1</a>	Initial Version
2	15 Jan 2025	<a href="#">66f9b8e7be833964c935cd43d1e8002b9f08d9ea</a>	Version updated during review
3	24 Feb 2025	<a href="#">f9894cb5ce8c0f6fd3dd4b233e8d72f953d2fb05</a>	First fixes iteration
4	31 Mar 2025	<a href="#">562ccc19210fe43c170c4451eb35fb786982ca43</a>	Final Version

#### Oracles

V	Date	Commit Hash	Note
1	8 Jan 2025	<a href="#">951b71670465e2f1b86d3140117d6e74cf55ee7a</a>	Initial Version
2	17 Mar 2025	<a href="#">fc8d3a0ab5bd7eb50ce3f6b87dde5cd3d887bafef</a>	Final Version

#### Integrations

V	Date	Commit Hash	Note
1	31 Mar 2025	<a href="#">361fb5c04df11a42d43bd46a2befcf98da6a8ce3</a>	Initial Version
2	31 Mar 2025	<a href="#">9e56cb66d59ab27bad0c04339d3b401c230e7ae2</a>	Final Version

For the `Core` solidity smart contracts, the compiler version `0.8.17` was chosen.

For the `Oracles` solidity smart contracts, the compiler version `0.8.23` was chosen.

For the `Integrations` solidity smart contracts, the compiler version `0.8.23` was chosen.

For the `Core`, the following contracts are in scope:

```
core:
  BotListV3.sol
  DefaultAccountFactoryV3.sol
  GearStakingV3.sol
  PriceOracleV3.sol
credit:
  CreditAccountV3.sol
  CreditConfiguratorV3.sol
  CreditFacadeV3.sol
```

```

    CreditManagerV3.sol
    CreditManagerV3_USDT.sol
libraries:
    BalancesLogic.sol
    BitMask.sol
    CollateralLogic.sol
    Constants.sol
    CreditAccountHelper.sol
    CreditLogic.sol
    QuotasLogic.sol
    USDTFees.sol
pool:
    GaugeV3.sol
    LinearInterestRateModelV3.sol
    PoolQuotaKeeperV3.sol
    PoolV3.sol
    PoolV3_USDT.sol
    TumblerV3.sol
traits:
    ACLTrait.sol
    ContractsRegisterTrait.sol
    PriceFeedValidationTrait.sol
    ReentrancyGuardTrait.sol
    SanityCheckTrait.sol
    USDT_Transfer.sol

```

For the Oracles, the following contracts are in scope:

```

oracles:
    balancer:
        BPTStablePriceFeed.sol
        BPTWeightedPriceFeed.sol
    curve:
        CurveCryptoLPPPriceFeed.sol
        CurveStableLPPPriceFeed.sol
        CurveUSDPriceFeed.sol
    erc4626:
        ERC4626PriceFeed.sol
    lido:
        WstETHPriceFeed.sol
    mellow:
        MellowLRTPriceFeed.sol
    updatable:
        RedstonePriceFeed.sol
    yearn:
        YearnPriceFeed.sol
    BoundedPriceFeed.sol
    CompositePriceFeed.sol
    LPPPriceFeed.sol
    PriceFeedParams.sol
    SingleAssetLPPPriceFeed.sol
    ZeroPriceFeed.sol

```

For the Integrations, the scope is limited to the scope defined in the dedicated report: <https://www.chainsecurity.com/security-audit/gearbox-v3-integrations>.

## 2.1.1 Excluded from scope

Any contracts not explicitly listed above are out of the scope of this review, especially `FixedPoint` and `LogExpMath` are assumed to work correctly, `PendleTWAPPTPriceFeed` and `PythPriceFeed` are out of the scope of this review. Third-party libraries are out of the scope of this review. In particular, OpenZeppelin libraries and the contracts inherited by the Redstone price oracle are considered to function as expected. The parameters of the system are assumed to be properly set. The system integrates with external protocols through adapters which are assumed to operate properly, their implementation is not part of the current review. A dedicated review for the adapters can be found at [Gearbox V3.10 Integrations](#). The compatibility with previously deployed contracts, typically core systems of Gearbox V1 and V2, is out of the scope of this review. The configuration of the system by its admins is beyond the scope of this review. Configuration includes but is not limited to the selection of tokens which will be added to the system as well as their parametrization. Economic attacks to the protocol are beyond the scope of this review.

This review focuses on the changes from V3 to V3.10, for a holistic understanding of the system, refer to the reports of V3:

- Core: <https://www.chainsecurity.com/security-audit/gearbox-v3-core>
- Integrations: <https://www.chainsecurity.com/security-audit/gearbox-v3-integrations>
- Oracles: <https://www.chainsecurity.com/security-audit/gearbox-v3-oracles-2>

## 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.

Gearbox offers an upgrade to Gearbox protocol from Version 3 (V3) to Version 3.10 (V3.10). The upgrade affects all components of the protocol, namely the core, the oracles and the integrations. This review is concerned with the core and the oracles.

### 2.2.1 Core

For the Core the most important changes are the following:

- Partial liquidations: Gearbox allows credit accounts to hold a great number of tokens as collateral. As some of the tokens might not be liquid, an account might not be fully liquidatable since this might not be profitable for the liquidators. In this case, a credit account can only be partially liquidated. Liquidation can happen either when an account is unhealthy or when the facade is expired. A partial liquidation consists of a user providing an amount of underlying in exchange for an amount of a token held by the credit account. For the liquidation to be profitable, the liquidator gets the token on a discount (`liquidationDiscount` or `liquidationDiscountExpired`) compared to its relative price with the underlying. Moreover, not the whole repaid amount goes to debt decrease. A part of it goes to the treasury as a fee (`feeLiquidationExpired` or `feeLiquidation`).
- All tokens are quoted: V3 introduced the mechanism of quoted tokens. Users have to buy the capacity to hold a quoted token. The mechanism is explained in detail in our [GearboxV3 report](#). In V3.10, all tokens but the underlying are quoted. This greatly simplifies how the protocol handles the enabled tokens for a credit account. In V3.10, the only way to enable or disable a token is by updating the quota of a credit account.

- Phantom tokens: These tokens are implemented by Gearbox to represent non-tokenized positions (see Zircuit integration). The protocol can now handle such tokens.
- Tumbler: V3.10 aims to be more flexible about how the quota rates are determined. The Tumbler is a simplified version of the Gauge contract which allows the configurator of the contract to arbitrarily specify the rates.
- Access control: It has been simplified as the governance update will take care of it.
- Sanity checks: Various sanity checks have been implemented. A multicall will revert if an external call requiring safe prices is done on a CA holding some forbidden tokens.

## 2.2.2 Oracles

The changes in the oracles are mostly related to the bigger upgrade in the governance (see relevant report). The contract type is standardized. The new type makes use of this format: `DOMAIN::POSTFIX`. For more information about how the contract type is used please refer to the governance report. Moreover, the most privileged role of the price oracles is the owner instead of some addresses set in the ACL (access control list) contract.

## 2.2.3 Trust Model and Roles

As discussed already, the access control has been simplified as it's handled by the governance module. We discuss the new trust model in the governance report.

## 2.2.4 Version 2

**Version 2** introduces `AliasedLossPolicyV3`. It introduces a fine-grained control over the conditions that allow for liquidations with loss. This is a loss policy that supports 3 different modes:

- **Forbidden:** it doesn't allow for liquidations.
- **Permissioned:** only users with `LOSS_LIQUIDATOR` role
- **Permissionless:** any user can liquidate. In such a case, the configurator can set some aliased price feeds that are used to check that an account is indeed liquidatable.

Moreover, the `CreditConfigurator` can now make all the tokens but the underlying quoted by calling `makeAllTokensQuoted()`.



### 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 any open findings. Findings that have been resolved have been moved to the [Resolved Findings](#) section. The findings are split into these different categories:

- **Design**: Architectural shortcomings and design inefficiencies
- **Correctness**: Mismatches between specification and implementation

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

<b>Critical</b> -Severity Findings	0
<b>High</b> -Severity Findings	0
<b>Medium</b> -Severity Findings	0
<b>Low</b> -Severity Findings	0

## 6 Resolved Findings

Here, we list findings that have been resolved during the course of the engagement. Their categories are explained in the [Open Findings](#) section.

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

<b>Critical</b> -Severity Findings	0
<b>High</b> -Severity Findings	0
<b>Medium</b> -Severity Findings	1
<ul style="list-style-type: none"><li>• <a href="#">Excessive Gas Burnt for Static Calls</a> <b>Code Corrected</b></li></ul>	
<b>Low</b> -Severity Findings	3
<ul style="list-style-type: none"><li>• <a href="#">Missing Input Sanitization</a> <b>Code Corrected</b></li><li>• <a href="#">PQK Reads the Wrong Gauge on Gauge Update</a> <b>Code Corrected</b></li><li>• <a href="#">Wrong Specifications</a> <b>Code Corrected</b></li></ul>	
Informational Findings	1
<ul style="list-style-type: none"><li>• <a href="#">Inconsistent "Is Contract" Checks</a> <b>Code Corrected</b></li></ul>	

### 6.1 Excessive Gas Burnt for Static Calls

**Design** **Medium** **Version 1** **Code Corrected**

CS-GEARPROTOV310-005

`_tryWithdrawPhantomToken()` performs a `getPhantomTokenInfo()` staticcall to an arbitrary address. For a staticcall, 63/64 of the gas available is given by default. If state change takes place, during a staticcall, the call reverts by consuming all the gas. This case is quite usual given that WETH is extensively used within the system. WETH implements a fallback function which is invoked during the static call and performs a state change i.e., mints WETH for the received ETH. As a result, paths such as collateral withdrawal or partial liquidations become very expensive.

#### Code corrected:

The call is done with `staticCallOptionalSafe()`, which will burn at most 30\_000 gas in case of failure.

### 6.2 Missing Input Sanitization

**Design** **Low** **Version 1** **Code Corrected**

CS-GEARPROTOV310-004

In `TumblerV3`, `epochLength_` is never checked to be in an acceptable range

#### Code corrected:

The `epochLength_` is enforced to be at most 28 days.

## 6.3 PQK Reads the Wrong Gauge on Gauge Update

**Correctness** **Low** **Version 1** **Code Corrected**

CS-GEARPROTOV310-003

In the function `PoolQuotaKeeper.setGauge()`, the old gauge is queried for `isTokenAdded()`, but this information should be queried from the new gauge instead.

---

### Code corrected:

The code has been updated to query the new gauge.

## 6.4 Wrong Specifications

**Correctness** **Low** **Version 1** **Code Corrected**

CS-GEARPROTOV310-006

The specs of the `CreditManagerV3` constructor specify:

Adds pool's underlying as collateral token with `LT = 0`.

But the constructor sets:

```
ltUnderlying = PERCENTAGE_FACTOR - _liquidationPremium - _feeLiquidation
```

---

### Code corrected:

The specs have been corrected to reflect the actual implementation.

## 6.5 Inconsistent "Is Contract" Checks

**Informational** **Version 1** **Code Corrected**

CS-GEARPROTOV310-002

The codebase uses two different ways to check for an address being a contract, even though the underlying checks are the same, using only one of them would make the codebase more consistent.

- `address(0xabc).code.length == 0 / address(0xabc).code.length > 0`
  - `!Address.isContract(0xabc) / Address.isContract(0xabc)`
- 

### Code corrected:

All the checks for contract addresses should be done using the `Address.isContract` function.

# 7 Informational

We utilize this section to point out informational findings that are less severe than issues. These informational issues allow us to point out more theoretical findings. Their explanation hopefully improves the overall understanding of the project's security. Furthermore, we point out findings which are unrelated to security.

## 7.1 Misleading Error Name

**Informational** **Version 1**

CS-GEARPROTOV310-001

1. The error `UnknownMethodException` emitted by `CreditFacadeV3._multicall()` in the case `flags & SKIP_COLLATERAL_CHECK_FLAG != 0` is misleading as the function selector is known in the system, but may be disabled in some callpaths