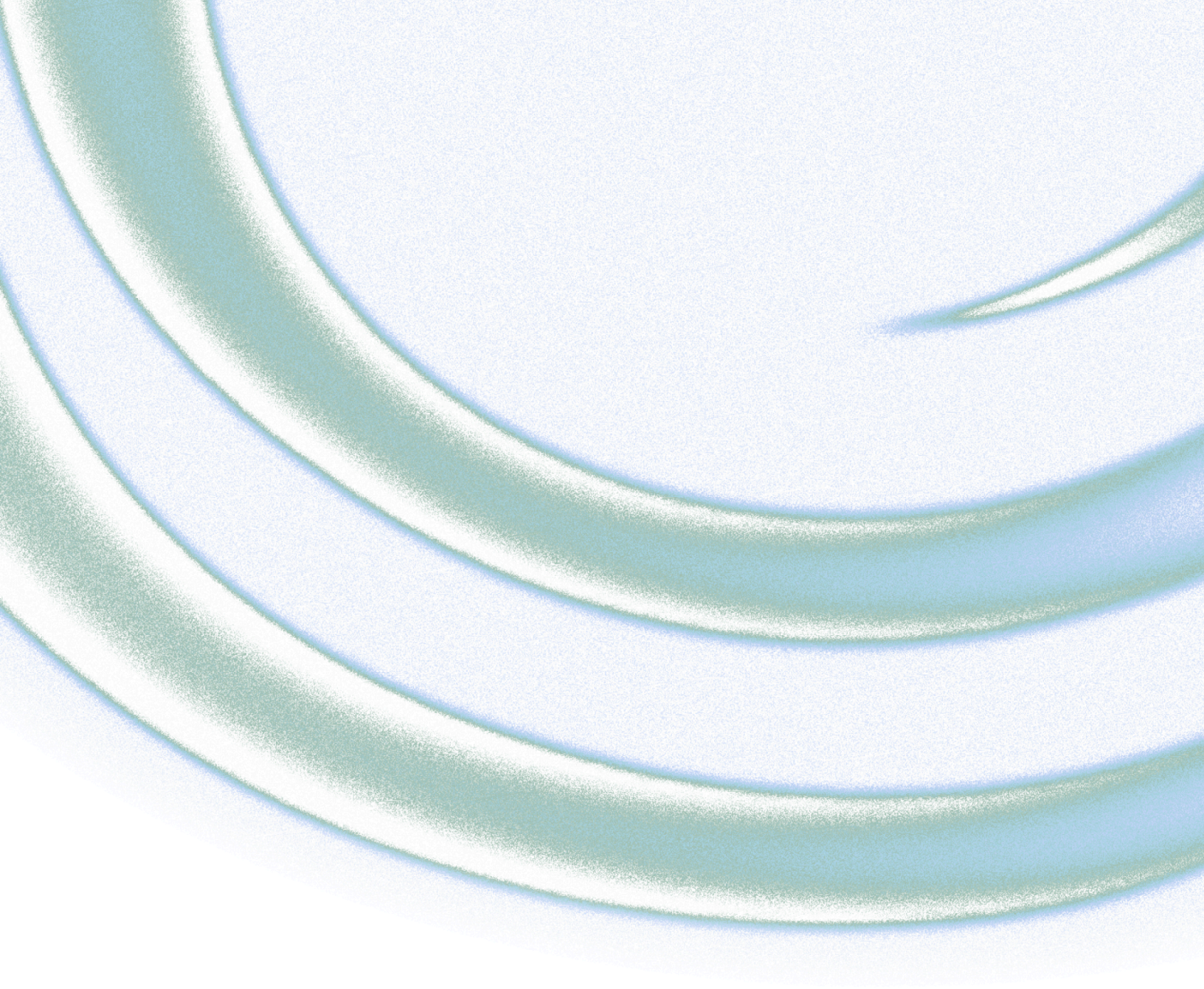


OtterSec

Fluid Oracle

Security Assessment

Prepared for
Fluid

Prepared by
Otter Audits, LLC

Prepared on
October 4th, 2026

Andreas Mantzoutas
andreas @osec.io

Thiago Tavares
thitav@osec.io

Table of Contents

1. Executive Summary 2
1.1. Overview 2
1.2. Key Findings 2
2. Scope 3
3. Findings 4
3.1. Findings Summary 4
4. Advisories 5
OS-FLR-ADV-00 Unversioned Oracle Quotes Bypass Security Upgrades 6
OS-FLR-ADV-01 Oracle Creation Allows Unusable Feed Configurations 7
5. Suggestions 8
OS-FLR-SUG-00 Clock Drift Rejects Fresh Oracle Prices 9
OS-FLR-SUG-01 Code Maturity 10
A. Vulnerability Rating Scale 11
B. Procedure 12

© 2026 Otter Audits LLC. All Rights Reserved.

1. Executive Summary

1.1. Overview

Fluid engaged OtterSec to conduct a security assessment of the `oracle` PR. This assessment was conducted between September 27th and September 29th, 2026. For information on our auditing methodology, refer to Appendix B.

1.2. Key Findings

We produced 4 findings throughout this audit engagement.

In particular, unversioned oracle view functions let callers use outdated package code to create type-identical PriceQuotes. Vaults may therefore accept prices computed with obsolete or vulnerable logic, bypassing later security fixes (● OS-FLR-ADV-00). Additionally, oracle creation does not verify that configured feeds exist or return valid prices, allowing an unusable oracle to be registered (● OS-FLR-ADV-01). Furthermore, strictly rejecting any future timestamp may discard valid, fresh prices when provider and Sui clocks drift slightly, potentially blocking vault operations and liquidations (● OS-FLR-SUG-00).

We also suggested emitting structured events for oracle initialization and every privileged role grant or revocation, rejecting grants to `0x0`, and making `AdminRole` implicitly satisfy `AuthorizeRole`, consistent with other Fluid packages (● OS-FLR-SUG-01).

© 2026 Otter Audits LLC. All Rights Reserved.

2. Scope

PR#36 replaces Fluid's Pyth-only multi-hop oracle with a dual-provider Pyth/RedStone design that returns the lower of the main and guard prices. It adds stricter price validation and graceful liquidation fallback, while making oracle configurations immutable and expanding integration tests.

The source code was delivered to us in a Git repository at `https://github.com/Instadapp/fluid-packages-sui`.

This audit was performed against PR#36.

© 2026 Otter Audits LLC. All Rights Reserved.

3. Findings

Findings were rated using the criteria in Appendix A.

We split the findings into **advisories** and **suggestions**. Advisories have an immediate impact and should be remediated as soon as possible. Suggestions do not have an immediate impact but will aid in mitigating future vulnerabilities.

Items are tracked with a stable identifier: `OS-FLR-ADV-*` for advisories and `OS-FLR-SUG-*` for suggestions.

3.1. Findings Summary

Severity	Count
Critical	0
High	0
Medium	0
Low	2
Informational	2

© 2026 Otter Audits LLC. All Rights Reserved.

4. Advisories

Here, we present a technical analysis of the 2 vulnerabilities we identified during our audit. These vulnerabilities have **immediate** security implications, and we recommend remediation as soon as possible.

ID	Severity	Status	Description
OS-FLR-ADV-00	Low	Resolved	Unversioned Oracle Quotes Bypass Security Upgrades
OS-FLR-ADV-01	Low	Ack'd	Oracle Creation Allows Unusable Feed Configurations

© 2026 Otter Audits LLC. All Rights Reserved.

Unversioned Oracle Quotes Bypass Security Upgrades

ID	Severity	Status
OS-FLR-ADV-00	Low	✓ Resolved

Description

Currently, the oracle implementation does not version-gate its functions. This is particularly impactful for view functions that compute prices and return a `PriceQuote`, as they remain callable regardless of the active oracle version. Functions such as `get_both_exchange_rate`, `get_exchange_rate_operate`, and `get_exchange_rate_liquidate` accept only `&Oracle`, `&PriceAdapter`, `&PriceInfoObject`, and `&Clock`, and neither `Oracle` nor `PriceQuote` carries version information. As a result, no read path invokes `registry.assert_version()`. Because Sui upgrades preserve type identity while keeping earlier package versions callable, users can invoke outdated view functions even after their pricing logic has been fixed. Those legacy functions may produce a `PriceQuote` indistinguishable from one generated by the latest implementation, allowing security fixes introduced by an upgrade to be bypassed.

Remediation

Require every quote function to call `registry.assert_version()`, and bind the validated version to `PriceQuote` so downstream vaults can verify that the quote was produced by the expected oracle version. Similarly, ensure that all state-mutating functions (such as authorization role setters) are version-gated.

Patch

Resolved in 95ac3ec.

© 2026 Otter Audits LLC. All Rights Reserved.

Oracle Creation Allows Unusable Feed Configurations

ID	Severity	Status
OS-FLR-ADV-01	Low	Ack'd

Description

Oracle creation validates source configuration structurally but does not confirm that the configured adapter, feed IDs, or price data are actually available and readable. An oracle can therefore be created successfully yet cause every subsequent quote request to abort, potentially rendering a bound vault market unusable.

Remediation

Attempt a validated read from each source so nonexistent feeds are rejected before the immutable oracle is registered.

Patch

Fluid acknowledged this issue.

© 2026 Otter Audits LLC. All Rights Reserved.

5. Suggestions

Here, we present a discussion of the 2 suggestions we identified during our audit. While these findings do not present an immediate security impact, they represent anti-patterns and may result in security issues in the future.

ID	Severity	Status	Description
OS-FLR-SUG-00	Informational	Ack'd	Clock Drift Rejects Fresh Oracle Prices
OS-FLR-SUG-01	Informational	Resolved	Code Maturity

© 2026 Otter Audits LLC. All Rights Reserved.

Clock Drift Rejects Fresh Oracle Prices

ID	Severity	Status
OS-FLR-SUG-00	Informational	Ack'd

Description

Both Pyth and RedStone reject a price whenever its publication timestamp is even slightly ahead of the Sui clock, despite benign clock drift being possible between these systems. Consequently, a fresh price may cause normal oracle reads to abort or be discarded by the liquidation fallback path. If both configured sources are marginally ahead, vault operations and liquidations can become temporarily unavailable.

Remediation

The validation should tolerate a bounded future offset and treat its age as zero, while continuing to reject timestamps beyond the permitted clock-skew allowance.

Patch

Fluid acknowledged this issue.

© 2026 Otter Audits LLC. All Rights Reserved.

Code Maturity

ID	Severity	Status
OS-FLR-SUG-01	Informational	✓ Resolved

Description

1. The oracle module emits no events when initializing oracle configurations or granting and revoking privileged roles. Consequently, off-chain indexers and monitoring systems cannot reliably track security-sensitive changes without parsing transactions or object-state mutations, reducing operational visibility and incident-response effectiveness.
2. The oracle's authorization behavior is inconsistent with the conventions followed by other Fluid packages. `auth::grant` allows roles to be assigned to `0x0`, creating unusable entries and potentially misleading role accounting. Moreover, `AdminRole` does not inherit `AuthorizeRole`, requiring administrators to grant themselves an additional role before creating an oracle. These differences increase operational complexity and the risk of governance or deployment misconfiguration.

Remediation

1. Emit structured events for oracle initialization and every privileged role grant or revocation.
2. Reject grants to `0x0` and make `AdminRole` implicitly satisfy `AuthorizeRole`, consistent with other Fluid packages.

Patch

Resolved in 95ac3ec.

© 2026 Otter Audits LLC. All Rights Reserved.

Appendix A.

Vulnerability Rating Scale

We rated our findings according to the following scale. Vulnerabilities have immediate security implications. Informational findings may be found in the general findings.

Severity	Description
Critical	Vulnerabilities that immediately result in a loss of availability, consensus divergence, or exploitable memory corruption with minimal preconditions.
High	Vulnerabilities that may result in consensus divergence or denial of service but require specific transaction or network conditions to exploit.
Medium	Vulnerabilities that may result in incorrect behavior or state divergence under specific configurations or edge cases.
Low	Low probability vulnerabilities, which are still exploitable but require extenuating circumstances or undue risk.
Informational	Best practices to mitigate future security risks. These are classified as general findings.

Resolution status is one of ✓ Resolved, ○ Ack'd, or × To-do.

© 2026 Otter Audits LLC. All Rights Reserved.

Appendix B.

Procedure

As part of our standard auditing procedure, we split our analysis into two main sections: design and implementation.

The design review asks whether the programs's architecture and operating assumptions support its intended behavior. We study the inputs it accepts, the decisions it makes, the information shared between components, and the services or participants it trusts. We then consider how it behaves when those inputs are unexpected or conflicting. The goal is to find conditions that can produce an incorrect decision, inconsistent state, or loss of service.

Design problems often arise from interactions rather than one defective function. Two components may each process a value correctly but interpret it differently. A failure may also leave one component ready to continue while another treats the same operation as incomplete. Local checks can look correct in both cases, so we follow the complete operation across component boundaries.

The implementation review asks whether the code matches the design. We trace untrusted data from its entry point to the point where it affects behavior, checking the assumptions made along the way. The review covers validation, error handling, state updates, concurrency, persistence, resource use, and cleanup. Its exact focus depends on the target's language, runtime, and architecture rather than a fixed list of bug classes.

The boundary between design and implementation is not exact. Missing validation may be a local coding error, while the same omission across several components may reflect an unclear rule. We classify a finding by its root cause, but judge its impact from the resulting behavior and the conditions needed to trigger it.

We begin by reading the target as a team and agreeing on its main components, trust boundaries, and expected behavior. Auditors then review different paths through the system and compare notes where those paths meet. We reproduce suspected issues with focused tests or traces when the setup allows. Before reporting a finding, another auditor reviews its trigger, impact, and proposed remediation.

While sometimes the line between design and implementation can be blurry, we hope this gives some insight into our auditing procedure and thought process.

© 2026 Otter Audits LLC. All Rights Reserved.