Yo Protocol
Security Review

Report Version 1.0

December 1, 2025

Conducted on **Aetheryc**

Table of Contents

1 About Aetheryc

Aetheryc is a leading network of security researchers specializing in smart contract code reviews.

For inquiries, visit aetheryc.com.

2 Disclaimer

Audits are a time-, resource-, and expertise-bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can reveal the presence of vulnerabilities, but cannot guarantee their absence.

3 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

3.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - involves a small loss of funds or affects a core functionality of the protocol.
- **Low** - encompasses any unexpected behavior that is non-critical.

3.2 Likelihood

- **High** - a direct attack vector; the cost is relatively low compared to the potential loss of funds.
- **Medium** - only a conditionally incentivized attack vector, with a moderate likelihood.
- **Low** - involves too many or unlikely assumptions; offers little to no incentive.

3.3 Actions required by severity level

- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

4 Executive summary

Overview

Project Name	Yo Protocol
Repository	https://github.com/yoprotocol/core-private
Commit hash	df8b7636ab1fcac9d10935b5ebe2388897d2ba21
Resolution	25736de4dd198f7b482e59563405c46b8074a67b
Methods	Manual review & testing

Scope

src/YoOracle.sol
src/YoToken.sol
src/YoVault_V2.sol

Issues Found

High risk	0
Medium risk	1
Low risk	1
Informational	1

5 Findings

5.1 Medium

5.1.1 Missing oracle share price validation

Severity: Medium

Description: The vault's conversion functions fetch prices from the oracle without validating that the price is non-zero. If the oracle returns a zero price (during initialization, malfunction, or manipulation), the conversions would return 0 assets when minting and redeeming effectively.

Recommendation: Add validation for non-zero price value.

Resolution: Resolved.

5.2 Low

5.2.1 Incorrectly tracking anchor price

Severity: Low

Description: When the anchor window expires, *d.anchorPrice* is set to *d.latestPrice* instead of *_sharePrice*:

```
if (nowTs - d.anchorTimestamp >= windowSeconds) {
    d.anchorPrice = d.latestPrice;
    d.anchorTimestamp = nowTs;
```

This creates a temporal inconsistency where the anchor price (taken from the previous update) is paired with the current block timestamp. The effective next window period would become longer than intended if the oracle fails to deliver updates for a certain period.

Recommendation: Consider using *_sharePrice* rather than *d.latestPrice*.

Resolution: Resolved.

5.3 Informational

5.3.1 Non-critical issues and suggestions

Severity: Informational

Description: The contracts contain one or more non-critical issues. In an effort to keep the report size reasonable, we enumerate these below:

1. *totalAssets()* could be overriden to return *_convertToAssets(totalSupply())* for more accurate result.

Recommendation: Consider fixing the above non-critical issues and suggestions.

Resolution: Acknowledged.