yAudit

February 2026

Prepared for
HAI

Audited by
Adriro
HHK

HAI CurveStableSwapNG Oracle Review

Smart Contract Security Assessment

Contents

1 Review Summary

1.1 Protocol Overview

HAI CurveStableSwapNG Oracle provides oracle infrastructure for price feeds. The reviewed component is a Curve StableSwap-NG relayer that combines Curve's EMA price oracle with external rate providers to deliver WAD-scaled price feeds via the IBaseOracle interface.

1.2 Audit Scope

This audit covers 2 smart contracts totaling approximately 180 lines of code across 1 day of review.

```
src/contracts/
       ├── oracles/CurveStableSwapNGRelayer.sol
       └── factories/CurveStableSwapNGRelayerFactory.sol
```

1.3 Risk Assessment Framework

1.3.1 Severity Classification

Severity	Description	Potential Impact
Critical	Immediate threat to user funds or protocol integrity	Direct loss of funds, protocol compromise
High	Significant security risk requiring urgent attention	Potential fund loss, major functionality disruption
Medium	Important issue that should be addressed	Limited fund risk, functionality concerns
Low	Minor issue with minimal impact	Best practice violations, minor inefficiencies
Undetermined	Findings whose impact could not be fully assessed within the time constraints of the engagement. These issues may range from low to critical severity, and although their exact consequences remain uncertain, they present a sufficient potential risk to warrant attention and remediation.	Varies based on actual severity
Gas	Findings that can improve the gas efficiency of the contracts.	Increased transaction costs
Informational	Code quality and best practice recommendations	Reduced maintainability and readability

Table 1: severity classification

1.4 Key Findings

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	1
Informational	3

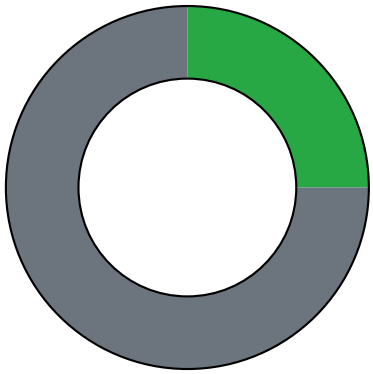

Figure 1: Distribution of security findings by impact level

1.5 Overall Assessment

The codebase is well-structured and minimal in scope. No critical or high severity issues were found. Findings are limited to a low severity precision loss, a gas optimization, and two informational issues around oracle reliability and factory ergonomics.

2 Audit Overview

2.1 Project Information

Protocol Name: HAI
Repository: https://github.com/hai-on-op/core
Commit Hash: e0f2bff8d8e95c2995da2df44e3b884fbdbfa5e6
Commit URL:
https://github.com/hai-on-op/core/blob/e0f2bff8d8e95c2995da2df44e3b884fbdbfa5e6

2.2 Audit Team

Adriro, HHK

2.3 Audit Timeline

The audit was conducted from February 12 to 13, 2026.

2.4 Audit Resources

Code repositories and documentation

2.5 Critical Findings

None.

2.6 High Findings

None.

2.7 Medium Findings

None.

2.8 Low Findings

2.8.1 Unnecessary precision loss in `_adjustForOracleRates()` due to intermediate WAD truncation

`_adjustForOracleRates()` computes `_price.wmul(_baseOracleRate).wdiv(_quoteOracleRate)`, which introduces an avoidable rounding step.

Technical Details

The expression `_price.wmul(_baseOracleRate).wdiv(_quoteOracleRate)` expands to:

```
((_price * _baseOracleRate / WAD) * WAD) / _quoteOracleRate
```

The intermediate division by `WAD` in `wmul()` truncates before the subsequent multiplication in `wdiv()`. This can be simplified to `_price * _baseOracleRate / _quoteOracleRate`, which avoids the intermediate truncation and preserves more precision. Since all three values are in WAD scale, the single division by `_quoteOracleRate` (1e18-denominated) produces the correct WAD-scaled result.

Impact

Low.

Recommendation

Replace the chained `wmul`/`wdiv` with a single multiplication and division:

```
1 return _price * _baseOracleRate / _quoteOracleRate;
```

Developer Response

Acknowledged. This is the same pattern used throughout the codebase.

2.9 Gas Savings Findings

2.9.1 State variables can be made immutable to save gas

All state variables in `CurveStableSwapNGRelayer` are set once in the constructor and never modified, but are not marked as `immutable`.

2.9.2 Technical Details

The following variables at CurveStableSwapNGRelayer.sol#L21-40 can be `immutable`: - `pool` - `baseToken` - `quoteToken` - `symbol` - `baseIndex` - `quoteIndex` - `baseRateMultiplier` - `quoteRateMultiplier`

2.9.3 Impact

Gas.

2.9.4 Recommendation

Consider making all variables `immutable`.

2.9.5 Developer Response

Acknowledged.
Originally I had these marked as immutable but if they are immutable we cannot use .runtimeCode in our unit tests.

2.10 Informational Findings

2.10.1 EMA price adjusted with current external rates can enable sudden price changes

The relayer uses Curve's manipulation-resistant EMA but adjusts it with current rates from external oracles, which can change suddenly.

2.10.2 Technical Details

The relayer combines two data sources (CurveStableSwapNGRelayer.sol#L82-94): - `price_oracle()`: EMA of historical prices (slow-moving, manipulation-resistant) - `stored_rates()`: Current rates via external calls (can change instantly)
For pools on Optimism with asset*type 1 (custom rate providers) or asset*type 3 (ERC4626 vaults), `stored_rates()` makes external calls to fetch current rates. The relayer multiplies the EMA by the current rate ratio.
Example: wstETH / ETH pool - EMA shows 1.0 (balanced pool in xp space) - Historical wstETH rate: 1.15 - Current wstETH rate: 1.20 (sudden increase from staking rewards or oracle update) - Relayer immediately reports adjusted price using rate 1.20
While this gives the correct current execution price, the EMA's manipulation resistance is partially bypassed since rates can spike instantly (e.g., ERC4626 donation inflation, or sudden oracle updates).

2.10.3 Impact

Informational. Low risk in most situation with battle-tested rate providers / high TVL ERC4626 vaults.

2.10.4 Recommendation

No ideal fix (can't dynamically bound rates without storing history). Consider: - Setting hardcoded boundaries, to restrict how high/low a rate can go - Only deploying relayers on pools with well-established ERC4626 vaults and robust rates manipulation protection

2.10.5 Developer Response

Acknowledged. This is mitigated through the selection of assets.

2.10.6 `getResultWithValidity()` may revert due to external dependency in `stored_rates()`

`CurveStableSwapNGRelayer.getResultWithValidity()` calls `pool.stored_rates()`, which depending on the pool's asset types may query an external oracle or invoke an ERC4626 `convertToAssets()` call. Either of these can revert, violating the `IBaseOracle` requirement that `getResultWithValidity()` should never revert.

Technical Details

Curve StableSwap-NG pools support multiple asset types. For type 1 (oracle) assets, `stored_rates()` queries an external price oracle; for type 3 (ERC4626) assets, it calls `convertToAssets()` on the vault. If the external oracle is paused, deprecated, or the ERC4626 vault reverts (e.g., due to a paused state), the entire `stored_rates()` call will revert. This causes `getResultWithValidity()` to revert instead of returning `(0, false)`, breaking the contract specified by `IBaseOracle` which states that this method should never revert.

Impact

Informational.

Recommendation

Consider returning `(0, false)` instead of bubbling the error if an external call reverts.

Developer Response

Fixed in commit 19ad26fe2d15e04c377531e76de3b9ec8ebbb2b7.

2.10.7 Missing membership getter for `_curveStableSwapNGRelayers` set

`CurveStableSwapNGRelayerFactory` stores deployed relayers in an `EnumerableSet.AddressSet` and exposes `curveStableSwapNGRelayersList()` to retrieve all entries, but provides no way to check whether a specific address belongs to the set.

Technical Details

The factory uses OpenZeppelin's `EnumerableSet` which natively supports `contains()`, but this function is never exposed. Consumers that need to verify whether a given oracle was deployed by the factory must iterate over the full list returned by `curveStableSwapNGRelayersList()`, which is gas-inefficient and impractical for on-chain callers.

Impact

Informational.

Recommendation

Add a public view that wraps `_curveStableSwapNGRelayers.contains()`:

```
function isCurveStableSwapNGRelayer(address _relayer) external view returns (bool) {
  return _curveStableSwapNGRelayers.contains(_relayer);
}
```

Developer Response

Fixed in commit fa1dff9fa41bd77c32201bced3a35ba661731a8d.

2.11 Final Remarks

The CurveStableSwapNG relayer is a compact and focused oracle integration. The main risk surface is the reliance on external rate providers via stored_rates(), which can revert or introduce sudden price changes. These risks are low in practice when restricted to battle-tested pools, as acknowledged by the team. The identified issues were promptly addressed. Overall the codebase is in good shape for its scope.