ENIGMA DARK

Securing the Shadows

Managed Security Review

# Alto Oracles + Staking

November 25, 2025

## Contents

## Summary

**Enigma Dark**
Enigma Dark is a web3 security firm leveraging the best talent in the space to secure all kinds of blockchain protocols and decentralized apps. Our team comprises experts who have honed their skills at some of the best auditing companies in the industry. With a proven track record as highly skilled white-hats, they bring a wealth of experience and a deep understanding of the technology and the ecosystem.

Learn more about us at enigmadark.com

**Alto**
Alto is a decentralized credit protocol built around DUSD. Isolated markets, a single stablecoin, and partial liquidations keep risk contained while giving you clear, sustainable ways to grow capital.

## Engagement Overview

Over the course of 1 week, the Enigma Dark team conducted a security review of the Alto project. The review was performed by Kiki.

The following repositories were reviewed at the specified commits:

| Repository | Commit |
|---|---|
| altomoney/v1 | `4bff7afb040ee83dce14016b2c206009ce91a759` |

## Risk Classification

| Severity | Description |
|---|---|
| Critical | Vulnerabilities that lead to a loss of a significant portion of funds of the system. |
| High | Exploitable, causing loss or manipulation of assets or data. |
| Medium | Risk of future exploits that may or may not impact the smart contract execution. |
| Low | Minor code errors that may or may not impact the smart contract execution. |
| Informational | Non-critical observations or suggestions for improving code quality, readability, or best practices. |

## Vulnerability Summary

| Severity | Count | Fixed | Acknowledged |
|---|---|---|---|
| Critical | 0 | 0 | 0 |
| High | 0 | 0 | 0 |
| Medium | 1 | 0 | 1 |
| Low | 3 | 1 | 2 |
| Informational | 7 | 5 | 2 |

## Findings

| Index | Issue Title | Status |
|---|---|---|
| M-01 | ETH feed misprices wstETH during depeg | Acknowledged |
| L-01 | Container removal distorts past epoch TVL proportions | Acknowledged |
| L-02 | Growth cap uses linear not compound interest | Acknowledged |
| L-03 | Cooldown can be set to zero enabling attacks | Fixed |
| I-01 | Typo in container activation comment | Fixed |
| I-02 | Redundant multiplier zero check in UpdateContainer | Fixed |
| I-03 | Lost voting power opportunity from deactivated containers | Acknowledged |
| I-04 | Off by one error in updatePrice function | Fixed |
| I-05 | Comment incorrectly claims code is unchanged | Fixed |
| I-06 | Oracle nodes lack token decimal validation | Acknowledged |
| I-07 | Swap event emits input not actual sold | Fixed |

# Detailed Findings

## High Risk

No issues found.

## Medium Risk

### M-01 - ETH feed misprices wstETH during depeg

**Severity**: Medium Risk

**Context**:

- ERC4626RatioChainlinkOracleWSTETH.sol#L27-L33

**Technical Details**:

The `ERC4626RatioChainlinkOracleWSTETH` oracle calculates wstETH price by multiplying the wstETH/stETH ratio from `stEthPerToken` with the configured `chainlinkNode` price feed. The oracle design assumes the Chainlink feed provides ETH/USD pricing, treating stETH as equivalent to ETH. During stETH depeg events where stETH trades below ETH, the oracle overvalues wstETH collateral by pricing it at the ETH rate rather than the actual stETH market rate. When stETH depegs to 0.95 ETH and wstETH ratio is 1.05, the oracle calculates wstETH value as 1.05 × ETH price instead of 1.05 × 0.95 × ETH price, inflating collateral value by approximately 5%. Arbitrageurs borrow against overvalued wstETH collateral at the inflated oracle price and immediately sell the borrowed assets for profit, using the protocol as exit liquidity while actual collateral value remains lower.

**Impact**:

Protocol accepts overvalued wstETH collateral during stETH depeg events, enabling arbitrageurs to extract value.

**Recommendation**:

Configure the oracle to use a stETH/USD Chainlink price feed instead of ETH/USD to accurately reflect stETH market price during depeg events.

**Developer Response**: Acknowledged. The above usage of oracle has been suggested by our risk management team who have done thorough analysis of it.

## Low Risk

### L-01 - Container removal distorts past epoch TVL proportions

**Severity**: Low Risk

**Context**:

- AltoStaking.sol#L251-L272

**Technical Details**:
The `_advanceEpoch` function processes unprocessed epochs by calling `_processEpoch` for each epoch. `_processEpoch` calculates total TVL via `_getTotalTvlForEpoch`, which sums TVL only from containers in `activeContainerIds`. When a container is deactivated via `deactivateContainer`, it is removed from `activeContainerIds`. If epochs are not advanced before deactivation, later `_advanceEpoch` processes past epochs using the current `activeContainerIds`, excluding containers that were active in those past epochs but are now deactivated. This excludes their TVL from the total, inflating the TVL share of remaining active containers. `_processContainer` then computes lock durations based on these incorrect proportions, causing `currentDurationEpochs` to increase more than it should have at those past epochs.

**Impact**:
Past epochs get processed with incorrect TVL proportions when containers are deactivated before epochs are advanced, leading to incorrect lock duration calculations for all containers.

**Recommendation**:
Advance epochs to current epoch before activating or deactivating containers in `activateContainer` and `deactivateContainer` functions to prevent past epoch processing errors.

**Developer Response**: Acknowledged. This will be documented.

## L-02 - Growth cap uses linear not compound interest

**Severity**: Low Risk

**Context**:

- ModuleERC4626Ratio.sol#L46

**Technical Details**:

The `_computeMaxRatio` function calculates the maximum allowed ratio growth using simple linear interest rather than compound interest. The calculation `maxAllowedGrowthRatio = maxGrowthPerYear * timePassed / MATH_PRECISION / SECONDS_PER_YEAR` applies the same fixed growth amount for each year based on `initialRatio`. With 5% annual growth starting at ratio 1.0, linear growth produces 1.05 after year one and 1.1 after year two, while compound growth produces 1.05 after year one and 1.1025 after year two. Liquid staking tokens naturally compound, meaning actual ratios follow compound growth patterns. Over time, the linear cap becomes increasingly restrictive relative to the actual compounded ratio, causing `readLatestRatio` to return the capped value instead of the true ratio, artificially suppressing collateral valuations in lending markets.

**Impact**:

Long-term LST collateral becomes systematically undervalued as the linear growth cap diverges from compound growth, reducing borrowing capacity and health of the account.

**Recommendation**:

Consider Implementing a `maxRatio` calculation to better align with the rate of change of LST's.

**Developer Response**: Acknowledged. We can update the ratio at periodic interval so that above issue won't happen.

## L-03 - Cooldown can be set to zero enabling attacks

**Severity**: Low Risk

**Context**:

- AltoRewardsOracle.sol#L113

**Technical Details**:

The `updatePriceUpdateCooldownPeriod` function validates only that the new `_priceUpdateCooldownPeriod` differs from the current value but does not prevent setting it to zero. When `priceUpdateCooldownPeriod` equals zero, the check `block.timestamp - lastPriceUpdateTimestamp < priceUpdateCooldownPeriod` in `updatePrice` always evaluates to false, allowing `updatePrice` to be called in consecutive blocks or even multiple times within the same block via different transactions. An attacker exploits this by using a flash loan to manipulate the Uniswap V3 pool price, calling `updatePrice` to cache the manipulated price, then using that cached price in `AltoRewardsDistributor` to purchase ALTO tokens at the distorted rate.

**Impact**:

Attackers manipulate the ALTO token price oracle using flash loans to purchase rewards at incorrect prices

**Recommendation**:

Add a validation check requiring `_priceUpdateCooldownPeriod` to be greater than zero in the `updatePriceUpdateCooldownPeriod` function.

**Developer Response**: Fixed at commit `288f91f`.

# Informational

## I-01 - Typo in container activation comment

**Severity**: Informational

**Context**:

- TVLWeightedContainers.sol#L352

**Technical Details**:
The NatSpec comment for `_activateContainer` contains a typo where "conainer" should be "container". The comment also contains "actiation" which should be "activation".

**Recommendation**:
Fix the typo by changing "conainer" to "container" and "actiation" to "activation" in the comment.

**Developer Response**: Fixed at commit `031ff1f`.

## I-02 - Redundant multiplier zero check in UpdateContainer

**Severity**: Informational

**Context**:

- AltoStaking.sol#L245

**Technical Details**:
The `updateContainer` function calls `_updateContainerMultiplier`, which reverts with `AltoStakingInvalidInput` if `multiplier` is zero meaning after this call, execution only continues if `multiplier` is non-zero. However, there is a check later in the function that is therefore always true, making it redundant.

```
if (multiplier != 0) {
    _updateMaxMultiplierStakingContainerId();
}
```

**Recommendation**:
Remove the redundant check and always call `_updateMaxMultiplierStakingContainerId` since `_updateContainerMultiplier` already validates the input is not zero.

**Developer Response**: Fixed at commit `be03d83`.

## I-03 - Lost voting power opportunity from deactivated containers

**Severity**: Informational

**Context**:

- AltoStaking.sol#L186

**Technical Details**:
When a container is deactivated via `deactivateContainer`, it is removed from the `activeContainerIds` array. The `getVotes` function iterates only through `activeContainerIds`, so positions in deactivated containers stop contributing to voting power immediately. However, the `unlock` function only permits unlocking when the position is expired via `_isExpiredPosition` or when the contract is in rescue mode. This means users with positions in deactivated containers lose voting power immediately but cannot unlock their tokens until `position.expiryTimestamp` is reached, leaving tokens locked without voting benefits until the timelock expires.

**Recommendation**:
Consider allowing early unlocks when container is deactivated. Or document this behavior for users.

**Developer Response**: Acknowledged. This is as per design and referenced in our documentation.

## I-04 - Off by one error in updatePrice function

**Severity**: Informational

**Context:**

AltoRewardsOracle.sol#L93

**Technical Details:**

The `updatePrice` function first validates that the cooldown has passed before any state changes occur. However, the check only validates that the time since the last update is less than `priceUpdateCooldownPeriod`. The issue with this is that the full duration of `priceUpdateCooldownPeriod` is not being utilized as this check will pass when the time delta is equal to `priceUpdateCooldownPeriod`. Meaning if the cooldown period was set to 60 seconds then the price can be updated at exactly 60 seconds making the cooldown check only valid for the first 59 seconds. And allowing users to update the price sooner than expected.

**Recommendation:**

Modify the cooldown check to use `<=` instead of `<`.

**Developer Response:** fixed at commit `a4a2dd2`.

## I-05 - Comment incorrectly claims code is unchanged

**Severity**: Informational

**Context**:

- OracleMath.sol#L8

**Technical Details**:

The `OracleMath` contract includes a NatSpec comment at line 8 stating the code is "Unchanged from Angle Core repository," but the implementation contains slight modifications to the `_getQuoteAtTick` function. Specifically the use of the enum `ConversionDirection` as opposed to the unit256 which is used as a boolean. Thought there is no logical change by doing this it should still be documented.

**Recommendation**:

Update the comment to indicate the contract is "Forked and modified from Angle Core repository" to accurately reflect that the implementation includes changes.

**Developer Response**: Fixed at commit `0affaf7`.

## I-06 - Oracle nodes lack token decimal validation

**Severity**: Informational

**Context**:

- AltoRewardsOracle.sol#L56
- AltoRewardsOracle.sol#L123
- AltoRewardsOracle.sol#L131

**Technical Details**:

The `AltoRewardsOracle` constructor and update functions set `chainlinkNode` and `uniswapNode` parameters without validating that decimal configurations match the actual Uniswap pool tokens. When `chainlinkNode` is set in the constructor or via `updateChainlinkNode`, the `_validateChainlinkParams` function checks basic parameter validity but does not verify that `chainlinkNode.inDecimals` matches the decimals of the correct token in the Uniswap pool.

**Recommendation**:

Consider adding validation in constructor and update functions to verify `chainlinkNode.inDecimals` matches the correct token decimals from the Uniswap pool.

**Developer Response**: Acknowledged. It will be ensured during deployment that proper parameters are set.

## I-07 - Swap event emits input not actual sold

**Severity**: Informational

**Context**:

- AltoLeverageSwapper.sol#L80

**Technical Details**:

The `Swap` event in the `swap` function emits `amountIn` as the amount of `sellToken` sold, but this value represents the input amount approved and transferred to `SWAP_TARGET`, not the actual amount consumed by the swap. The 0x protocol swap executed via `SWAP_TARGET.call` may not utilize the full `amountIn`, with any leftover tokens returned to the caller via `sellTokenBalanceLeftover`. Off-chain indexers and monitoring services consuming this event will record potentially inflated swap volumes, as they receive the input amount rather than the executed amount. While the event does include `sellTokenBalanceLeftover` as a separate parameter, services expecting `amountIn` to represent the actual sold amount will misinterpret the data.

**Recommendation**:

Add NatSpec documentation to the `Swap` event clarifying that `amountIn` represents the input amount approved for the swap, and may not be the actual amount sold.

**Developer Response**: Fixed at commit `73bb6cf`.

## Disclaimer