Audit Report: GPT-4o - 1

Title

Audit Report GPT-4o - 1: ARSUSDTOracle Contract

Summary

This audit analyzes the `ARSUSDTOracle` smart contract developed for securely managing and publishing ARS/USDT exchange rates on-chain. The contract enables off-chain sources (via an owner or multisig) to update prices, ensuring low operational costs compared to external oracles such as Chainlink Functions.

Scope

- Contract: `ARSUSDTOracle`
- Language: Solidity ^0.8.20
- Auditor: GPT-4o
- Date: July 2025

Architecture Overview

The `ARSUSDTOracle` contract is designed to: - Store a price (`latestAnswer`) manually updated by an owner (or multisig). - Emit an event when the price is updated. - Allow consumers to query the latest price and check data freshness.

Security Review

Access Control

✅ Uses `Ownable` from OpenZeppelin to restrict `updatePrice` to the contract owner. ✅Clear, minimal ownership model reduces attack surface. ⚠️Recommendation: Use a multisig wallet or a timelock mechanism for additional security.

Data Integrity

✅Implements an `isStale` check to ensure consumers can verify if data is fresh. ✅Provides revert conditions for stale data and zero price (`ARSUSDTOracle__StaleData` , `ARSUSDTOracle__PriceError`).

Event Emission

✅Emits `PriceUpdated` event, providing transparency and off-chain indexing support.

Error Handling

✅Uses custom errors for gas efficiency and clarity.

Upgradeability

⚠️Contract is not upgradeable; this is acceptable for an oracle with simple logic but should be noted.

Denial-of-Service Risks

✅No loops or external calls that could lead to DoS vectors. ✅Gas usage is predictable and minimal.

Gas Efficiency

✅Efficient use of storage: Only two state variables (`latestAnswer`, `lastUpdateTimestamp`). ✅Proper use of custom errors instead of revert strings to save gas. ✅Minimal logic in `updatePrice` and view functions, reducing transaction costs.

Code Quality and Best Practices

✅Follows Solidity style conventions and explicit visibility. ✅Well-structured and readable. ✅Uses `immutable` constructor ownership setup (via `Ownable`). ⚠️Comment: Comments could include examples for scaled price formats for clarity.

Recommendations

- Consider implementing a multisig owner or timelock for `updatePrice`.
- Provide off-chain monitoring to ensure the `updatePrice` function is called regularly to maintain fresh data.
- Document price scaling details clearly for integrators.
- Potentially include versioning or future upgrade hooks if the oracle design needs to evolve.

Conclusion

The `ARSUSDTOracle` contract is secure, simple, and gas-efficient. It is well-suited as a manual or semi-automated oracle for ARS/USDT price feeds, providing a practical alternative to more expensive automated oracles.

Audit Result: PASSED with minor recommendations