Hourglass
Euler Price Oracle
Security Audit Report

PREPARED FOR:

Charlie Pyle

Hourglass Foundation

ARCADIA CONTACT INFO

Email: audits@arcadiamgroup.com

Telegram: https://t.me/thearcadiagroup

Revision history

Date	Reason	Commit
1/14/2025	Initial Audit Scope	#3c393c2c49504c1d91549aa1c6697ef589f90487
1/20/2025	Delivery	

THE ARCADIA GROUP

Table of Contents

Executive Summary

1. Introduction and Audit Scope

Hourglass Foundation engaged Arcadia to perform a security audit of their PT & CT Linear Discount Rate Oracle; our review of their codebase occurred in the repo euler-xyz/euler-price-oracle on the commit hash #3c393c2c49504c1d91549aa1c6697ef589f90487. The review consisted of only the Hourglass Oracle Implementation on the Pull Request #77.

a. Review Team

Van Cam Pham - Lead Security Engineer

b. Project Background

Hourglass is a protocol that facilitates liquidity for time-locked and semi-fungible assets.

c. Coverage

For this audit, we performed research, test coverage, investigation, and review of Hourglass's Euler Oracle Implementation, followed by issue reporting and mitigation and remediation instructions as outlined in this report. The following code repositories, files, and/or libraries are considered in scope for the review.

Files
src/adapter/hourglass/HourglassOracle.sol
src/adapter/hourglass/IHourglassDepositor.sol
src/adapter/hourglass/IHourglassERC20TBT.sol
test/adapter/hourglass/HourglassAddresses.sol
test/adapter/hourglass/HourglassOracle.fork.t.sol
test/adapter/hourglass/HourglassOracle.prop.t.sol

test/adapter/hourglass/HourglassOracle.unit.t.sol
test/adapter/hourglass/HourglassOracleHelper.sol

2. Audit Summary

a. Audit Methodology

Arcadia completed this security review using various methods, primarily consisting of dynamic and static analysis. This process included a line-by-line analysis of the in-scope contracts, optimization analysis, analysis of key functionalities and limiters, and reference against intended functionality.

The following are the steps performed while auditing the smart contracts:

- Investigating the project and its technical architecture overview through its documentation
- Understanding the overview of the smart contracts, the functions of the contracts, the inheritance, and how the contracts interface with each other thanks to the graph created by Solidity Visual Developer
- Manual smart contract audit:
    - Review the code to find any issue that could be exploited by known attacks listed by Consensys
    - Identifying which existing projects the smart contracts are built upon and what are the known vulnerabilities and remediations to the existing projects
    - Line-by-line manual review of the code to find any algorithmic, design and/or arithmetic-related vulnerabilities compared to what should be done based on the project's documentation
    - Find any potential code that could be refactored to save gas
    - Run through the unit-tests and test-coverage if exists

- Static Analysis:
    - Scanning for vulnerabilities in the smart contracts using Static Code Analysis Software
    - Making a static analysis of the smart contracts using Slither
- Additional review: a follow-up review is done when the smart contracts have any new updates. The follow-up is done by reviewing all changes compared to the audited commit revision and its impact on the existing source code and found issues.

b. Summary

We thoroughly reviewed the PT & CT Linear Discount Rate Oracle. Our findings indicated no findings in the code of the Oracle smart contracts.

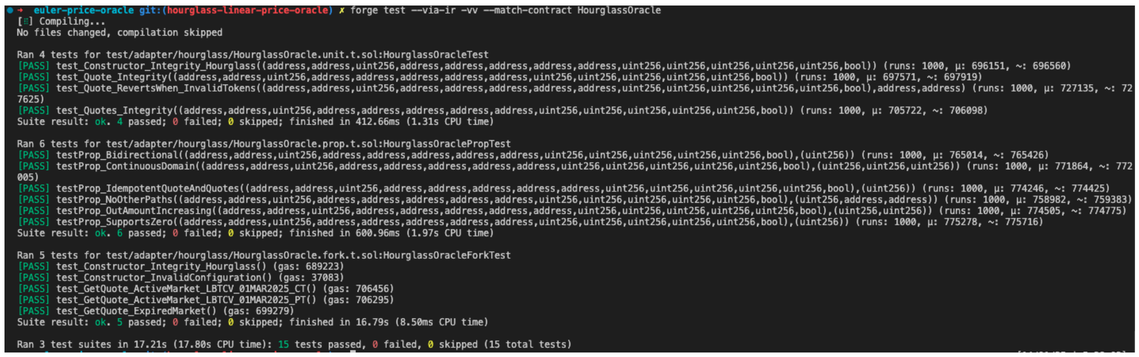
One element we flagged on an informational level during the engagement (already known to the client) is that a number of the contracts used as data sources are either upgradeable or have privileged functionality through roles that could lead to unintended data inputs or failures through upgrades of underlying contracts or the use of other privileged roles. Most of the contracts deployed are done so by the Hourglass Foundation and utilize OpenZeppelin's upgrade safe Upgradeable Plugin and are unlikely to introduce breaking changes that could disrupt data flow. Nevertheless, we have outlined the contracts below for the sake of completeness.

Contract Name	Contract Address	Owner	Description
HOURGLASS_LBTCV_01MAR2025_DEPOSITOR	0xf06617fBECF1BdEa2D62079bdab9595f86801604	Hourglass Foundation	OZ Upgradeable Beacon Proxy
HOURGLASS_LBTCV_01MAR2025_CT	0xe6dA3BD04cEEE35D6A52fF329e57cC2220a669b1	Hourglass Foundation	OZ ERC20Upgradeable
HOURGLASS_LBTCV_01M	0x97955073caA92028a86Cd	Hourglass	OZ ERC20Upgradeable

AR2025_PT	3F660FE484d6B89B938	Foundation	
HOURGLASS_LBTCV_01DEC2024_DEPOSITOR	0xA285bca8f01c8F18953443e645ef2786D31ada99	Hourglass Foundation	OZ Upgradeable Beacon Proxy
HOURGLASS_LBTCV_01DEC2024_CT	0x0CB35DC9ADDce18669E2Fd5db4B405Ea655e98Bd	Hourglass Foundation	OZ ERC20Upgradeable
HOURGLASS_LBTCV_01DEC2024_PT	0xDB0Ee7308cF1F5A3f376D015a1545B4cB9A878D9	Hourglass Foundation	OZ ERC20Upgradeable
LBTCV	0x5401b8620E5FB570064CA9114fd1e135fd77D57c	ether.fi	Privileged Roles on Contract (i.e. enter minter)

c. Unit Tests

At the hash reviewed by Arcadia, the Unit Tests (described in the scope) were found to cover the Oracle implementation sufficiently and were all passing.



```
● ➜  euler-price-oracle git:(hourglass-linear-price-oracle) ✗ forge test --via-ir -vv --match-contract HourglassOracle
 [⠊] Compiling...
 No files changed, compilation skipped

 Ran 4 tests for test/adapter/hourglass/HourglassOracle.unit.t.sol:HourglassOracleTest
 [PASS] test_Constructor_Integrity_Hourglass((address,address,uint256,address,address,address,address,address,uint256,uint256,uint256,uint256,uint256,bool)) (runs: 1000, μ: 696151, ~: 696560)
 [PASS] test_Quote_Integrity((address,address,uint256,address,address,address,address,address,uint256,uint256,uint256,uint256,uint256,bool)) (runs: 1000, μ: 697571, ~: 697919)
 [PASS] test_Quote_RevertsWhen_InvalidTokens((address,address,uint256,address,address,address,address,address,uint256,uint256,uint256,uint256,uint256,bool),address,address) (runs: 1000, μ: 727135, ~: 72
 7625)
 [PASS] test_Quotes_Integrity((address,address,uint256,address,address,address,address,address,uint256,uint256,uint256,uint256,uint256,bool)) (runs: 1000, μ: 705722, ~: 706098)
 Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 412.66ms (1.31s CPU time)

 Ran 6 tests for test/adapter/hourglass/HourglassOracle.prop.t.sol:HourglassOraclePropTest
 [PASS] testProp_Bidirectional((address,address,uint256,address,address,address,address,address,uint256,uint256,uint256,uint256,uint256,bool),(uint256)) (runs: 1000, μ: 765014, ~: 765426)
 [PASS] testProp_ContinuousDomain((address,address,uint256,address,address,address,address,address,uint256,uint256,uint256,uint256,uint256,bool),(uint256,uint256,uint256)) (runs: 1000, μ: 771864, ~: 772
 005)
 [PASS] testProp_IdempotentQuoteAndQuotes((address,address,uint256,address,address,address,address,address,uint256,uint256,uint256,uint256,uint256,bool),(uint256)) (runs: 1000, μ: 774246, ~: 774425)
 [PASS] testProp_NoOtherPaths((address,address,uint256,address,address,address,address,address,uint256,uint256,uint256,uint256,uint256,bool),(uint256,address,address)) (runs: 1000, μ: 758982, ~: 759383)
 [PASS] testProp_OutAmountIncreasing((address,address,uint256,address,address,address,address,address,uint256,uint256,uint256,uint256,uint256,bool),(uint256,uint256)) (runs: 1000, μ: 774505, ~: 774775)
 [PASS] testProp_SupportsZero((address,address,uint256,address,address,address,address,address,uint256,uint256,uint256,uint256,uint256,bool),(uint256)) (runs: 1000, μ: 775278, ~: 775716)
 Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 600.96ms (1.97s CPU time)

 Ran 5 tests for test/adapter/hourglass/HourglassOracle.fork.t.sol:HourglassOracleForkTest
 [PASS] test_Constructor_Integrity_Hourglass() (gas: 689223)
 [PASS] test_Constructor_InvalidConfiguration() (gas: 37083)
 [PASS] test_GetQuote_ActiveMarket_LBTCV_01MAR2025_CT() (gas: 706456)
 [PASS] test_GetQuote_ActiveMarket_LBTCV_01MAR2025_PT() (gas: 706295)
 [PASS] test_GetQuote_ExpiredMarket() (gas: 699279)
 Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 16.79s (8.50ms CPU time)

 Ran 3 test suites in 17.21s (17.80s CPU time): 15 tests passed, 0 failed, 0 skipped (15 total tests)
```

Disclaimer