Lien Protocol

New Volatility Oracle

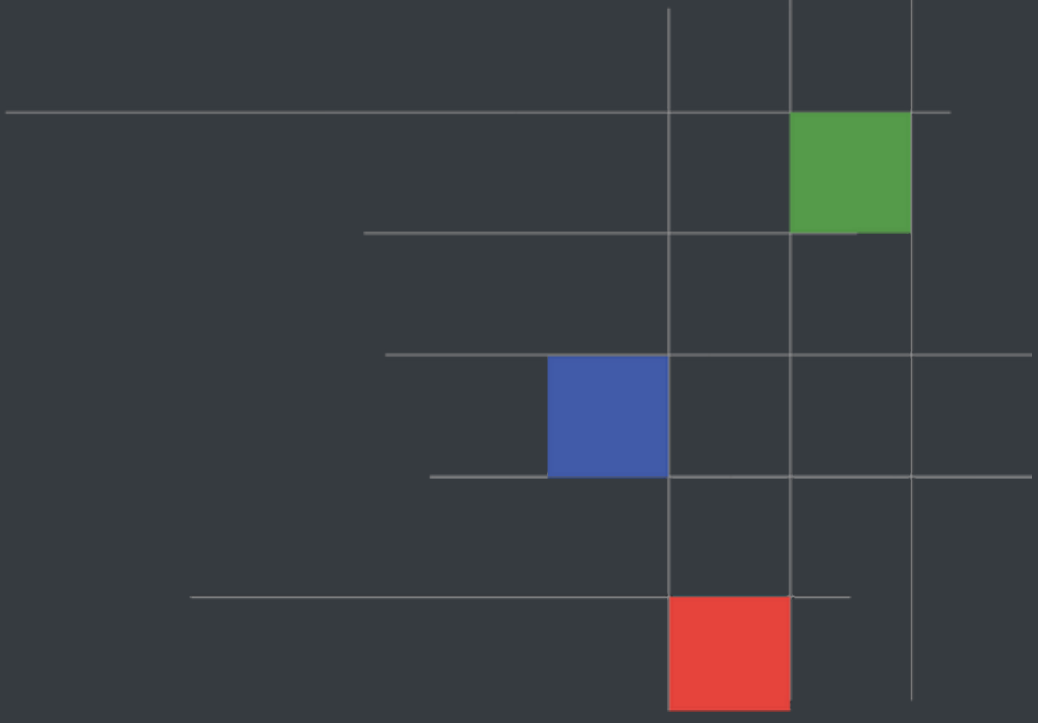
Security Assessment

March 6th, 2021

By:
Alex Papageorgiou @ CertiK
alex.papageorgiou@certik.org

Camden Smallwood @ CertiK
camden.smallwood@certik.org

Sheraz Arshad @ CertiK
sheraz.arshad@certik.org



Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Lien Protocol - NewVolatilityOracle Smart Contracts
Description	Smart contracts of the NewVolatilityOracle repository.
Platform	Ethereum; Solidity
Codebase	GitHub Repository
Commits	1. 38f36065ae51841c6fcf444a074579a3f2f76dc8 2. f1b6997333f6b3131d2de9524525d5ff2485893f 3. ce7bd2fbc4506410ec54cfcdf4459c11e32cb7c8

Audit Summary

Delivery Date	Mar. 06, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	3
Timeline	Feb. 08, 2021 - Mar. 06, 2021

Vulnerability Summary

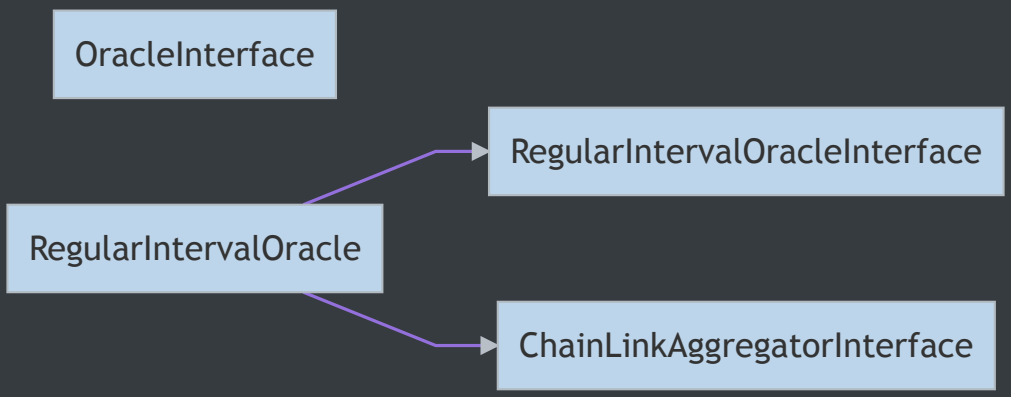
Total Issues	38 - *36 Resolved, 2 Acknowledged*
🔴 **Total Critical**	0
🟠 **Total Major**	2 - *2 Resolved*
🟡 **Total Medium**	5 - *5 Resolved*
🔵 **Total Minor**	17 - *16 Resolved, 1 Acknowledged*
🟢 **Total Informational**	14 - *13 Resolved, 1 Acknowledged*

Executive Summary

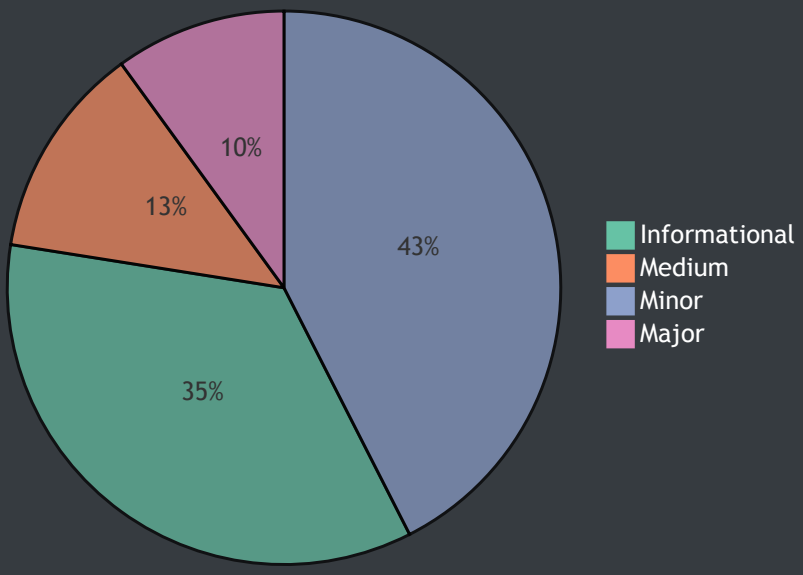
The code within the NewVolatilityOracle repository was primarily found to be well written, with the exception of the implementation of the RegularIntervalOracle contract where we identified multiple cases of unchecked primitive arithmetic operations, missing parameter sanitization in the constructor, an unsound square root implementation, two unspecified cases of open access control, and multiple points where integer values can overflow and underflow or truncate. These issues should be investigated prior to deployment. See the Findings section for more information.

Files In Scope

ID	Contract	Location
CLA	ChainLinkAggregatorInterface	contracts/ChainLinkAggregator/ChainLinkAggregatorInterface.sol
ORI	OracleInterface	contracts/oracle/OracleInterface.sol
RIO	RegularIntervalOracle	contracts/oracle/RegularIntervalOracle.sol
RII	RegularIntervalOracleInterface	contracts/oracle/RegularIntervalOracleInterface.sol



Findings



ID	Title	Type	Severity	Resolved
RIO-01	Unnecessary relative import path	Implementation	Informational	⏲
RIO-02	State variable lacks immutability and visibility	Implementation	Informational	✓
RIO-03	Unspecified state variable visibility	Implementation	Informational	✓
RIO-04	Unspecified state variable visibility	Implementation	Informational	✓
RIO-05	Unspecified state variable visibility	Implementation	Informational	✓

RIO-06	Unspecified state variable visibility	Implementation	Informational	✓
RIO-07	Unspecified state variable visibility	Implementation	Informational	✓
RIO-08	Unspecified state variable visibility	Implementation	Informational	✓
RIO-09	Unspecified state variable visibility	Implementation	Informational	✓
RIO-10	Unspecified state variable visibility	Implementation	Informational	✓
RIO-11	Constructor lacks parameter value check	Volatile Code	Minor	✓
RIO-12	Constructor lacks address verification	Volatile Code	Minor	✓
RIO-13	Constructor lacks address verification	Volatile Code	Minor	✓
RIO-14	Constructor lacks parameter value check	Volatile Code	Minor	✓
RIO-15	Lack of access restriction on setPrice	Control Flow	Major	✓
RIO-16	Potential integer overflow in setPrice	Arithmetic	Minor	✓
RIO-17	Lack of access restriction on setSequentialPrices	Control Flow	Major	✓
RIO-18	Unverified primitive arithmetic in setSequentialPrices	Arithmetic	Medium	✓
RIO-19	Inefficient loop over array length	Gas Optimization	Informational	✓
RIO-20	Unverified primitive arithmetic in setOptimizedParameters	Arithmetic	Medium	✓
RIO-21	No event emitted for external state variable change	Implementation	Minor	✓
RIO-22	Unverified primitive arithmetic in setOptimizedParameters	Arithmetic	Medium	✓
RIO-23	Potential integer overflow in setOptimizedParameters	Arithmetic	Minor	✓
RIO-24	Potential integer overflow in	Arithmetic	Minor	✓

	setOptimizedParameters			
RIO-25	Potential integer overflow in setOptimizedParameters	Arithmetic	Minor	✓
RIO-26	Lack of address verification in updateQuantsAddress	Volatile Code	Minor	✓
RIO-27	No event emitted for external state variable change	Implementation	Minor	✓
RIO-28	Unverified primitive arithmetic in _getEwmaVolatility	Arithmetic	Medium	✓
RIO-29	Potential interger overflow in _sqrt	Arithmetic	Minor	✓
RIO-30	Potential integer underflow in _getValidRoundID	Arithmetic	Minor	✓
RIO-31	Potential integer overflow in _getValidRoundID	Arithmetic	Minor	✓
RIO-32	Potential integer underflow in _setPrice	Arithmetic	Minor	✓
RIO-33	Potential integer underflow in _getPriceFromChainlink	Arithmetic	Minor	⏲
RIO-34	Unverified primitive arithmetic in getNormalizedTimeStamp	Arithmetic	Medium	✓
RIO-35	Potential integer overflow in getNormalizedTimeStamp	Arithmetic	Minor	✓
RIO-36	getInfo should be re-declared as external	Implementation	Informational	✓
RIO-37	getVolatility should be re-declared as external	Implementation	Informational	✓
RIO-38	getVolatilityTimeOf should be re-declared as external	Implementation	Informational	✓

RIO-01: Unnecessary relative import path

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L6

Description:

The `SafeCast` library is imported using a relative path, which is unnecessary.

Recommendation:

Consider replacing the relative import path with a module import path, starting at the `@` character.

Alleviation:

The recommendation was not taken into account.

RIO-02: State variable lacks immutability and visibility

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L23

Description:

The `_chainlinkOracle` state variable in the `RegularIntervalOracle` contract should be declared immutable and have its visibility specified.

Recommendation:

Consider declaring the `_chainlinkOracle` state variable in the `RegularIntervalOracle` contract as immutable.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-03: Unspecified state variable visibility

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L24

Description:

The `_interval` state variable in the `RegularIntervalOracle` contract should have its visibility specified.

Recommendation:

Consider specifying the visibility of the `_interval` state variable in the `RegularIntervalOracle` contract as internal or private.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-04: Unspecified state variable visibility

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L25

Description:

The `_decimals` state variable in the `RegularIntervalOracle` contract should have its visibility specified.

Recommendation:

Consider specifying the visibility of the `_decimals` state variable in the `RegularIntervalOracle` contract as internal or private.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-05: Unspecified state variable visibility

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L26

Description:

The `_timeCorrectionFactor` state variable in the `RegularIntervalOracle` contract should have its visibility specified.

Recommendation:

Consider specifying the visibility of the `_timeCorrectionFactor` state variable in the `RegularIntervalOracle` contract as internal or private.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-06: Unspecified state variable visibility

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L27

Description:

The `_oldestTimestamp` state variable in the `RegularIntervalOracle` contract should have its visibility specified.

Recommendation:

Consider specifying the visibility of the `_oldestTimestamp` state variable in the `RegularIntervalOracle` contract as internal or private.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-07: Unspecified state variable visibility

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L28

Description:

The `_dataNum` state variable in the `RegularIntervalOracle` contract should have its visibility specified.

Recommendation:

Consider specifying the visibility of the `_dataNum` state variable in the `RegularIntervalOracle` contract as internal or private.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-08: Unspecified state variable visibility

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L32

Description:

The `_quantsAddress` state variable in the `RegularIntervalOracle` contract should have its visibility specified.

Recommendation:

Consider specifying the visibility of the `_quantsAddress` state variable in the `RegularIntervalOracle` contract as internal or private.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-09: Unspecified state variable visibility

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L33

Description:

The `_latestTimestamp` state variable in the `RegularIntervalOracle` contract should have its visibility specified.

Recommendation:

Consider specifying the visibility of the `_latestTimestamp` state variable in the `RegularIntervalOracle` contract as internal or private.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-10: Unspecified state variable visibility

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L36

Description:

The `lambdaE4` state variable in the `RegularIntervalOracle` contract should have its visibility specified.

Recommendation:

Consider specifying the visibility of the `lambdaE4` state variable in the `RegularIntervalOracle` contract as internal or public.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-11: Constructor lacks parameter value check

Type	Severity	Location
Volatile Code	Minor	contracts/oracle/RegularIntervalOracle.sol L48, L63

Description:

The constructor of the `RegularIntervalOracle` contract does not perform a value check on the supplied `decimals` parameter before assigning it to the `_decimals` state variable.

Recommendation:

Consider adding a requirement that value of the supplied `decimals` parameter should be greater than zero.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-12: Constructor lacks address verification

Type	Severity	Location
Volatile Code	Minor	contracts/oracle/RegularIntervalOracle.sol L52, L60

Description:

The constructor of the `RegularIntervalOracle` contract does not perform a zero address check on the supplied `quantsAddress` parameter before assigning it to the `_quantsAddress` state variable.

Recommendation:

Consider adding a requirement that the supplied `quantsAddress` parameter should be non-zero.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-13: Constructor lacks address verification

Type	Severity	Location
Volatile Code	Minor	contracts/oracle/RegularIntervalOracle.sol L53, L61

Description:

The constructor of the `RegularIntervalOracle` contract does not perform a zero address check on the supplied `chainlinkOracleAddress` parameter before assigning it to the `_chainlinkOracle` state variable.

Recommendation:

Consider adding a requirement that the supplied `chainlinkOracleAddress` should be non-zero.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-14: Constructor lacks parameter value check

Type	Severity	Location
Volatile Code	Minor	contracts/oracle/RegularIntervalOracle.sol L55, L62

Description:

The constructor of the `RegularIntervalOracle` contract does not perform a value check on the supplied `interval` parameter before assigning it to the `_interval` state variable.

Recommendation:

Consider adding a requirement that the supplied `interval` parameter should be greater than zero.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-15: Lack of access restriction on setPrice

Type	Severity	Location
Control Flow	Major	contracts/oracle/RegularIntervalOracle.sol L82

Description:

The public `setPrice` function in the `RegularIntervalOracle` contract does not contain access restriction, which allows anyone to call it and potentially continually increment the `_latestTimestamp` state variable by the current value of the `_interval` state variable, before calling the internal `_getValidRoundID` and `_setPrice` functions with the modified `_latestTimestamp` value.

Recommendation:

Determine the accessibility of the `setPrice` function and consider refactoring in order to prevent manipulating the price by repetitive calls to the `setPrice` function.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-16: Potential integer overflow in setPrice

Type	Severity	Location
Arithmetic	Minor	contracts/oracle/RegularIntervalOracle.sol L83

Description:

The public `setPrice` function in the `RegularIntervalOracle` contract performs a primitive incrementation on the `_latestTimestamp` state variable with the `_interval` state variable without ensuring that either value is valid beforehand, which has the potential to overflow.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of an overflow.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-17: Lack of access restriction on setSequentialPrices

Type	Severity	Location
Control Flow	Major	contracts/oracle/RegularIntervalOracle.sol L92

Description:

The external `setSequentialPrices` function in the `RegularIntervalOracle` contract does not contain access restriction, which allows anyone to call it and potentially continually set prices for multiple rounds.

Recommendation:

Determine the accessibility of the `setSequentialPrices` function and consider refactoring in order to prevent manipulating the price by repetitive calls to the `setSequentialPrices` function.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-18: Unverified primitive arithmetic in setSequentialPrices

Type	Severity	Location
Arithmetic	Medium	contracts/oracle/RegularIntervalOracle.sol L104

Description:

The external `setSequentialPrices` function in the `RegularIntervalOracle` contract performs a primitive division on the result of a primitive subtraction without checking any of the values beforehand, which has the potential to result in underflow and division by zero.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of underflow or division by zero.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-19: Inefficient loop over array length

Type	Severity	Location
Gas Optimization	Informational	contracts/oracle/RegularIntervalOracle.sol L111

Description:

The external `setSequentialPrices` function in the `RegularIntervalOracle` contract performs a loop over the supplied `roundIds` array parameter while querying the length of the array during each iteration, which is inefficient.

Recommendation:

Consider storing the length of the supplied `roundIds` in a local `roundIdCount` variable and referencing this variable in the for loop requirement on line 111 in order to save on the overall cost of gas.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-20: Unverified primitive arithmetic in setOptimizedParameters

Type	Severity	Location
Arithmetic	Medium	contracts/oracle/RegularIntervalOracle.sol L133

Description:

The external `setOptimizedParameters` function in the `RegularIntervalOracle` contract performs a primitive division on the result of a primitive subtraction without checking any of the values beforehand, which has the potential to result in underflow and division by zero.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of underflow or division by zero.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-21: No event emitted for external state variable change

Type	Severity	Location
Implementation	Minor	contracts/oracle/RegularIntervalOracle.sol L136

Description:

The external `setOptimizedParameters` function in the `RegularIntervalOracle` contract assigns the supplied `newLambdaE4` parameter to the public `lambdaE4` state variable variable without emitting an event, which makes it difficult to track off-chain.

Recommendation:

Consider creating and emitting an event in order to track when the state of the `lambdaE4` state variable changes from within the `RegularIntervalOracle` contract.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-22: Unverified primitive arithmetic in setOptimizedParameters

Type	Severity	Location
Arithmetic	Medium	contracts/oracle/RegularIntervalOracle.sol L137

Description:

The external `setOptimizedParameters` function in the `RegularIntervalOracle` contract performs a primitive multiplication on the result of a primitive subtraction without checking any of the values beforehand, which has the potential to underflow and result in undesired values.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of underflow or division by zero.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-23: Potential integer overflow in setOptimizedParameters

Type	Severity	Location
Arithmetic	Minor	contracts/oracle/RegularIntervalOracle.sol L138

Description:

The external `setOptimizedParameters` function in the `RegularIntervalOracle` contract performs a primitive addition between the local `oldTimestamp` variable and the `_interval` state variable without checking either value beforehand, which has the potential to overflow.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of overflow.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-24: Potential integer overflow in setOptimizedParameters

Type	Severity	Location
Arithmetic	Minor	contracts/oracle/RegularIntervalOracle.sol L142

Description:

The external `setOptimizedParameters` function in the `RegularIntervalOracle` contract performs a primitive incrementation on the local `oldTimestamp` variable with the `_interval` state variable without checking either value beforehand, which has the potential to overflow.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of overflow.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-25: Potential integer overflow in setOptimizedParameters

Type	Severity	Location
Arithmetic	Minor	contracts/oracle/RegularIntervalOracle.sol L143

Description:

The external `setOptimizedParameters` function in the `RegularIntervalOracle` contract performs a primitive addition between the local `oldTimestamp` variable and the `_interval` state variable without checking either value beforehand, which has the potential to overflow.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of overflow.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-26: Lack of address verification in updateQuantsAddress

Type	Severity	Location
Volatile Code	Minor	contracts/oracle/RegularIntervalOracle.sol L154, L160

Description:

The external `updateQuantsAddress` function in the `RegularIntervalOracle` contract does not perform a zero address check on the supplied `quantsAddress` before assigning it to the `_quantsAddress` state variable.

Recommendation:

Consider adding a requirement that the supplied `quantsAddress` should be non-zero.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-27: No event emitted for external state variable change

Type	Severity	Location
Implementation	Minor	contracts/oracle/RegularIntervalOracle.sol L160

Description:

The external `updateQuantsAddress` function in the `RegularIntervalOracle` contract assigns the supplied `quantsAddress` to the `_quantsAddress` state variable without emitting an event, which makes it difficult to track off-chain.

Recommendation:

Consider creating and emitting an event in order to track when the `_quantsAddress` state variable is changed from within the `RegularIntervalOracle` contract.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-28: Unverified primitive arithmetic in _getEwmaVolatility

Type	Severity	Location
Arithmetic	Medium	contracts/oracle/RegularIntervalOracle.sol L198-L201

Description:

The internal `_getEwmaVolatility` function in the `RegularIntervalOracle` contract performs primitive arithmetic without checking if any of the values are valid across their domain beforehand.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of unsafe arithmetic operations.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-29: Potential interger overflow in _sqrt

Type	Severity	Location
Arithmetic	Minor	contracts/oracle/RegularIntervalOracle.sol L208-L216

Description:

The internal `_sqrt` function in the `RegularIntervalOracle` contract incorrectly implements the Babylonian method for calculating the square root of an integer due to the initial value of `z = (x + 1) / 2` on line 210, which can overflow in the case that `x` is the maximum `uint256` value, leading `z` to be set to zero and causing a division by zero on line 214, causing the function to revert unnecessarily.

Recommendation:

Consider refactoring the `_sqrt` implementation to be well-defined over its entire domain without reverting:

```
function _sqrt(uint256 x) internal pure returns (uint256 y) {
  if (x > 3) {
    uint z = x / 2 + 1;
    y = x;
    while (z < y) {
      y = z;
      z = (x / z + z) / 2;
    }
  } else if (x != 0) {
    y = 1;
  }
}
```

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-30: Potential integer underflow in _getValidRoundID

Type	Severity	Location
Arithmetic	Minor	contracts/oracle/RegularIntervalOracle.sol L231

Description:

The internal `_getValidRoundID` function in the `RegularIntervalOracle` contract performs a primitive subtraction on the supplied `hintID` value without checking its value beforehand, which has the potential to underflow.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of an integer underflow.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-31: Potential integer overflow in _getValidRoundID

Type	Severity	Location
Arithmetic	Minor	contracts/oracle/RegularIntervalOracle.sol L234

Description:

The internal `_getValidRoundID` function in the `RegularIntervalOracle` contract performs a primitive addition on the local `index` variable without checking its value beforehand, which has the potential to overflow.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of an integer overflow.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-32: Potential integer underflow in _setPrice

Type	Severity	Location
Arithmetic	Minor	contracts/oracle/RegularIntervalOracle.sol L245, L247

Description:

The internal `_setPrice` function in the `RegularIntervalOracle` contract performs a primitive subtraction between the supplied `timeStamp` parameter and the `_interval` state variable, which has the potential to underflow.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of an integer underflow.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-33: Potential integer underflow in _getPriceFromChainlink

Type	Severity	Location
Arithmetic	Minor	contracts/oracle/RegularIntervalOracle.sol L265

Description:

The internal `_getPriceFromChainlink` function in the `RegularIntervalOracle` contract performs a primitive decrement on the supplied `roundId` parameter without checking either value beforehand, which has the potential to underflow.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of an integer underflow.

Alleviation:

The recommendation was not taken into account.

RIO-34: Unverified primitive arithmetic in getNormalizedTimeStamp

Type	Severity	Location
Arithmetic	Medium	contracts/oracle/RegularIntervalOracle.sol L281

Description:

The public `getNormalizedTimeStamp` function in the `RegularIntervalOracle` performs a primitive subtraction between the supplied `timestamp` parameter and the `_timeCorrectionFactor` state variable without checking either value beforehand, which has the potential to result in underflow or division by zero, before performing a primitive multiplication on the result of the division, which has the potential to truncate, before performing a primitive addition without checking either value beforehand, which has the potential to overflow.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of integer over/underflow and division/multiplication by zero.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-35: Potential integer overflow in getNormalizedTimeStamp

Type	Severity	Location
Arithmetic	Minor	contracts/oracle/RegularIntervalOracle.sol L282-L283

Description:

The public `getNormalizedTimeStamp` function in the `RegularIntervalOracle` performs a primitive addition between the `_interval` and `_timeCorrectionFactor` state variables without checking either value beforehand, which has the potential to overflow.

Recommendation:

Since the project already imports the `@openzeppelin/contracts` node module, consider importing and utilizing the `SafeMath` library within the `RegularIntervalOracle` contract in order to safely revert in the event of integer overflow.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-36: getInfo should be re-declared as external

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L286

Description:

The public `getInfo` function in the `RegularIntervalOracle` contract should be re-declared as external.

Recommendation:

Consider re-declaring the public `getInfo` function in the `RegularIntervalOracle` contract as external.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-37: getVolatility should be re-declared as external

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L365

Description:

The public `getVolatility` function in the `RegularIntervalOracle` contract should be re-declared as external.

Recommendation:

Consider re-declaring the public `getVolatility` function in the `RegularIntervalOracle` contract as external.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

RIO-38: getVolatilityTimeOf should be re-declared as external

Type	Severity	Location
Implementation	Informational	contracts/oracle/RegularIntervalOracle.sol L385

Description:

The public `getVolatilityTimeOf` function in the `RegularIntervalOracle` contract should be re-declared as external.

Recommendation:

Consider re-declaring the public `getVolatilityTimeOf` function in the `RegularIntervalOracle` contract as external.

Alleviation:

The recommendation was found to be applied as of commit f1b6997333f6b3131d2de9524525d5ff2485893f.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Arithmetic

Arithmetic exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an in-storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.