STATE MIND

Sidechain oracles

30-11-2022 - 09-12-2022

Table of contents

1. Project Brief

Title	Description
Client	Keep3r
Project name	Sidechain oracles
Timeline	30-11-2022 - 09-12-2022
Initial commit	da7cf7d15fca848828f3a2c6e0e8c55e0dd76841
Final commit	477bb912bd2245d17d37f25f2434315a01483d74

Short Overview

The Keep3r Sidechain Oracles project is a set of contracts that provide UniswapV3's price history to chains where Uniswap pools are unavailable or don't have healthy liquidity to rely on.

Project Scope

The audit covered the following files:

- StrategyJob.sol
- DataFeedStrategy.sol
- BridgeReceiverAdapter.sol
- ConnextReceiverAdapter.sol
- ConnextSenderAdapter.sol
- OracleFactory.sol
- OracleSidechain.sol
- PipelineManagement.sol
- Governable.sol
- Keep3rJob.sol
- DataReceiver.sol
- DataFeed.sol
- Create2Address.sol

2. Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on its potential severity and has the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Informational	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

3. Summary of findings

Severity	# of Findings
Critical	0
High	1
Medium	2
Informational	19

4. Conclusion

Deployment

File name	Contract deployed on mainnet
DataFeed.sol (ethereum)	0x1ce8129OEb4c10cC9Fa71256799665423e87b628
DataFeedStrategy.sol (ethereum)	0xd8f44421aO44AC88a6F67ff6BDAO07b63adfEd5C
StrategyJob.sol (ethereum)	0x1f5fODA9391ABO8c7FO15Od45B41F69O0fb4FdOC
ConnextSenderAdapter.sol (ethereum)	0x5b9315CE13O4DF3B2A83B2O74cbF849D16O642Ab
DataReceiver.sol (optimism)	0x5b9315CE13O4DF3B2A83B2O74cbF849D16O642Ab
OracleFactory.sol (optimism)	0x1ce8129OEb4c10cC9Fa71256799665423e87b628
OracleSidechain.sol (optimism)	0x6AO6OBF6579318c1513816OEe1f1d225fcC9D4O9
ConnextReceiverAdapter.sol (optimism)	0x5bBa978A823a8f1fBFBb61d44Oea1483Ad1737a6
DataReceiver.sol (polygon)	0x5b9315CE13O4DF3B2A83B2O74cbF849D16O642Ab
OracleFactory.sol (polygon)	0x1ce8129OEb4c10cC9Fa71256799665423e87b628

File name	Contract deployed on mainnet
OracleSidechain.sol (polygon)	0x6A060BF6579318c15138160Ee1f1d225fcC9D409
ConnextReceiverAdapter.sol (polygon)	0x5bBa978A823a8f1fBFBb61d440ea1483Ad1737a6

5. Findings report

High

External access in strategicFetchObservation function	Acknowledged

Description

Function `strategicFetchObservation` is granted with external access, consequently anyone can call it without any restrictions.
DataFeedStrategy.sol#L42
So, it can be used to make incorrect price calculation, using several flashloans in a row, trying to call fetchObservations with `TriggerReason.TWAP` reason every twapLength period. This will lead to incorrect price readings. Moreover, there's another function `forceFetchObservations` for extreme situations, which can be called if something will go wrong with keep3r's workers.

Recommendation

We recommend to create `onlyStrategyJob` modifier, and grant access to call this function only from `StrategyJob` contract, so valid keep3rs will call this function without any bad intensions.

Client's comments

> The DataFeedStrategy sets the conditions in which the pool history should be updated, any non-reverting transaction is considered to be 'without bad intentions'. The contract should be openly callable in the case StrategyJob fails. In every situation, TWAP is being quoted and broadcast, so flashloans should not affect the reading. The strategy defines which observations and how should be made, every time it allows a fetch, it's because the state should be updated (by one of the reasons: TIME, TWAP, OLD). So any signer could do the fetch, not being possible for it to misbehave.

Medium

Whitelisting the same pipeline can disable writing to the oracle	Fixed at O1dOO2

Description

At the lines contracts/peripherals/PipelineManagement.sol#L139-L144:

```
function _whitelistPipeline(uint32 _chainId, bytes32 _poolSalt) internal {
  (uint24 _lastPoolNonceObserved, , , ) = IDataFeed(address(this)).lastPoolStateObserved(_poolSalt);
  whitelistedNonces[_chainId][_poolSalt] = _lastPoolNonceObserved + 1;
  _whitelistedPools.add(_poolSalt);
  emit PipelineWhitelisted(_chainId, _poolSalt, _lastPoolNonceObserved + 1);
}
```

Scenario:

1. A pipeline is whitelisted, `whitelistedNonces[_chainId][_poolSalt] = 1`.
2. A keeper/user fetches and sends observations for the pool with `_poolNonce = 1`. Then `poolNonce = 1` in `OracleSidechain`.
3. The keeper/user fetches observations with `_poolNonce = 2`.
4. The same pipeline is whitelisted again, `whitelistedNonces[_chainId][_poolSalt] = 3`.
5. The keeper/user cannot send observations with `_poolNonce = 2` since `_whitelistedNonce > _poolNonce`.

The observations with `_poolNonce >= 3` can be sent, but they will not be written since `poolNonce = 1` in `OracleSidechain`.

Recommendation

It is recommended to check if the same pipeline has already been whitelisted.

Bridged nonces are not tracked properly	Acknowledged

Description

At the lines contracts/StrategyJob.sol#L76-L88:

```
  function _workable(
    uint32 _chainId,
    bytes32 _poolSalt,
    uint24 _poolNonce
  ) internal view returns (bool _isWorkable) {
    uint24 _lastPoolNonceBridged = lastPoolNonceBridged[_chainId][_poolSalt];
    if (_lastPoolNonceBridged == 0) {
      (uint24 _lastPoolNonceObserved, , , ) = dataFeed.lastPoolStateObserved(_poolSalt);
      return _poolNonce == _lastPoolNonceObserved;
    } else {
      return _poolNonce == ++_lastPoolNonceBridged;
    }
  }
```

Scenario 1:

1. A keeper fetches and sends observations for a new pool with `_poolNonce = 1`. Then `lastPoolNonceBridged[_chainId][_poolSalt] = 1`.
2. A user fetches and sends observations for the same pool with `_poolNonce = 2`.
3. The keeper fetches and tries to send observations with `_poolNonce = 3`, but it fails because `_poolNonce != ++_lastPoolNonceBridged`.

The keeper has to send observations with `_poolNonce = 2` that will revert in the sidechain to synchronize nonces.

Scenario 2:

1. A user fetches observations for a new pool three times such that `dataFeed.lastPoolStateObserved(_poolSalt) = 3`.
2. The user sends observations with `_poolNonce = 1`. Then `poolNonce = 1` in `OracleSidechain`.
3. The keeper sends observations with `_poolNonce = 3` with `lastPoolNonceBridged[_chainId][_poolSalt] = 3`, but it reverts in the sidechain because `_poolNonce != ++poolNonce`.

The user has to send observations with `_poolNonce = 2` to synchronize nonces.

Recommendation

It is recommended to save `lastPoolNonceBridged` in `DataFeed` to keep track of all bridged nonces.

Client's comments

> The reward layer of the setup (StrategyJob) should be completely independent from the Strategy behaviour. In Scenario 1.2, the keeper will also be able to send observations _poolNonce = 2, despite they will revert in the sidechain (because they were already bridged). With Scenario 2, where 3 fetch tx are made ‘at the same time’, if a user chooses to bridge n3 first, then n1 and n2 (and again n3) are also going to be bridged (keepers will work it), but revert on the sidechain.

Informational

Self static call from the inhereted contract	Fixed at 410c35

Description

`DataFeed` is inheriting from the `PipelineManagement` contract. At the same time, `PipelineManagement`'s `whitelistPipeline` function makes an STATICCALL to his child contract (himself).

Recommendation

We recommend moving part of the missing internal functionality from the `DataFeed` contract to the `PipelineManagement` or vice versa to avoid making redundant calls.

Explicit bytes return value	Fixed at 4b21b3

Description

In the `ConnextReceiverAdapter` function `xReceive` returns `bytes(abi.encode(''))` which is essentially `64` bytes of zero. As it seems like the return value is irrelevant, without an explicit return it will output an empty array which reduces the cost of execution.

Recommendation

We recommend removing explicit return value.

Client's comments

TODO: #62

Observation overwriting	Acknowledged

Description

Number of observations is based on period passed since last observation. It is calculated at Line 99. In some cases this number could exceed pool cardinality and some observations will be overwritten.

Recommendation

It is recommended to take into account `cardinality` while setting `strategyCooldown` and `periodTime` parameters. In general, `strategyCooldown / periodTime` should be less than `cardinality`.

Client's comments

Pool history is supposed to be updated every cooldown period, and be filled with as many periodDuration fit inside. Current deployed settings have a cooldown of 2 days, and a periodDuration of 4hs: that uses 12 slots, while the initialCardinality is set to 144. From mainnet side, the cardinality of the pools or the factory is not reachable, so settings must optimistically take them into account.

Cardinality check	Acknowledged

Description

In setInitialCardinality() there is no restriction for setting `initialCardinality` variable. If zero is passed, then whole system will be useless.

Recommendation

It is recommended to add `require` to check if `_initialCardinality` equals zero.

Client's comments

TODO: #66

Revert transaction	Acknowledged

Description

At Line 119 if comparision is not passed, transaction will simply return nothing. It is better to revert with error, because at the moment building and sending transaction error can be detected and falsy transaction won't be sent.

Recommendation

It is recommended to revert transaction instead of empty return.

Client's comments

> Discuss: #88 Verify reverting logic at increaseCardinality

TODO comments	Fixed at 044a23

Description

At the lines:

- contracts/DataFeedStrategy.sol#L168
- contracts/StrategyJob.sol#L40
- contracts/bridges/ConnextSenderAdapter.sol#L25
- interfaces/IOracleSidechain.sol#L16

`TODO` comments are irrelevant and should be removed before deployment.

Recommendation

It is recommended to remove `TODO` comments.

Period duration can be set to 0	Fixed at 713879

Description

At the line contracts/DataFeedStrategy.sol#L193, there is no check if `_periodDuration` is bigger than `0`.

Recommendation

It is recommended to check if `_periodDuration > 0`.

Helper functions for pool salt — Acknowledged

Description

1. Currently, in the contracts a pool salt is used to identify a UniswapV3 pool. For users to calculate pool salt they have to get `keccak256(abi.encode(token0, token1, fee)` externally. There is `getPoolSalt()` in `OracleFactory.sol` already, but it would be useful to have it also on the mainnet side.
2. To use `initializePoolInfo()` in `OracleSidechain.sol` users have to know `token0, token1, fee` to match the `poolSalt`. It would be useful to have a helper function that given the pool salt would return these values from the UniswapV3 pool.

Recommendation

It is recommended to add helper functions.

Client's comments

> keepers should listen to the events and push the emitted data, not caring about which pool it corresponds to. Users that choose to initializePoolInfo are users of the Oracle, that found it by querying the Factory with token0, token1, fee, so if required, they would know what parameters to input to initialize the pool info.

Missing NatSpec comments — Fixed at 6723d1

Description

At the lines:

- interfaces/bridges/IBridgeReceiverAdapter.sol#L10
- interfaces/bridges/IBridgeSenderAdapter.sol#L9
- interfaces/bridges/IConnextReceiverAdapter.sol#L10-L14
- interfaces/bridges/IConnextSenderAdapter.sol#L11-L13
- interfaces/peripherals/IPipelineManagement.sol#L10-L87

There are missing NatSpec comments.

Recommendation

It is recommended to add NatSpec comments.

Event parameters can be indexed — Acknowledged

Description

At the lines interfaces/peripherals/IPipelineManagement.sol#L20-L26, the parameters `_chainId`, `_bridgeSenderAdapter`, `_destinationDomainId`, `_dataReceiver` can be indexed.
At the line interfaces/IDataFeed.sol#L51, the parameters `_chainId`, `_dataReceiver` can be indexed.
At the lines interfaces/IDataReceiver.sol#L36-L46, the parameters `_receiverAdapter`, `_adapter` can be indexed.
At the line interfaces/IOracleSidechain.sol#L80, the parameters `token0`, `token1` can be indexed.

Recommendation

It is recommended to change event parameters to `indexed` where appropriate.

Client's comments

> indexed parameters are good to filter and search for specific events, but are hard to read from those events. Keepers need to read the information emitted, so that they can broadcast the data.

Functions can be external	Fixed at a73230

Description

At the lines:

- contracts/peripherals/Keep3rJob.sol#L12
- contracts/StrategyJob.sol#L57
- contracts/StrategyJob.sol#L72

The functions can be made `external` to optimize gas.

Recommendation

It is recommended to change these functions to `external`.

slot0.unlocked in OracleSidechain has opposite meaning	Acknowledged

Description

At the lines contracts/OracleSidechain.sol#L67, contracts/OracleSidechain.sol#L77, contracts/OracleSidechain.sol#L85.

The variable `slot0.unlocked` is set to `true` in the constructor and then to `false` after initialization. This means that the oracle is "locked" after initialization, but it should be "unlocked".

Recommendation

It is recommended to set `slot0.unlocked = false` in the constructor and then to `true` after initialization. Change the check to

```
if (slot0.unlocked) revert AI();
```

Client's comments

> locked means that the method to be called is unaccesible (because it's locked), while unlocked means the opposite. An oracle is locked when it already has the initialized variables. slot0.unlocked was used for this to keep UniV3 slot0, none of the non-reentrant calls from UniV3 are present in the contract.

minLastOracleDelta and periodDuration are independent from each other	Acknowledged

Description

At the lines:

- contracts/DataFeed.sol#L116
- contracts/DataFeed.sol#L153
- contracts/DataFeedStrategy.sol#L240

If `minLastOracleDelta` is set to be bigger than `periodDuration` then every `fetchObservations()` would revert. Alternatively, if `periodDuration` is set to be smaller than `minLastOracleDelta` then every `fetchObservations()` would revert.

Recommendation

It is recommended to check that `minLastOracleDelta` cannot be set to be bigger than `periodDuration` and `periodDuration` cannot be set to be smaller than `minLastOracleDelta`.

Pending old observations sending DoS

Acknowledged

Description

After sending observation with the highest nonce when there are more unsent observations with lower nonce on undeployed sidechain oracle, it will be impossible to send observations with lower nonce despite nonces are whitelisted. OracleSidechain.sol#L101

Recommendation

We recommend checking last bridged nonces before sending and initializing with validated nonce.

Client's comments

> If the oracle is new, any nonce (higher than whitelisted one) will suffice to deploy and initialize the oracle, and all following nonces must be sequential. Highest nonce information relates to 'most recent', so any information previous to that can be dismissed.

Unused variable ORACLE_INIT_CODE_HASH

Fixed at 31a737

Description

The above mentioned variable is not used throughout the contract DataReceiver.sol.

Recommendation

Delete this variable if it is not necessary in the contract.

Superfluous functions

Acknowledged

Description

The function `whitelistPipeline` is almost a total copy of `whitelistPipelines`, with only difference, that parameters are not arrays. Thus, it could be disposed of not damaging contract's functionaity. Same holds for `whitelistAdapter`, `setDestinationDomainId`, `setReceiver`.

Recommendation

Optimize code by deleting excessive functions.

Client's comments

> Adding an array way of whitelisting pipelines can save a lot of gas when whitelisting more than 1, since it will use initialized data slots.

Missing event in forceFetchObservations

Fixed at 7edbbd

Description

`strategicFetchObservations` is emitting StrategicFetch event which can be used in the future for analysis, but there's no event in case fetch was made by governor.

Recommendation

We recommend to add new event and emit it in this function.

Add zero address checks in constructors and setters	Fixed at 5ba851

Description

Here, there's no zero address check on `_dataReceiver` address.
BridgeReceiverAdapter.sol#L11
Here, there's no zero address checks on `_source` and `_connext` addresses.
ConnextReceiverAdapter.sol#L22
Here, there's no zero address checks on `_ataFeed` and `_connext` addresses.
ConnextSenderAdapter.sol#L15
Here, there's no zero address check on `_dataFeed` address.
DataFeedStrategy.sol#L34
Here, there's no zero address check on `_oracleFactory` address.
DataReceiver.sol#L24
Here, there's no zero address check on `_dataReceiver` address.
OracleFactory.sol#L26
Here, there's no zero address check in `_setStrategy()`. In order to turn off this contract `_setStrategy()` can be separated into two functions, one to set with zero address check and one to abandon the strategy setting it to zero address.
DataFeed.sol#L132
Here, there're no zero address checks on `_dataFeeStrategy`, `_dataFeed` and `_defaultBridgeSenderAdapter` addresses.
StrategyJob.sol#L28-30

Recommendation

We recommend to add some zero address checks.

Payable send to bridge	Acknowledged

Description

Currently `bridgeObservations` is set payable, though the only possible executor `DataFeed` can not send ether.

Recommendation

Remove payable keyword or add logic for paying connext.

Client's comments

> TODO: #68

6. Appendix A. Linter

Error/ordering

- solidity/contracts/bridges/ConnextReceiverAdapter.sol:44 - Function order is incorrect, modifier definition can not go after external function (line 27)
- solidity/contracts/bridges/ConnextSenderAdapter.sol:40 - Function order is incorrect, modifier definition can not go after external payable function (line 20)
- solidity/contracts/DataFeed.sol:137 - Function order is incorrect, modifier definition can not go after private function (line 132)
- solidity/contracts/DataFeedStrategy.sol:167 - Function order is incorrect, internal view function can not go after internal pure function (line 156)
- solidity/contracts/DataReceiver.sol:57 - Function order is incorrect, external function can not go after internal function (line 35)
- solidity/contracts/OracleFactory.sol:81 - Function order is incorrect, modifier definition can not go after public pure function (line 72)
- solidity/contracts/OracleSidechain.sol:16 - Function order is incorrect, struct definition can not go after state variable declaration (line 14)
- solidity/contracts/peripherals/Governable.sol:40 - Function order is incorrect, modifier definition can not go after internal function (line 34)
- solidity/contracts/peripherals/Keep3rJob.sol:25 - Function order is incorrect, modifier definition can not go after internal function (line 21)
- solidity/contracts/peripherals/PipelineManagement.sol:169 - Function order is incorrect, modifier definition can not go after internal function (line 160)
- solidity/contracts/StrategyJob.sol:67 - Function order is incorrect, external view function can not go after public view function (line 57)
- solidity/interfaces/IDataFeed.sol:51 - Function order is incorrect, event definition can not go after external view function (line 33)
- solidity/interfaces/IDataFeedStrategy.sol:51 - Function order is incorrect, event definition can not go after external view function (line 44)
- solidity/interfaces/IDataReceiver.sol:36 - Function order is incorrect, event definition can not go after external view function (line 27)
- solidity/interfaces/IOracleFactory.sol:47 - Function order is incorrect, event definition can not go after external view function (line 39)

- solidity/interfaces/IOracleSidechain.sol:80 - Function order is incorrect, event definition can not go after external view function (line 63)
- solidity/interfaces/IStrategyJob.sol:31 - Function order is incorrect, event definition can not go after external view function (line 25)
- solidity/interfaces/peripherals/IGovernable.sol:18 - Function order is incorrect, event definition can not go after external view function (line 11)
- solidity/interfaces/peripherals/IKeep3rJob.sol:17 - Function order is incorrect, event definition can not go after external view function (line 11)
- solidity/interfaces/peripherals/IPipelineManagement.sol:20 - Function order is incorrect, event definition can not go after external view function (line 16)

Error/function-max-lines

- solidity/contracts/DataFeed.sol:46 - Function body contains 78 lines but allowed no more than 40 lines

Error/not-rely-on-time

- solidity/contracts/DataFeed.sol:54 - Avoid to make time-based decisions in your business logic
- solidity/contracts/DataFeedStrategy.sol:96 - Avoid to make time-based decisions in your business logic
- solidity/contracts/DataFeedStrategy.sol:122 - Avoid to make time-based decisions in your business logic
- solidity/contracts/OracleSidechain.sol:52 - Avoid to make time-based decisions in your business logic

Error/max-states-count

- solidity/contracts/DataFeedStrategy.sol:10 - Contract has 4 states declarations but allowed no more than 3
- solidity/contracts/OracleSidechain.sol:10 - Contract has 6 states declarations but allowed no more than 3
- solidity/contracts/peripherals/PipelineManagement.sol:9 - Contract has 5 states declarations but allowed no more than 3

7. Appendix B. Slither

High/High/uninitialized-state

OracleSidechain.observations is never initialized. It is used in: - OracleSidechain.observe(uint32[]) - OracleSidechain._write(IOracleSidechain.ObservationData)

Informational/Medium/similar-names

Variable PipelineManagement.setDestinationDomainId(IBridgeSenderAdapter,uint32,uint32)._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable DataFeed.sendObservations(IBridgeSenderAdapter,uint32,bytes32,uint24,IOracleSidechain.ObservationData[])._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable IPipelineManagement.receivers(IBridgeSenderAdapter,uint32)._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable IPipelineManagement.whitelistedNonces(uint32,bytes32)._whitelistedNonce is too similar to PipelineManagement.whitelistedNonces

Variable PipelineManagement.validateSenderAdapter(IBridgeSenderAdapter,uint32)._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable PipelineManagement._setDestinationDomainId(IBridgeSenderAdapter,uint32,uint32)._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable IPipelineManagement.setDestinationDomainId(IBridgeSenderAdapter,uint32,uint32)._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable PipelineManagement.setReceiver(IBridgeSenderAdapter,uint32,address)._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable IPipelineManagement.setReceiver(IBridgeSenderAdapter,uint32,address)._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable IPipelineManagement.validateSenderAdapter(IBridgeSenderAdapter,uint32)._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable IDataReceiver.deployedOracles(bytes32)._deployedOracle is too similar to DataReceiver.deployedOracles

Variable IPipelineManagement.setDestinationDomainIds(IBridgeSenderAdapter[],uint32[],uint32[])._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable PipelineManagement._setReceiver(IBridgeSenderAdapter,uint32,address)._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable IPipelineManagement.destinationDomainIds(IBridgeSenderAdapter,uint32)._destinationDomainId is too similar to PipelineManagement.destinationDomainIds

Variable DataFeedStrategy._computeTwap(int56,int56,uint32)._tickCumulative1 is too similar to DataFeedStrategy._computeTwap(int56,int56,uint32)._tickCumulative2

Informational/Medium/too-many-digits

OracleFactory.slitherConstructorConstantVariables() uses literals with too many digits: - ORACLE_INIT_CODE_HASH = keccak256(bytes)(type()(OracleSidechain).creationCode)

Low/Medium/calls-loop

PipelineManagement._whitelistPipeline(uint32,bytes32) has external calls inside a loop: (_lastPoolNonceObserved) = IDataFeed(address(this)).lastPoolStateObserved(_poolSalt)

OracleSidechain._write(IOracleSidechain.ObservationData) has external calls inside a loop: (_indexUpdated,_cardinalityUpdated) = observations.write(slot0.observationIndex,_observationData.blockTimestamp,slot0.tick,0,slot0.observationCardinality,slot0.observationCardinalityNext)

Low/Medium/missing-zero-check

ConnextReceiverAdapter.constructor(IDataReceiver,address,uint32,IConnext)._source lacks a zero-check on : - source = _source

Low/Medium/reentrancy-benign

Reentrancy in StrategyJob.work(uint32,bytes32,uint24,IOracleSidechain.ObservationData[]): External calls: - upkeep() - ! keep3r.isKeeper(_keeper) - keep3r.worked(msg.sender) State variables written after the call(s): - lastPoolNonceBridged[_chainId][_poolSalt] = _poolNonce

Low/Medium/reentrancy-events

Reentrancy in DataFeedStrategy.strategicFetchObservations(bytes32,IDataFeedStrategy.TriggerReason): External calls: - dataFeed.fetchObservations(_poolSalt,_secondsAgos) Event emitted after the call(s): - StrategicFetch(_poolSalt,_reason)

Reentrancy in DataFeed.sendObservations(IBridgeSenderAdapter,uint32,bytes32,uint24,IOracleSidechain.ObservationData[]):

External calls: - _bridgeSenderAdapter.bridgeObservations(_dataReceiver,_destinationDomainId,_observationsData,_poolSalt,_poolNonce) Event emitted after the call(s): - DataBroadcast(_poolSalt,_poolNonce,_chainId,_dataReceiver,_bridgeSenderAdapter)

Reentrancy in DataReceiver._addObservations(IOracleSidechain.ObservationData[],bytes32,uint24): External calls: - _oracle = oracleFactory.deployOracle(_poolSalt,_poolNonce) - _oracle.write(_observationsData,_poolNonce) Event emitted after the call(s): - ObservationsAdded(_poolSalt,_poolNonce,_observationsData,msg.sender)

Low/Medium/timestamp

DataFeedStrategy._isStrategic(bytes32,IDataFeed.PoolState,IDataFeedStrategy.TriggerReason) uses timestamp for comparisons Dangerous comparisons: - _secondsNow >= _lastPoolStateObserved.blockTimestamp + strategyCooldown

DataFeedStrategy._computeTwap(int56,int56,uint32) uses timestamp for comparisons Dangerous comparisons: - _tickCumulativesDelta < 0 && (_tickCumulativesDelta % int32(_delta) != 0)

DataFeedStrategy.calculateSecondsAgos(uint32) uses timestamp for comparisons Dangerous comparisons: - _remainder != 0 - _timeSinceLastObservation > 0

Medium/High/locked-ether

Contract locking ether found: Contract ConnextSenderAdapter has payable functions: - IBridgeSenderAdapter.bridgeObservations(address,uint32,IOracleSidechain.ObservationData[],bytes32,uint24) - ConnextSenderAdapter.bridgeObservations(address,uint32,IOracleSidechain.ObservationData[],bytes32,uint24) But does not have a function to withdraw the ether

Medium/Medium/reentrancy-no-eth

Reentrancy in DataReceiver._addObservations(IOracleSidechain.ObservationData[],bytes32,uint24): External calls: - _oracle = oracleFactory.deployOracle(_poolSalt,_poolNonce) State variables written after the call(s): - deployedOracles[_poolSalt] = _oracle

Reentrancy in OracleSidechain._write(IOracleSidechain.ObservationData): External calls: - (_indexUpdated,_cardinalityUpdated) = observations.write(slot0.observationIndex,_observationData.blockTimestamp,slot0.tick,0,slot0.observationCardinality,slot0.observationCardinalityNext) State variables written after the call(s): - (slot0.observationIndex,slot0.observationCardinality) = (_indexUpdated,_cardinalityUpdated) - slot0.tick = _observationData.tick

Medium/Medium/uninitialized-local

DataReceiver.whitelistAdapters(IBridgeReceiverAdapter[],bool[])._i is a local variable never initialized

PipelineManagement.whitelistAdapters(IBridgeSenderAdapter[],bool[])._i is a local variable never initialized

DataFeedStrategy.calculateSecondsAgos(uint32)._i is a local variable never initialized

PipelineManagement.whitelistPipelines(uint32[],bytes32[])._i is a local variable never initialized

DataFeed.fetchObservations(bytes32,uint32[])._i is a local variable never initialized

DataFeedStrategy.strategicFetchObservations(bytes32,IDataFeedStrategy.TriggerReason)._lastPoolStateObserved is a local variable never initialized

PipelineManagement.setDestinationDomainIds(IBridgeSenderAdapter[],uint32[],uint32[])._i is a local variable never initialized

OracleSidechain.write(IOracleSidechain.ObservationData[],uint24)._i is a local variable never initialized

DataFeedStrategy.isStrategic(bytes32,IDataFeedStrategy.TriggerReason)._lastPoolStateObserved is a local variable never initialized

PipelineManagement.setReceivers(IBridgeSenderAdapter[],uint32[],address[])._i is a local variable never initialized

Medium/Medium/unused-return

ConnextSenderAdapter.bridgeObservations(address,uint32,IOracleSidechain.ObservationData[],bytes32,uint24) ignores return value by connext.xcall(_destinationDomainId,_to,address(0),address(0),0,0,_callData)

PipelineManagement._whitelistPipeline(uint32,bytes32) ignores return value by _whitelistedPools.add(_poolSalt)

Optimization/High/external-function

workable(bytes32,IDataFeedStrategy.TriggerReason) should be declared external: - StrategyJob.workable(bytes32,IDataFeedStrategy.TriggerReason)

setKeep3r(IKeep3r) should be declared external: - Keep3rJob.setKeep3r(IKeep3r)

workable(uint32,bytes32,uint24) should be declared external: - StrategyJob.workable(uint32,bytes32,uint24)

8. Appendix C. Tests

Tests result

```
287 passing (1m)
1 pending
1 failing
```

Tests coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	99.15	100	100	
DataFeed.sol	100	100	100	100	
DataFeedStrategy.sol	100	97.22	100	100	
DataReceiver.sol	100	100	100	100	
OracleFactory.sol	100	100	100	100	
OracleSidechain.sol	100	100	100	100	
StrategyJob.sol	100	100	100	100	
contracts/bridges/	100	75	100	100	
BridgeReceiverAdapter.sol	100	100	100	100	
ConnextReceiverAdapter.sol	100	62.5	100	100	
ConnextSenderAdapter.sol	100	100	100	100	
contracts/peripherals/	100	98.21	100	100	
Governable.sol	100	100	100	100	
Keep3rJob.sol	100	75	100	100	
PipelineManagement.sol	100	100	100	100	
libraries/	100	100	100	100	
Create2Address.sol	100	100	100	100	

STATE
MIND