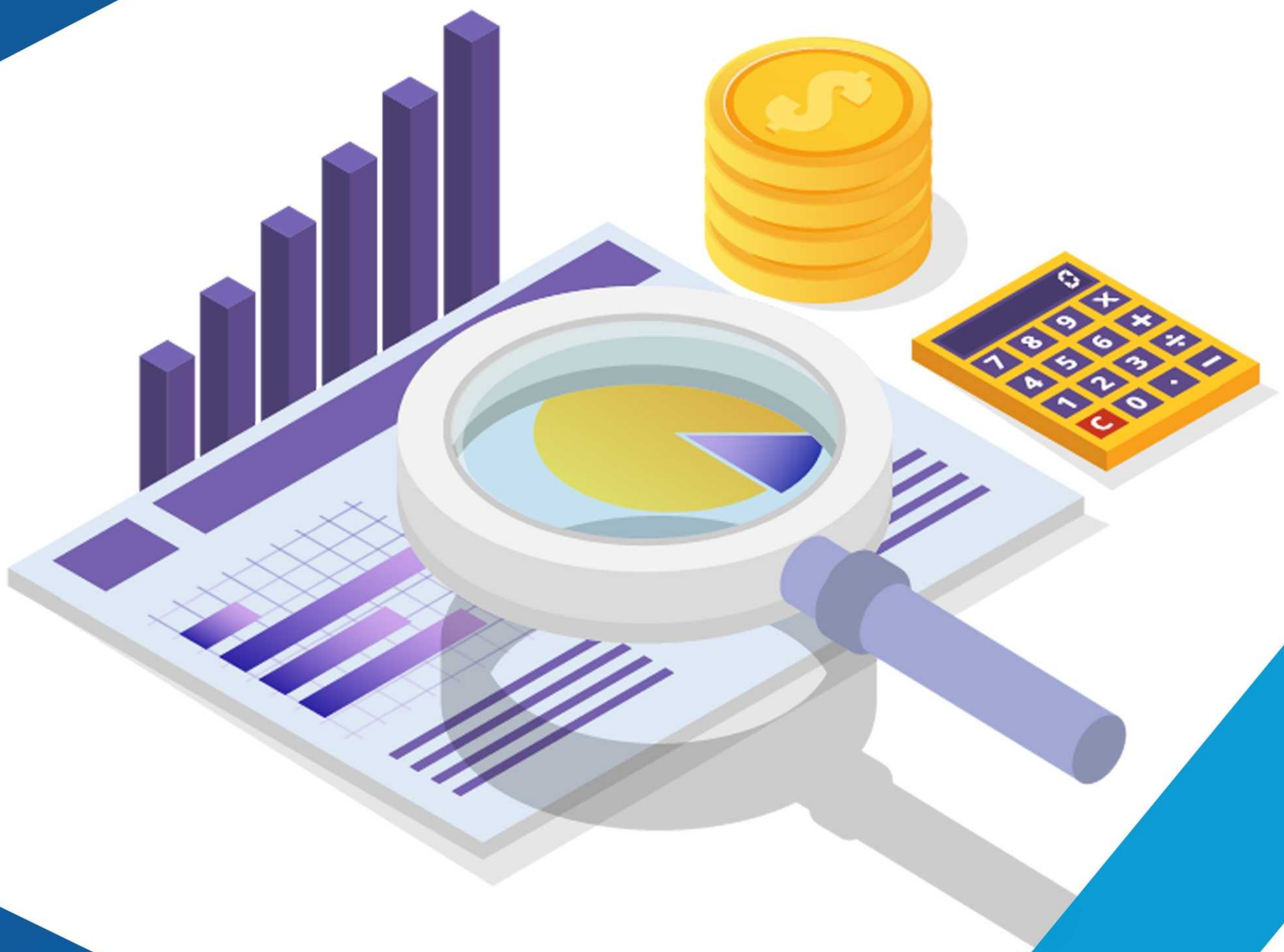
QuillAudits



Audit Report
August, 2021

OCP

OMNI CONSUMER PROTOCOLS

Contents

Scope of Audit

The scope of this audit was to analyze and document the OmniOracle smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	1	0	0
Closed	0	0	2	0

Introduction

During the period of **August 16, 2021 to August 24, 2021** - QuillAudits Team performed a security audit for OmniOracle smart contracts.
Our scope was limited to:
- UniswapOracleTWAP.sol
- UniswapConfig.sol

The rest of the git is an exact replication of compound.

The code for the audit was taken from the following official repo of OmniOracle:
https://github.com/omni-corp-protocols/omni-oracle/tree/main/contracts

Note	Date	Commit hash
Version 1	August	37e7f7bc5beacdeb4e703e9b964ca8196db82805
Version 2	August	a60159b282d9b2fa968fc7fc2988baf8ddd39d78

Issues Found – Code Review / Manual Testing

A. Contract - UniswapOracleTWAP.sol

High severity issues

No issues were found.

Medium severity issues

1. Usage of block.timestamp

```
UniswapOracleTWAP: 314
    bytes32 symbolHash = config.symbolHash;
        uint cumulativePrice = currentCumulativePrice(config);
        oldObservations[symbolHash].timestamp = block.timestamp;
        newObservations[symbolHash].timestamp = block.timestamp;
        oldObservations[symbolHash].acc = cumulativePrice;
        newObservations[symbolHash].acc = cumulativePrice;
        emit UniswapWindowUpdated(symbolHash, block.timestamp,
    block.timestamp, cumulativePrice, cumulativePrice);
```

```
UniswapOracleTWAP: 280
function fetchAnchorPrice(
TokenConfig memory config, uint conversionFactor
) internal virtual returns (uint) {
        (uint nowCumulativePrice, uint oldCumulativePrice, uint oldTimestamp) =
pokeWindowValues(config);
        // This should be impossible, but better safe than sorry
        require(block.timestamp > oldTimestamp, "now must come after before");
        uint timeElapsed = block.timestamp - oldTimestamp;
```

```
UniswapOracleTWAP: 313
function pokeWindowValues(
TokenConfig memory config
) internal returns (uint, uint, uint) {
        bytes32 symbolHash = config.symbolHash;
        uint cumulativePrice = currentCumulativePrice(config);

        Observation memory newObservation = newObservations[symbolHash];

        // Update new and old observations if elapsed time is greater than or
equal to anchor period
        uint timeElapsed = block.timestamp - newObservation.timestamp;
        if (timeElapsed >= anchorPeriod) {
            oldObservations[symbolHash].timestamp = newObservation.timestamp;
            oldObservations[symbolHash].acc = newObservation.acc;
            newObservations[symbolHash].timestamp = block.timestamp;
            newObservations[symbolHash].acc = cumulativePrice;
            emit UniswapWindowUpdated(
config.symbolHash, newObservation.timestamp, block.timestamp,
newObservation.acc, cumulativePrice);
        }
```

Description
Block.timestamp is used in the contract. The variable block is a set of variables. The timestamp does not always reflect the current time and may be inaccurate. The value of a block can be influenced by miners. Maximal Extractable Value attacks require a timestamp of up to 900 seconds. There is no guarantee that the value is right; all that is guaranteed is that it is higher than the timestamp of the previous block.

Remediation
You can use an Oracle to get the exact time or verify if a delay of 900 seconds won't destroy the logic of the staking contract.

Status: Acknowledged by the Auditee.
The OmniOracle team has acknowledged the risk for the reason that the 900 seconds delay won't impact the logic of the smart contract.

Low level severity issues

2. Public Function That Can Be Declared External

Description
The following public functions that are never called by the contract should be declared external to save gas:

- getSymbolHash ()
- updateAllPrices ()
- _setUnderlyingForCTokens()
- _setPrice()

Remediation
Use the external attribute for functions that are not called from the contract.

Status: Fixed
The OmniOracle team has fixed the issue by changing the visibility of the functions that are not called from the contract to external instead of public.

3. Floating Pragma

```
Line 3:
    pragma solidity ^0.6.10;
```

Description

The contract makes use of the floating-point pragma 0.6.10. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps to ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

Remediation

Consider locking the pragma version. It is advised that floating pragma not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status: Fixed

The OmniOracle team has fixed the issue by locking the pragma solidity version to 0.6.12.

Informational

No issues were found.

A. Contract - UniswapOracleTWAP.sol

High severity issues

No issues were found.

Medium level severity issues

No issues were found.

Low level severity issues

No issues were found.

Informational

No issues were found.

Functional test

Function Names	Testing results
_setConfig	Passed
_acceptAdmin()	Passed
_setConfigInternal	Passed
_setConfigs	Passed
_setPendingAdmin()	Passed
_setPoster()	Passed
_setPrice	Passed
_setUnderlyingForCToken	Passed
_setUnderlyingForCTokens	Passed
getSymbolHash	Passed
getTokenConfig	Passed
getTokenConfigBySymbol	Passed
getTokenConfigBySymbolHash	Passed
getTokenConfigByUnderlying	Passed
getUnderlyingPrice	Passed
price(symbol)	Passed
price(underlying)	Passed
updateAllPrices()	Passed
updateEthPrice()	Passed
updatePrice(cToken)	Passed
updatePrice(symbol	Passed
updateUnderlyingPrice	Passed

Automated Testing

Slither

```
INFO:Detectors:
UniswapV2OracleLibrary.currentBlockTimestamp() (UniswapHelper.sol#38-40) uses a weak PRNG: "uint32(block.timestamp % 2 ** 32) (UniswapHelper.sol#39)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
INFO:Detectors:
UniswapOracleTWAP.mul(uint256,uint256) (UniswapOracleTWAP.sol#337-342) uses a dangerous strict equality:
        - a == 0 (UniswapOracleTWAP.sol#338)
UniswapOracleTWAP.mul(uint256,uint256) (UniswapOracleTWAP.sol#337-342) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,multiplication overflow) (UniswapOracleTWAP.sol#340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Administrable._setPendingAdmin(address).newPendingAdmin (Administrable.sol#37) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (Administrable.sol#47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
UniswapV2OracleLibrary.currentCumulativePrices(address) (UniswapHelper.sol#43-61) uses timestamp for comparisons
        Dangerous comparisons:
        - blockTimestampLast != blockTimestamp (UniswapHelper.sol#52)
UniswapOracleTWAP.fetchAnchorPrice(UniswapConfig.TokenConfig,uint256) (UniswapOracleTWAP.sol#280-307) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > oldTimestamp,now must come after before) (UniswapOracleTWAP.sol#284)
UniswapOracleTWAP.pokeWindowValues(UniswapConfig.TokenConfig) (UniswapOracleTWAP.sol#313-330) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed >= anchorPeriod (UniswapOracleTWAP.sol#321)
UniswapOracleTWAP.mul(uint256,uint256) (UniswapOracleTWAP.sol#337-342) uses timestamp for comparisons
        Dangerous comparisons:
        - a == 0 (UniswapOracleTWAP.sol#338)
        - require(bool,string)(c / a == b,multiplication overflow) (UniswapOracleTWAP.sol#340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Pragma version^0.6.10 (Administrable.sol#3) allows old versions
Pragma version^0.6.10 (IERC20Extended.sol#3) allows old versions
Pragma version^0.6.10 (IExternalOracle.sol#3) allows old versions
Pragma version^0.6.10 (PosterAccessControl.sol#3) allows old versions
Pragma version^0.6.10 (UniswapConfig.sol#3) allows old versions
Pragma version^0.6.10 (UniswapHelper.sol#3) allows old versions
Pragma version^0.6.10 (UniswapOracleTWAP.sol#3) allows old versions
solc-0.6.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function Administrable._setPendingAdmin(address) (Administrable.sol#37-53) is not in mixedCase
Function Administrable._acceptAdmin() (Administrable.sol#60-80) is not in mixedCase
Function PosterAccessControl._setPoster(address) (PosterAccessControl.sol#16-20) is not in mixedCase
Constant UniswapConfig.maxTokens (UniswapConfig.sol#39) is not in UPPER_CASE_WITH_UNDERSCORES
Struct FixedPoint.uq112x112 (UniswapHelper.sol#11-13) is not in CapWords
Function UniswapOracleTWAP._setConfig(UniswapConfig.TokenConfig) (UniswapOracleTWAP.sol#66-98) is not in mixedCase
Function UniswapOracleTWAP._setConfigs(UniswapConfig.TokenConfig[]) (UniswapOracleTWAP.sol#100-104) is not in mixedCase
Function UniswapOracleTWAP._setPrice(address,string,uint256) (UniswapOracleTWAP.sol#106-116) is not in mixedCase
Function UniswapOracleTWAP._setUnderlyingForCToken(address,address) (UniswapOracleTWAP.sol#118-128) is not in mixedCase
Function UniswapOracleTWAP._setUnderlyingForCTokens(address[],address[]) (UniswapOracleTWAP.sol#130-135) is not in mixedCase
Parameter UniswapOracleTWAP._setUnderlyingForCTokens(address[],address[])._cTokens (UniswapOracleTWAP.sol#130) is not in mixedCase
Parameter UniswapOracleTWAP._setUnderlyingForCTokens(address[],address[])._underlyings (UniswapOracleTWAP.sol#130) is not in mixedCase
Constant UniswapOracleTWAP.ethBaseUnit (UniswapOracleTWAP.sol#22) is not in UPPER_CASE_WITH_UNDERSCORES
Constant UniswapOracleTWAP.expScale (UniswapOracleTWAP.sol#25) is not in UPPER_CASE_WITH_UNDERSCORES
Variable UniswapOracleTWAP.ETH (UniswapOracleTWAP.sol#54) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable UniswapV2OracleLibrary.currentCumulativePrices(address).price0Cumulative (UniswapHelper.sol#45) is too similar to UniswapV2OracleLibrary.currentCumulativePrices(address).price1Cumulative (UniswapHelper.
sol#45)
Variable UniswapOracleTWAP.currentCumulativePrice(UniswapConfig.TokenConfig).cumulativePrice0 (UniswapOracleTWAP.sol#261) is too similar to UniswapOracleTWAP.currentCumulativePrice(UniswapConfig.TokenConfig).cum
ulativePrice1 (UniswapOracleTWAP.sol#261)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
_setPendingAdmin(address) should be declared external:
        - Administrable._setPendingAdmin(address) (Administrable.sol#37-53)
_acceptAdmin() should be declared external:
        - Administrable._acceptAdmin() (Administrable.sol#60-80)
_setPrice(address,string,uint256) should be declared external:
        - UniswapOracleTWAP._setPrice(address,string,uint256) (UniswapOracleTWAP.sol#106-116)
_setUnderlyingForCTokens(address[],address[]) should be declared external:
        - UniswapOracleTWAP._setUnderlyingForCTokens(address[],address[]) (UniswapOracleTWAP.sol#130-135)
updateAllPrices() should be declared external:
        - UniswapOracleTWAP.updateAllPrices() (UniswapOracleTWAP.sol#223-227)
getSymbolHash(string) should be declared external:
        - UniswapOracleTWAP.getSymbolHash(string) (UniswapOracleTWAP.sol#332-334)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:UniswapOracleTWAP.sol analyzed (9 contracts with 75 detectors), 39 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

MythX

```
Found 3 job(s). Submit? [y/N]: y
  [####################################]  100%
Report for UniswapConfig.sol
https://dashboard.mythx.io/#/console/analyses/b355f695-a6d1-4c20-a8b3-8d95ce845c7c
```

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

```
Report for UniswapHelper.sol
https://dashboard.mythx.io/#/console/analyses/3eb296f6-db0c-4404-a68d-3fa69328242e
```

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

```
Report for UniswapOracleTWAP.sol
https://dashboard.mythx.io/#/console/analyses/798dc4d9-9318-4692-81bd-f168f077f29a
```

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
54	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the OmniOracle Contracts. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the OmniOracle team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract.

A few issues were reported during the initial Audit. However, all of the issues are now either fixed or acknowledged by the OmniOracle team.

OCP

OMNI CONSUMER PROTOCOLS

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com