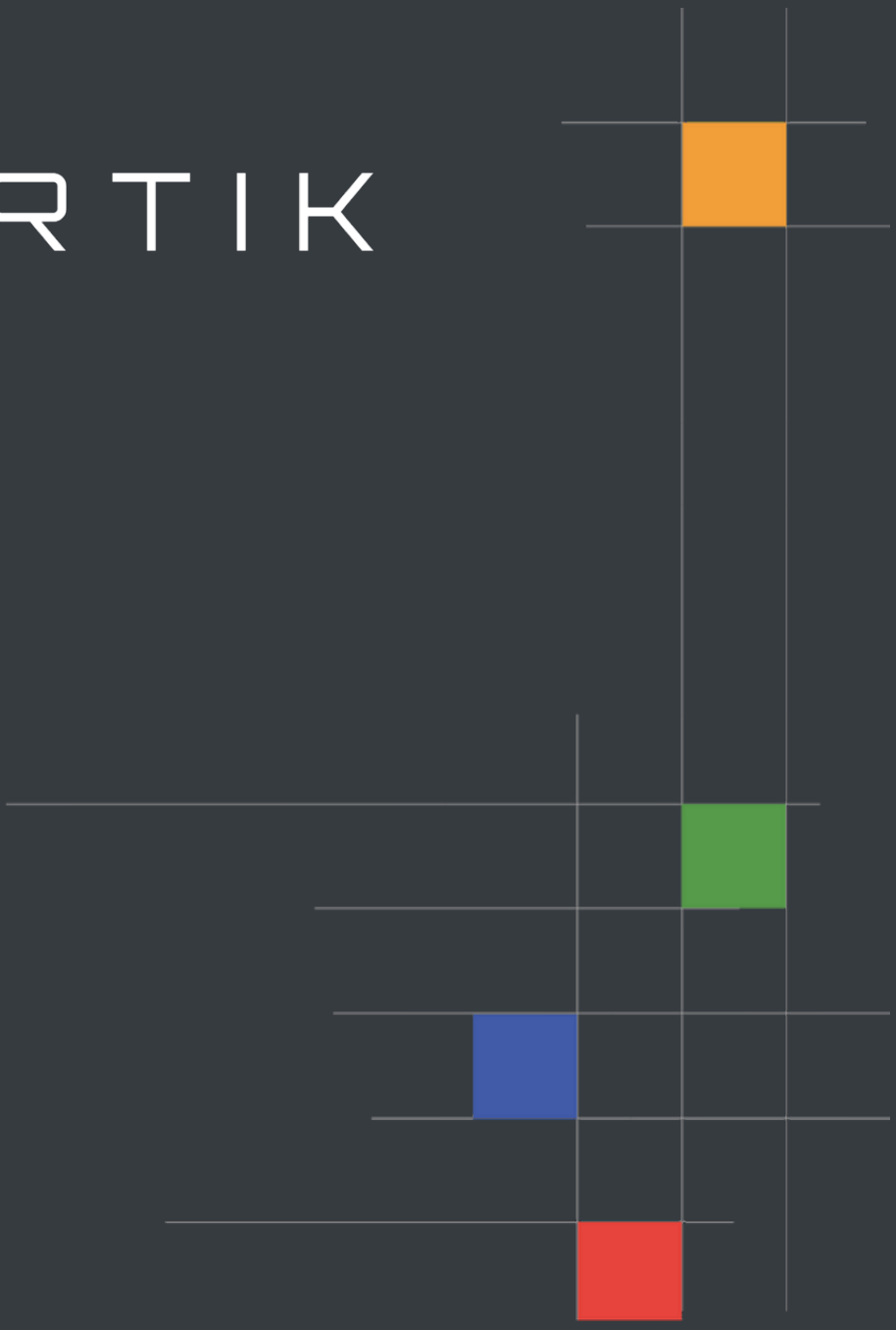
OMS Finance

OMS

Security Assessment

April 20th, 2021

Audited By:
Sheraz Arshad @ CertiK
sheraz.arshad@certik.org
Reviewed By:
Camden Smallwood @ CertiK
camden.smallwood@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	OMS Finance - OMS
Description	The codebase comprise an Ethereum based elastic supply token designed to rebase (inflate/deflate) supply through automated distribution proportional to percentage of supply held.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 40245b9b181a9631681cf5d700b02ff71763d538 2. 752e4216f87207fcad09e16402052d7457cde3b6

Audit Summary

Delivery Date	April 20th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	1
Timeline	April 5th, 2021 - April 20th, 2021

Vulnerability Summary

Total Issues	17
Total Critical	0
Total Major	1
Total Medium	2
Total Minor	5
Total Informational	9

Executive Summary

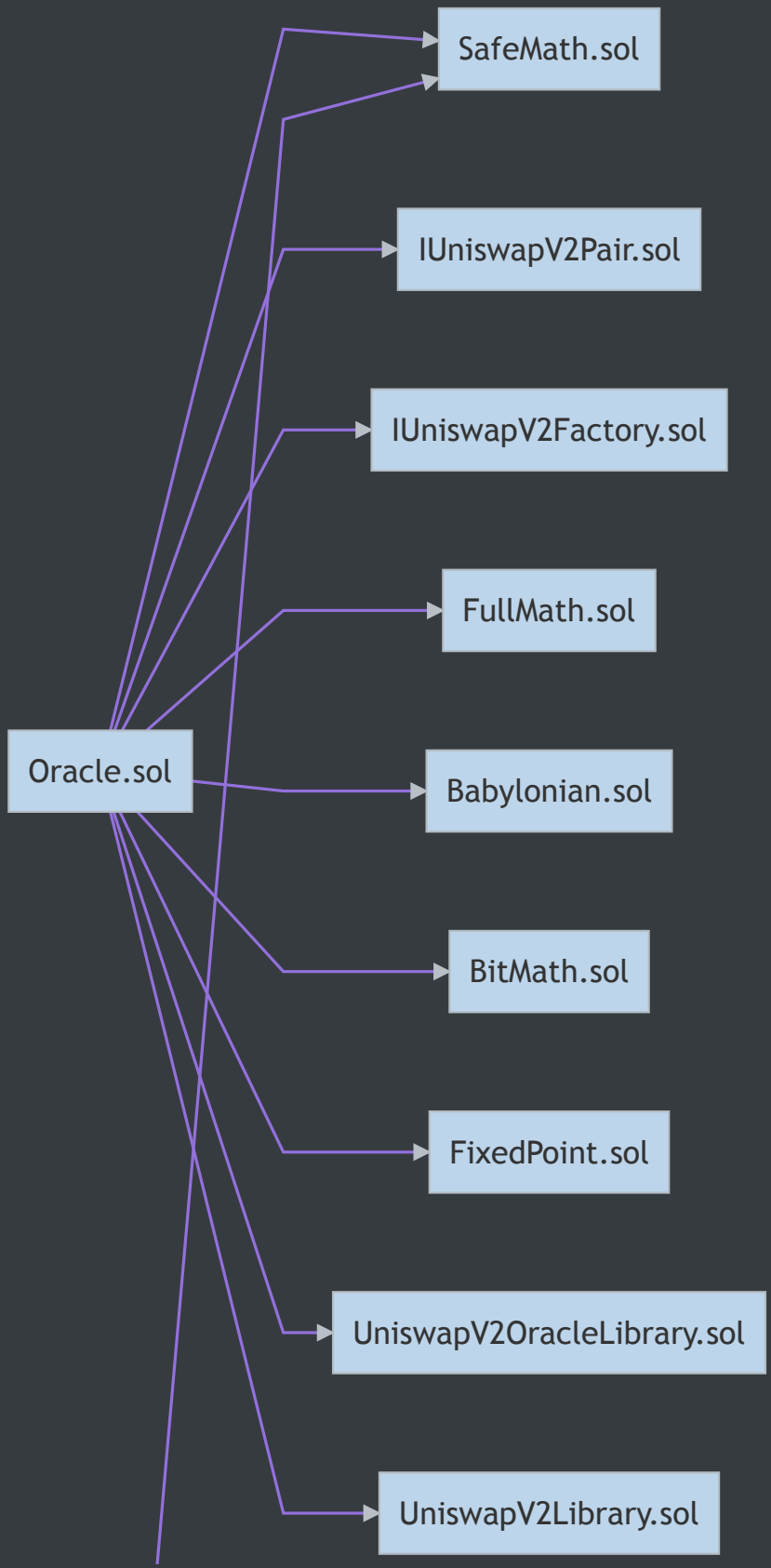
This report represents the results of CertiK's engagement with OMS on their implementation of the OMS core smart contracts.

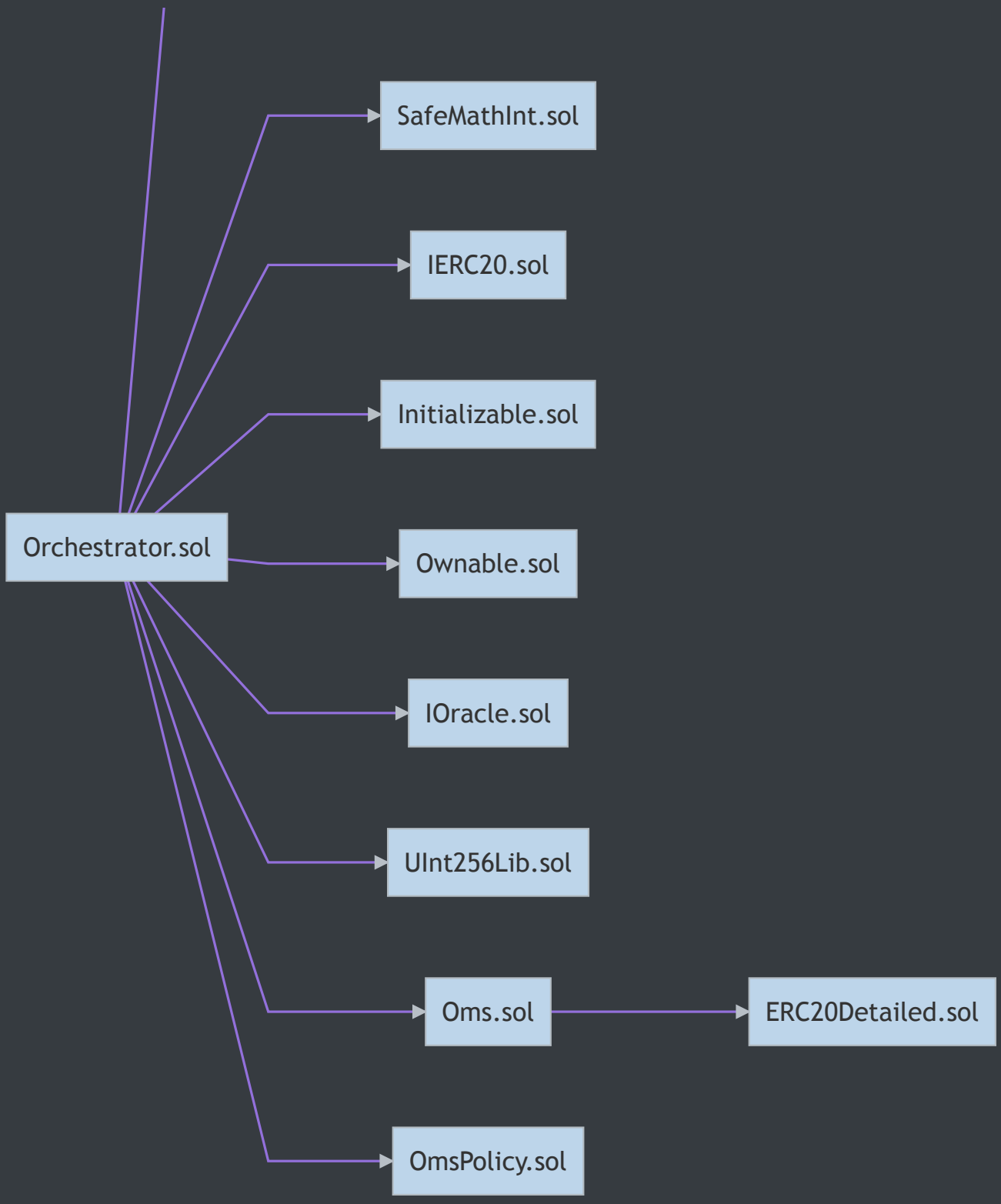
The manual and static analysis were performed in the audit. Our findings major, medium, minor and informational issues. The major issue comprises the incorrect price calculation in the `Oracle` contract and, medium and minor issues comprise the non-checking of addresses that initialize the contracts states against zero address value. All of the findings are remediated as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6`.

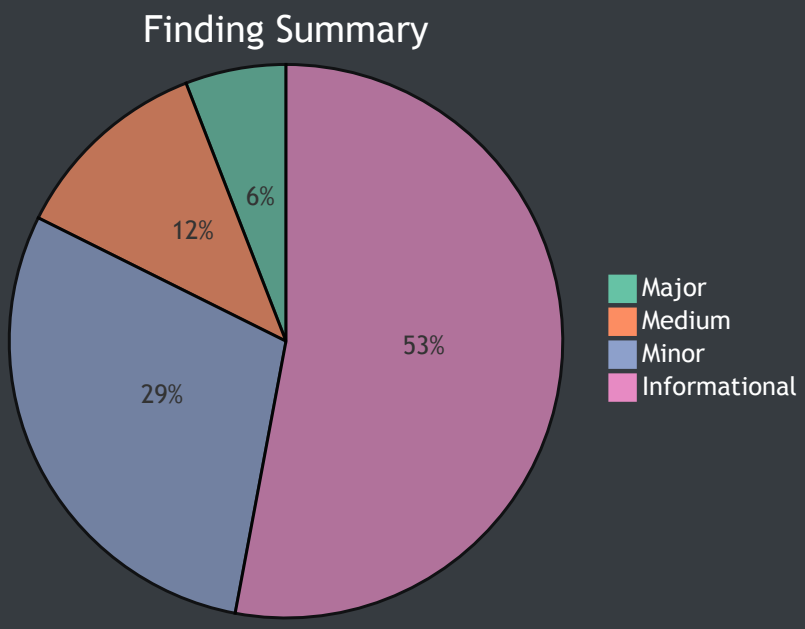
Files In Scope

ID	Contract	Location
OMS	Oms.sol	contracts/v4/Oms.sol
OPY	OmsPolicy.sol	contracts/v4/OmsPolicy.sol
ORC	Orchestrator.sol	contracts/v4/Orchestrator.sol
ORA	Oracle.sol	contracts/v6/Oracle.sol
ERC	ERC20Detailed.sol	contracts/v4/common/ERC20Detailed.sol
INI	Initializable.sol	contracts/v4/common/Initializable.sol
OWN	Ownable.sol	contracts/v4/common/Ownable.sol
IER	IERC20.sol	contracts/v4/interface/IERC20.sol
IOE	IOracle.sol	contracts/v4/interface/IOracle.sol
UIL	UInt256Lib.sol	contracts/v4/interface/UInt256Lib.sol
SMH	SafeMath.sol	contracts/v4/library/SafeMath.sol
SMI	SafeMathInt.sol	contracts/v4/library/SafeMathInt.sol

File Dependency Graph




SafeMathInt.sol
IERC20.sol
Initializable.sol
Orchestrator.sol
Ownable.sol
IOracle.sol
UInt256Lib.sol
Oms.sol
ERC20Detailed.sol
OmsPolicy.sol


Finding Summary
6%
12%
29%
53%
Major
Medium
Minor
Informational

Manual Review Findings

ID	Title	Type	Severity	Resolved
OMS-01	Lack of verification for the function parameters	Logical Issue	Medium	✓
OMS-02	Lack of verification for the function parameter	Logical Issue	Minor	✓
OMS-03	Return Variable Utilization	Gas Optimization	Informational	✓
OMS-04	`require` statements are missing reason `string`	Language Specific	Informational	✓
OPY-01	Lack of verification for function parameter	Logical Issue	Minor	✓
OPY-02	Lack of verification for function parameter	Logical Issue	Minor	✓
OPY-03	Lack of verification for function parameter	Logical Issue	Minor	✓
OPY-04	Lack of verification for function parameter	Logical Issue	Minor	✓
OPY-05	`require` statements are missing reason `string`	Language Specific	Informational	✓
OPY-06	Redundant Variable Initialization	Coding Style	Informational	✓
OPY-07	Redundant Statements	Dead Code	Informational	✓
ORC-01	Lack of verification for constructor parameter	Logical Issue	Medium	✓
ORC-02	Unlocked Compiler Version	Language Specific	Informational	✓
ORC-03	`require` statement is missing reason `string`	Language Specific	Informational	✓
ORC-04	Volatile code	Volatile Code	Informational	✓
ORA-01	Incorrect `price` calculation	Logical Issue	Major	✓
ORA-02	Redundant variable initialization	Dead Code	Informational	✓

OMS-01: Lack of verification for the function parameters

Type	Severity	Location
Logical Issue	Medium	Oms.sol L134

Description:

The function parameters of `owner_` and `reserve_` on the aforementioned line are not validated against zero address value. If zero address value is provided for any of the aforementioned parameters, it will result in funds being assigned to zero address.

Recommendation:

We advise to validate the parameters of `owner_` and `reserve_` against zero address values.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6`.

OMS-02: Lack of verification for the function parameter

Type	Severity	Location
Logical Issue	Minor	Oms.sol L68

Description:

The function parameter of `omsPolicy_` on the aforementioned line is not validated against zero address value, which when passed to the function, will result in unwanted state of the contract.

Recommendation:

We advise to validate the parameter of `omsPolicy_` against zero address value.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6` .

OMS-03: Redundant default assignment

Type	Severity	Location
Gas Optimization	Informational	Oms.sol L138-L139

Description:

The linked lines perform redundant assignment default assignment of variables.

Recommendation:

We advise to remove the redundant default assignment of variables on the aforementioned lines..

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6`.

OMS-04: `require` statements are missing reason `string`

Type	Severity	Location
Language Specific	Informational	Oms.sol L23, L32, L37, L42-L43

Description:

The `require` statements on the aforementioned lines are missing reason strings.

Recommendation:

We advise to provide reason strings for the `require` statements on the aforementioned lines to increase the legibility of the codebase.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6` .

OPY-01: Lack of verification for function parameter

Type	Severity	Location
Logical Issue	Minor	OmsPolicy.sol L127

Description:

The address type parameter of function on the aforementioned line is assigned to a variable in the contract's storage. The parameter is not validated against zero address value, which, when provided, results in unwanted state of the contract.

Recommendation:

We advise to validate the function parameter on the aforementioned line against zero address value.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6` .

OPY-02: Lack of verification for function parameter

Type	Severity	Location
Logical Issue	Minor	OmsPolicy.sol L138

Description:

The address type parameter of function on the aforementioned line is assigned to a variable in the contract's storage. The parameter is not validated against zero address value, which, when provided, results in unwanted state of the contract.

Recommendation:

We advise to validate the function parameter on the aforementioned line against zero address value.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6` .

OPY-03: Lack of verification for function parameter

Type	Severity	Location
Logical Issue	Minor	OmsPolicy.sol L205

Description:

The address type parameter of function on the aforementioned line is assigned to a variable in the contract's storage. The parameter is not validated against zero address value, which, when provided, results in unwanted state of the contract.

Recommendation:

We advise to validate the function parameter on the aforementioned line against zero address value.

Alleviation:

The relevant code is removed from the codebase rendering this exhibit ineffectual.

OPY-04: Lack of verification for function parameter

Type	Severity	Location
Logical Issue	Minor	OmsPolicy.sol L218

Description:

The address type parameter of function on the aforementioned line is assigned to a variable in the contract's storage. The parameter is not validated against zero address value, which, when provided, results in unwanted state of the contract.

Recommendation:

We advise to validate the function parameter on the aforementioned line against zero address value.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6` .

OPY-05: `require` statements are missing reason `string`

Type	Severity	Location
Language Specific	Informational	OmsPolicy.sol L75, L90, L103, L87, L170

Description:

The `require` statements on the aforementioned lines are missing reason strings.

Recommendation:

We advise to provide reason strings for the `require` statements on the aforementioned lines to increase the legibility of the codebase.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6` .

OPY-06: Redundant Variable Initialization

Type	Severity	Location
Coding Style	Informational	OmsPolicy.sol L232-L233

Description:

All variable types within Solidity are initialized to their default "empty" value, which is usually their zeroed out representation. Particularly:

- `uint` / `int` : All `uint` and `int` variable types are initialized at `0`
- `address` : All `address` types are initialized to `address(0)`
- `byte` : All `byte` types are initialized to their `byte(0)` representation
- `bool` : All `bool` types are initialized to `false`
- `ContractType` : All contract types (i.e. for a given `contract ERC20 {}` its contract type is `ERC20`) are initialized to their zeroed out address (i.e. for a given `contract ERC20 {}` its default value is `ERC20(address(0))`)
- `struct` : All `struct` types are initialized with all their members zeroed out according to this table

Recommendation:

We advise that the linked initialization statements are removed from the codebase to increase legibility.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6` .

OPY-07: Redundant Statements

Type	Severity	Location
Dead Code	Informational	OmsPolicy.sol L31, L205-L211

Description:

The linked statements do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

Recommendation:

We advise that they are removed to better prepare the code for production environments.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6`.

ORC-01: Lack of verification for constructor parameter

Type	Severity	Location
Logical Issue	Medium	Orchestrator.sol L31

Description:

The constructor on the aforementioned line does not validate address type parameter against zero address value before assigning it to state variable. If constructor parameter is passed as zero address then it will result in unwanted state of contract and once set, the involved state variable cannot be changed after the contract's deployment.

Recommendation:

We advise to introduce checks inside the constructor to validate the parameter against zero address value.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6`.

ORC-02: Unlocked Compiler Version

Type	Severity	Location
Language Specific	Informational	Orchestrator.sol L1

Description:

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation:

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.6.2` the contract should contain the following line:

```
pragma solidity 0.6.2;
```

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6`.

ORC-03: `require` statement is missing reason `string`

Type	Severity	Location
Language Specific	Informational	Orchestrator.sol L48

Description:

The `require` statement on the aforementioned line is missing reason `string`.

Recommendation:

We advise to provide reason string for the `require` statement on the aforementioned line to increase the legibility of the codebase.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6`.

ORC-04: Volatile code

Type	Severity	Location
Volatile Code	Informational	Orchestrator.sol L141-L145

Description:

The code on the aforementioned lines specify the gas to forward with the message call. As the transaction reverts in the event when any of the message calls on `L140` return `false`, it makes keeping the gas the in the caller contract ineffectual which would have only been useful if the transaction was not reverted.

Recommendation:

We advise to pass all the remaining gas to the message call on `L145` to increase the legibility of the codebase.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6`.

ORA-01: Incorrect `price` calculation

Type	Severity	Location
Logical Issue	Major	Oracle.sol L64-L71

Description:

The aforementioned code lines calculate price of `OMS` token in `USDC`. The `price0Average` and `price1Average` denote the `OMS` token price in `USDC` having 6 decimal points. As the `exchangeRate` assignment on `L102` in `OmsPolicy` contract expects the price to have 18 decimals, the price in `USDC` has to be appended with further 12 decimal points to satisfy it. However, the `L65` and `L69` appends 18 decimals points to the price resulting in large enough value resulting in `exchangeRate` set to `MAX_RATE` on `L105-L107` in `OmsPolicy` contract.

Recommendation:

We advise to remove the appending of `18` decimals points on `L65` and `L69` to have the price correctly denoted in `18` decimals.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6`.

ORA-02: Redundant variable initialization

Type	Severity	Location
Dead Code	Informational	Oracle.sol L63

Description:

The local variable `price` on the aforementioned line is assigned a value of `10**18`. This value is replaced in the following `if-else` block rendering the initialization on the aforementioned line ineffectual.

Recommendation:

We recommend to remove the ineffectual initialization on the aforementioned line to increase the legibility of the codebase.

Alleviation:

Alleviations were applied as of commit hash `752e4216f87207fcad09e16402052d7457cde3b6`.

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.

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.

Volatile Code

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

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.

Dead Code

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