# Ruler Protocol

## Security Assessment

February 21st, 2021

For :
Ruler Protocol

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

Georgios Delkos @ CertiK
georgios.delkos@certik.io

# 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 indeed 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 | Ruler Protocol |
| --- | --- |
| Description | A typical ERC20 implementation with enhanced features. |
| Platform | Ethereum; Solidity, Yul |
| Codebase | GitHub Repository |
| Commits | 1. 03276fa9def018df430906adc6e2b8a7a04edd0c<br>2. bb80a8864500656752331628664 90a5cc0c8c10b<br>3. bd0ca68307b1fa74be3941ace9a42c0a0f74267b |

## Audit Summary

| Delivery Date | February 21st, 2021 |
| --- | --- |
| Method of Audit | Static Analysis, Manual Review |
| Consultants Engaged | 2 |
| Timeline | February 1, 2021 - February 14, 2021 |

## Vulnerability Summary

| Total Issues | 14 |
| --- | --- |
| Total Critical | 0 |
| Total Major | 1 |
| Total Medium | 1 |
| Total Minor | 9 |
| Total Informational | 3 |

## Executive Summary

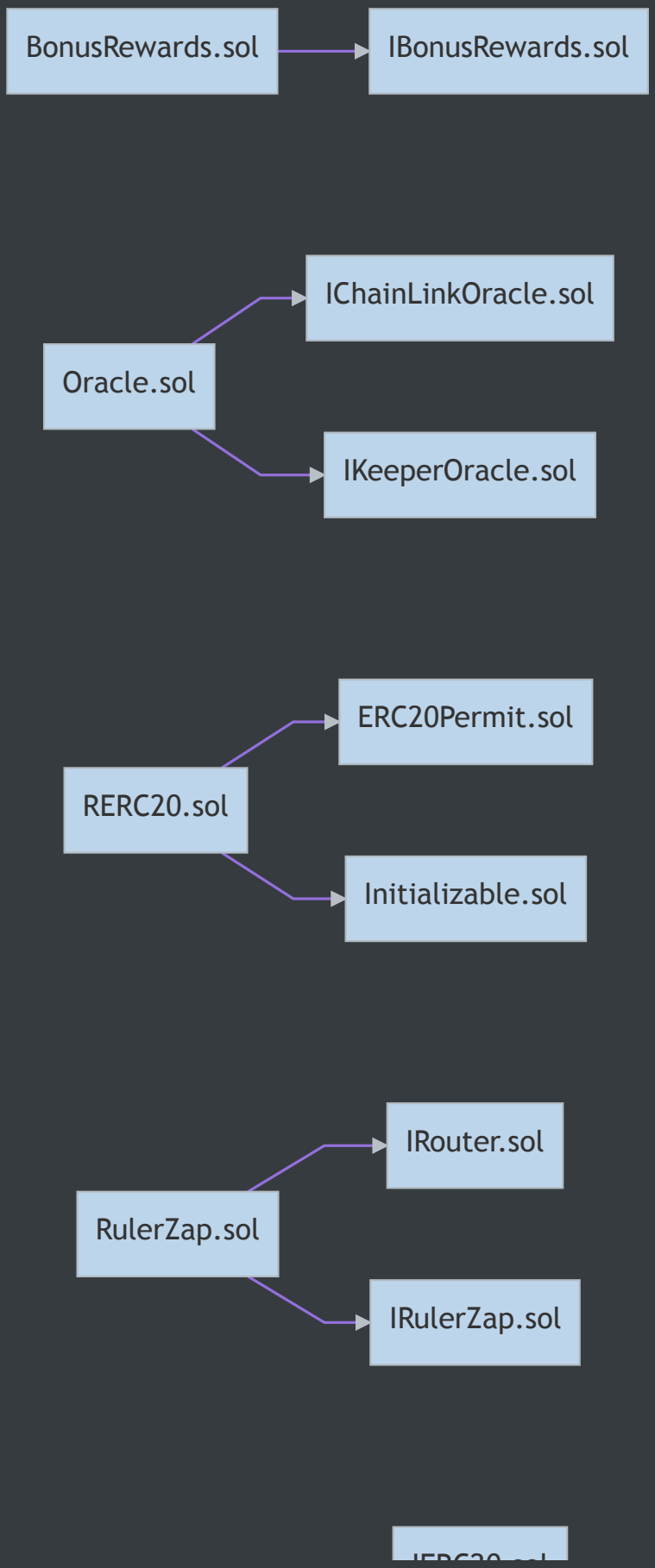
Ruler Protocol requested for CertiK to perform an audit in their new smart contract system implementation. The auditing team conducted the audit in the timeframe between February 1, 2021, and February 14, 2021, with two engineers. The auditing process evaluated code implementation against provided specifications, examining language-specific issues, and performed manual examination of the code.

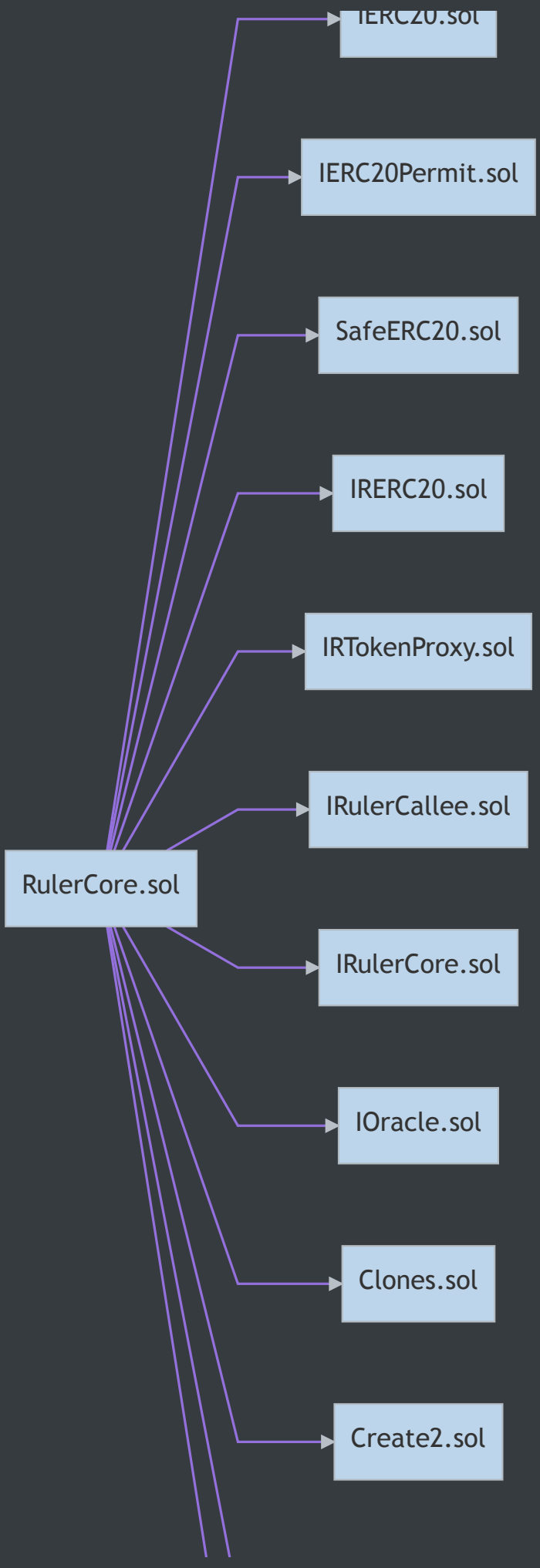
The code's examination revealed issues that the auditing team discussed with the development team and were either acknowledged or addressed in the alleviation iteration.

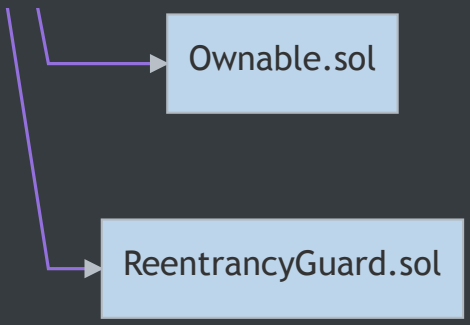
## Files In Scope

| ID | Contract | Location |
| --- | --- | --- |
| BRS | BonusRewards.sol | contracts/BonusRewards.sol |
| ORA | Oracle.sol | contracts/Oracle.sol |
| RUL | RULER.sol | contracts/RULER.sol |
| RER | RERC20.sol | contracts/RERC20.sol |
| RZP | RulerZap.sol | contracts/RulerZap.sol |
| RCE | RulerCore.sol | contracts/RulerCore.sol |

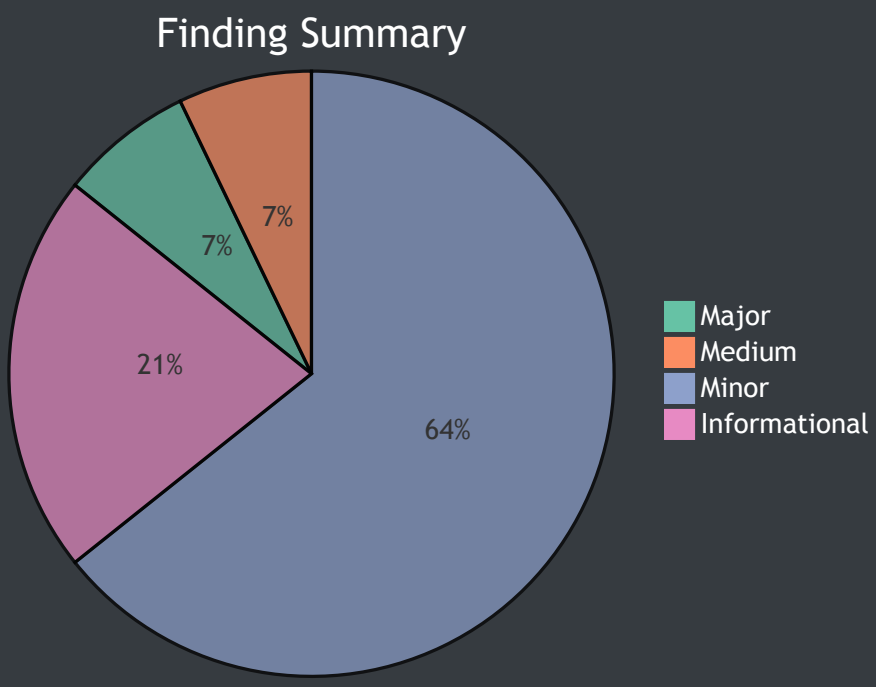
# File Dependency Graph




IERC20.sol
IERC20Permit.sol
SafeERC20.sol
IRERC20.sol
IRTokenProxy.sol
IRulerCallee.sol
RulerCore.sol
IRulerCore.sol
IOracle.sol
Clones.sol
Create2.sol


Ownable.sol
ReentrancyGuard.sol

# Findings



| ID | Title | Type | Severity | Resolved |
|---|---|---|---|---|
| BRS-01 | Unlocked Compiler Version | Language Specific | Informational |  |
| BRS-02 | Magic Number to `constant` | Coding Style | Informational |  |
| BRS-03 | Usage of `transfer()` for sending Ether | Volatile Code | Minor |  |
| BRS-04 | Requisite Value of ERC-20 `transferFrom()` / `transfer()` Call | Logical Issue | Minor |  |
| BRS-05 | Missing `event` Emmision | Language Specific | Minor |  |
| BRS-06 | Inefficient Comparison | Gas Optimization | Informational |  |
| ORA-01 | Ambiguous Implementation | Logical Issue | Medium |  |
| ORA-02 | No Check Against the Zero Address | Control Flow | Minor |  |
| RZP-01 | Add Documentation | Logical Issue | Minor |  |
| RZP-02 | Requisite Value of ERC-20 `transferFrom()` / `transfer()` Call | Logical Issue | Minor |  |
| RZP-03 | Check `internal` Functionality | Volatile Code | Minor |  |
| RZP-04 | Ambiguous `transfer` Amount | Logical Issue | Minor |  |
| RZP-05 | Unused `return` Value | Volatile Code | Minor |  |
| RCE-01 | Dangerous `if` block | Volatile Code | Major |  |

## BRS-01: Unlocked Compiler Version

| Type | Severity | Location |
| --- | --- | --- |
| Language Specific | Informational | BonusRewards.sol L3 |

### 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:

The Ruler Protocol development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

## BRS-02: Magic Number to `constant`

| Type | Severity | Location |
| --- | --- | --- |
| Coding Style | Informational | BonusRewards.sol L203 |

### Description:

The code contains a hard coded number for decimals.

### Recommendation:

We advise that the code should introduce a new `constant` variable for the decimals.

### Alleviation:

The Ruler Protocol development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

## BRS-03: Usage of `transfer()` for sending Ether

| Type | Severity | Location |
| --- | --- | --- |
| Volatile Code | Minor | BonusRewards.sol L224 |

### Description:

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

### Recommendation:

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

### Alleviation:

The development team opted to consider our references, imported OpenZeppelin's `Address` library and used its `sendValue()` function in the linked statement.

## BRS-04: Requisite Value of ERC-20 `transferFrom()` / `transfer()` Call

| Type | Severity | Location |
| --- | --- | --- |
| Logical Issue | Minor | BonusRewards.sol L236 |

### Description:

While the ERC-20 implementation does necessitate that the `transferFrom()` / `transfer()` function returns a `bool` variable yielding `true`, many token implementations do not return anything i.e. Tether (USDT) leading to unexpected halts in code execution.

### Recommendation:

We advise that the `SafeERC20.sol` library is utilized by OpenZeppelin to ensure that the `transferFrom()` / `transfer()` function is safely invoked in all circumstances.

### Alleviation:

The development team opted to consider our references, imported OpenZeppelin's `SafeERC20` library and used its `safeTransfer()` function in the linked statement.

## BRS-05: Missing `event` Emmision

| Type | Severity | Location |
| --- | --- | --- |
| Language Specific | Minor | BonusRewards.sol L244-L247 |

### Description:

The code pauses the system without providing any event that the system is paused.

### Recommendation:

We advise that the code emits an `event` when the state of the system is paused.

### Alleviation:

The development team opted to consider our references and added the `PausedStatusUpdated` event.

## BRS-06: Inefficient Comparison

| Type | Severity | Location |
| --- | --- | --- |
| Gas Optimization | Informational | BonusRewards.sol L269 |

### Description:

The linked comparison with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

### Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

### Alleviation:

The Ruler Protocol development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

## ORA-01: Ambiguous Implementation

| Type | Severity | Location |
| --- | --- | --- |
| Logical Issue | Medium | Oracle.sol L32-L46 |

### Description:

The code does not match the comments / intended functionality.

### Recommendation:

We advise to revise the linked code.

### Alleviation:

The development team opted to consider our references and added in-line documentation on the returned amounts.

## ORA-02: No Check Against the Zero Address

| Type | Severity | Location |
| --- | --- | --- |
| Control Flow | Minor | Oracle.sol L56-58 |

### Description:

The code does not check if the address for the oracle is equal to the zero address.

### Recommendation:

We advise to perform the check so that the oracle cannot be set to the zero address.

### Alleviation:

The development team opted to consider our references and added a `require` statement, checking the input value against the zero address.

## RZP-01: Add Documentation

| Type | Severity | Location |
| --- | --- | --- |
| Logical Issue | Minor | RulerZap.sol L225 |

### Description:

The linked statement arbitrarily chooses the last array value returned from the `getAmountsOut()` function.

### Recommendation:

We advise to document the intended funcionality.

### Alleviation:

The Ruler Protocol development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

## RZP-02: Requisite Value of ERC-20 `transferFrom()` / `transfer()` Call

| Type | Severity | Location |
| --- | --- | --- |
| Logical Issue | Minor | RulerZap.sol L302, L306 |

### Description:

While the ERC-20 implementation does necessitate that the `transferFrom()` / `transfer()` function returns a `bool` variable yielding `true`, many token implementations do not return anything i.e. Tether (USDT) leading to unexpected halts in code execution.

### Recommendation:

We advise that the `SafeERC20.sol` library is utilized by OpenZeppelin to ensure that the `transferFrom()` / `transfer()` function is safely invoked in all circumstances.

### Alleviation:

The development team opted to consider our references, imported OpenZeppelin's `SafeERC20` library and used its `safeTransfer()` function in the linked statements.

## RZP-03: Check `internal` Functionality

| Type | Severity | Location |
| --- | --- | --- |
| Volatile Code | Minor | RulerZap.sol L278, L299, L360, L410, L437, L443, L455 |

### Description:

The linked `internal` functions do not properly sanitize the input values.

### Recommendation:

We advise to add `require` checks to the linked `internal` functions, especially the ones that do not return a value, to ensure correct execution.

### Alleviation:

The Ruler Protocol development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase.

## RZP-04: Ambiguous `transfer` Amount

| Type | Severity | Location |
| --- | --- | --- |
| Logical Issue | Minor | RulerZap.sol L406, L407 |

### Description:

The linked statements return the amount of token in the pool before the deposit from the caller.

### Recommendation:

We advise to add descriptive documentation regarding this process.

### Alleviation:

The development team opted to consider our references and added descriptive documentation for the linked functionality.

Text

## RZP-05: Unused `return` Value

| Type | Severity | Location |
| --- | --- | --- |
| Volatile Code | Minor | RulerZap.sol L458 |

### Description:

The `_transferRem()` function omits the returned value of `transfer()`.

### Recommendation:

We advise to use `safeTansfer()` instead.

### Alleviation:

The development team opted to consider our references, imported OpenZeppelin's `SafeERC20` library and used its `safeTransfer()` function in the linked statement.

## RCE-01: Dangerous `if` block

| Type | Severity | Location |
| --- | --- | --- |
| Volatile Code | Major | RulerCore.sol L472-L481 |

### Description:

The linked code segment should be protected with checks that will ensure oracle is set properly.

### Recommendation:

We advise to change all the `if` statements to `require` ones.

### Alleviation:

The development team opted to consider our references and added descriptive documentation for the "no oracle" edge case.

"descriptive documentation" from Code:

" // Oracle price is not required, the consequence is low since it will just allow users to deposit collateral (which can be collected thro repay before expiry. If default, early repayments will be diluted"

# 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.

### Mathematical Operations

Mathematical Operation 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.