EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Amplify Protocol
Platform:	Cronos Blockchain
Language:	Solidity
Date:	April 23rd, 2022

Table of contents

Introduction

EtherAuthority was contracted by Amplify to perform the Security audit of the Amplify Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on April 23rd, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

The Amplify Contracts have functions like createVaultToken, addLiquidity, reinvest, redeem, mint, sync, skim, borrow, createCollateral, canBorrow, deployCollateral, etc.

Audit scope

Name	**Code Review and Security Analysis Report for Amplify Protocol Smart Contracts**
Platform	**Cronos / Solidity**
File 1	BAllowance.sol
File 1 MD5 Hash	3037CD07486AC11E486706363A61705E
File 2	BDeployer.sol
File 2 MD5 Hash	F5E9E9E044066B702061ED2F70F70886
File 3	BInterestRateModel.sol
File 3 MD5 Hash	145CC218BC4156995B3C4340812BEFBA
File 4	Borrowable.sol
File 4 MD5 Hash	D1C74897867CFC80CC68C50B7705EEF1
File 5	BSetter.sol
File 5 MD5 Hash	C0A8E066647627566E97F1057538FF28

File 6	BStorage.sol
File 6 MD5 Hash	013D8BD79C889528C8C75000DC4FAD36
File 7	CDeployer.sol
File 7 MD5 Hash	1CF3DD65C627C335C3FB431418A9A1F3
File 8	Collateral.sol
File 8 MD5 Hash	F76A7E47A6F43B08EACB228F6CA4758D
File 9	CSetter.sol
File 9 MD5 Hash	7A1C408CFD52CB44650026781726962C
File 10	CStorage.sol
File 10 MD5 Hash	930F31DC19A5E1B67C609E6E4626FD94
File 11	EleosERC20.sol
File 11 MD5 Hash	1E12219135BA114154FF3AEE3CE10707
File 12	EleosPriceOracle.sol
File 12 MD5 Hash	1A7CFC71811F512D1F737D8377B4CF1C
File 13	Factory.sol
File 13 MD5 Hash	4830BDCBD9ECC64D1E5649AAE3F10535
File 14	PoolToken.sol
File 14 MD5 Hash	839D027AC3B62D900369467CE554B62F
File 15	Router02.sol
File 15 MD5 Hash	3725CD2EEE7EE6900621D9BFA2F042D5
File 16	VaultToken.sol
File 16 MD5 Hash	5EB119311125FC93EFC8DD6B7BA81CB0
File 17	VaultTokenFactory.sol
File 17 MD5 Hash	E54EF0F2523535DC387AEB197B4242B7
Audit Date	April 23rd,2022

Claimed Smart Contract Features

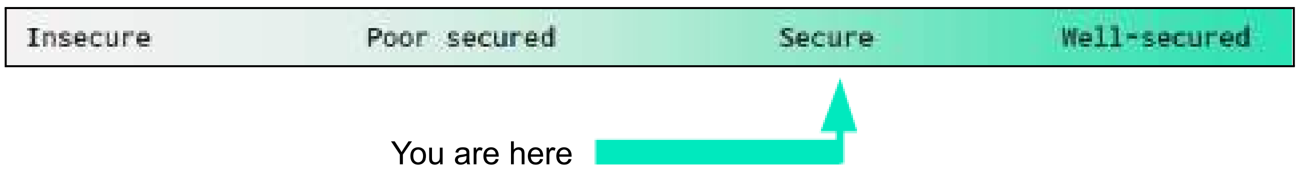
Claimed Feature Detail	Our Observation
File 1 BAllowance.sol ● Decimals: 18 ● Reserve Factor: 0.1 ● Minimum Liquidity: 1000 ● KinkUtilization:0.7 ● Borrow Index: 1	**YES, This is valid.**
File 2 BDeployer.sol ● Borrow Fee: 0.1%	**YES, This is valid.**
File 3 BInterestRateModel.sol ● Kink Multiplier: 5 ● Kink Borrow Rate Max: 100% Per Year ● Kink Borrow Rate min: 1% Per Year	**YES, This is valid.**
File 4 Borrowable.sol ● Borrow Fee: 0.1% ● Decimals: 18 ● Kink UR Maximum: 1 ● Kink UR Min: 0.5 ● KinkUtilization: 0.7 ● Minimum Liquidity: 10000	**YES, This is valid.**
File 5 BSetter.sol ● Reserve Factor Max: 20% ● Kink Ur Min: 50% ● Kink Ur Max: 99% ● Adjust Speed Min: 0.5% Per Day ● Adjust Speed Max: 50% Per Day	**YES, This is valid.**
File 6 BStorage.sol ● kink Borrow Rate: 10% per year	**YES, This is valid.**

● Reserve Factor: 10% ● Kink Utilization Rate: 70% ● Adjust Speed: 5% per day	
File 7 CDeployer.sol ● The CDeployer contract is used by the Factory to deploy Collateral(s).	**YES, This is valid.**
File 8 Collateral.sol ● Liquidation Incentive Min: 100% ● Liquidation Incentive Max: 105% ● Safety Margin Min: 100% ● Safety Margin Max: 250%	**YES, This is valid.**
File 9 CSetter.sol ● Liquidation Incentive Min: 100% ● Liquidation Incentive Max: 105% ● Safety Margin Min: 100% ● Safety Margin Max: 250%	**YES, This is valid.**
File 10 CStorage.sol ● Safety Margin Sqrt: 250% ● Liquidation Incentive: 4%	**YES, This is valid.**
File 11 EleosERC20.sol ● Decimals: 18	**YES, This is valid.**
File 12 EleosPriceOracle.sol ● Min T: 1200	**YES, This is valid.**
File 13 Factory.sol ● Factory has functions like: _createLendingPool, createCollateral, createBorrowable0, etc.	**YES, This is valid.**
File 14 PoolToken.sol ● Decimals: 18	**YES, This is valid.**

● Minimum Liquidity: 1000	
File 15 Router02.sol ● Router02 has functions like: mint, mintETH, mintCollateral, Redeem, etc.	**YES, This is valid.**
File 16 VaultToken.sol ● Name: Eleos Vault Token ● Symbol: vELEOS ● Decimals: 18 ● Reinvest Bounty: 0.1	**YES, This is valid.**
File 17 VaultTokenFactory.sol ● VaultTokenFactory has functions like: allVaultTokensLength, createVaultToken, etc.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. Also, these contracts do not contain owner control, which does make them fully decentralized.



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 1 low and some very low level issues.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	"Out of Gas" Issue	Passed
	High consumption 'for/while' loop	Passed
	High consumption 'storage' storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	"Short Address" Attack	Passed
	"Double Spend" Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 17 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the Amplify Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Amplify Protocol.

The Amplify team has not provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **not** well commented on smart contracts.

Documentation

We were given an Amplify Protocol smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented. So it is not easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

BAllowance.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	safe112	internal	Passed	No Issue
3	_setFactory	external	Passed	No Issue
4	_update	internal	Passed	No Issue
5	exchangeRate	write	Passed	No Issue
6	mint	external	Passed	No Issue
7	redeem	read	Passed	No Issue
8	skim	external	Passed	No Issue
9	sync	external	Passed	No Issue
10	_safeTransfer	internal	Passed	No Issue
11	nonReentrant	modifier	Passed	No Issue
12	update	modifier	Passed	No Issue
13	_borrowApprove	write	Passed	No Issue
14	borrowApprove	external	Passed	No Issue
15	_checkBorrowAllowance	internal	Passed	No Issue
16	borrowPermit	external	Passed	No Issue

BDeployer.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	deployBorrowable	external	Passed	No Issue

BInterestRateModel.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	safe112	internal	Passed	No Issue
3	_setFactory	external	Passed	No Issue
4	_update	internal	Passed	No Issue
5	exchangeRate	write	Passed	No Issue
6	mint	external	Passed	No Issue
7	redeem	read	Passed	No Issue
8	skim	external	Passed	No Issue
9	sync	external	Passed	No Issue
10	_safeTransfer	internal	Passed	No Issue

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	safe112	internal	Passed	No Issue
3	_setFactory	external	Passed	No Issue
4	_update	internal	Passed	No Issue
5	exchangeRate	write	Passed	No Issue
6	mint	external	Passed	No Issue
7	nonReentrant	modifier	Passed	No Issue
8	update	modifier	Passed	No Issue
9	_calculateBorrowRate	internal	Passed	No Issue
10	accrueInterest	write	Passed	No Issue
11	getBlockTimestamp	read	Passed	No Issue

Borrowable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	safe112	internal	Passed	No Issue
3	_setFactory	external	Passed	No Issue
4	_update	internal	Passed	No Issue
5	exchangeRate	write	Passed	No Issue
6	mint	external	Passed	No Issue
7	redeem	read	Passed	No Issue
8	skim	external	Passed	No Issue
9	sync	external	Passed	No Issue
10	_safeTransfer	internal	Passed	No Issue
11	nonReentrant	modifier	Passed	No Issue
12	update	modifier	Passed	No Issue
13	_initialize	external	Passed	No Issue
14	_setReserveFactor	external	Passed	No Issue
15	_setKinkUtilizationRate	external	Passed	No Issue
16	_setAdjustSpeed	external	Passed	No Issue
17	_setBorrowTracker	external	Passed	No Issue
18	_checkSetting	internal	Passed	No Issue
19	_checkAdmin	internal	Passed	No Issue
20	_borrowApprove	write	Passed	No Issue
21	borrowApprove	external	Passed	No Issue
22	_checkBorrowAllowance	internal	Passed	No Issue
23	borrowPermit	external	Passed	No Issue
24	_calculateBorrowRate	internal	Passed	No Issue
25	accrueInterest	write	Passed	No Issue
26	getBlockTimestamp	read	Passed	No Issue
27	_update	internal	Passed	No Issue
28	_mintReserves	internal	Passed	No Issue
29	exchangeRate	write	Passed	No Issue

30	sync	external	Passed	No Issue
31	borrowBalance	read	Passed	No Issue
32	_trackBorrow	internal	Passed	No Issue
33	_updateBorrow	write	Passed	No Issue
34	borrow	external	Passed	No Issue
35	liquidate	external	Passed	No Issue
36	trackBorrow	external	Passed	No Issue
37	accrue	modifier	Passed	No Issue

BSetter.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	safe112	internal	Passed	No Issue
3	_setFactory	external	Passed	No Issue
4	_update	internal	Passed	No Issue
5	exchangeRate	write	Passed	No Issue
6	mint	external	Passed	No Issue
7	redeem	read	Passed	No Issue
8	skim	external	Passed	No Issue
9	sync	external	Passed	No Issue
10	_safeTransfer	internal	Passed	No Issue
11	nonReentrant	modifier	Passed	No Issue
12	update	modifier	Passed	No Issue
13	_initialize	external	Passed	No Issue
14	_setReserveFactor	external	Passed	No Issue
15	_setKinkUtilizationRate	external	Passed	No Issue
16	_setAdjustSpeed	external	Passed	No Issue
17	_setBorrowTracker	external	Passed	No Issue
18	_checkSetting	internal	Passed	No Issue
19	_checkAdmin	internal	Passed	No Issue

BStorage.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	safe112	internal	Passed	No Issue

CDeployer.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Warning: Visibility for constructor is ignored	Refer Audit Findings
2	deployCollateral	external	Passed	No Issue

Collateral.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getPrices	write	Passed	No Issue
3	_calculateLiquidityAndShortfall	read	Passed	No Issue
4	_calculateLiquidity	internal	Passed	No Issue
5	_transfer	internal	Passed	No Issue
6	tokensUnlocked	write	Passed	No Issue
7	accountLiquidityAmounts	write	Passed	No Issue
8	accountLiquidity	write	Passed	No Issue
9	exchangeRate	read	Passed	No Issue
10	_computePrice	read	Passed	No Issue
11	accountLiquidityStale	read	Passed	No Issue
12	canBorrow	write	Passed	No Issue
13	seize	external	Passed	No Issue
14	flashRedeem	external	Passed	No Issue

CSetter.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_initialize	external	Passed	No Issue
3	_setSafetyMarginSqrt	external	Passed	No Issue
4	_setLiquidationIncentive	external	Passed	No Issue
5	_checkSetting	internal	Passed	No Issue
6	_checkAdmin	internal	Passed	No Issue
7	_setFactory	external	Passed	No Issue
8	_update	internal	Passed	No Issue
9	exchangeRate	write	Passed	No Issue
10	mint	external	Passed	No Issue
11	redeem	external	Passed	No Issue
12	skim	external	Passed	No Issue

13	sync	external	Passed	No Issue
14	_safeTransfer	internal	Passed	No Issue
15	nonReentrant	modifier	Passed	No Issue
16	update	modifier	Passed	No Issue

EleosERC20.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_setName	internal	Passed	No Issue
3	_mint	internal	Passed	No Issue
4	_burn	internal	Passed	No Issue
5	_approve	write	Passed	No Issue
6	_transfer	internal	Passed	No Issue
7	approve	external	Passed	No Issue
8	transfer	external	Passed	No Issue
9	transferFrom	external	Passed	No Issue
10	_checkSignature	internal	Passed	No Issue
11	permit	external	Passed	No Issue

EleosPriceOracle.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	toUint224	internal	Passed	No Issue
3	getPriceCumulativeCurrent	internal	Passed	No Issue
4	initialize	external	Passed	No Issue
5	getResultStale	external	Passed	No Issue
6	getResult	external	Passed	No Issue
7	getBlockTimestamp	read	Passed	No Issue

Factory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	allLendingPoolsLength	external	Passed	No Issue
3	_getTokens	read	Passed	No Issue
4	_createLendingPool	write	Passed	No Issue
5	createCollateral	external	Passed	No Issue
6	createBorrowable0	external	Passed	No Issue

7	createBorrowable1	external	Passed	No Issue
8	initializeLendingPool	external	Passed	No Issue
9	_setPendingAdmin	external	Passed	No Issue
10	_acceptAdmin	external	Passed	No Issue
11	_setReservesPendingAdmin	external	Passed	No Issue
12	_acceptReservesAdmin	external	Passed	No Issue
13	_setReservesManager	external	Passed	No Issue

PoolToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_setName	internal	Passed	No Issue
3	_mint	internal	Passed	No Issue
4	_burn	internal	Passed	No Issue
5	_approve	write	Passed	No Issue
6	_transfer	write	Passed	No Issue
7	approve	external	Passed	No Issue
8	transfer	external	Passed	No Issue
9	transferFrom	external	Passed	No Issue
10	_checkSignature	internal	Passed	No Issue
11	permit	external	Passed	No Issue
12	_setFactory	external	Anyone can call setFactory() external function	Refer Audit Findings
13	_update	internal	Passed	No Issue
14	exchangeRate	write	Passed	No Issue
15	mint	external	Passed	No Issue
16	redeem	external	Passed	No Issue
17	skim	external	Passed	No Issue
18	sync	external	Passed	No Issue
19	_safeTransfer	internal	Passed	No Issue
20	nonReentrant	modifier	Passed	No Issue
21	update	modifier	Passed	No Issue

Router02.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Warning: Visibility for constructor is ignored	Refer Audit Findings
2	ensure	modifier	Passed	No Issue
3	checkETH	modifier	Passed	No Issue

4	receive	external	Passed	No Issue
5	_mint	internal	Passed	No Issue
6	mint	external	Passed	No Issue
7	mintETH	external	Passed	No Issue
8	mintCollateral	external	Passed	No Issue
9	redeem	write	Passed	No Issue
10	redeemETH	write	Passed	No Issue
11	borrow	write	Passed	No Issue
12	borrowETH	write	Passed	No Issue
13	_repayAmount	internal	Passed	No Issue
14	repay	external	Passed	No Issue
15	repayETH	external	Passed	No Issue
16	liquidate	external	Passed	No Issue
17	liquidateETH	external	Passed	No Issue
18	_leverage	internal	Passed	No Issue
19	leverage	external	Passed	No Issue
20	_addLiquidityAndMint	internal	Passed	No Issue
21	deleverage	external	Passed	No Issue
22	_removeLiqAndRepay	internal	Passed	No Issue
23	_repayAndRefund	internal	Passed	No Issue
24	eleosBorrow	external	Passed	No Issue
25	eleosRedeem	external	Passed	No Issue
26	_permit	internal	Passed	No Issue
27	_borrowPermit	internal	Passed	No Issue
28	_optimalLiquidity	read	Passed	No Issue
29	_quote	internal	Passed	No Issue
30	isVaultToken	read	Passed	No Issue
31	getUniswapV2Pair	read	Passed	No Issue
32	getBorrowable	read	Passed	No Issue
33	getCollateral	read	Passed	No Issue
34	getLendingPool	read	Passed	No Issue

VaultToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_setFactory	external	Passed	No Issue
3	_update	internal	Passed	No Issue
4	exchangeRate	write	Passed	No Issue
5	mint	external	Passed	No Issue
6	redeem	external	Passed	No Issue
7	skim	external	Passed	No Issue
8	sync	external	Passed	No Issue
9	_safeTransfer	internal	Passed	No Issue
10	nonReentrant	modifier	Passed	No Issue
11	update	modifier	Passed	No Issue

12	_initialize	external	Passed	No Issue
13	_update	internal	Passed	No Issue
14	mint	external	Passed	No Issue
15	redeem	external	Passed	No Issue
16	_optimalDepositA	internal	Passed	No Issue
17	approveRouter	internal	Passed	No Issue
18	swapExactTokensForTokens	internal	Passed	No Issue
19	addLiquidity	internal	Passed	No Issue
20	reinvest	external	Passed	No Issue
21	getReserves	external	Passed	No Issue
22	price0CumulativeLast	external	Passed	No Issue
23	price1CumulativeLast	external	Passed	No Issue
24	safe112	internal	Passed	No Issue

VaultTokenFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Warning: Visibility for constructor is ignored	Refer Audit Findings
2	allVaultTokensLength	external	Passed	No Issue
3	createVaultToken	external	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

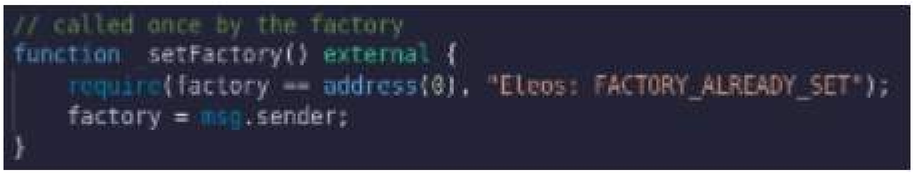
No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Anyone can call setFactory() external function: **PoolToken.sol**



```
// called once by the factory
function  setFactory() external {
    require(factory == address(0), "Eleos: FACTORY_ALREADY_SET");
    factory = msg.sender;
}
```

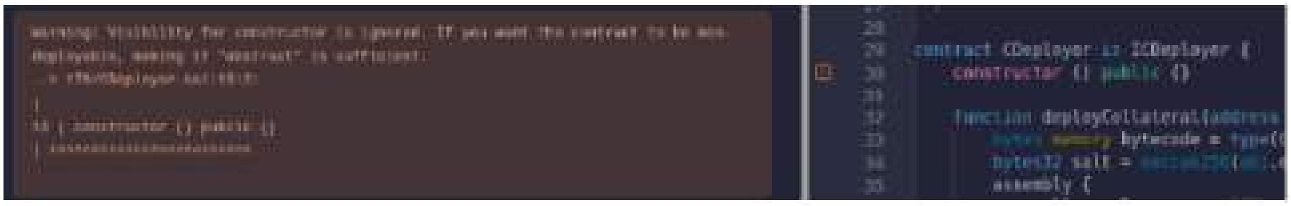
There is an external function _setFactory(), Anyone can call that function and update the factory address owner itself.

Resolution: Deployer has to confirm before deploying the contract to production.

Very Low / Informational / Best practices:

(1) Warning: Visibility for constructor is ignored:

CDeployer.sol

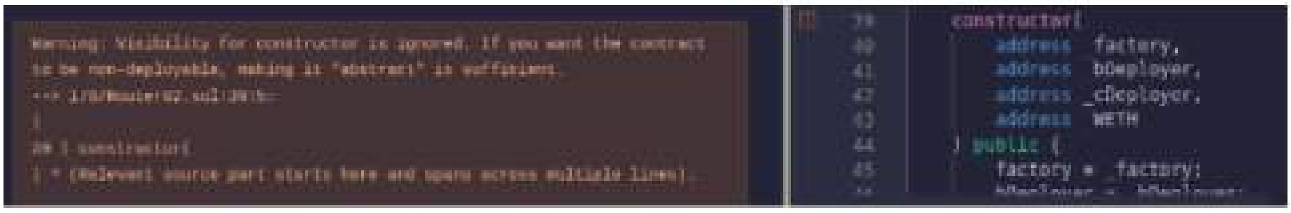


```
Warning: Visibility for constructor is ignored. If you want the contract to be non-
deployable, making it "abstract" is sufficient.
  --> [illegible]
   |
30 | constructor () public {}
   | ^^^^^^^^^^^^^^^^^^^^^^^^
```

```
contract CDeployer is ICDeployer {
    constructor () public {}

    function deployCollateral(address
        bytes memory bytecode = type(
        bytes32 salt = keccak256(
        assembly {
```

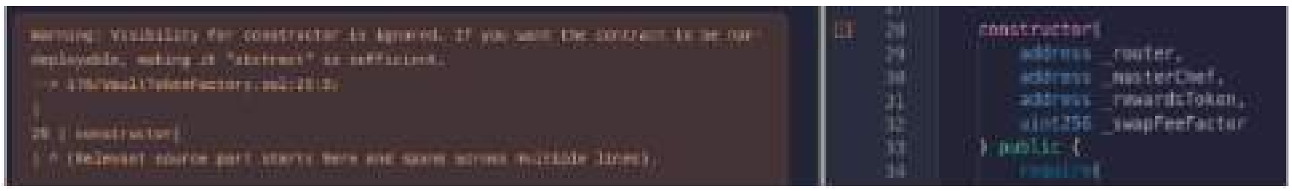
Router02.sol



```
Warning: Visibility for constructor is ignored. If you want the contract
to be non-deployable, making it "abstract" is sufficient.
--> [illegible]/Router02.sol:[illegible]
|
[illegible] | constructor(
| ^ (Relevant source part starts here and spans across multiple lines).
```

```
constructor(
    address _factory,
    address _bDeployer,
    address _cDeployer,
    address _WETH
) public {
    factory = _factory;
```

VaultTokenFactory.sol



```
Warning: Visibility for constructor is ignored. If you want the contract to be non-
deployable, making it "abstract" is sufficient.
--> [illegible]/VaultTokenFactory.sol:[illegible]
|
28 | constructor(
| ^ (Relevant source part starts here and spans across multiple lines).
```

```
constructor(
    address _router,
    address _masterChef,
    address _rewardsToken,
    uint256 _swapFeeFactor
) public {
    require(
```

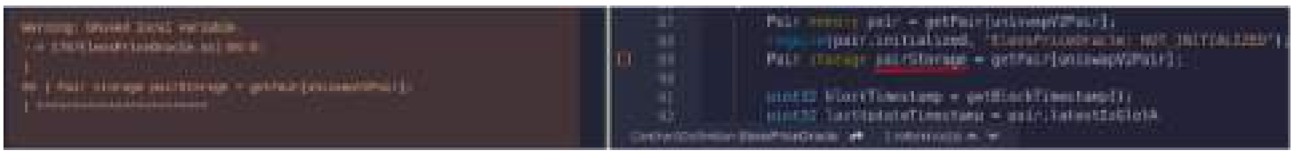
Resolution: Warning: Visibility for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient.

(2) SafeMath Library: **EleosERC20.sol, PoolToken.sol**

SafeMath Library is used in this contract code, but the compiler version is greater than or equal to 0.8.0, Then it will not be required to use, solidity automatically handles overflow / underflow.

Resolution: We suggest removing the SafeMath library and use normal math operators, It will improve code size, and less gas consumption.

(3) Warning: Unused local variable: **EleosPriceOracle.sol**



```
Warning: Unused local variable.
--> [illegible]/EleosPriceOracle.sol:[illegible]
|
[illegible] | Pair storage pairStorage = getPair[uniswapV2Pair];
| ^^^^^^^^^^^^^^^^^^^^^^^^
```

```
Pair memory pair = getPair[uniswapV2Pair];
require(pair.initialized, "EleosPriceOracle: NOT_INITIALIZED");
Pair storage pairStorage = getPair[uniswapV2Pair];

uint32 blockTimestamp = getBlockTimestamp();
uint32 lastUpdateTimestamp = pair.latestIsSlotA
```

Warning: Unused local variable.

Pair storage pairStorage = getPair[uniswapV2Pair];

Resolution: We suggest removing unused variables from code.

Conclusion

We were given a contract code in the form of files. And we have used all possible tests based on given objects as files. We had observed some issues in the smart contracts, but they were resolved in the revised smart contract code. **So, the smart contracts are ready for the mainnet deployment**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured".**

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:
In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:
Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

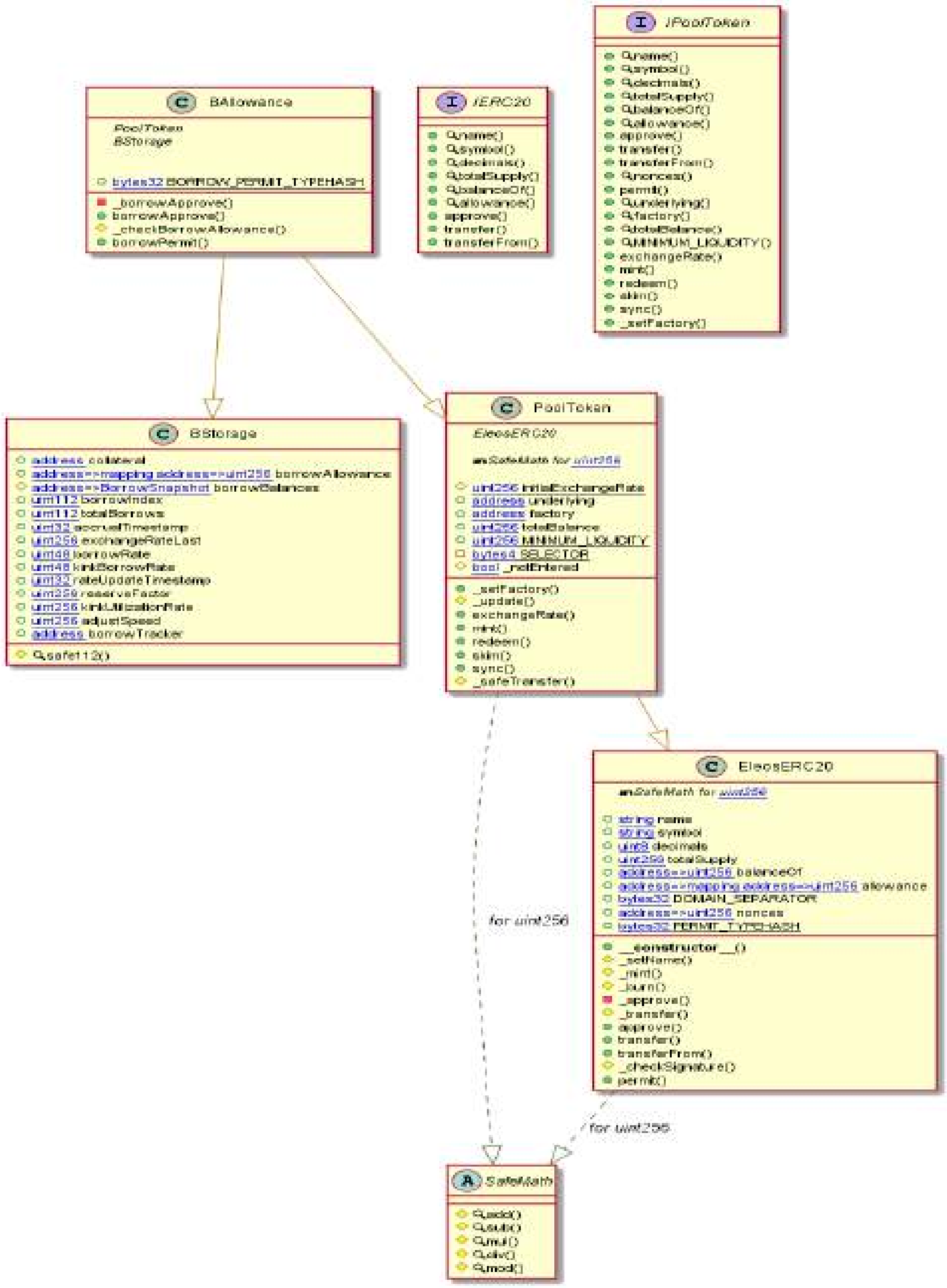
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

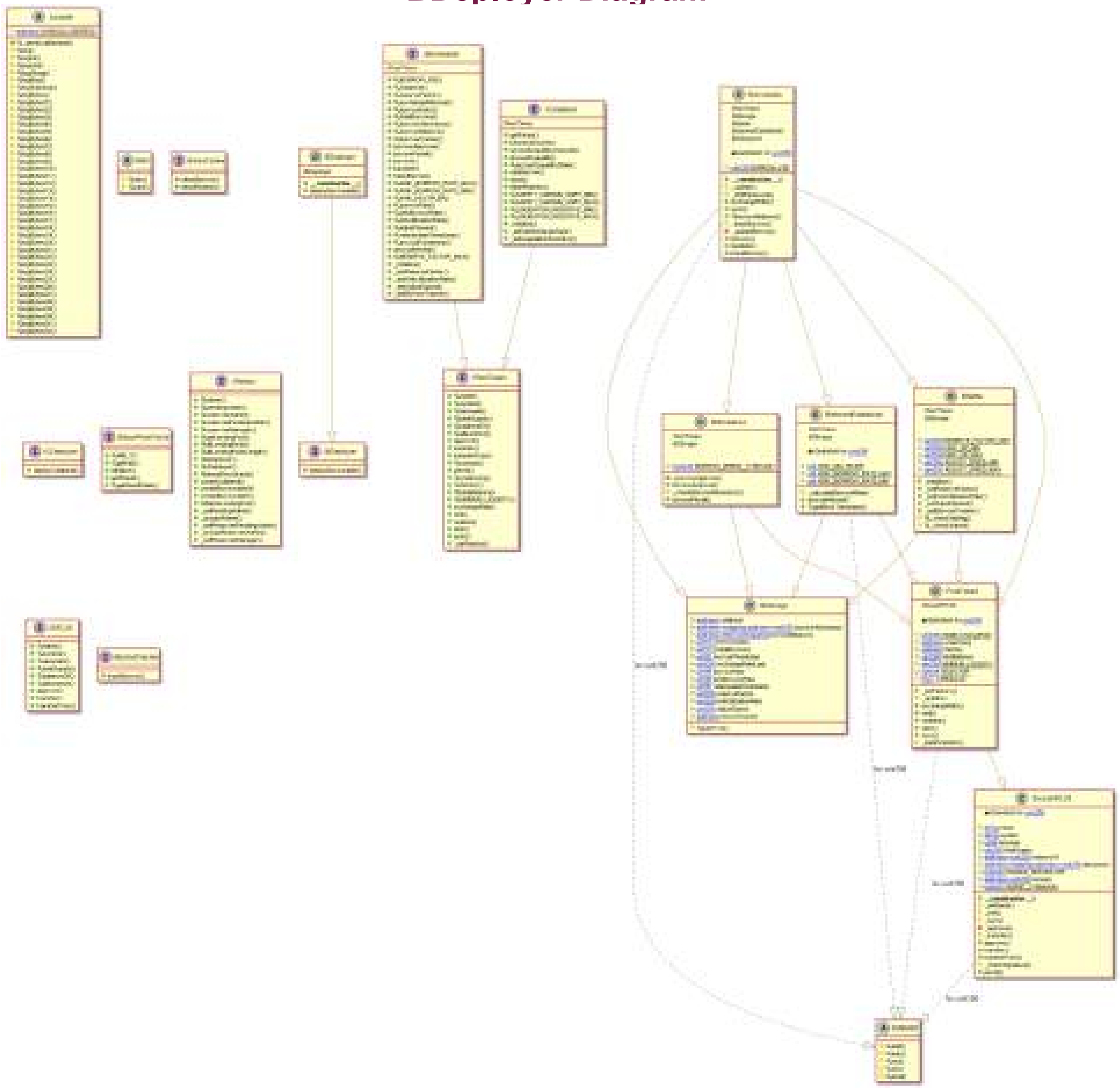
Appendix

Code Flow Diagram - Amplify Protocol

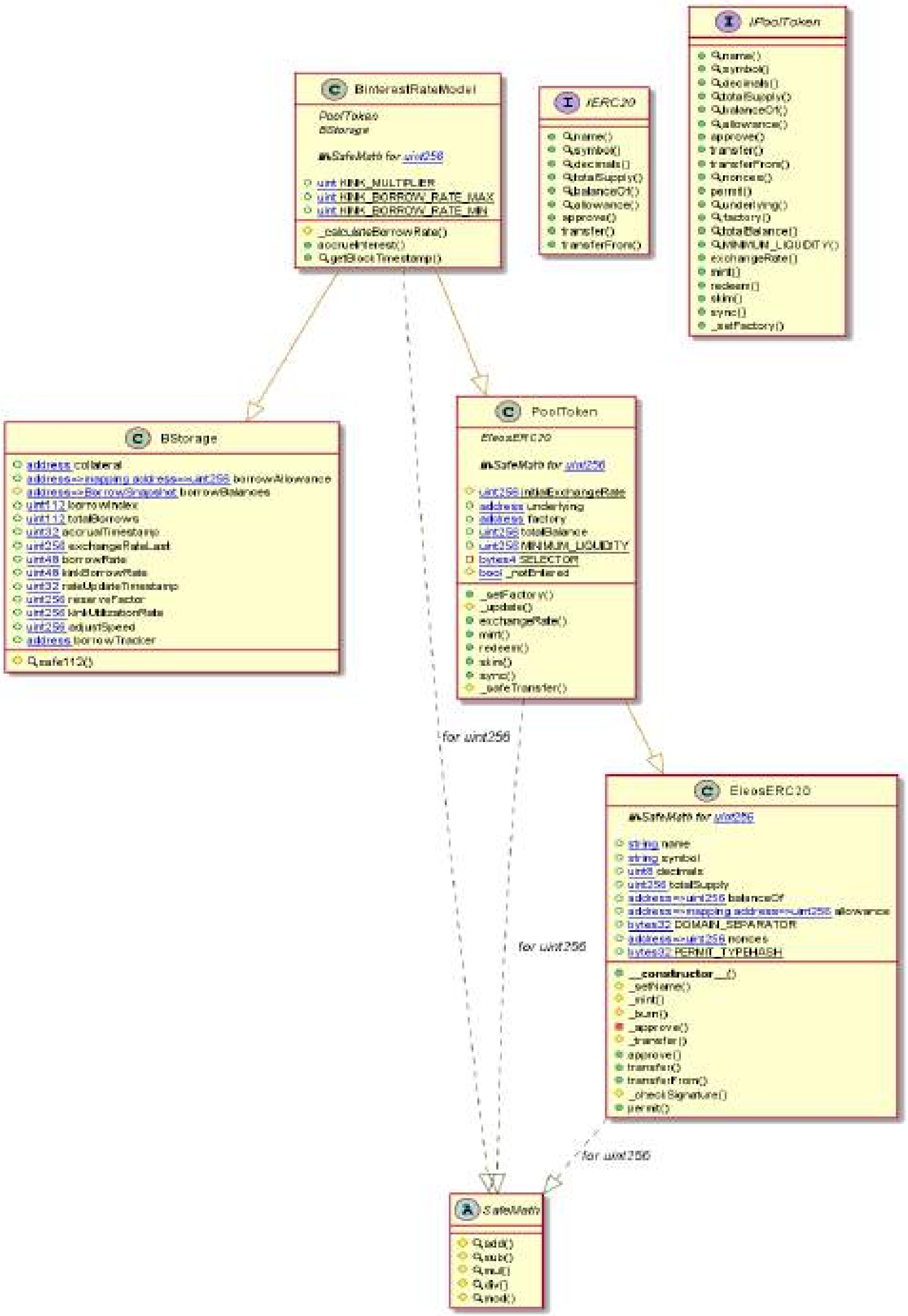
BAllowance Diagram



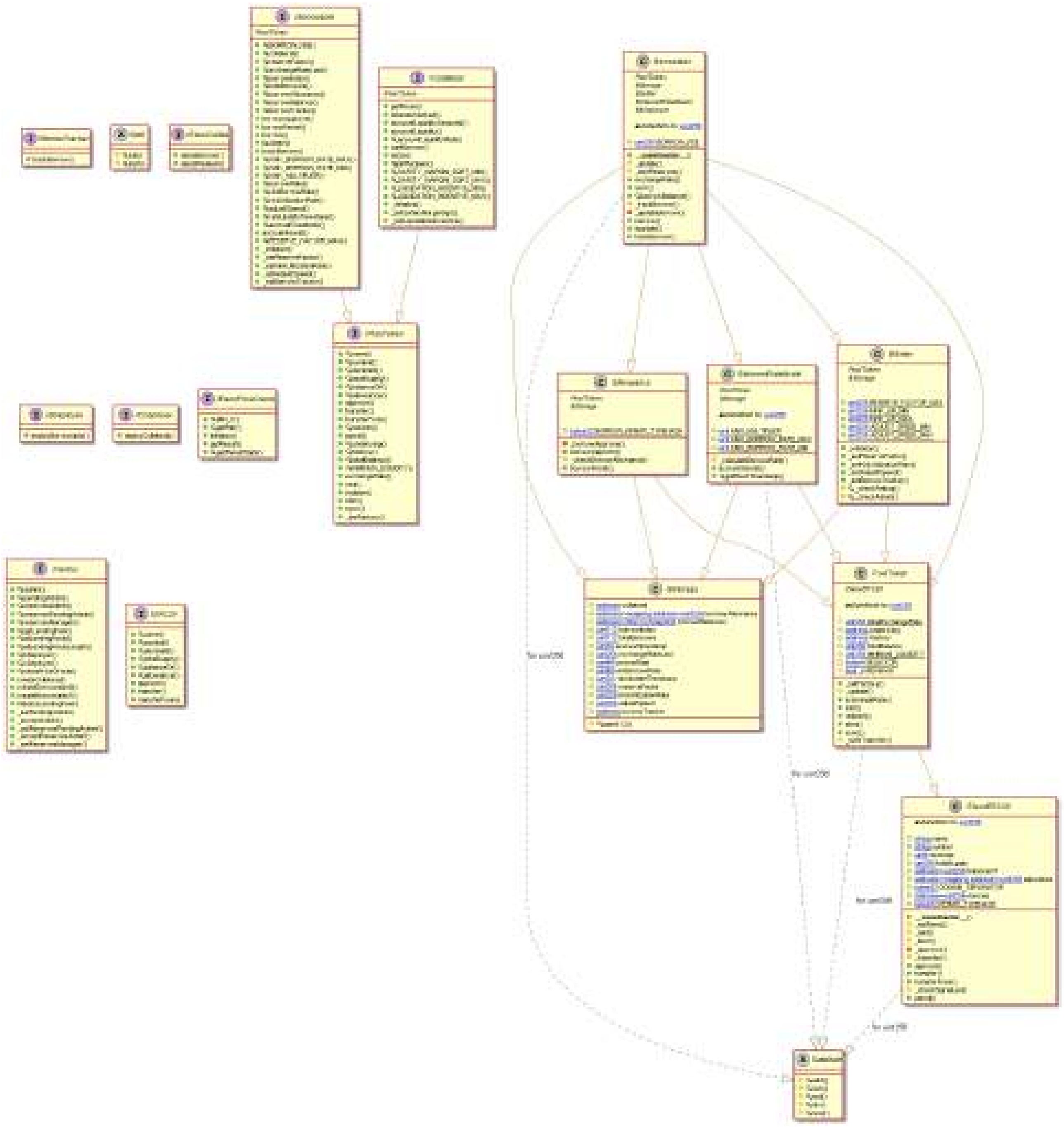
BDeployer Diagram



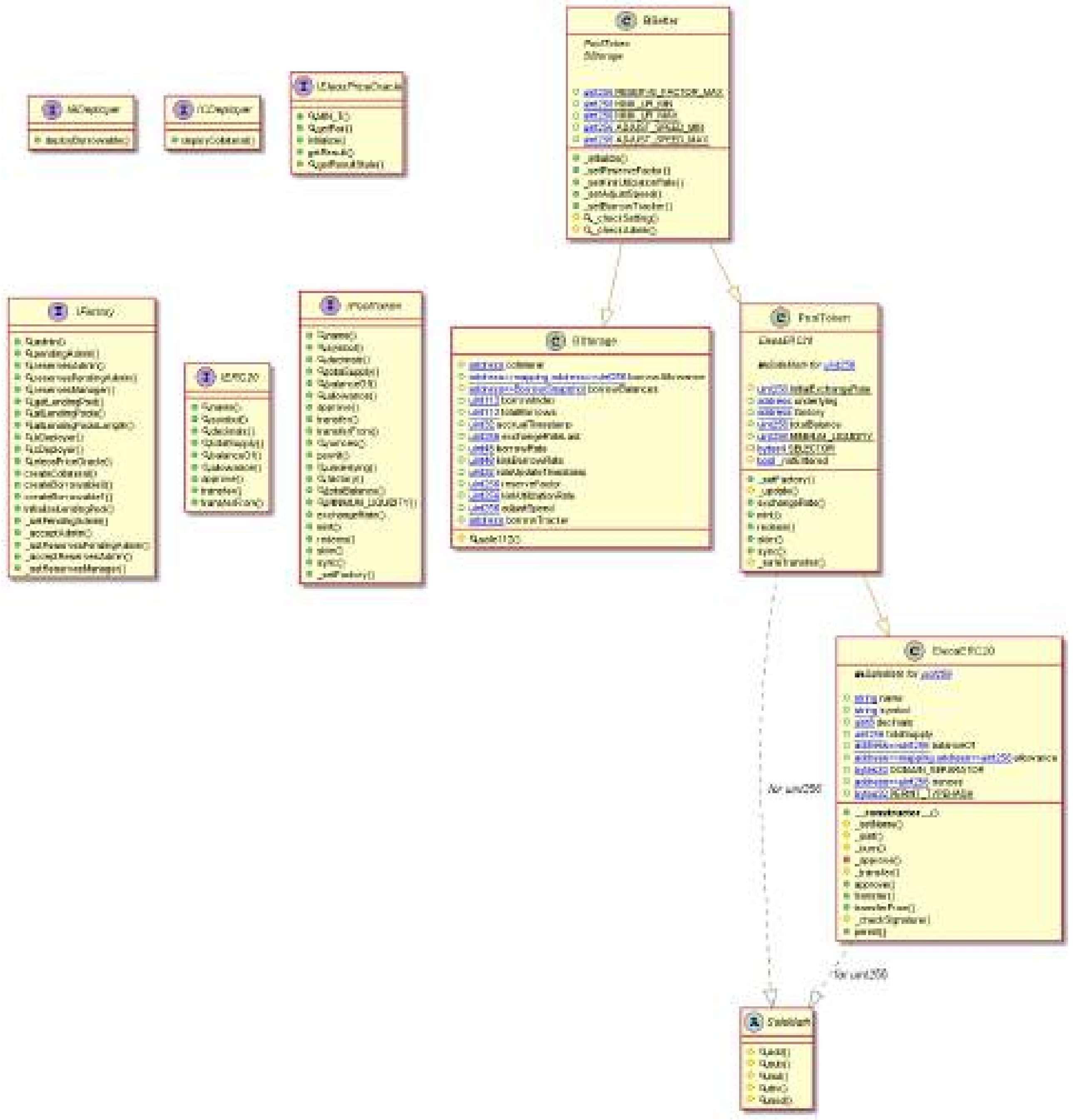
BInterestRateModel Diagram



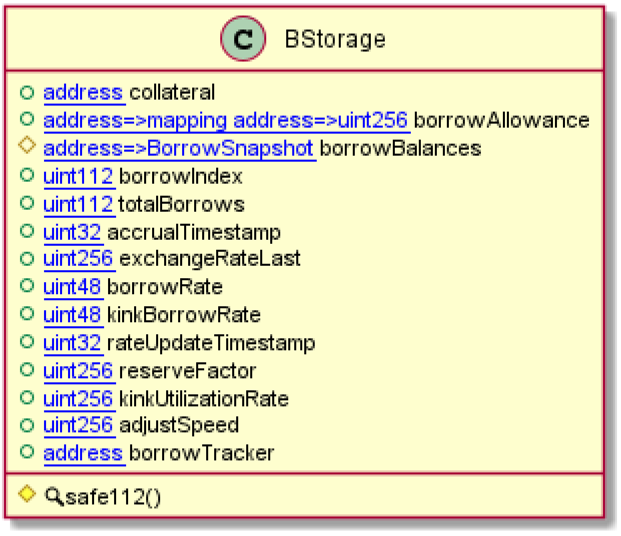
Borrowable Diagram



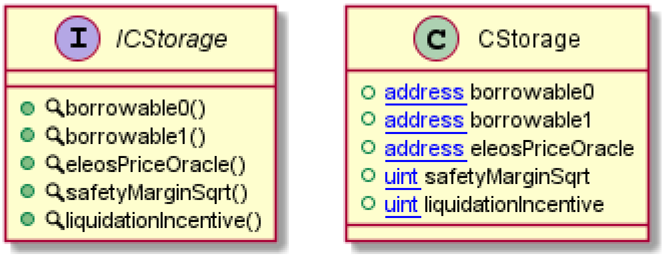
BSetter Diagram



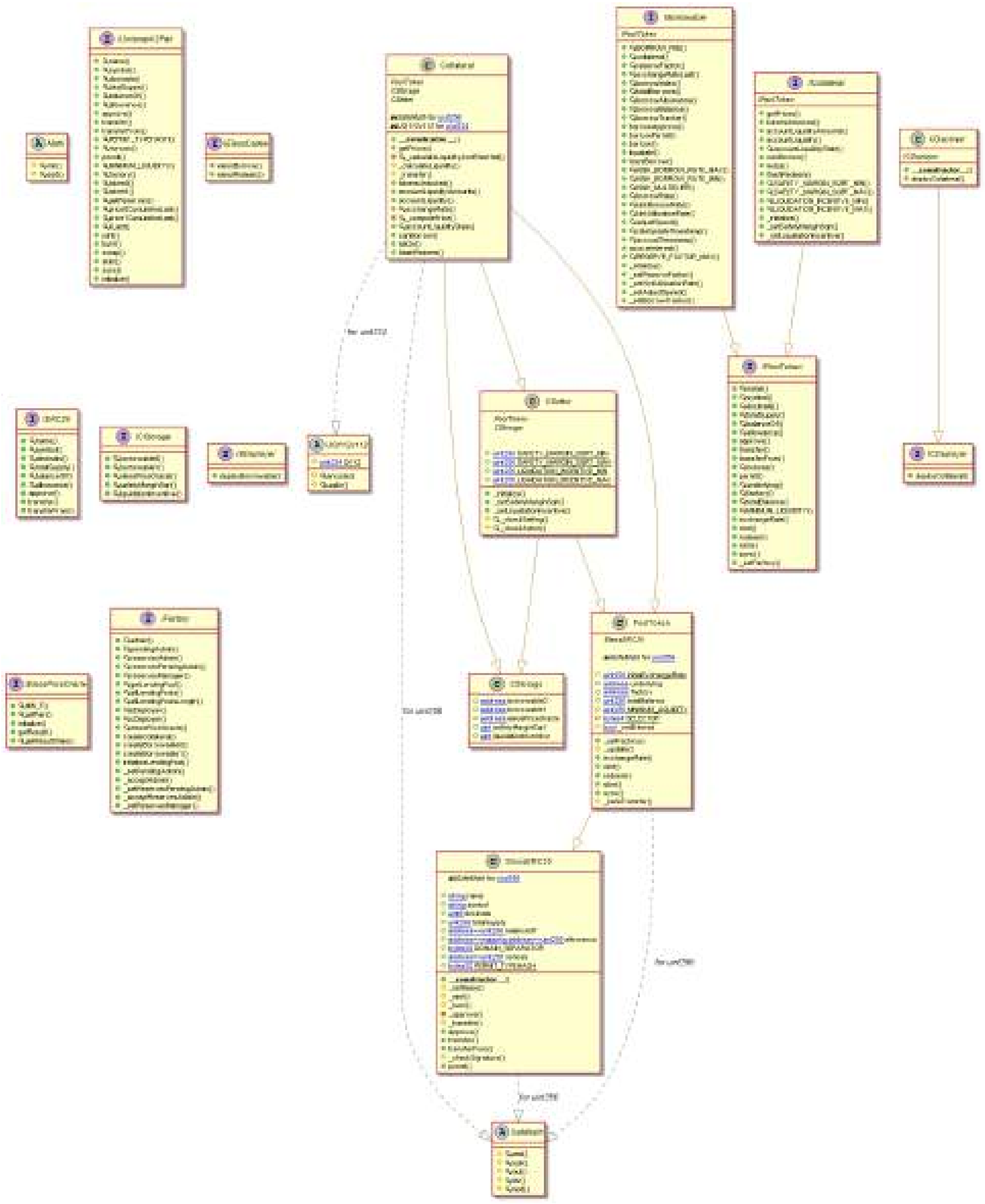
BStorage Diagram



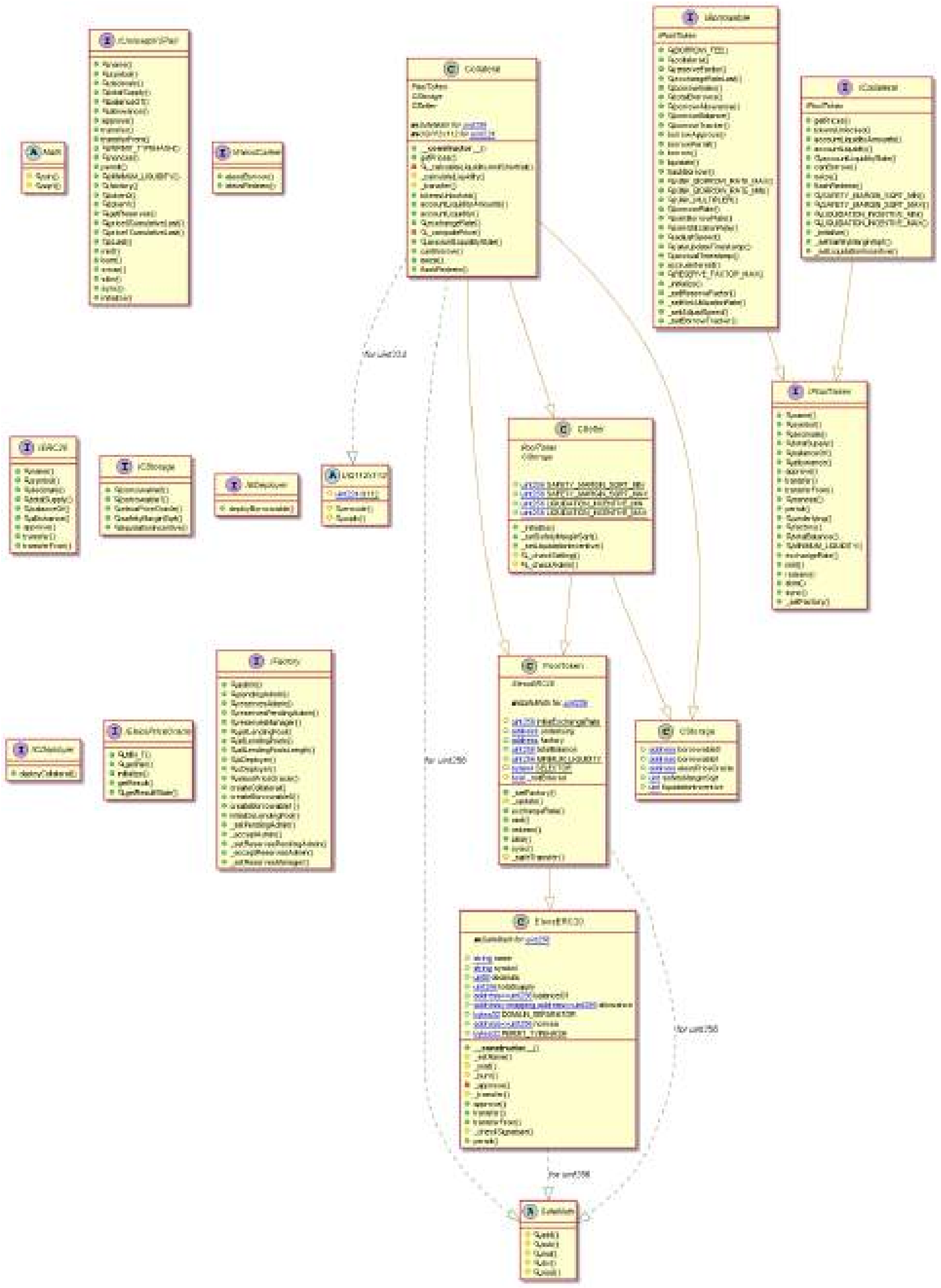
CStorage Diagram



CDeployer Diagram

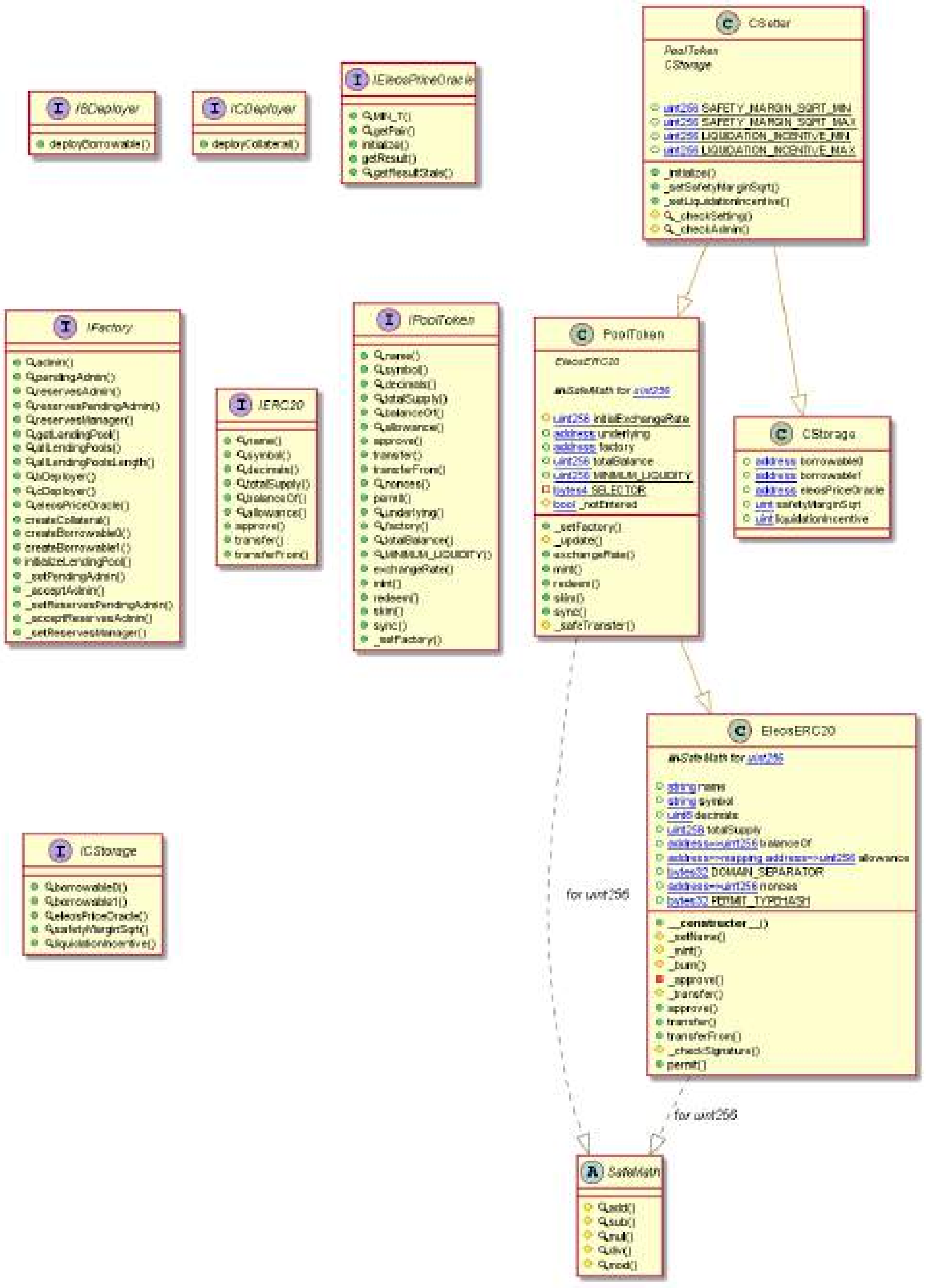


Collateral Diagram

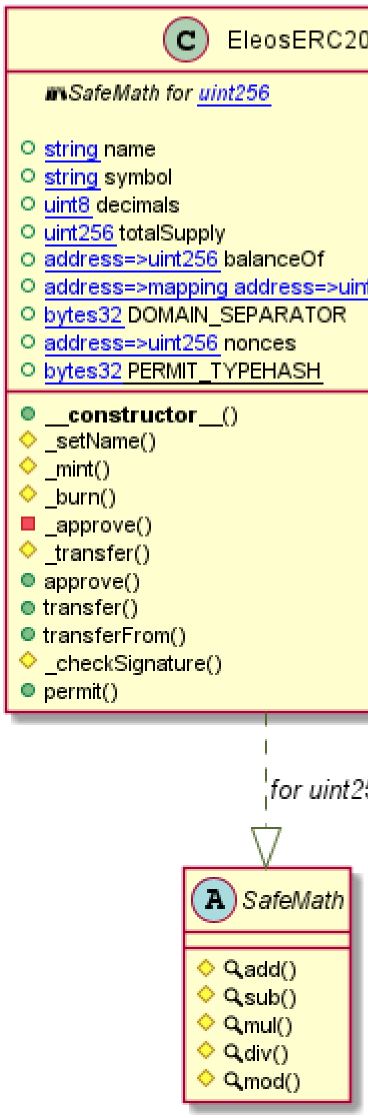


Email: audit@EtherAuthority.io

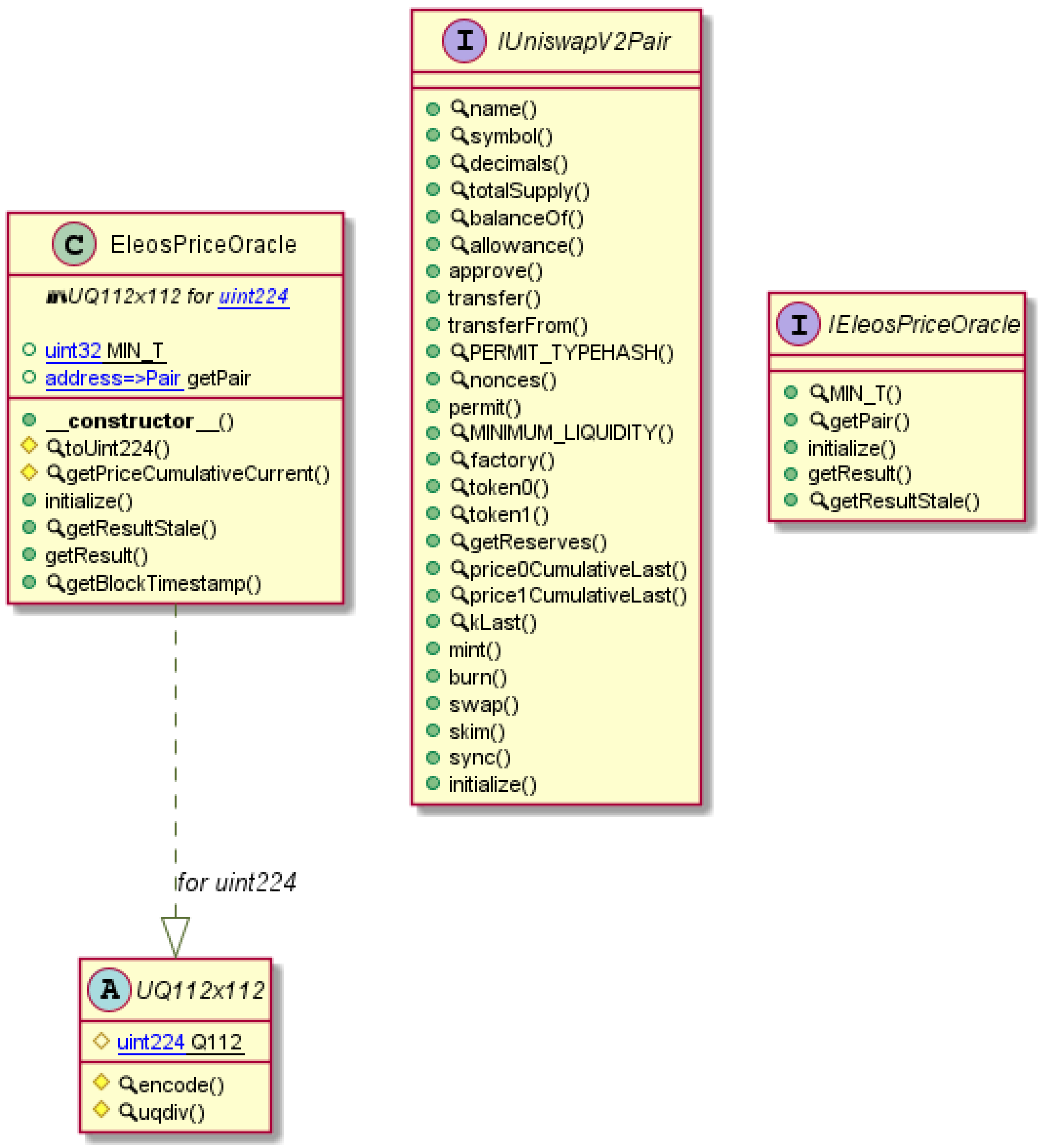
CSetter Diagram



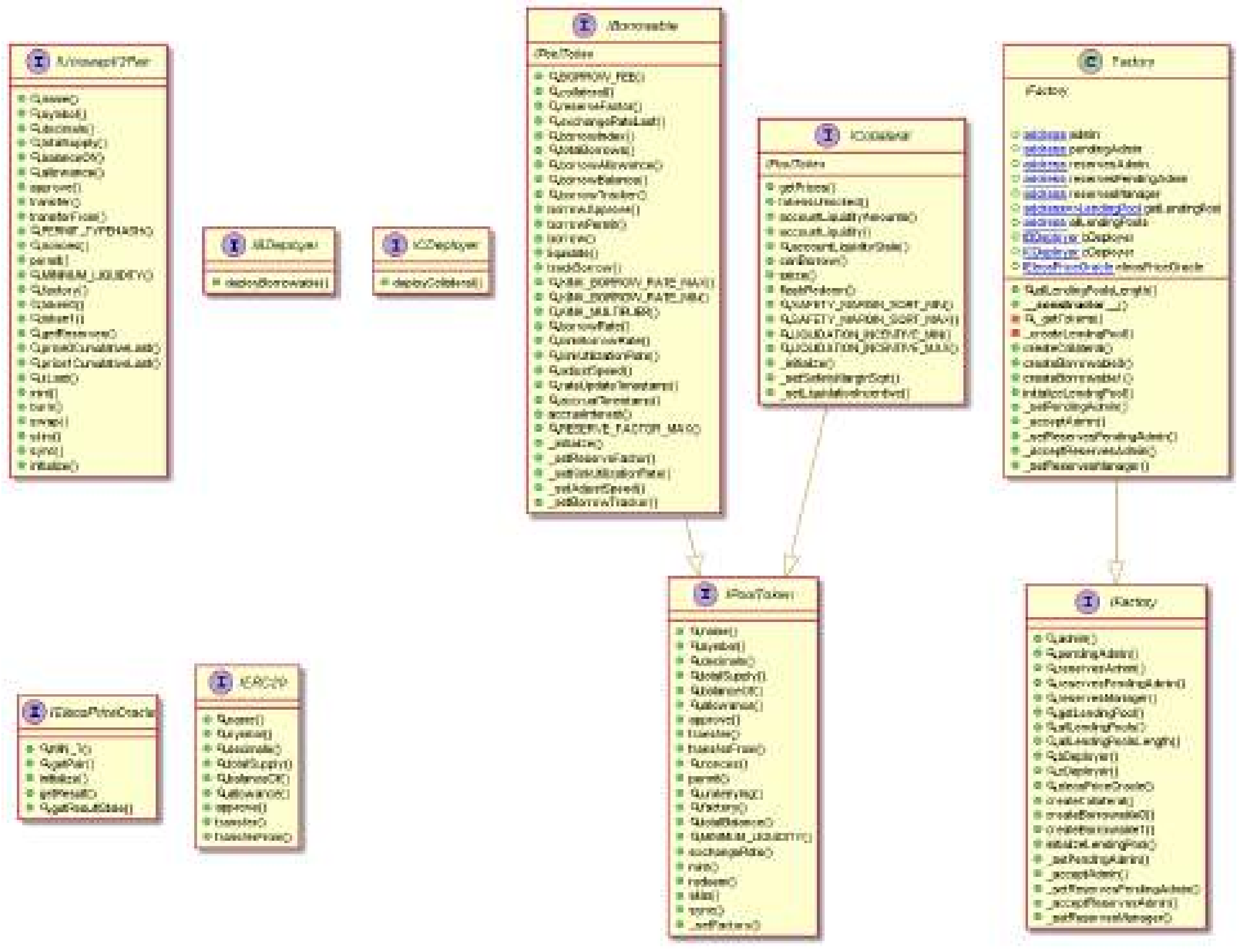
EleosERC20 Diagram



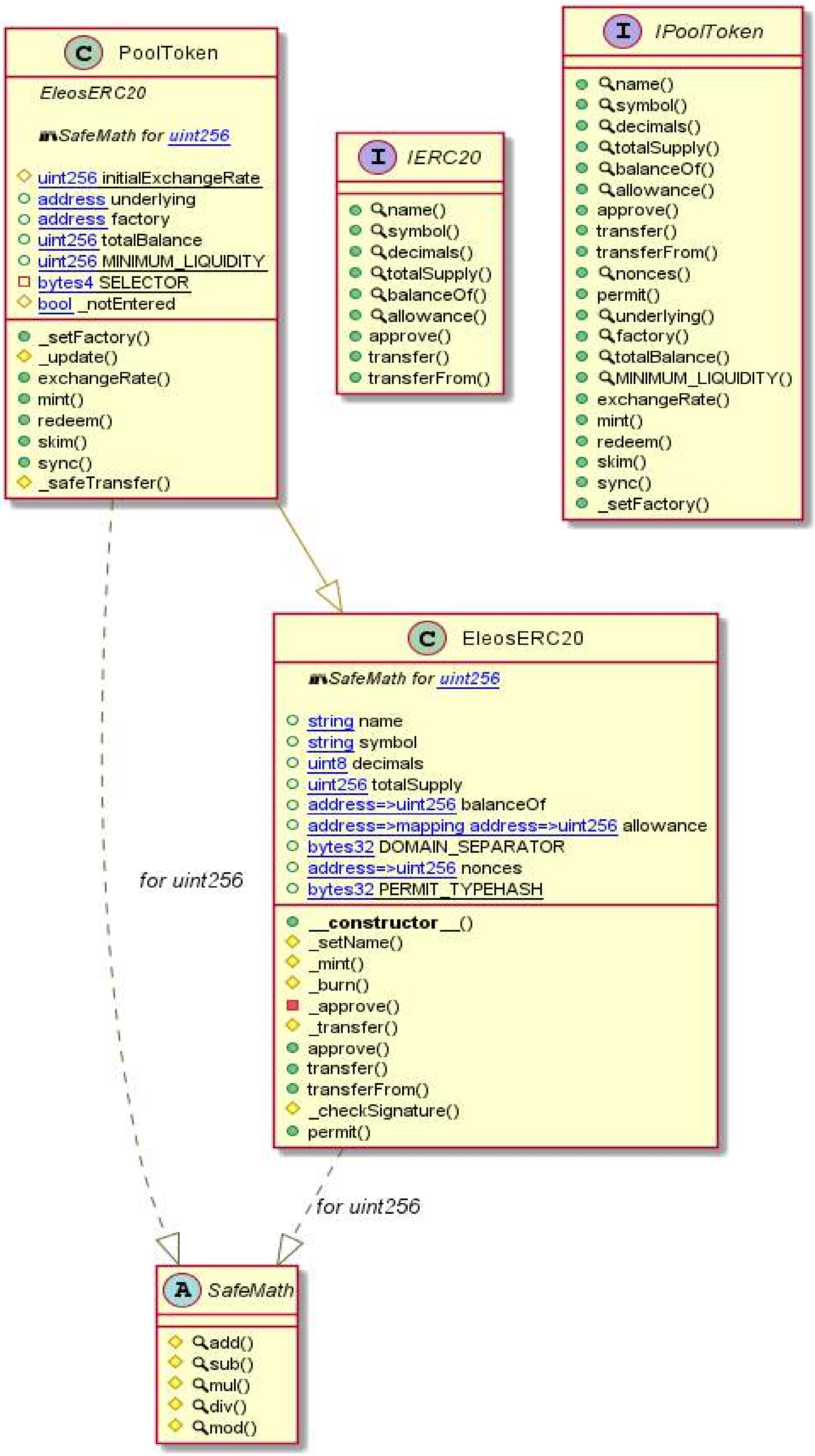
EleosPriceOracle Diagram



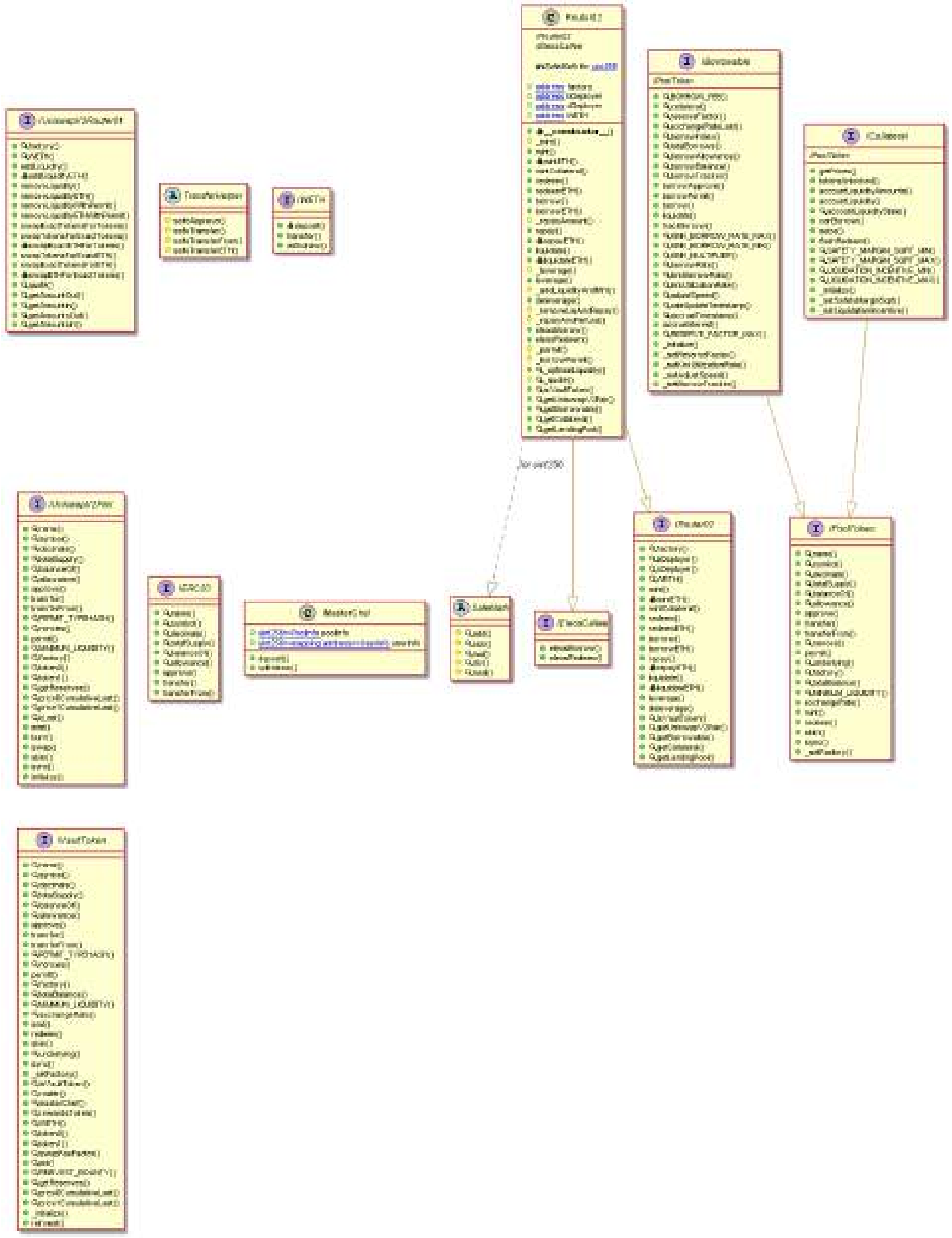
Factory Diagram



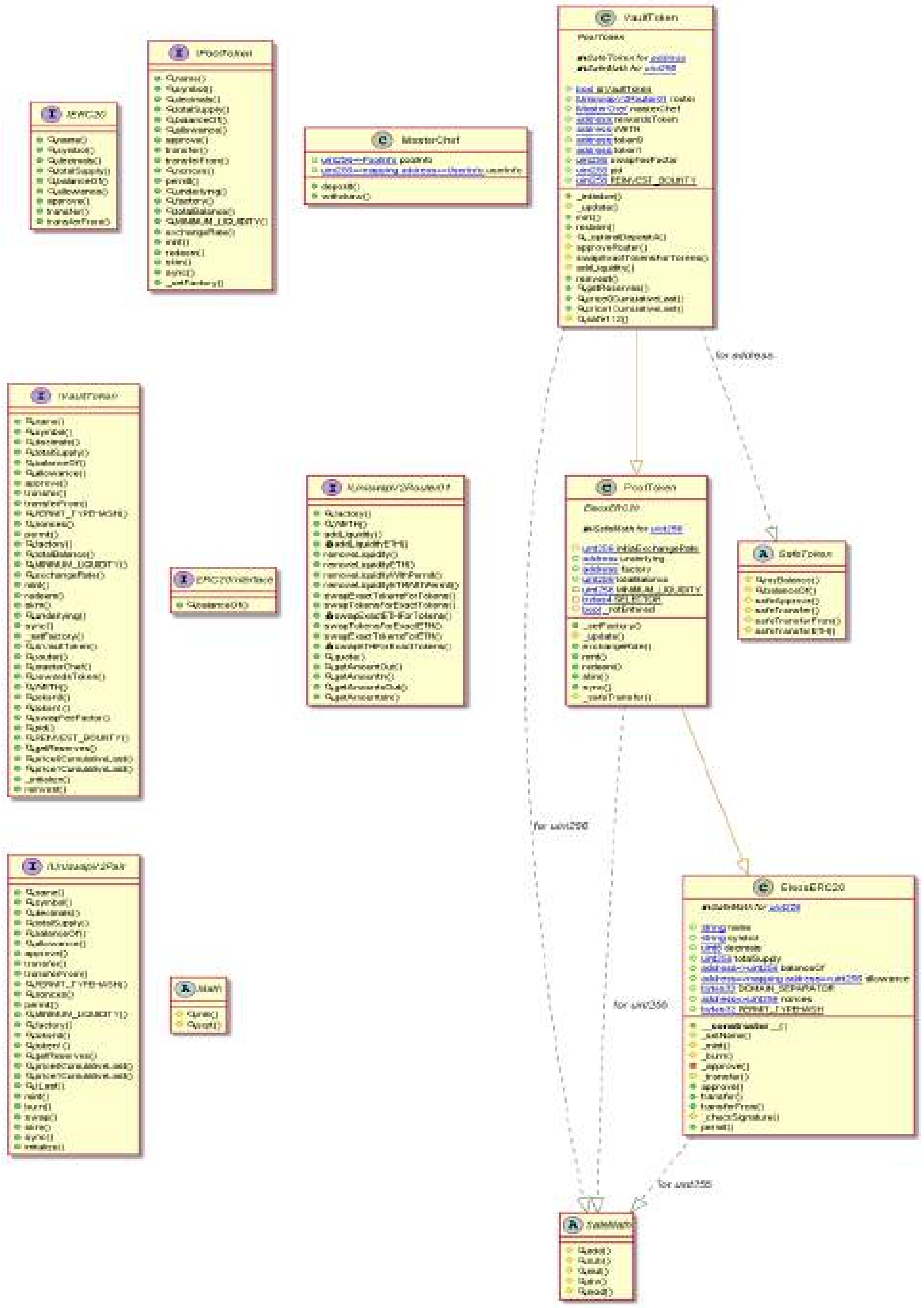
PoolToken Diagram



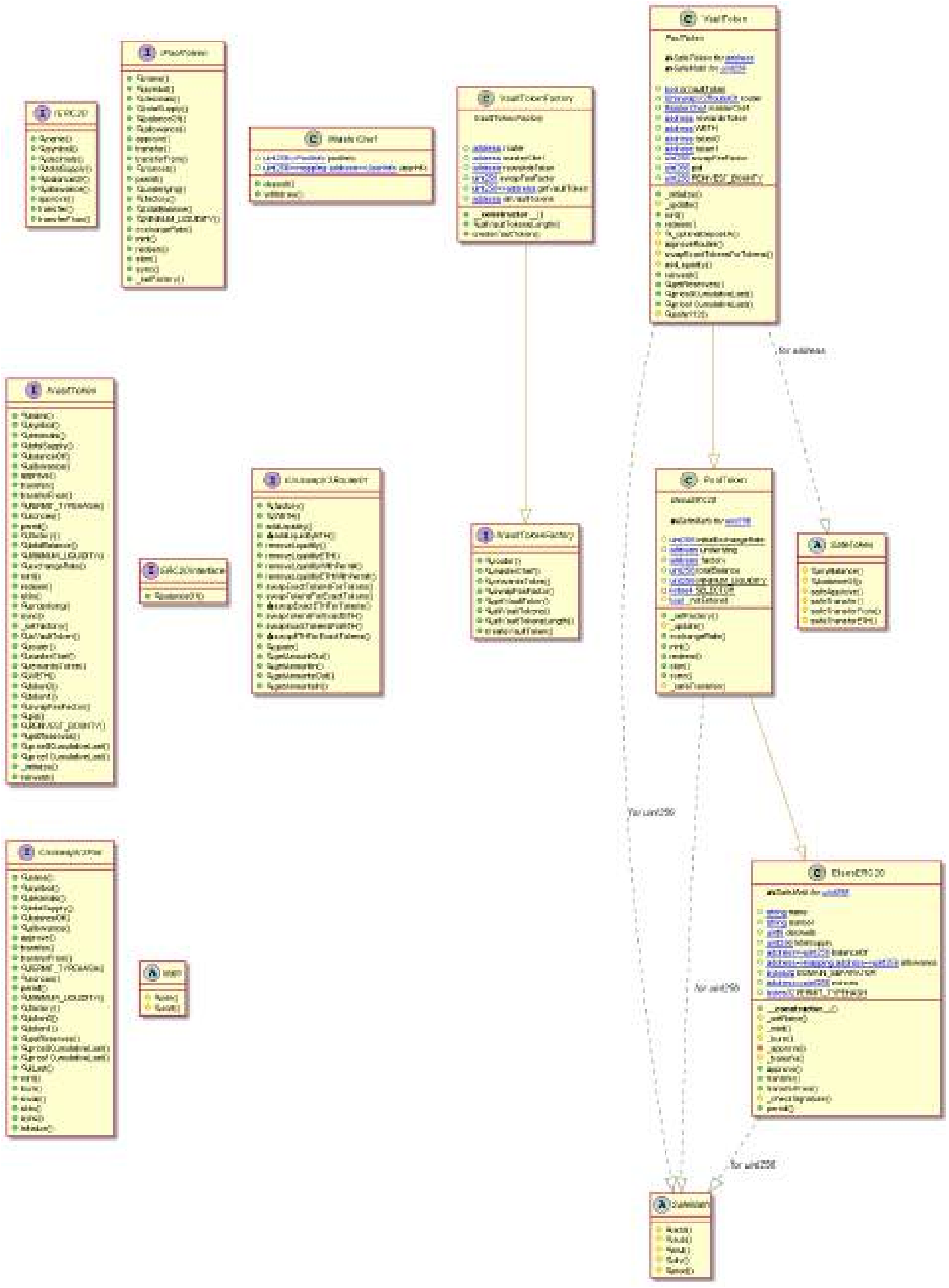
Router02 Diagram



VaultToken Diagram

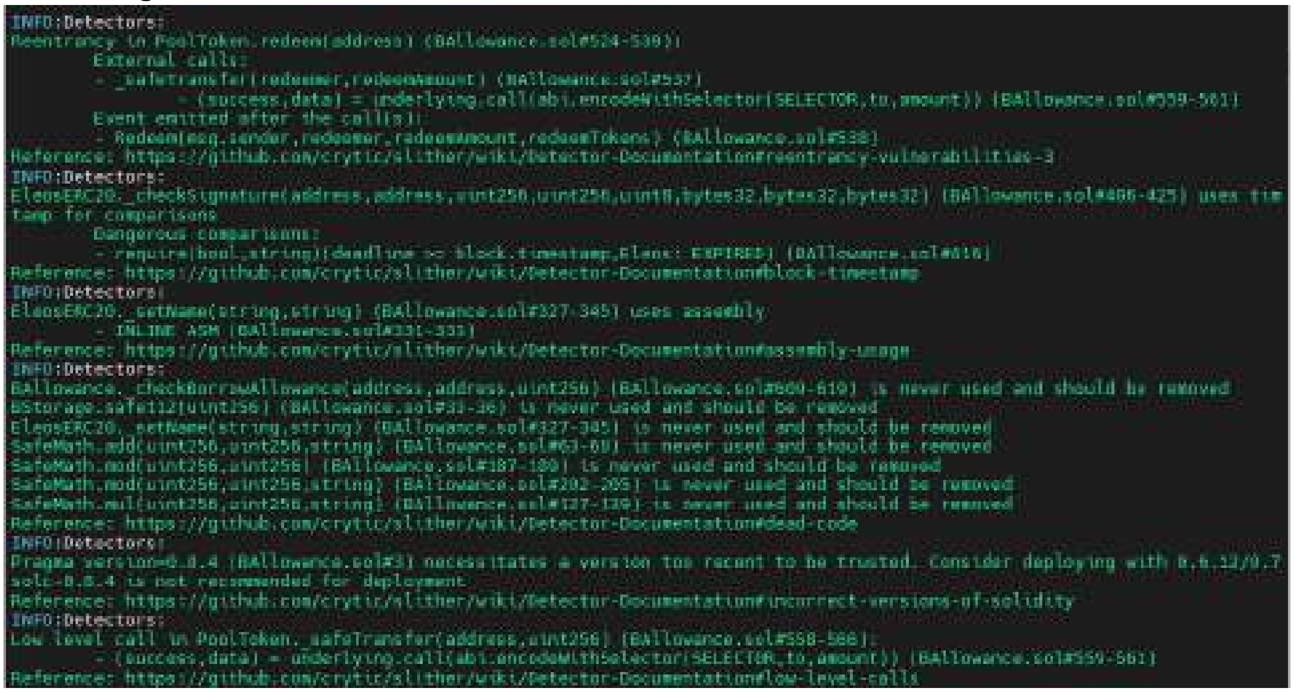


VaultTokenFactory Diagram

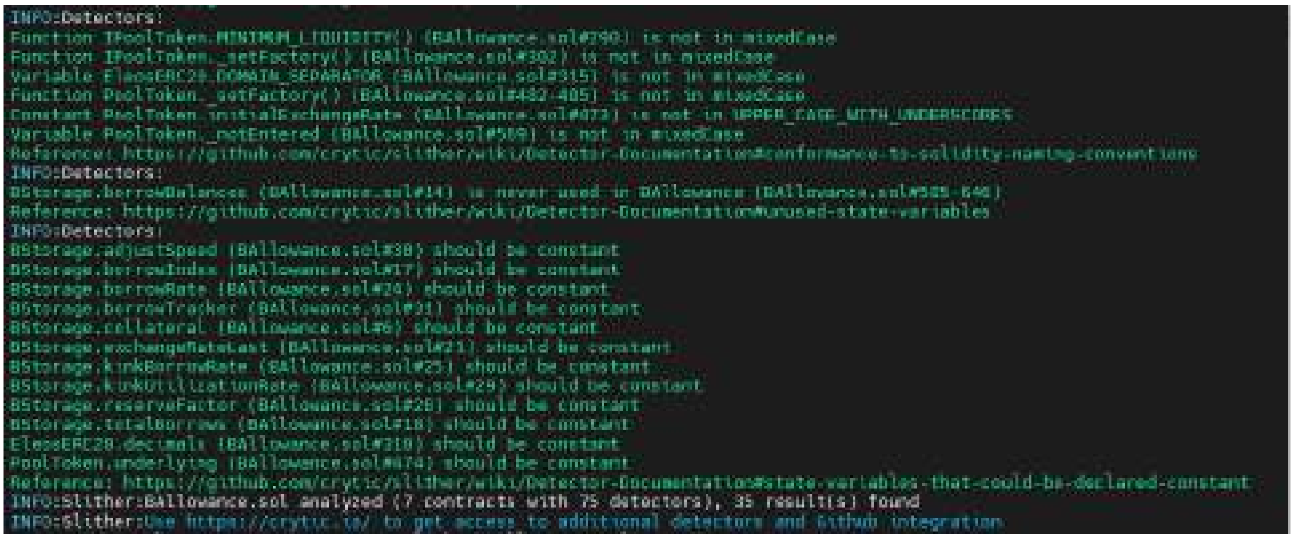


Slither Results Log

Slither log >> BAllowance.sol

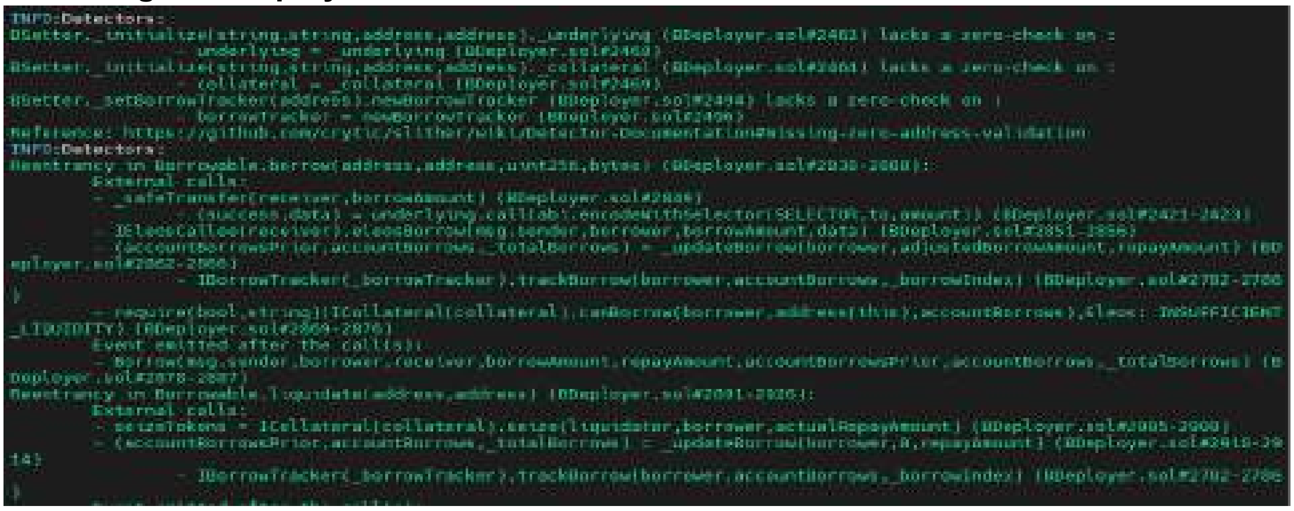


```
INFO:Detectors:
Reentrancy in PoolToken.redeem(address) (BAllowance.sol#524-539):
        External calls:
        - _safeTransfer(redeemer,redeemAmount) (BAllowance.sol#537)
                - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (BAllowance.sol#559-561)
        Event emitted after the call(s):
        - Redeem(msg.sender,redeemer,redeemAmount,redeemTokens) (BAllowance.sol#538)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ElaosERC20._checkSignature(address,address,uint256,uint256,uint8,bytes32,bytes32,bytes32) (BAllowance.sol#406-425) uses tim
tamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Elaos: EXPIRED) (BAllowance.sol#416)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
ElaosERC20._setName(string,string) (BAllowance.sol#327-345) uses assembly
        - INLINE ASM (BAllowance.sol#331-333)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
BAllowance._checkBorrowAllowance(address,address,uint256) (BAllowance.sol#609-619) is never used and should be removed
BStorage.safe112(uint256) (BAllowance.sol#33-36) is never used and should be removed
ElaosERC20._setName(string,string) (BAllowance.sol#327-345) is never used and should be removed
SafeMath.add(uint256,uint256,string) (BAllowance.sol#63-69) is never used and should be removed
SafeMath.mod(uint256,uint256) (BAllowance.sol#187-189) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (BAllowance.sol#202-205) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (BAllowance.sol#127-139) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.4 (BAllowance.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in PoolToken._safeTransfer(address,uint256) (BAllowance.sol#558-568):
        - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (BAllowance.sol#559-561)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

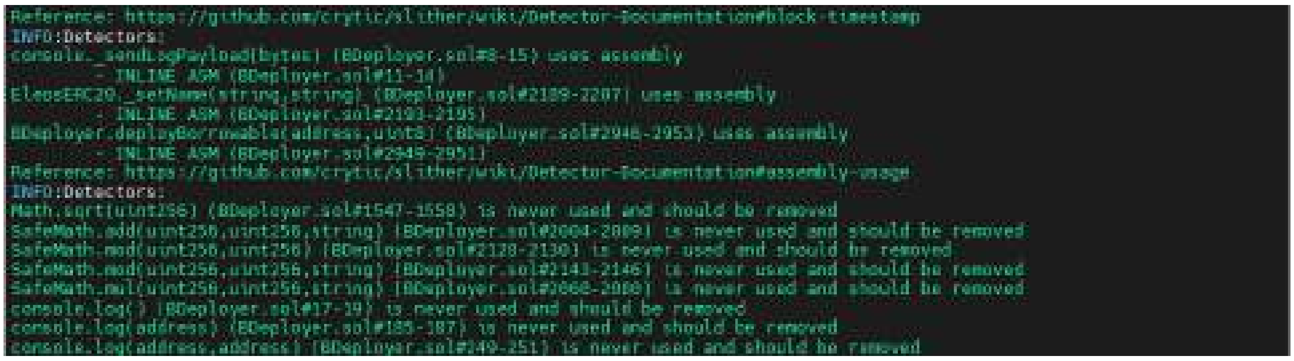


```
INFO:Detectors:
Function IPoolToken.MINIMUM_LIQUIDITY() (BAllowance.sol#290) is not in mixedCase
Function IPoolToken._setFactory() (BAllowance.sol#302) is not in mixedCase
Variable ElaosERC20.DOMAIN_SEPARATOR (BAllowance.sol#315) is not in mixedCase
Function PoolToken._setFactory() (BAllowance.sol#482-485) is not in mixedCase
Constant PoolToken.initialExchangeRate (BAllowance.sol#472) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PoolToken._notEntered (BAllowance.sol#569) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
BStorage.borrowBalances (BAllowance.sol#14) is never used in BAllowance (BAllowance.sol#585-646)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
BStorage.adjustSpeed (BAllowance.sol#30) should be constant
BStorage.borrowIndex (BAllowance.sol#17) should be constant
BStorage.borrowRate (BAllowance.sol#24) should be constant
BStorage.borrowTracker (BAllowance.sol#31) should be constant
BStorage.collateral (BAllowance.sol#6) should be constant
BStorage.exchangeRateLast (BAllowance.sol#21) should be constant
BStorage.kinkBorrowRate (BAllowance.sol#25) should be constant
BStorage.kinkUtilizationRate (BAllowance.sol#29) should be constant
BStorage.reserveFactor (BAllowance.sol#28) should be constant
BStorage.totalBorrows (BAllowance.sol#18) should be constant
ElaosERC20.decimals (BAllowance.sol#310) should be constant
PoolToken.underlying (BAllowance.sol#474) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:BAllowance.sol analyzed (7 contracts with 75 detectors), 35 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

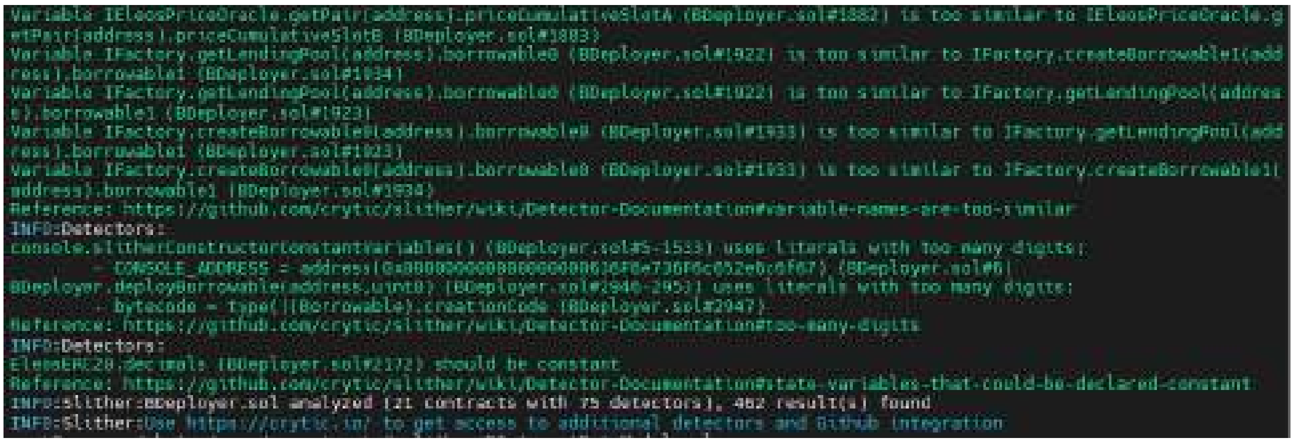
Slither log >> BDeployer.sol



```
INFO:Detectors:
BSetter._initialize(string,string,address,address)._underlying (BDeployer.sol#2463) lacks a zero-check on :
                - underlying = _underlying (BDeployer.sol#2468)
BSetter._initialize(string,string,address,address)._collateral (BDeployer.sol#2464) lacks a zero-check on :
                - collateral = _collateral (BDeployer.sol#2469)
BSetter._setBorrowTracker(address).newBorrowTracker (BDeployer.sol#2494) lacks a zero-check on :
                - borrowTracker = newBorrowTracker (BDeployer.sol#2496)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Borrowable.borrow(address,address,uint256,bytes) (BDeployer.sol#2838-2888):
        External calls:
        - _safeTransfer(receiver,borrowAmount) (BDeployer.sol#2866)
                - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (BDeployer.sol#2421-2423)
        - IElaosCallee(receiver).elaosBorrow(msg.sender,borrower,borrowAmount,data) (BDeployer.sol#2851-2856)
        - (accountBorrowsPrior,accountBorrows,_totalBorrows) = _updateBorrow(borrower,adjustedBorrowAmount,repayAmount) (BD
eployer.sol#2862-2866)
                - IBorrowTracker(_borrowTracker).trackBorrow(borrower,accountBorrows,_borrowIndex) (BDeployer.sol#2782-2786
)
        - require(bool,string)(ICollateral(collateral).canBorrow(borrower,address(this),accountBorrows),Elaos: INSUFFICIENT
_LIQUIDITY) (BDeployer.sol#2869-2876)
        Event emitted after the call(s):
        - Borrow(msg.sender,borrower,receiver,borrowAmount,repayAmount,accountBorrowsPrior,accountBorrows,_totalBorrows) (B
Deployer.sol#2878-2887)
Reentrancy in Borrowable.liquidate(address,address) (BDeployer.sol#2891-2926):
        External calls:
        - seizeTokens = ICollateral(collateral).seize(liquidator,borrower,actualRepayAmount) (BDeployer.sol#2905-2909)
        - (accountBorrowsPrior,accountBorrows,_totalBorrows) = _updateBorrow(borrower,0,repayAmount) (BDeployer.sol#2910-29
14)
                - IBorrowTracker(_borrowTracker).trackBorrow(borrower,accountBorrows,_borrowIndex) (BDeployer.sol#2782-2786
)
```

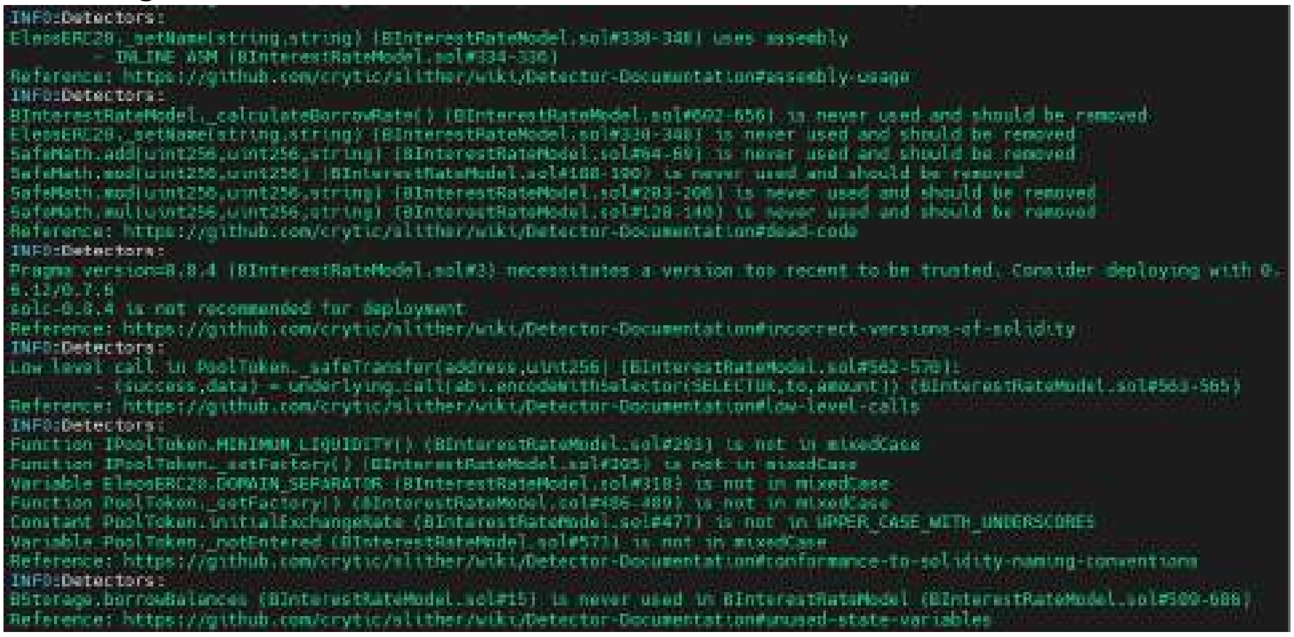


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
console._sendLogPayload(bytes) (BDeployer.sol#8-15) uses assembly
        - INLINE ASM (BDeployer.sol#11-14)
EleosERC20._setName(string,string) (BDeployer.sol#2189-2207) uses assembly
        - INLINE ASM (BDeployer.sol#2193-2195)
BDeployer.deployBorrowable(address,uint8) (BDeployer.sol#2946-2953) uses assembly
        - INLINE ASM (BDeployer.sol#2949-2951)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Math.sqrt(uint256) (BDeployer.sol#1547-1558) is never used and should be removed
SafeMath.add(uint256,uint256,string) (BDeployer.sol#2004-2009) is never used and should be removed
SafeMath.mod(uint256,uint256) (BDeployer.sol#2128-2130) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (BDeployer.sol#2143-2146) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (BDeployer.sol#2068-2080) is never used and should be removed
console.log() (BDeployer.sol#17-19) is never used and should be removed
console.log(address) (BDeployer.sol#185-187) is never used and should be removed
console.log(address,address) (BDeployer.sol#249-251) is never used and should be removed
```

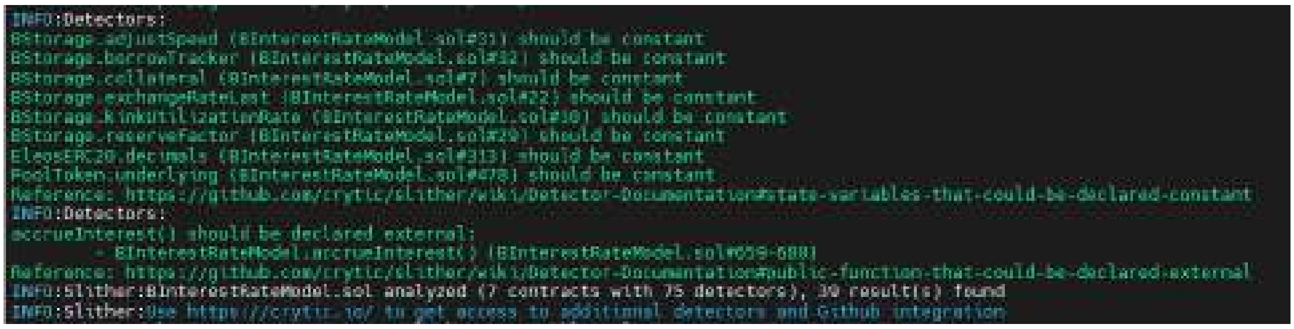


```
Variable IEleosPriceOracle.getPair(address).priceCumulativeSlotA (BDeployer.sol#1882) is too similar to IEleosPriceOracle.g
etPair(address).priceCumulativeSlotB (BDeployer.sol#1883)
Variable IFactory.getLendingPool(address).borrowable0 (BDeployer.sol#1922) is too similar to IFactory.createBorrowable1(add
ress).borrowable1 (BDeployer.sol#1934)
Variable IFactory.getLendingPool(address).borrowable0 (BDeployer.sol#1922) is too similar to IFactory.getLendingPool(addres
s).borrowable1 (BDeployer.sol#1923)
Variable IFactory.createBorrowable0(address).borrowable0 (BDeployer.sol#1933) is too similar to IFactory.getLendingPool(add
ress).borrowable1 (BDeployer.sol#1923)
Variable IFactory.createBorrowable0(address).borrowable0 (BDeployer.sol#1933) is too similar to IFactory.createBorrowable1(
address).borrowable1 (BDeployer.sol#1934)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
console.slitherConstructorConstantVariables() (BDeployer.sol#5-1533) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (BDeployer.sol#6)
BDeployer.deployBorrowable(address,uint8) (BDeployer.sol#2946-2953) uses literals with too many digits:
        - bytecode = type()(Borrowable).creationCode (BDeployer.sol#2947)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
EleosERC20.decimals (BDeployer.sol#2172) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:BDeployer.sol analyzed (21 contracts with 75 detectors), 482 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> BInterestRateModel.sol

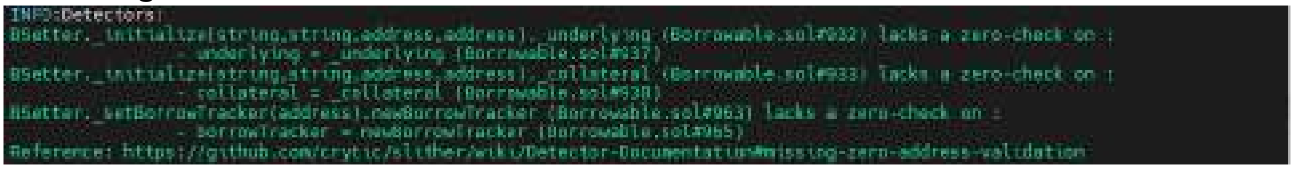


```
INFO:Detectors:
EleosERC20._setName(string,string) (BInterestRateModel.sol#330-348) uses assembly
        - INLINE ASM (BInterestRateModel.sol#334-336)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
BInterestRateModel._calculateBorrowRate() (BInterestRateModel.sol#602-656) is never used and should be removed
EleosERC20._setName(string,string) (BInterestRateModel.sol#330-348) is never used and should be removed
SafeMath.add(uint256,uint256,string) (BInterestRateModel.sol#64-69) is never used and should be removed
SafeMath.mod(uint256,uint256) (BInterestRateModel.sol#188-190) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (BInterestRateModel.sol#203-206) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (BInterestRateModel.sol#128-140) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.4 (BInterestRateModel.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.
6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in PoolToken._safeTransfer(address,uint256) (BInterestRateModel.sol#562-570):
        - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (BInterestRateModel.sol#563-565)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IPoolToken.MINIMUM_LIQUIDITY() (BInterestRateModel.sol#293) is not in mixedCase
Function IPoolToken._setFactory() (BInterestRateModel.sol#305) is not in mixedCase
Variable EleosERC20.DOMAIN_SEPARATOR (BInterestRateModel.sol#318) is not in mixedCase
Function PoolToken._setFactory() (BInterestRateModel.sol#486-489) is not in mixedCase
Constant PoolToken.initialExchangeRate (BInterestRateModel.sol#477) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PoolToken._notEntered (BInterestRateModel.sol#571) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
BStorage.borrowBalances (BInterestRateModel.sol#15) is never used in BInterestRateModel (BInterestRateModel.sol#589-688)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
```

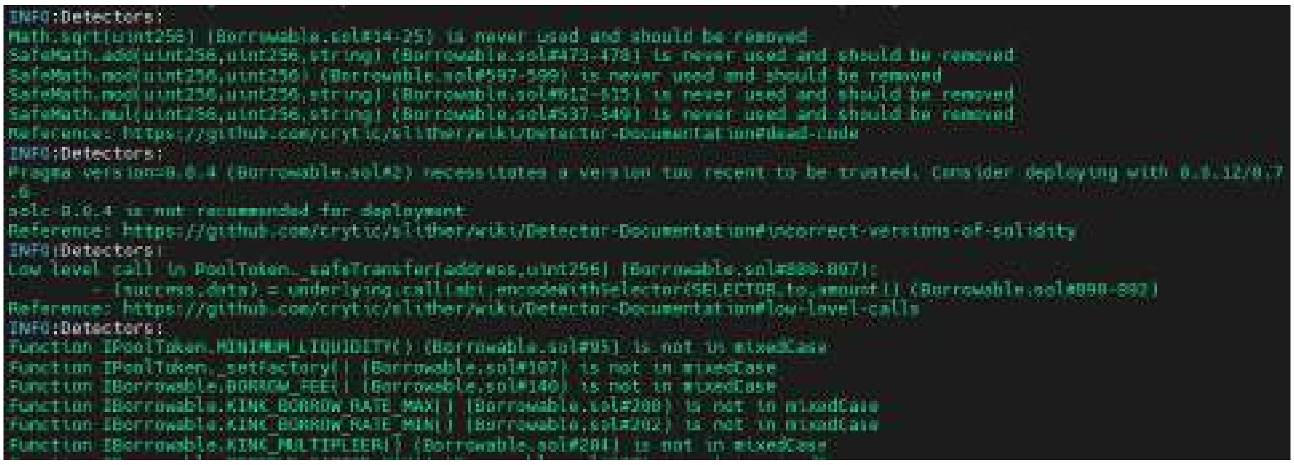


```
INFO:Detectors:
BStorage.adjustSpeed (BInterestRateModel.sol#31) should be constant
BStorage.borrowTracker (BInterestRateModel.sol#32) should be constant
BStorage.collateral (BInterestRateModel.sol#7) should be constant
BStorage.exchangeRateLast (BInterestRateModel.sol#22) should be constant
BStorage.kinkUtilizationRate (BInterestRateModel.sol#30) should be constant
BStorage.reserveFactor (BInterestRateModel.sol#29) should be constant
EleosERC20.decimals (BInterestRateModel.sol#313) should be constant
PoolToken.underlying (BInterestRateModel.sol#478) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
accrueInterest() should be declared external:
        - BInterestRateModel.accrueInterest() (BInterestRateModel.sol#659-688)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:BInterestRateModel.sol analyzed (7 contracts with 75 detectors), 30 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

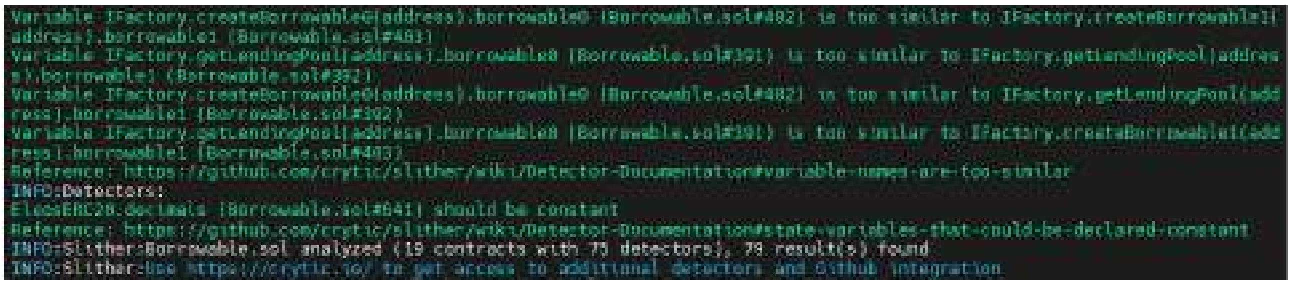
Slither log >> Borrowable.sol



```
INFO:Detectors:
BSetter._initialize(string,string,address,address)._underlying (Borrowable.sol#932) lacks a zero-check on :
                - underlying = _underlying (Borrowable.sol#937)
BSetter._initialize(string,string,address,address)._collateral (Borrowable.sol#933) lacks a zero-check on :
                - collateral = _collateral (Borrowable.sol#938)
BSetter._setBorrowTracker(address).newBorrowTracker (Borrowable.sol#963) lacks a zero-check on :
                - borrowTracker = newBorrowTracker (Borrowable.sol#965)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

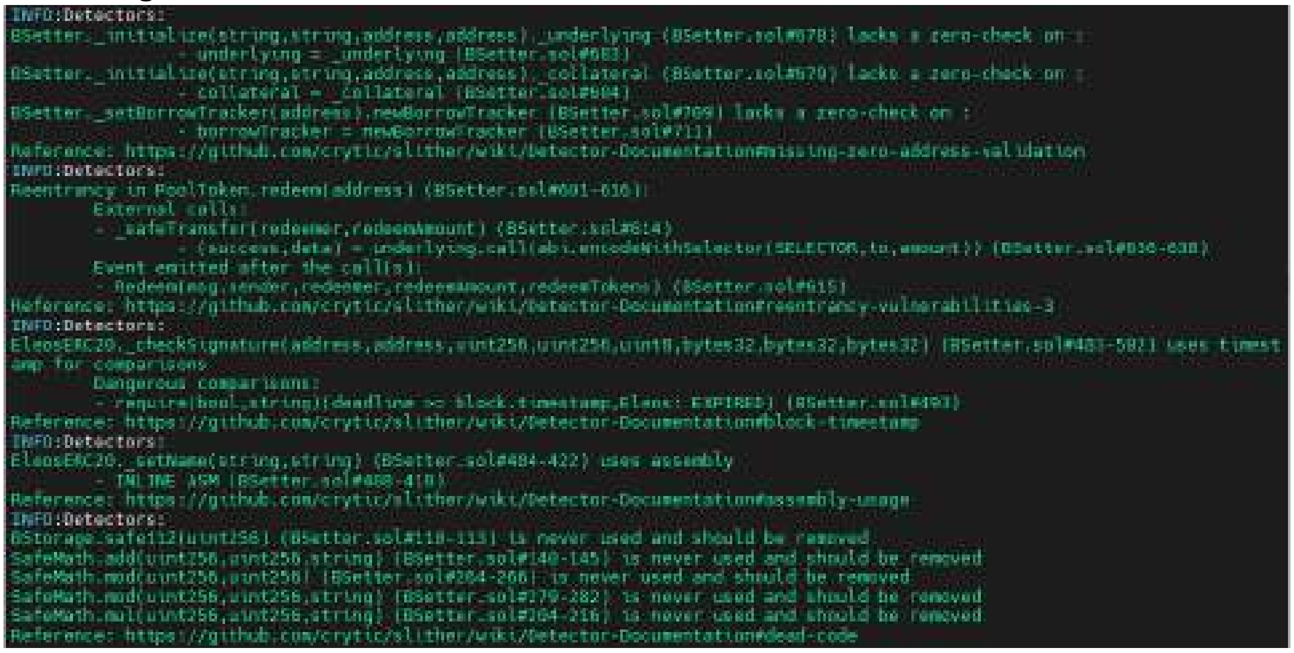


```
INFO:Detectors:
Math.sqrt(uint256) (Borrowable.sol#14-25) is never used and should be removed
SafeMath.add(uint256,uint256,string) (Borrowable.sol#473-478) is never used and should be removed
SafeMath.mod(uint256,uint256) (Borrowable.sol#597-599) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Borrowable.sol#612-615) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (Borrowable.sol#537-549) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.4 (Borrowable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in PoolToken._safeTransfer(address,uint256) (Borrowable.sol#889-897):
        - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (Borrowable.sol#890-892)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IPoolToken.MINIMUM_LIQUIDITY() (Borrowable.sol#95) is not in mixedCase
Function IPoolToken._setFactory() (Borrowable.sol#107) is not in mixedCase
Function IBorrowable.BORROW_FEE() (Borrowable.sol#140) is not in mixedCase
Function IBorrowable.KINK_BORROW_RATE_MAX() (Borrowable.sol#200) is not in mixedCase
Function IBorrowable.KINK_BORROW_RATE_MIN() (Borrowable.sol#202) is not in mixedCase
Function IBorrowable.KINK_MULTIPLIER() (Borrowable.sol#204) is not in mixedCase
```

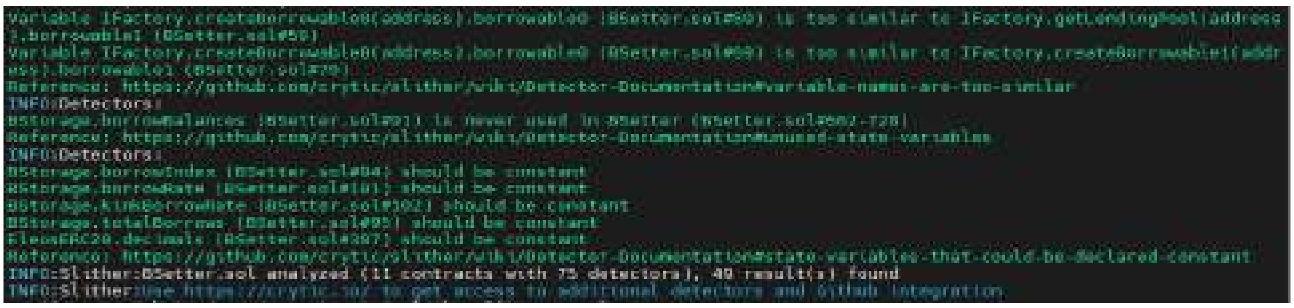


```
Variable IFactory.createBorrowable0(address).borrowable0 (Borrowable.sol#482) is too similar to IFactory.createBorrowable1(
address).borrowable1 (Borrowable.sol#483)
Variable IFactory.getLendingPool(address).borrowable0 (Borrowable.sol#391) is too similar to IFactory.getLendingPool(addres
s).borrowable1 (Borrowable.sol#392)
Variable IFactory.createBorrowable0(address).borrowable0 (Borrowable.sol#482) is too similar to IFactory.getLendingPool(add
ress).borrowable1 (Borrowable.sol#392)
Variable IFactory.getLendingPool(address).borrowable0 (Borrowable.sol#391) is too similar to IFactory.createBorrowable1(add
ress).borrowable1 (Borrowable.sol#483)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
EleosERC20.decimals (Borrowable.sol#641) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:Borrowable.sol analyzed (19 contracts with 75 detectors), 79 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> BSetter.sol

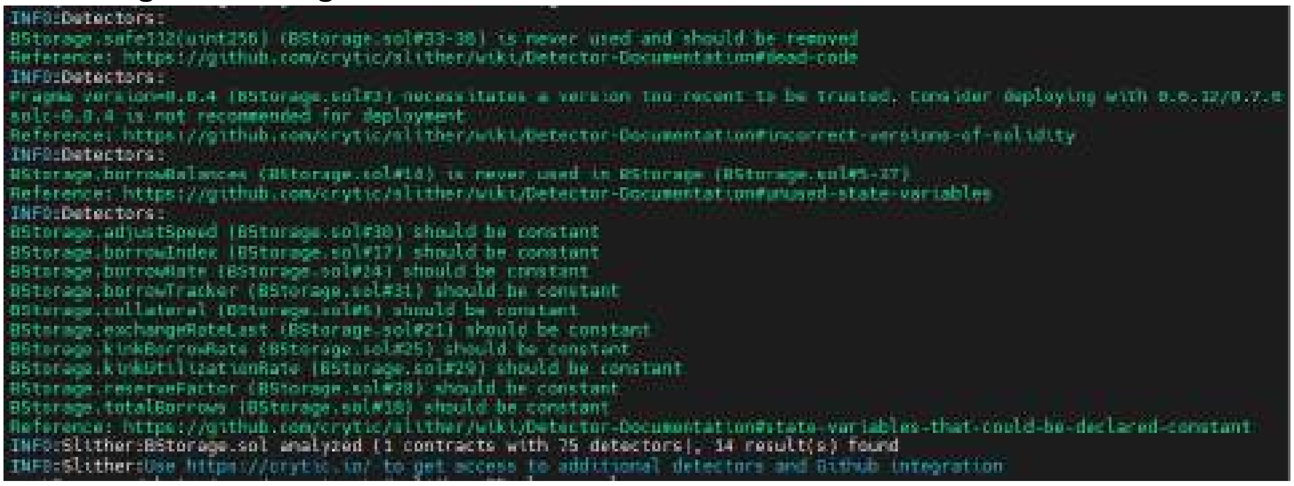


```
INFO:Detectors:
BSetter._initialize(string,string,address,address)._underlying (BSetter.sol#678) lacks a zero-check on :
                - underlying = _underlying (BSetter.sol#683)
BSetter._initialize(string,string,address,address)._collateral (BSetter.sol#679) lacks a zero-check on :
                - collateral = _collateral (BSetter.sol#684)
BSetter._setBorrowTracker(address).newBorrowTracker (BSetter.sol#709) lacks a zero-check on :
                - borrowTracker = newBorrowTracker (BSetter.sol#711)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in PoolToken.redeem(address) (BSetter.sol#601-616):
        External calls:
        - _safeTransfer(redeemer,redeemAmount) (BSetter.sol#614)
                - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (BSetter.sol#636-638)
        Event emitted after the call(s):
        - Redeem(msg.sender,redeemer,redeemAmount,redeemTokens) (BSetter.sol#615)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
EleosERC20._checkSignature(address,address,uint256,uint256,uint8,bytes32,bytes32,bytes32) (BSetter.sol#481-502) uses timest
amp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Eleos: EXPIRED) (BSetter.sol#493)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
EleosERC20._setName(string,string) (BSetter.sol#404-422) uses assembly
        - INLINE ASM (BSetter.sol#408-418)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
BStorage.safe112(uint256) (BSetter.sol#110-113) is never used and should be removed
SafeMath.add(uint256,uint256,string) (BSetter.sol#140-145) is never used and should be removed
SafeMath.mod(uint256,uint256) (BSetter.sol#264-266) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (BSetter.sol#279-282) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (BSetter.sol#204-216) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```



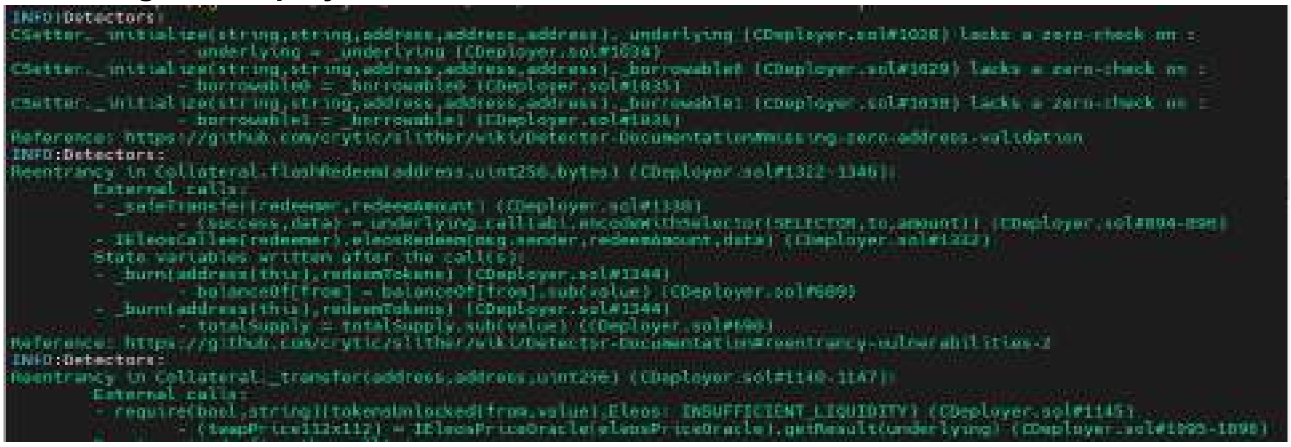
```
Variable IFactory.createBorrowable0(address).borrowable0 (BSetter.sol#60) is too similar to IFactory.getLendingPool(address
).borrowable1 (BSetter.sol#59)
Variable IFactory.createBorrowable0(address).borrowable0 (BSetter.sol#69) is too similar to IFactory.createBorrowable1(addr
ess).borrowable1 (BSetter.sol#70)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
BStorage.borrowBalances (BSetter.sol#91) is never used in BSetter (BSetter.sol#667-728)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
BStorage.borrowIndex (BSetter.sol#94) should be constant
BStorage.borrowRate (BSetter.sol#101) should be constant
BStorage.kinkBorrowRate (BSetter.sol#102) should be constant
BStorage.totalBorrows (BSetter.sol#95) should be constant
EleosERC20.decimals (BSetter.sol#387) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:BSetter.sol analyzed (11 contracts with 75 detectors), 49 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> BStorage.sol

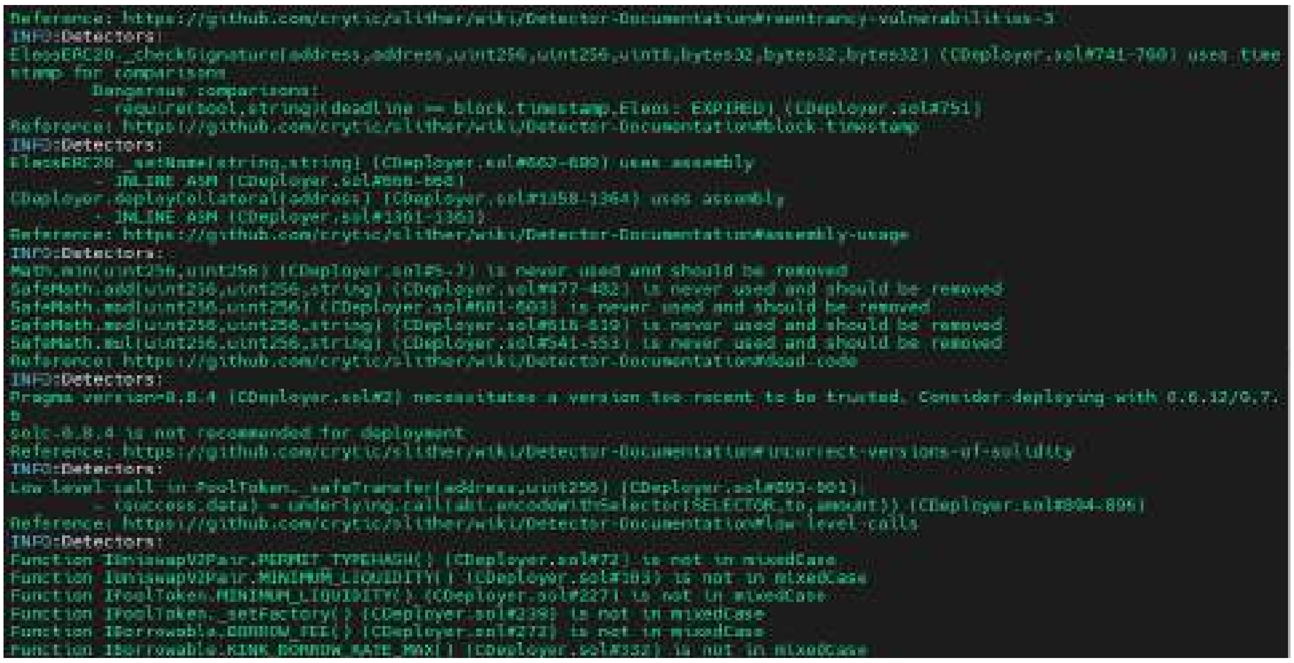


```
INFO:Detectors:
BStorage.safe112(uint256) (BStorage.sol#33-38) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.4 (BStorage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
BStorage.borrowBalances (BStorage.sol#16) is never used in BStorage (BStorage.sol#5-37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
BStorage.adjustSpeed (BStorage.sol#30) should be constant
BStorage.borrowIndex (BStorage.sol#17) should be constant
BStorage.borrowRate (BStorage.sol#24) should be constant
BStorage.borrowTracker (BStorage.sol#31) should be constant
BStorage.collateral (BStorage.sol#6) should be constant
BStorage.exchangeRateLast (BStorage.sol#21) should be constant
BStorage.kinkBorrowRate (BStorage.sol#25) should be constant
BStorage.kinkUtilizationRate (BStorage.sol#29) should be constant
BStorage.reserveFactor (BStorage.sol#28) should be constant
BStorage.totalBorrows (BStorage.sol#18) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:BStorage.sol analyzed (1 contracts with 75 detectors), 14 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

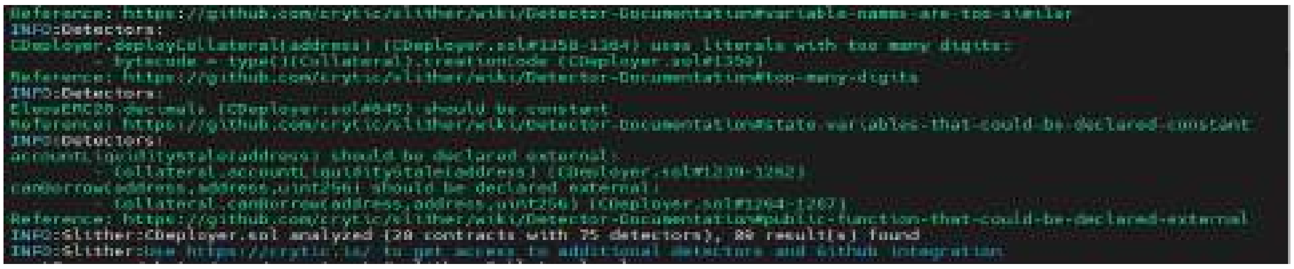
Slither log >> CDeployer.sol



```
INFO:Detectors:
CSetter._initialize(string,string,address,address,address)._underlying (CDeployer.sol#1028) lacks a zero-check on :
                - underlying = _underlying (CDeployer.sol#1034)
CSetter._initialize(string,string,address,address,address)._borrowable0 (CDeployer.sol#1029) lacks a zero-check on :
                - borrowable0 = _borrowable0 (CDeployer.sol#1035)
CSetter._initialize(string,string,address,address,address)._borrowable1 (CDeployer.sol#1030) lacks a zero-check on :
                - borrowable1 = _borrowable1 (CDeployer.sol#1036)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Collateral.flashRedeem(address,uint256,bytes) (CDeployer.sol#1322-1346):
        External calls:
        - _safeTransfer(redeemer,redeemAmount) (CDeployer.sol#1338)
                - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (CDeployer.sol#894-896)
        - IElocoCallee(redeemer).elocoRedeem(msg.sender,redeemAmount,data) (CDeployer.sol#1322)
        State variables written after the call(s):
        - _burn(address(this),redeemTokens) (CDeployer.sol#1344)
                - balanceOf[from] = balanceOf[from].sub(value) (CDeployer.sol#689)
        - _burn(address(this),redeemTokens) (CDeployer.sol#1344)
                - totalSupply = totalSupply.sub(value) (CDeployer.sol#690)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Collateral._transfer(address,address,uint256) (CDeployer.sol#1140-1147):
        External calls:
        - require(bool,string)(tokensUnlocked(from,value),Eleos: INSUFFICIENT_LIQUIDITY) (CDeployer.sol#1145)
                - (twapPrice112x112) = IElocoPriceOracle(elocoPriceOracle).getResult(underlying) (CDeployer.sol#1095-1098)
```

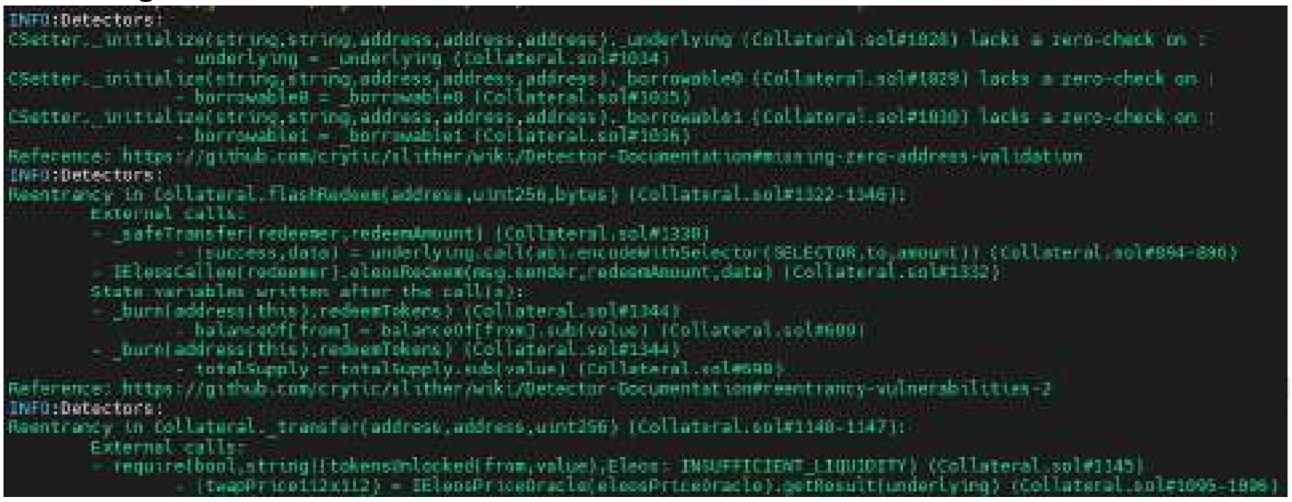


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
EleosERC20._checkSignature(address,address,uint256,uint256,uint8,bytes32,bytes32,bytes32) (CDeployer.sol#741-760) uses time
stamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Eleos: EXPIRED) (CDeployer.sol#751)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
EleosERC20._setName(string,string) (CDeployer.sol#662-680) uses assembly
        - INLINE ASM (CDeployer.sol#666-682)
CDeployer.deployCollateral(address) (CDeployer.sol#1358-1364) uses assembly
        - INLINE ASM (CDeployer.sol#1361-1363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Math.min(uint256,uint256) (CDeployer.sol#5-7) is never used and should be removed
SafeMath.add(uint256,uint256,string) (CDeployer.sol#477-482) is never used and should be removed
SafeMath.mod(uint256,uint256) (CDeployer.sol#601-603) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (CDeployer.sol#616-619) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (CDeployer.sol#541-553) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.4 (CDeployer.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in PoolToken._safeTransfer(address,uint256) (CDeployer.sol#893-901):
        - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (CDeployer.sol#894-896)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.PERMIT_TYPEHASH() (CDeployer.sol#72) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (CDeployer.sol#103) is not in mixedCase
Function IPoolToken.MINIMUM_LIQUIDITY() (CDeployer.sol#227) is not in mixedCase
Function IPoolToken._setFactory() (CDeployer.sol#238) is not in mixedCase
Function IBorrowable.BORROW_FEE() (CDeployer.sol#272) is not in mixedCase
Function IBorrowable.KINK_BORROW_RATE_MAX() (CDeployer.sol#332) is not in mixedCase
```

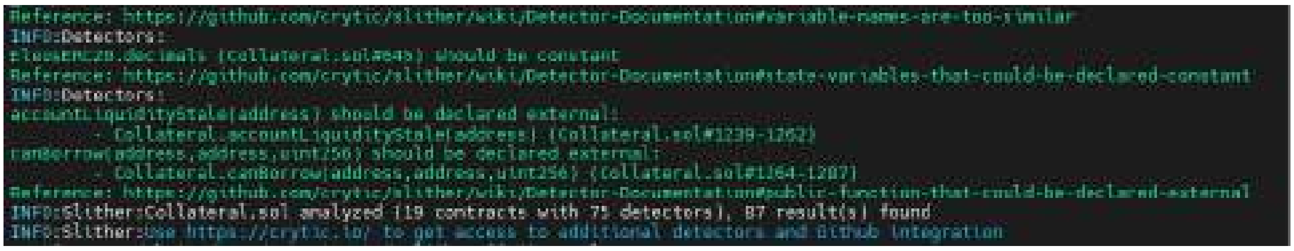


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
CDeployer.deployCollateral(address) (CDeployer.sol#1358-1364) uses literals with too many digits:
        - bytecode = type()(Collateral).creationCode (CDeployer.sol#1359)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
EleosERC20.decimals (CDeployer.sol#645) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
accountLiquidityStale(address) should be declared external:
        - Collateral.accountLiquidityStale(address) (CDeployer.sol#1239-1262)
canBorrow(address,address,uint256) should be declared external:
        - Collateral.canBorrow(address,address,uint256) (CDeployer.sol#1264-1267)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CDeployer.sol analyzed (28 contracts with 75 detectors), 88 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Collateral.sol

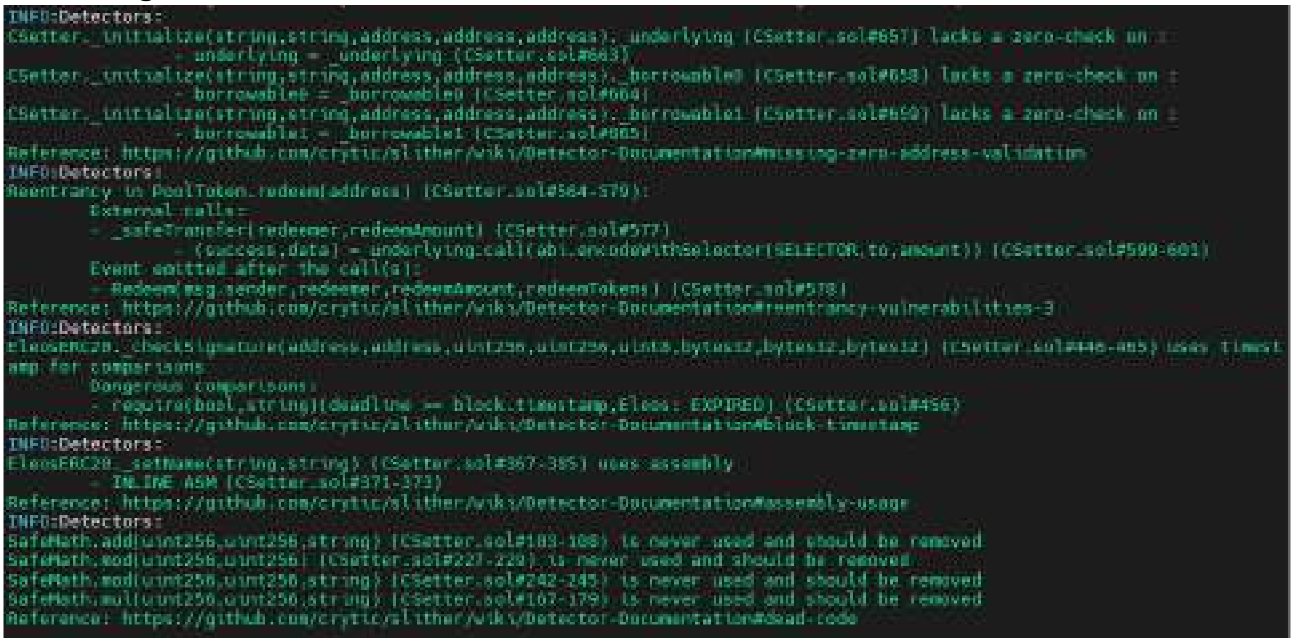


```
INFO:Detectors:
CSetter._initialize(string,string,address,address,address)._underlying (Collateral.sol#1028) lacks a zero-check on :
                - underlying = _underlying (Collateral.sol#1034)
CSetter._initialize(string,string,address,address,address)._borrowable0 (Collateral.sol#1029) lacks a zero-check on :
                - borrowable0 = _borrowable0 (Collateral.sol#1035)
CSetter._initialize(string,string,address,address,address)._borrowable1 (Collateral.sol#1030) lacks a zero-check on :
                - borrowable1 = _borrowable1 (Collateral.sol#1036)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Collateral.flashRedeem(address,uint256,bytes) (Collateral.sol#1322-1346):
        External calls:
        - _safeTransfer(redeemer,redeemAmount) (Collateral.sol#1330)
                - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (Collateral.sol#894-896)
        - IEleosCallee(redeemer).eleosRedeem(msg.sender,redeemAmount,data) (Collateral.sol#1332)
        State variables written after the call(s):
        - _burn(address(this),redeemTokens) (Collateral.sol#1344)
                - balanceOf[from] = balanceOf[from].sub(value) (Collateral.sol#609)
        - _burn(address(this),redeemTokens) (Collateral.sol#1344)
                - totalSupply = totalSupply.sub(value) (Collateral.sol#608)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Collateral._transfer(address,address,uint256) (Collateral.sol#1140-1147):
        External calls:
        - require(bool,string)(tokensUnlocked(from,value),Eleos: INSUFFICIENT_LIQUIDITY) (Collateral.sol#1145)
                - (twapPrice112x112) = IEleosPriceOracle(eleosPriceOracle).getResult(underlying) (Collateral.sol#1095-1096)
```

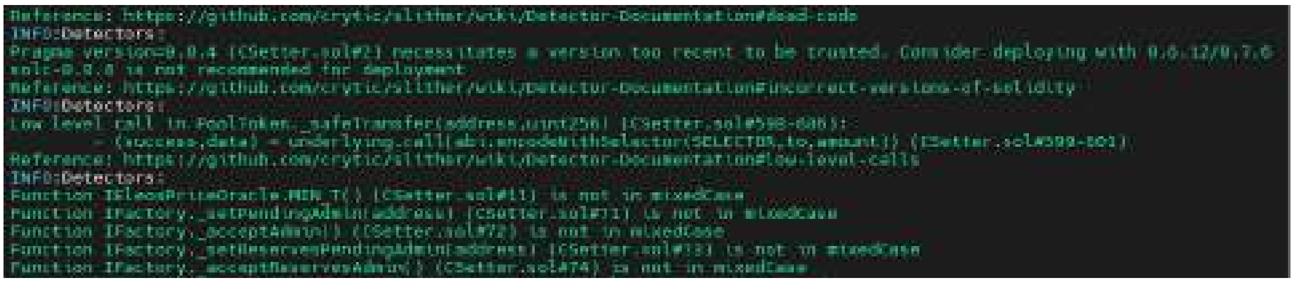


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
EleosERC20.decimals (Collateral.sol#645) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
accountLiquidityStale(address) should be declared external:
        - Collateral.accountLiquidityStale(address) (Collateral.sol#1239-1262)
canBorrow(address,address,uint256) should be declared external:
        - Collateral.canBorrow(address,address,uint256) (Collateral.sol#1264-1287)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Collateral.sol analyzed (19 contracts with 75 detectors), 87 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

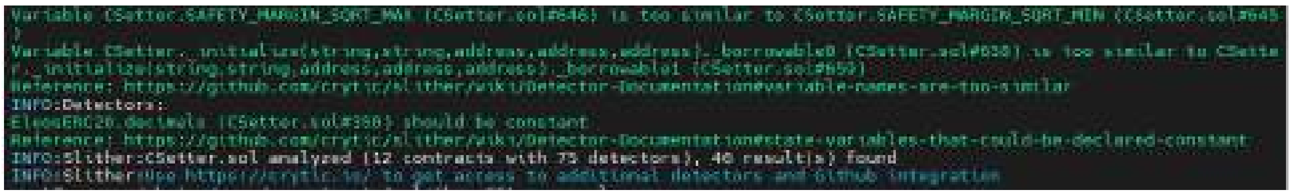
Slither log >> CSetter.sol



```
INFO:Detectors:
CSetter._initialize(string,string,address,address,address)._underlying (CSetter.sol#657) lacks a zero-check on :
                - underlying = _underlying (CSetter.sol#663)
CSetter._initialize(string,string,address,address,address)._borrowable0 (CSetter.sol#658) lacks a zero-check on :
                - borrowable0 = _borrowable0 (CSetter.sol#664)
CSetter._initialize(string,string,address,address,address)._borrowable1 (CSetter.sol#659) lacks a zero-check on :
                - borrowable1 = _borrowable1 (CSetter.sol#665)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in PoolToken.redeem(address) (CSetter.sol#564-579):
        External calls:
        - _safeTransfer(redeemer,redeemAmount) (CSetter.sol#577)
                - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (CSetter.sol#599-601)
        Event emitted after the call(s):
        - Redeem(msg.sender,redeemer,redeemAmount,redeemTokens) (CSetter.sol#578)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
EleosERC20._checkSignature(address,address,uint256,uint256,uint8,bytes32,bytes32,bytes32) (CSetter.sol#446-465) uses timest
amp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Eleos: EXPIRED) (CSetter.sol#456)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
EleosERC20._setName(string,string) (CSetter.sol#367-385) uses assembly
        - INLINE ASM (CSetter.sol#371-373)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
SafeMath.add(uint256,uint256,string) (CSetter.sol#183-188) is never used and should be removed
SafeMath.mod(uint256,uint256) (CSetter.sol#227-229) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (CSetter.sol#242-245) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (CSetter.sol#167-179) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

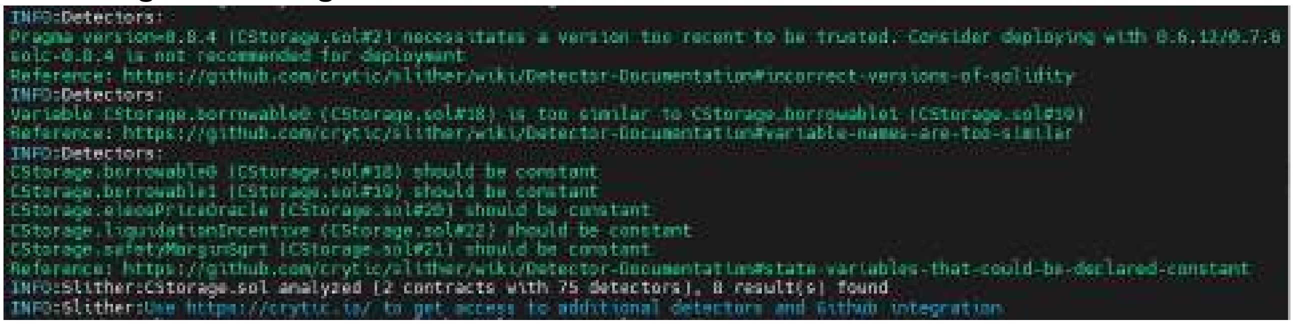


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.4 (CSetter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.8 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in PoolToken._safeTransfer(address,uint256) (CSetter.sol#598-606):
        - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (CSetter.sol#599-601)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IEleosPriceOracle.MIN_T() (CSetter.sol#11) is not in mixedCase
Function IFactory._setPendingAdmin(address) (CSetter.sol#71) is not in mixedCase
Function IFactory._acceptAdmin() (CSetter.sol#72) is not in mixedCase
Function IFactory._setReservesPendingAdmin(address) (CSetter.sol#73) is not in mixedCase
Function IFactory._acceptReservesAdmin() (CSetter.sol#74) is not in mixedCase
```



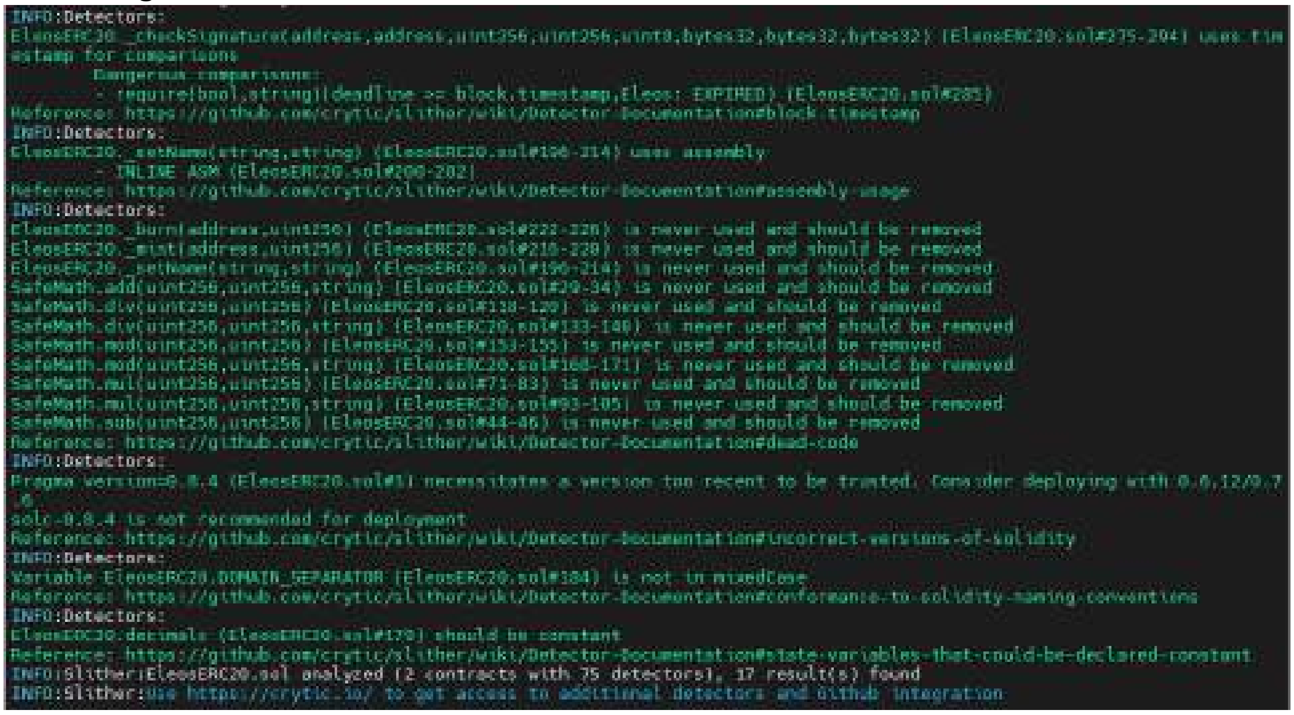
```
Variable CSetter.SAFETY_MARGIN_SQRT_MAX (CSetter.sol#646) is too similar to CSetter.SAFETY_MARGIN_SQRT_MIN (CSetter.sol#645
)
Variable CSetter._initialize(string,string,address,address,address)._borrowable0 (CSetter.sol#658) is too similar to CSette
r._initialize(string,string,address,address,address)._borrowable1 (CSetter.sol#659)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
EleosERC20.decimals (CSetter.sol#388) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:CSetter.sol analyzed (12 contracts with 75 detectors), 40 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> CStorage.sol



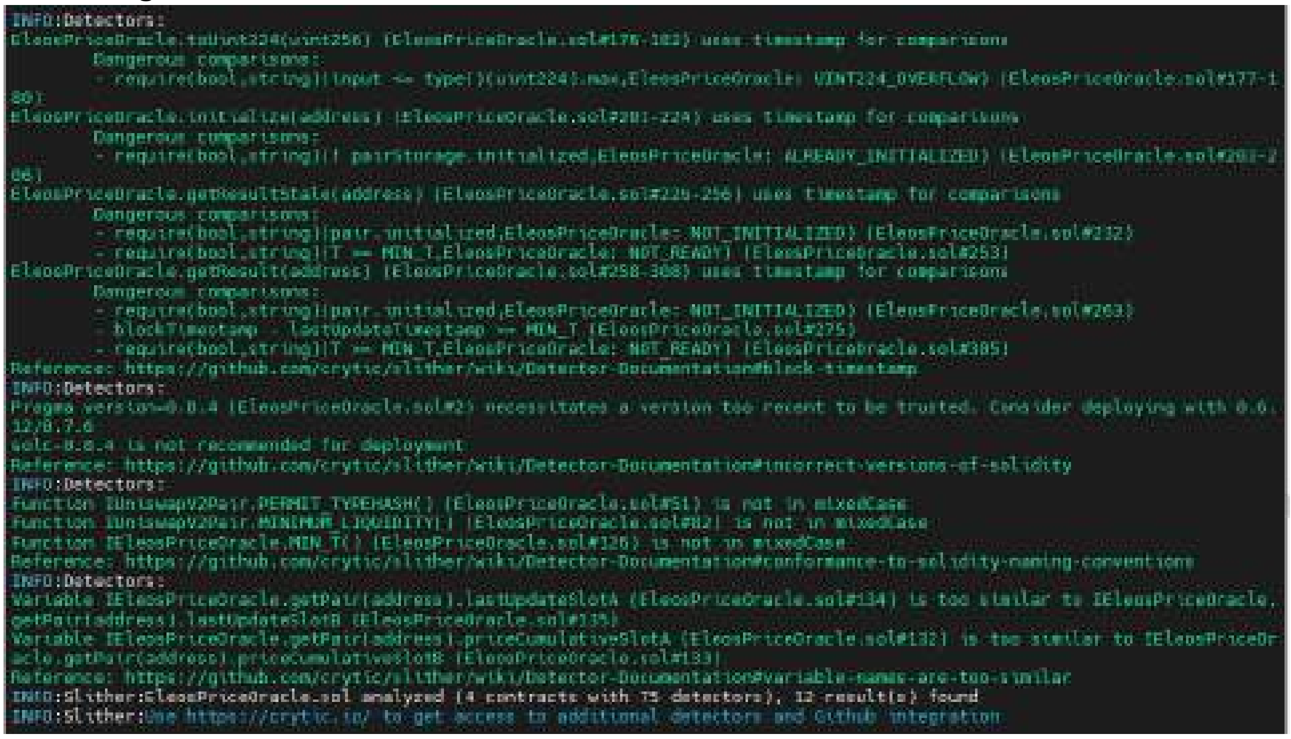
```
INFO:Detectors:
Pragma version=0.8.4 (CStorage.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable CStorage.borrowable0 (CStorage.sol#18) is too similar to CStorage.borrowable1 (CStorage.sol#19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
CStorage.borrowable0 (CStorage.sol#18) should be constant
CStorage.borrowable1 (CStorage.sol#19) should be constant
CStorage.eleosPriceOracle (CStorage.sol#20) should be constant
CStorage.liquidationIncentive (CStorage.sol#22) should be constant
CStorage.safetyMarginSqrt (CStorage.sol#21) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:CStorage.sol analyzed (2 contracts with 75 detectors), 8 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> EleosERC20.sol



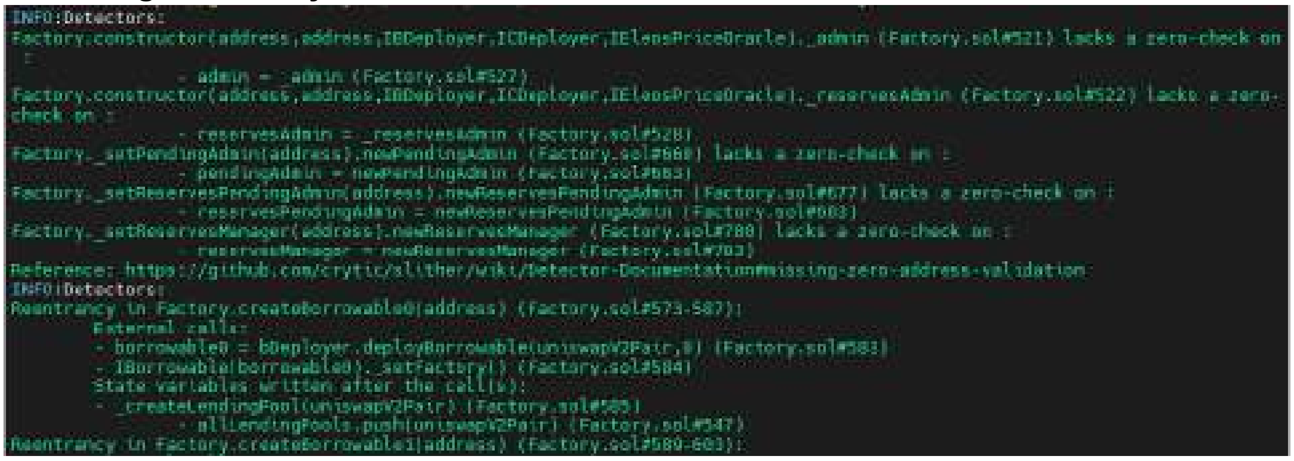
```
INFO:Detectors:
EleosERC20._checkSignature(address,address,uint256,uint256,uint8,bytes32,bytes32,bytes32) (EleosERC20.sol#275-294) uses tim
estamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Eleos: EXPIRED) (EleosERC20.sol#285)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
EleosERC20._setName(string,string) (EleosERC20.sol#196-214) uses assembly
        - INLINE ASM (EleosERC20.sol#200-202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
EleosERC20._burn(address,uint256) (EleosERC20.sol#222-228) is never used and should be removed
EleosERC20._mint(address,uint256) (EleosERC20.sol#216-220) is never used and should be removed
EleosERC20._setName(string,string) (EleosERC20.sol#196-214) is never used and should be removed
SafeMath.add(uint256,uint256,string) (EleosERC20.sol#29-34) is never used and should be removed
SafeMath.div(uint256,uint256) (EleosERC20.sol#118-120) is never used and should be removed
SafeMath.div(uint256,uint256,string) (EleosERC20.sol#133-140) is never used and should be removed
SafeMath.mod(uint256,uint256) (EleosERC20.sol#153-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (EleosERC20.sol#168-171) is never used and should be removed
SafeMath.mul(uint256,uint256) (EleosERC20.sol#71-83) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (EleosERC20.sol#93-105) is never used and should be removed
SafeMath.sub(uint256,uint256) (EleosERC20.sol#44-46) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.4 (EleosERC20.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable EleosERC20.DOMAIN_SEPARATOR (EleosERC20.sol#184) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
EleosERC20.decimals (EleosERC20.sol#170) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:EleosERC20.sol analyzed (2 contracts with 75 detectors), 17 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> EleosPriceOracle.sol

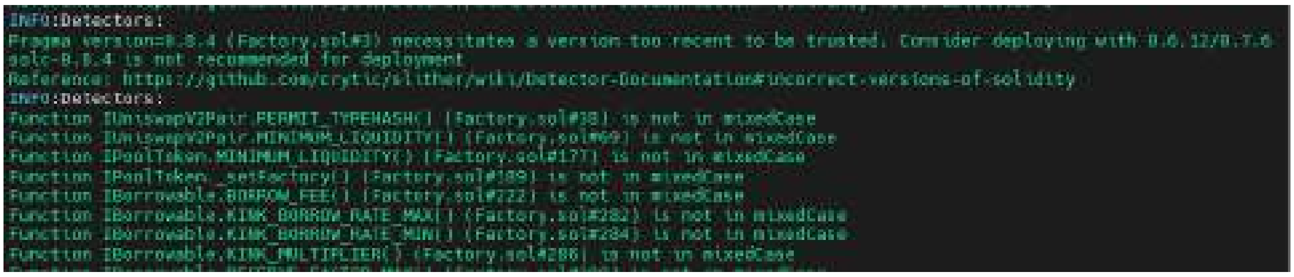


```
INFO:Detectors:
EleosPriceOracle.toUint224(uint256) (EleosPriceOracle.sol#176-182) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(input <= type()(uint224).max,EleosPriceOracle: UINT224_OVERFLOW) (EleosPriceOracle.sol#177-1
80)
EleosPriceOracle.initialize(address) (EleosPriceOracle.sol#201-224) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(! pairStorage.initialized,EleosPriceOracle: ALREADY_INITIALIZED) (EleosPriceOracle.sol#203-2
06)
EleosPriceOracle.getResultState(address) (EleosPriceOracle.sol#226-256) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(pair.initialized,EleosPriceOracle: NOT_INITIALIZED) (EleosPriceOracle.sol#232)
        - require(bool,string)(T >= MIN_T,EleosPriceOracle: NOT_READY) (EleosPriceOracle.sol#253)
EleosPriceOracle.getResult(address) (EleosPriceOracle.sol#258-308) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(pair.initialized,EleosPriceOracle: NOT_INITIALIZED) (EleosPriceOracle.sol#263)
        - blockTimestamp - lastUpdateTimestamp >= MIN_T (EleosPriceOracle.sol#275)
        - require(bool,string)(T >= MIN_T,EleosPriceOracle: NOT_READY) (EleosPriceOracle.sol#305)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Pragma version=0.8.4 (EleosPriceOracle.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.
12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Pair.PERMIT_TYPEHASH() (EleosPriceOracle.sol#51) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (EleosPriceOracle.sol#82) is not in mixedCase
Function IEleosPriceOracle.MIN_T() (EleosPriceOracle.sol#126) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IEleosPriceOracle.getPair(address).lastUpdateSlotA (EleosPriceOracle.sol#134) is too similar to IEleosPriceOracle.
getPair(address).lastUpdateSlotB (EleosPriceOracle.sol#135)
Variable IEleosPriceOracle.getPair(address).priceCumulativeSlotA (EleosPriceOracle.sol#132) is too similar to IEleosPriceOr
acle.getPair(address).priceCumulativeSlotB (EleosPriceOracle.sol#133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Slither:EleosPriceOracle.sol analyzed (4 contracts with 75 detectors), 12 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

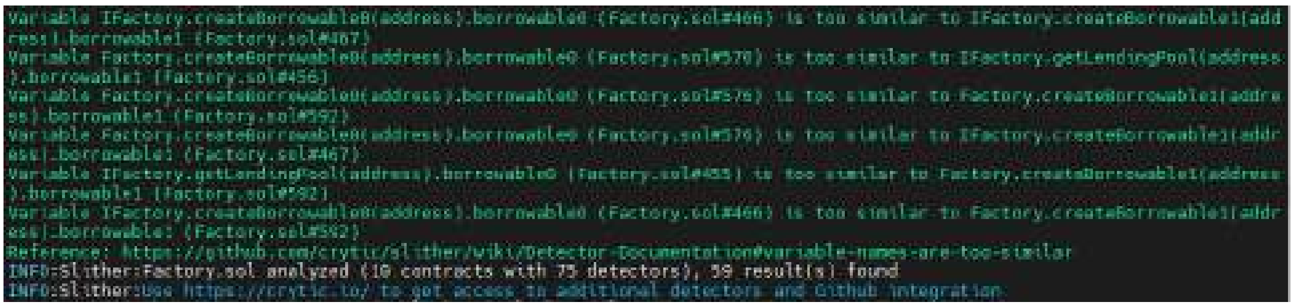
Slither log >> Factory.sol



```
INFO:Detectors:
Factory.constructor(address,address,IBDeployer,ICDeployer,IEleosPriceOracle)._admin (Factory.sol#521) lacks a zero-check on
 :
                - admin = _admin (Factory.sol#527)
Factory.constructor(address,address,IBDeployer,ICDeployer,IEleosPriceOracle)._reservesAdmin (Factory.sol#522) lacks a zero-
check on :
                - reservesAdmin = _reservesAdmin (Factory.sol#528)
Factory._setPendingAdmin(address).newPendingAdmin (Factory.sol#660) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (Factory.sol#663)
Factory._setReservesPendingAdmin(address).newReservesPendingAdmin (Factory.sol#677) lacks a zero-check on :
                - reservesPendingAdmin = newReservesPendingAdmin (Factory.sol#683)
Factory._setReservesManager(address).newReservesManager (Factory.sol#700) lacks a zero-check on :
                - reservesManager = newReservesManager (Factory.sol#703)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Factory.createBorrowable0(address) (Factory.sol#573-587):
        External calls:
        - borrowable0 = bDeployer.deployBorrowable(uniswapV2Pair,0) (Factory.sol#583)
        - IBorrowable(borrowable0)._setFactory() (Factory.sol#584)
        State variables written after the call(s):
        - _createLendingPool(uniswapV2Pair) (Factory.sol#585)
                - allLendingPools.push(uniswapV2Pair) (Factory.sol#547)
Reentrancy in Factory.createBorrowable1(address) (Factory.sol#589-603):
```

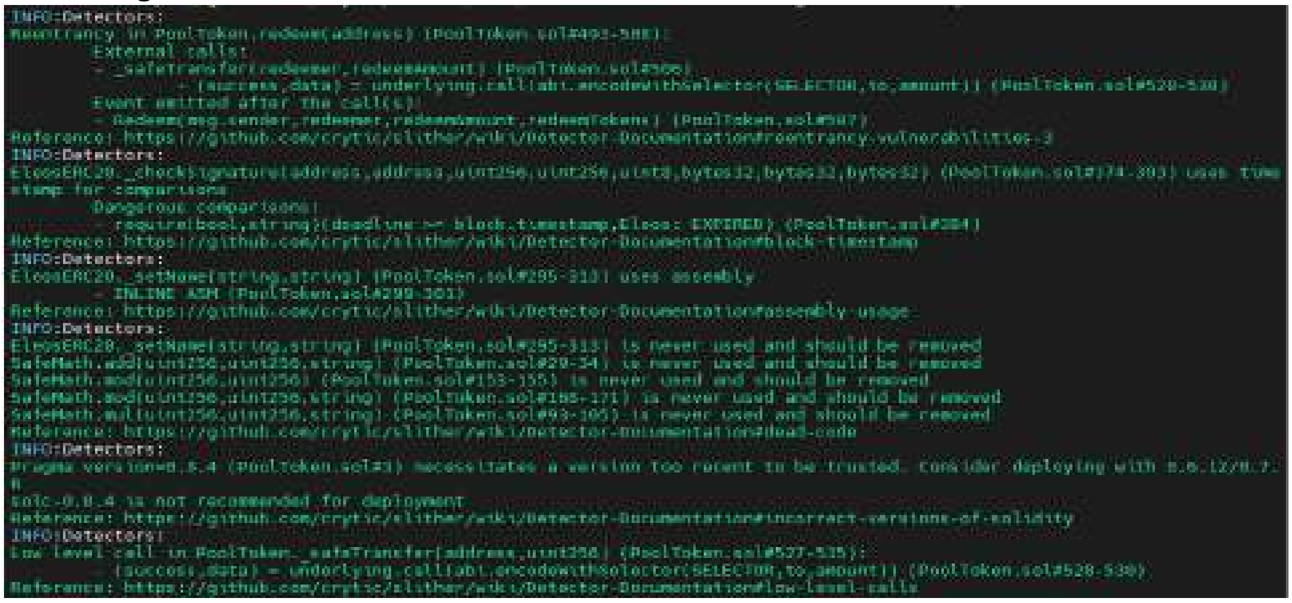


```
INFO:Detectors:
Pragma version=0.8.4 (Factory.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Pair.PERMIT_TYPEHASH() (Factory.sol#38) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (Factory.sol#69) is not in mixedCase
Function IPoolToken.MINIMUM_LIQUIDITY() (Factory.sol#177) is not in mixedCase
Function IPoolToken._setFactory() (Factory.sol#189) is not in mixedCase
Function IBorrowable.BORROW_FEE() (Factory.sol#222) is not in mixedCase
Function IBorrowable.KINK_BORROW_RATE_MAX() (Factory.sol#282) is not in mixedCase
Function IBorrowable.KINK_BORROW_RATE_MIN() (Factory.sol#284) is not in mixedCase
Function IBorrowable.KINK_MULTIPLIER() (Factory.sol#286) is not in mixedCase
```

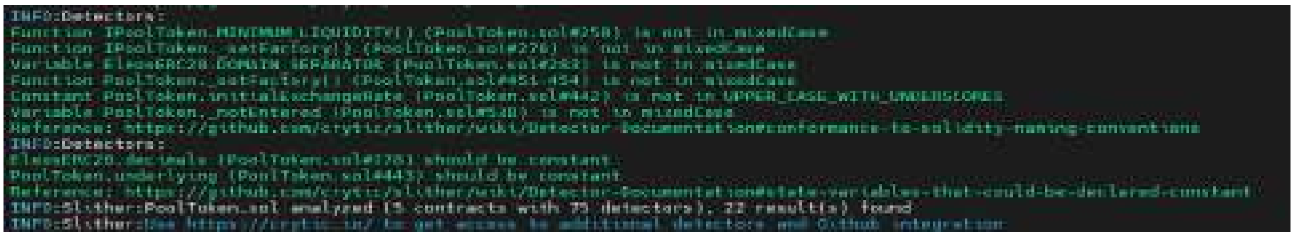


```
Variable IFactory.createBorrowable0(address).borrowable0 (Factory.sol#466) is too similar to IFactory.createBorrowable1(add
ress).borrowable1 (Factory.sol#467)
Variable Factory.createBorrowable0(address).borrowable0 (Factory.sol#576) is too similar to IFactory.getLendingPool(address
).borrowable1 (Factory.sol#456)
Variable Factory.createBorrowable0(address).borrowable0 (Factory.sol#576) is too similar to Factory.createBorrowable1(addre
ss).borrowable1 (Factory.sol#592)
Variable Factory.createBorrowable0(address).borrowable0 (Factory.sol#576) is too similar to IFactory.createBorrowable1(addr
ess).borrowable1 (Factory.sol#467)
Variable IFactory.getLendingPool(address).borrowable0 (Factory.sol#455) is too similar to Factory.createBorrowable1(address
).borrowable1 (Factory.sol#592)
Variable IFactory.createBorrowable0(address).borrowable0 (Factory.sol#466) is too similar to Factory.createBorrowable1(addr
ess).borrowable1 (Factory.sol#592)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Slither:Factory.sol analyzed (10 contracts with 75 detectors), 59 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> PoolToken.sol

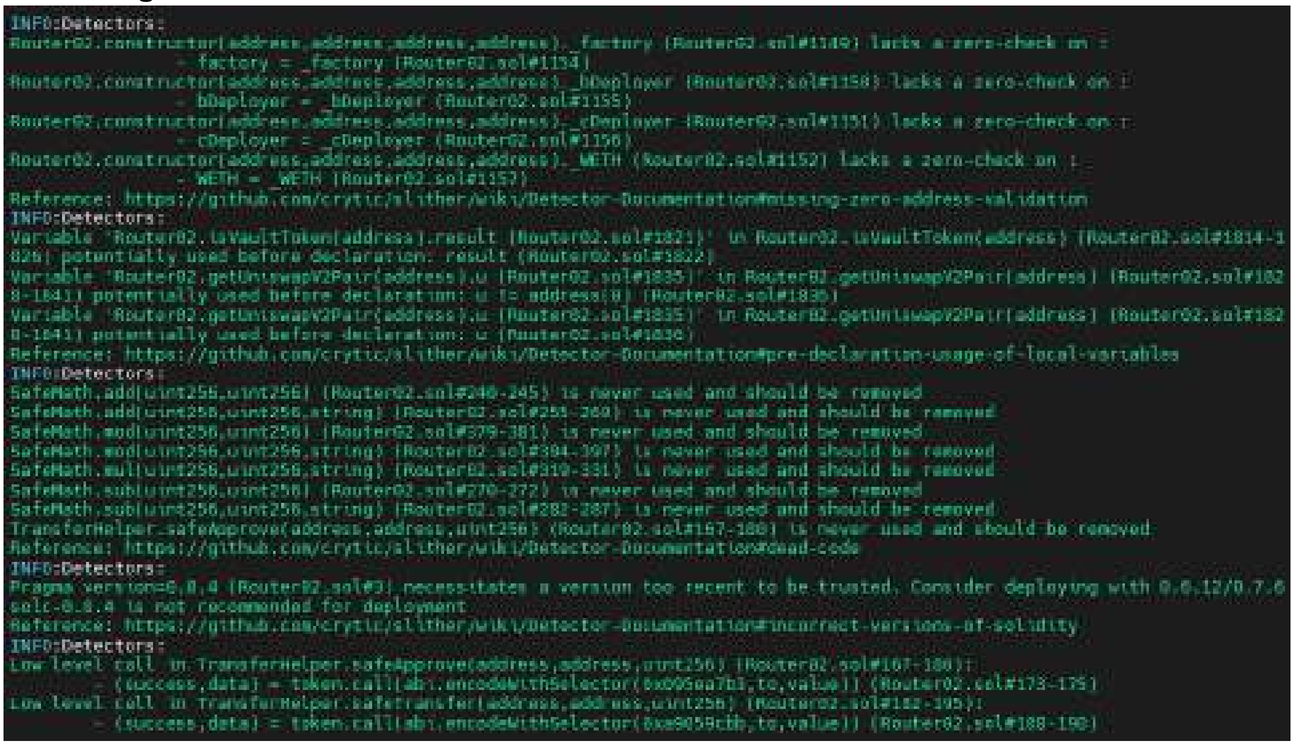


```
INFO:Detectors:
Reentrancy in PoolToken.redeem(address) (PoolToken.sol#491-508):
        External calls:
        - _safeTransfer(redeemer,redeemAmount) (PoolToken.sol#506)
                - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (PoolToken.sol#528-530)
        Event emitted after the call(s):
        - Redeem(msg.sender,redeemer,redeemAmount,redeemTokens) (PoolToken.sol#507)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
EleosERC20._checkSignature(address,address,uint256,uint256,uint8,bytes32,bytes32,bytes32) (PoolToken.sol#374-393) uses time
stamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Eleos: EXPIRED) (PoolToken.sol#384)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
EleosERC20._setName(string,string) (PoolToken.sol#295-313) uses assembly
        - INLINE ASM (PoolToken.sol#299-301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
EleosERC20._setName(string,string) (PoolToken.sol#295-313) is never used and should be removed
SafeMath.add(uint256,uint256,string) (PoolToken.sol#29-34) is never used and should be removed
SafeMath.mod(uint256,uint256) (PoolToken.sol#153-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (PoolToken.sol#165-171) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (PoolToken.sol#93-105) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.4 (PoolToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in PoolToken._safeTransfer(address,uint256) (PoolToken.sol#527-535):
        - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (PoolToken.sol#528-530)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

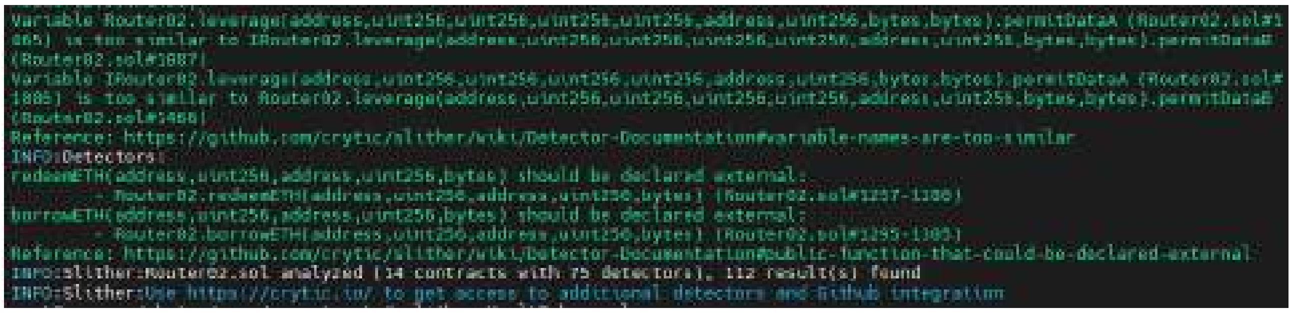


```
INFO:Detectors:
Function IPoolToken.MINIMUM_LIQUIDITY() (PoolToken.sol#258) is not in mixedCase
Function IPoolToken._setFactory() (PoolToken.sol#270) is not in mixedCase
Variable EleosERC20.DOMAIN_SEPARATOR (PoolToken.sol#283) is not in mixedCase
Function PoolToken._setFactory() (PoolToken.sol#451-454) is not in mixedCase
Constant PoolToken.initialExchangeRate (PoolToken.sol#442) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PoolToken._notEntered (PoolToken.sol#520) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
EleosERC20.decimals (PoolToken.sol#278) should be constant
PoolToken.underlying (PoolToken.sol#443) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:PoolToken.sol analyzed (5 contracts with 75 detectors), 22 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Router02.sol

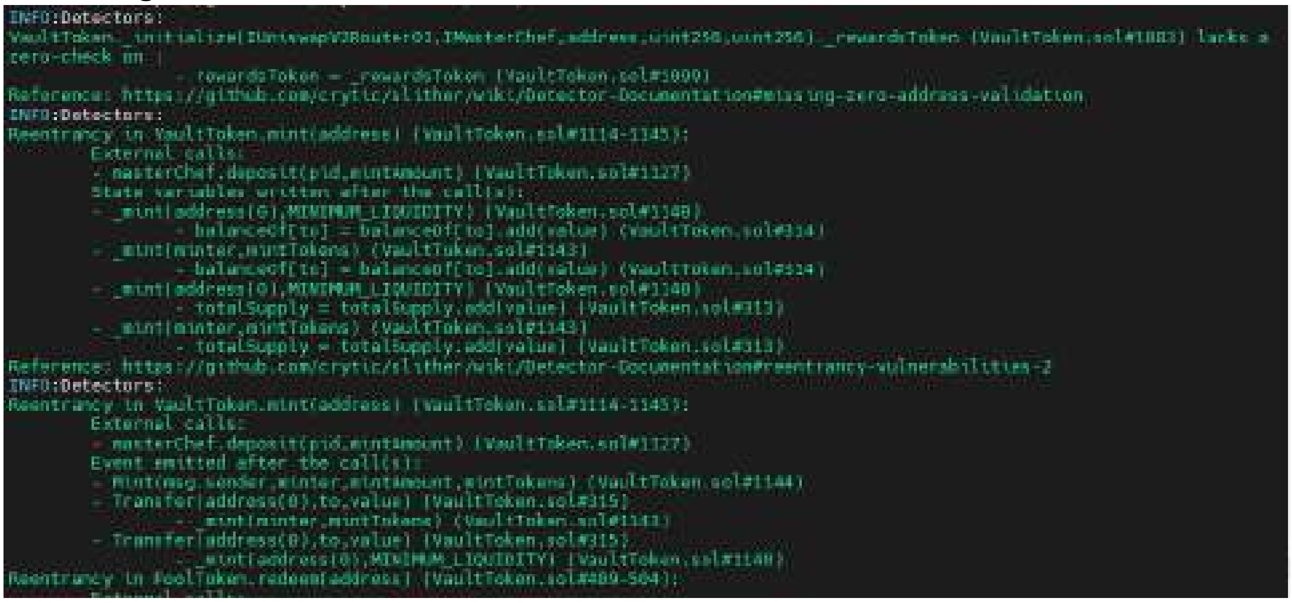


```
INFO:Detectors:
Router02.constructor(address,address,address,address)._factory (Router02.sol#1149) lacks a zero-check on :
                - factory = _factory (Router02.sol#1154)
Router02.constructor(address,address,address,address)._bDeployer (Router02.sol#1150) lacks a zero-check on :
                - bDeployer = _bDeployer (Router02.sol#1155)
Router02.constructor(address,address,address,address)._cDeployer (Router02.sol#1151) lacks a zero-check on :
                - cDeployer = _cDeployer (Router02.sol#1156)
Router02.constructor(address,address,address,address)._WETH (Router02.sol#1152) lacks a zero-check on :
                - WETH = _WETH (Router02.sol#1157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'Router02.isVaultToken(address).result (Router02.sol#1821)' in Router02.isVaultToken(address) (Router02.sol#1814-1
826) potentially used before declaration: result (Router02.sol#1822)
Variable 'Router02.getUniswapV2Pair(address).u (Router02.sol#1835)' in Router02.getUniswapV2Pair(address) (Router02.sol#182
8-1841) potentially used before declaration: u != address(0) (Router02.sol#1836)
Variable 'Router02.getUniswapV2Pair(address).u (Router02.sol#1835)' in Router02.getUniswapV2Pair(address) (Router02.sol#182
8-1841) potentially used before declaration: u (Router02.sol#1836)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
SafeMath.add(uint256,uint256) (Router02.sol#240-245) is never used and should be removed
SafeMath.add(uint256,uint256,string) (Router02.sol#255-260) is never used and should be removed
SafeMath.mod(uint256,uint256) (Router02.sol#379-381) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Router02.sol#394-397) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (Router02.sol#319-331) is never used and should be removed
SafeMath.sub(uint256,uint256) (Router02.sol#270-272) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (Router02.sol#282-287) is never used and should be removed
TransferHelper.safeApprove(address,address,uint256) (Router02.sol#167-180) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.4 (Router02.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in TransferHelper.safeApprove(address,address,uint256) (Router02.sol#167-180):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (Router02.sol#173-175)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (Router02.sol#182-195):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (Router02.sol#188-190)
```

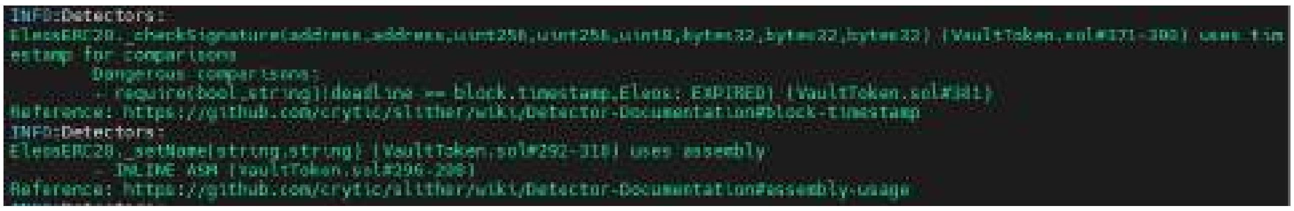


```
Variable Router02.leverage(address,uint256,uint256,uint256,uint256,address,uint256,bytes,bytes).permitDataA (Router02.sol#1
465) is too similar to IRouter02.leverage(address,uint256,uint256,uint256,uint256,address,uint256,bytes,bytes).permitDataB
(Router02.sol#1087)
Variable IRouter02.leverage(address,uint256,uint256,uint256,uint256,address,uint256,bytes,bytes).permitDataA (Router02.sol#
1086) is too similar to Router02.leverage(address,uint256,uint256,uint256,uint256,address,uint256,bytes,bytes).permitDataB
(Router02.sol#1466)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
redeemETH(address,uint256,address,uint256,bytes) should be declared external:
        - Router02.redeemETH(address,uint256,address,uint256,bytes) (Router02.sol#1257-1280)
borrowETH(address,uint256,address,uint256,bytes) should be declared external:
        - Router02.borrowETH(address,uint256,address,uint256,bytes) (Router02.sol#1295-1305)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Router02.sol analyzed (14 contracts with 75 detectors), 112 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

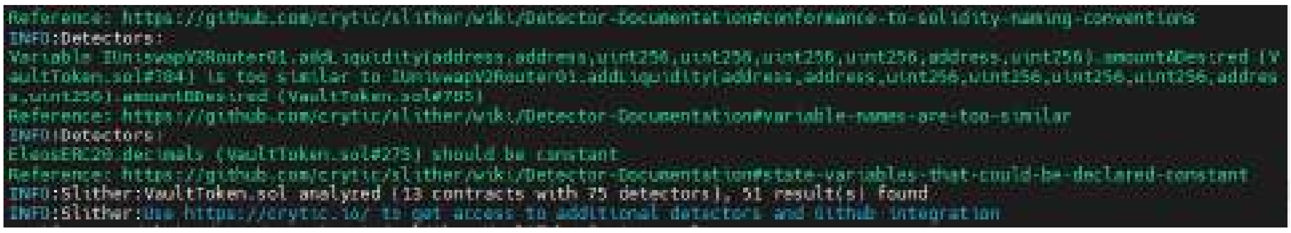
Slither log >> VaultToken.sol



```
INFO:Detectors:
VaultToken._initialize(IUniswapV2Router01,IMasterChef,address,uint256,uint256)._rewardsToken (VaultToken.sol#1083) lacks a
zero-check on :
                - rewardsToken = _rewardsToken (VaultToken.sol#1090)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in VaultToken.mint(address) (VaultToken.sol#1114-1145):
        External calls:
        - masterChef.deposit(pid,mintAmount) (VaultToken.sol#1127)
        State variables written after the call(s):
        - _mint(address(0),MINIMUM_LIQUIDITY) (VaultToken.sol#1140)
                - balanceOf[to] = balanceOf[to].add(value) (VaultToken.sol#314)
        - _mint(minter,mintTokens) (VaultToken.sol#1143)
                - balanceOf[to] = balanceOf[to].add(value) (VaultToken.sol#314)
        - _mint(address(0),MINIMUM_LIQUIDITY) (VaultToken.sol#1140)
                - totalSupply = totalSupply.add(value) (VaultToken.sol#313)
        - _mint(minter,mintTokens) (VaultToken.sol#1143)
                - totalSupply = totalSupply.add(value) (VaultToken.sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in VaultToken.mint(address) (VaultToken.sol#1114-1145):
        External calls:
        - masterChef.deposit(pid,mintAmount) (VaultToken.sol#1127)
        Event emitted after the call(s):
        - Mint(msg.sender,minter,mintAmount,mintTokens) (VaultToken.sol#1144)
        - Transfer(address(0),to,value) (VaultToken.sol#315)
                - _mint(minter,mintTokens) (VaultToken.sol#1143)
        - Transfer(address(0),to,value) (VaultToken.sol#315)
                - _mint(address(0),MINIMUM_LIQUIDITY) (VaultToken.sol#1140)
Reentrancy in PoolToken.redeem(address) (VaultToken.sol#489-504):
```

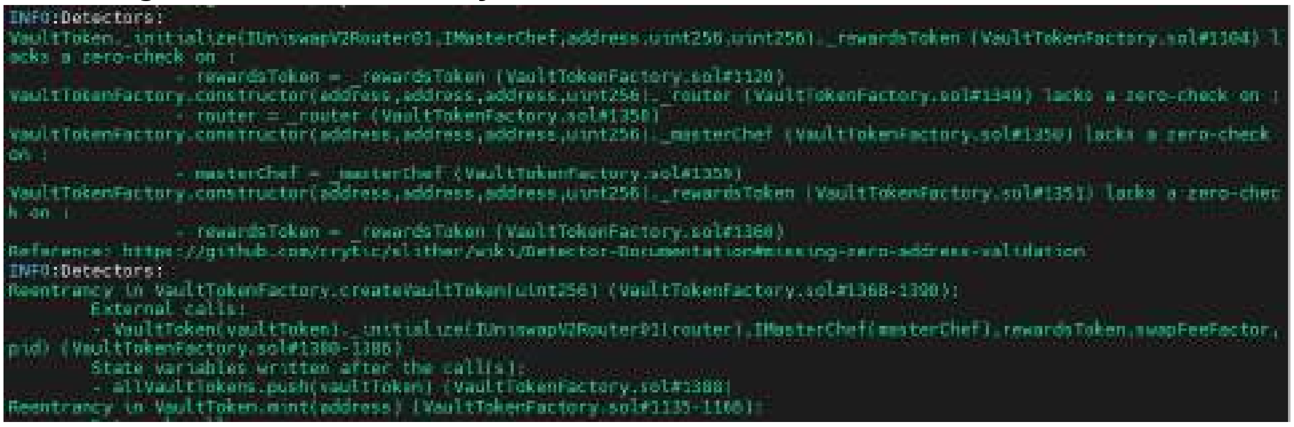


```
INFO:Detectors:
ElessERC20._checkSignature(address,address,uint256,uint256,uint8,bytes32,bytes32,bytes32) (VaultToken.sol#371-390) uses tim
estamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Eless: EXPIRED) (VaultToken.sol#381)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
ElessERC20._setName(string,string) (VaultToken.sol#292-310) uses assembly
        - INLINE ASM (VaultToken.sol#296-298)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

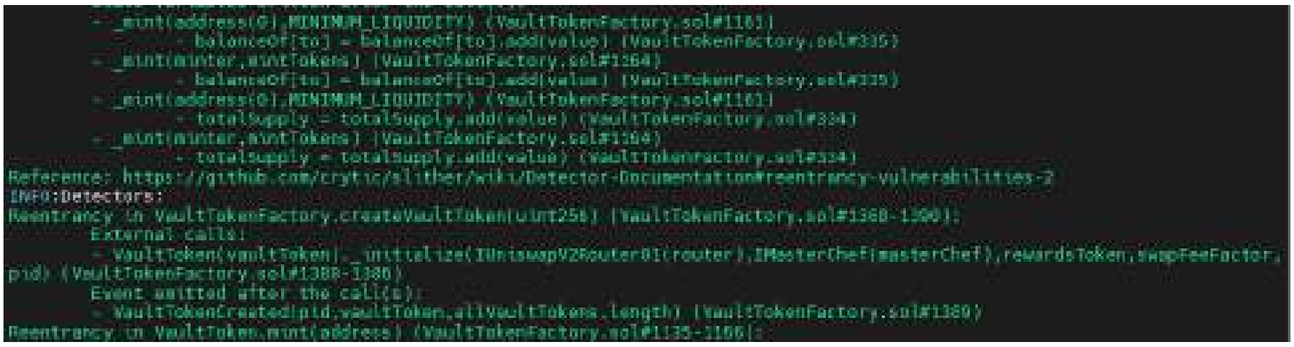


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (V
aultToken.sol#784) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,addres
s,uint256).amountBDesired (VaultToken.sol#785)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
ElossERC20.decimals (VaultToken.sol#275) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:VaultToken.sol analyzed (13 contracts with 75 detectors), 51 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

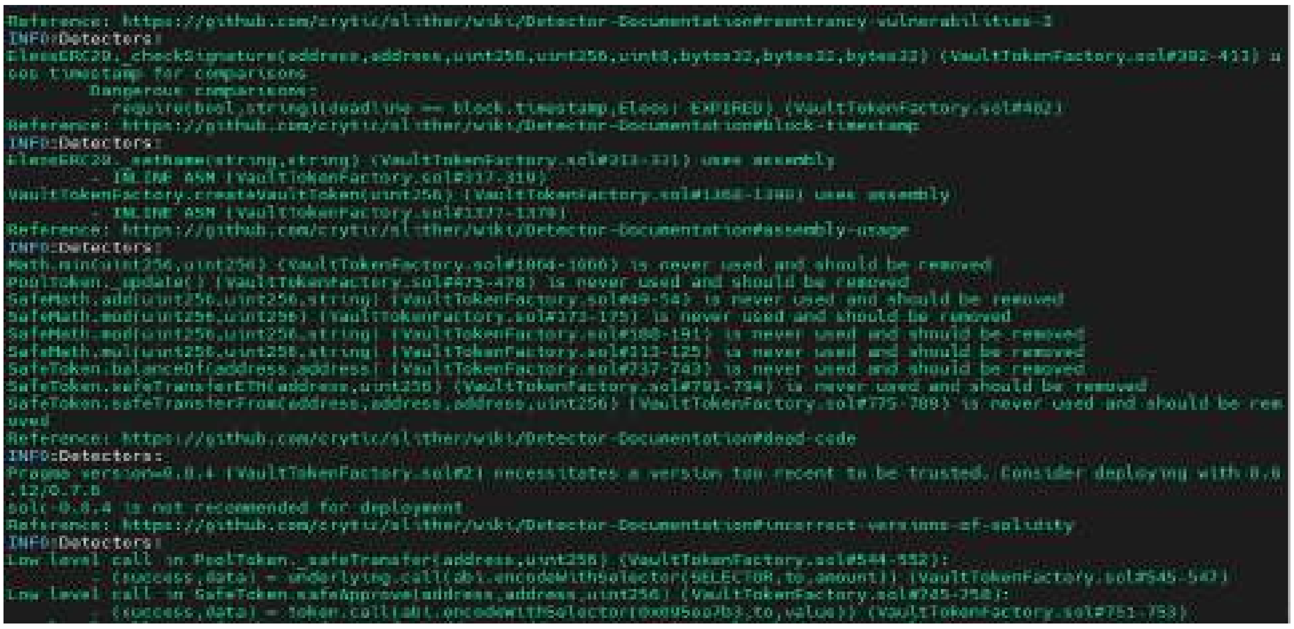
Slither log >> VaultTokenFactory.sol



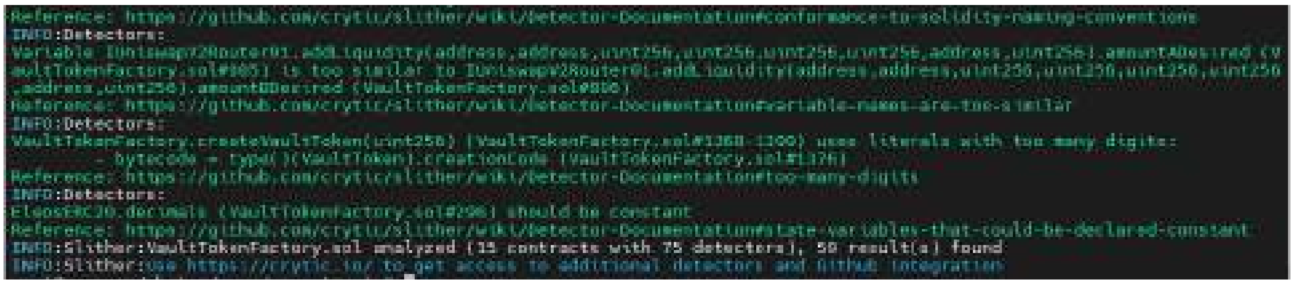
```
INFO:Detectors:
VaultToken._initialize(IUniswapV2Router01,IMasterChef,address,uint256,uint256)._rewardsToken (VaultTokenFactory.sol#1104) l
acks a zero-check on :
                - rewardsToken = _rewardsToken (VaultTokenFactory.sol#1120)
VaultTokenFactory.constructor(address,address,address,uint256)._router (VaultTokenFactory.sol#1349) lacks a zero-check on :
                - router = _router (VaultTokenFactory.sol#1358)
VaultTokenFactory.constructor(address,address,address,uint256)._masterChef (VaultTokenFactory.sol#1350) lacks a zero-check
on :
                - masterChef = _masterChef (VaultTokenFactory.sol#1359)
VaultTokenFactory.constructor(address,address,address,uint256)._rewardsToken (VaultTokenFactory.sol#1351) lacks a zero-chec
k on :
                - rewardsToken = _rewardsToken (VaultTokenFactory.sol#1360)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in VaultTokenFactory.createVaultToken(uint256) (VaultTokenFactory.sol#1368-1390):
        External calls:
        - VaultToken(vaultToken)._initialize(IUniswapV2Router01(router),IMasterChef(masterChef),rewardsToken,swapFeeFactor,
pid) (VaultTokenFactory.sol#1380-1386)
        State variables written after the call(s):
        - allVaultTokens.push(vaultToken) (VaultTokenFactory.sol#1388)
Reentrancy in VaultToken.mint(address) (VaultTokenFactory.sol#1135-1166):
```



```
        - _mint(address(0),MINIMUM_LIQUIDITY) (VaultTokenFactory.sol#1161)
                - balanceOf[to] = balanceOf[to].add(value) (VaultTokenFactory.sol#335)
        - _mint(minter,mintTokens) (VaultTokenFactory.sol#1164)
                - balanceOf[to] = balanceOf[to].add(value) (VaultTokenFactory.sol#335)
        - _mint(address(0),MINIMUM_LIQUIDITY) (VaultTokenFactory.sol#1161)
                - totalSupply = totalSupply.add(value) (VaultTokenFactory.sol#334)
        - _mint(minter,mintTokens) (VaultTokenFactory.sol#1164)
                - totalSupply = totalSupply.add(value) (VaultTokenFactory.sol#334)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in VaultTokenFactory.createVaultToken(uint256) (VaultTokenFactory.sol#1368-1390):
        External calls:
        - VaultToken(vaultToken)._initialize(IUniswapV2Router01(router),IMasterChef(masterChef),rewardsToken,swapFeeFactor,
pid) (VaultTokenFactory.sol#1380-1386)
        Event emitted after the call(s):
        - VaultTokenCreated(pid,vaultToken,allVaultTokens.length) (VaultTokenFactory.sol#1389)
Reentrancy in VaultToken.mint(address) (VaultTokenFactory.sol#1135-1166):
```



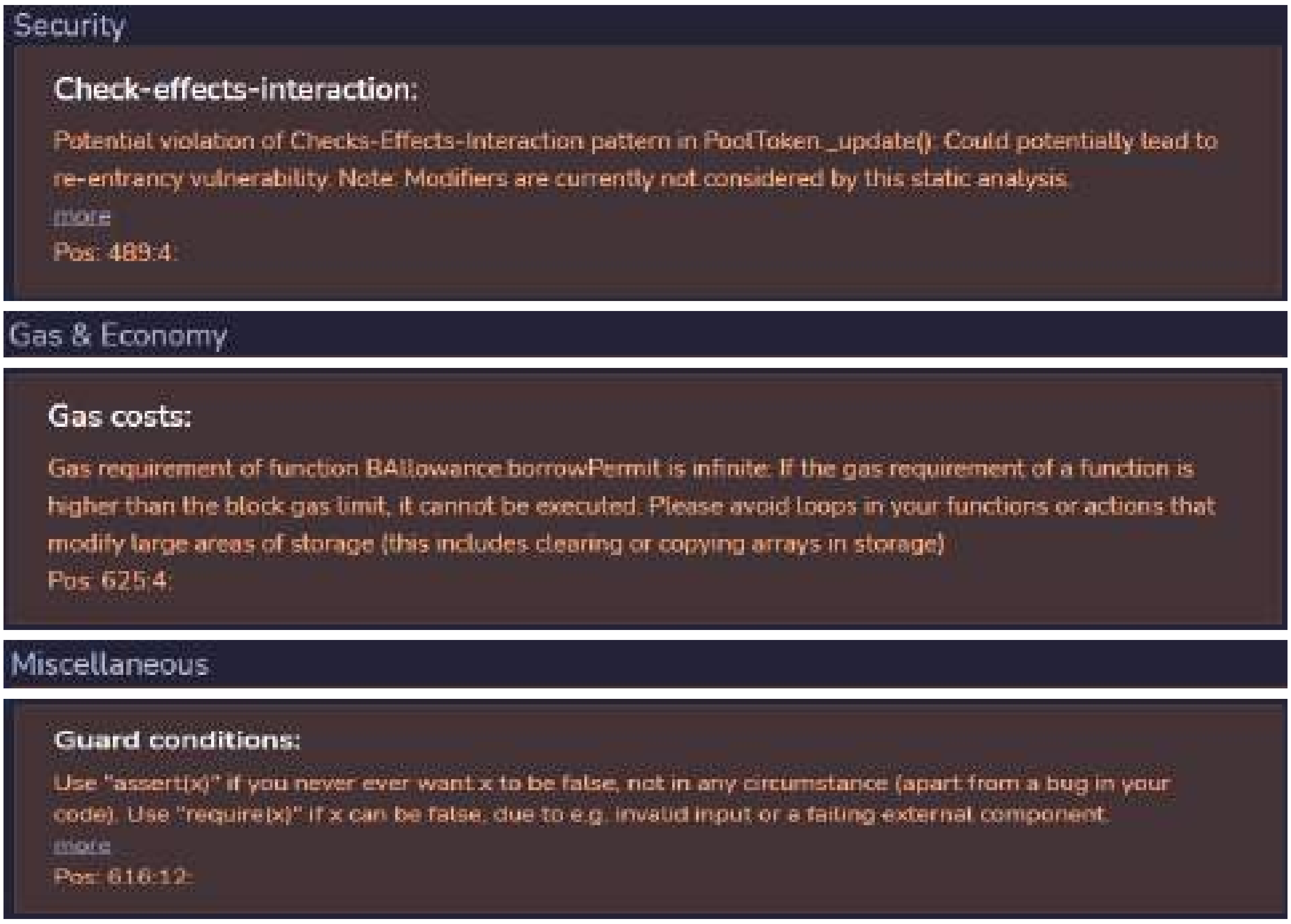
```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ElossERC20._checkSignature(address,address,uint256,uint256,uint8,bytes32,bytes32,bytes32) (VaultTokenFactory.sol#392-413) u
ses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Eloss: EXPIRED) (VaultTokenFactory.sol#402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
ElossERC20._setName(string,string) (VaultTokenFactory.sol#313-331) uses assembly
        - INLINE ASM (VaultTokenFactory.sol#317-319)
VaultTokenFactory.createVaultToken(uint256) (VaultTokenFactory.sol#1368-1390) uses assembly
        - INLINE ASM (VaultTokenFactory.sol#1377-1379)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Math.min(uint256,uint256) (VaultTokenFactory.sol#1004-1006) is never used and should be removed
PoolToken._update() (VaultTokenFactory.sol#475-478) is never used and should be removed
SafeMath.add(uint256,uint256,string) (VaultTokenFactory.sol#49-54) is never used and should be removed
SafeMath.mod(uint256,uint256) (VaultTokenFactory.sol#173-175) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (VaultTokenFactory.sol#188-191) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (VaultTokenFactory.sol#113-125) is never used and should be removed
SafeToken.balanceOf(address,address) (VaultTokenFactory.sol#737-743) is never used and should be removed
SafeToken.safeTransferETH(address,uint256) (VaultTokenFactory.sol#791-794) is never used and should be removed
SafeToken.safeTransferFrom(address,address,address,uint256) (VaultTokenFactory.sol#775-789) is never used and should be rem
oved
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.4 (VaultTokenFactory.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6
.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in PoolToken._safeTransfer(address,uint256) (VaultTokenFactory.sol#544-552):
        - (success,data) = underlying.call(abi.encodeWithSelector(SELECTOR,to,amount)) (VaultTokenFactory.sol#545-547)
Low level call in SafeToken.safeApprove(address,address,uint256) (VaultTokenFactory.sol#745-758):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (VaultTokenFactory.sol#751-753)
```



```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (V
aultTokenFactory.sol#805) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256
,address,uint256).amountBDesired (VaultTokenFactory.sol#806)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
VaultTokenFactory.createVaultToken(uint256) (VaultTokenFactory.sol#1368-1390) uses literals with too many digits:
        - bytecode = type()(VaultToken).creationCode (VaultTokenFactory.sol#1376)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ElossERC20.decimals (VaultTokenFactory.sol#296) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:VaultTokenFactory.sol analyzed (15 contracts with 75 detectors), 50 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

BAllowance.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in PoolToken._update(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 489:4:

Gas & Economy

Gas costs:

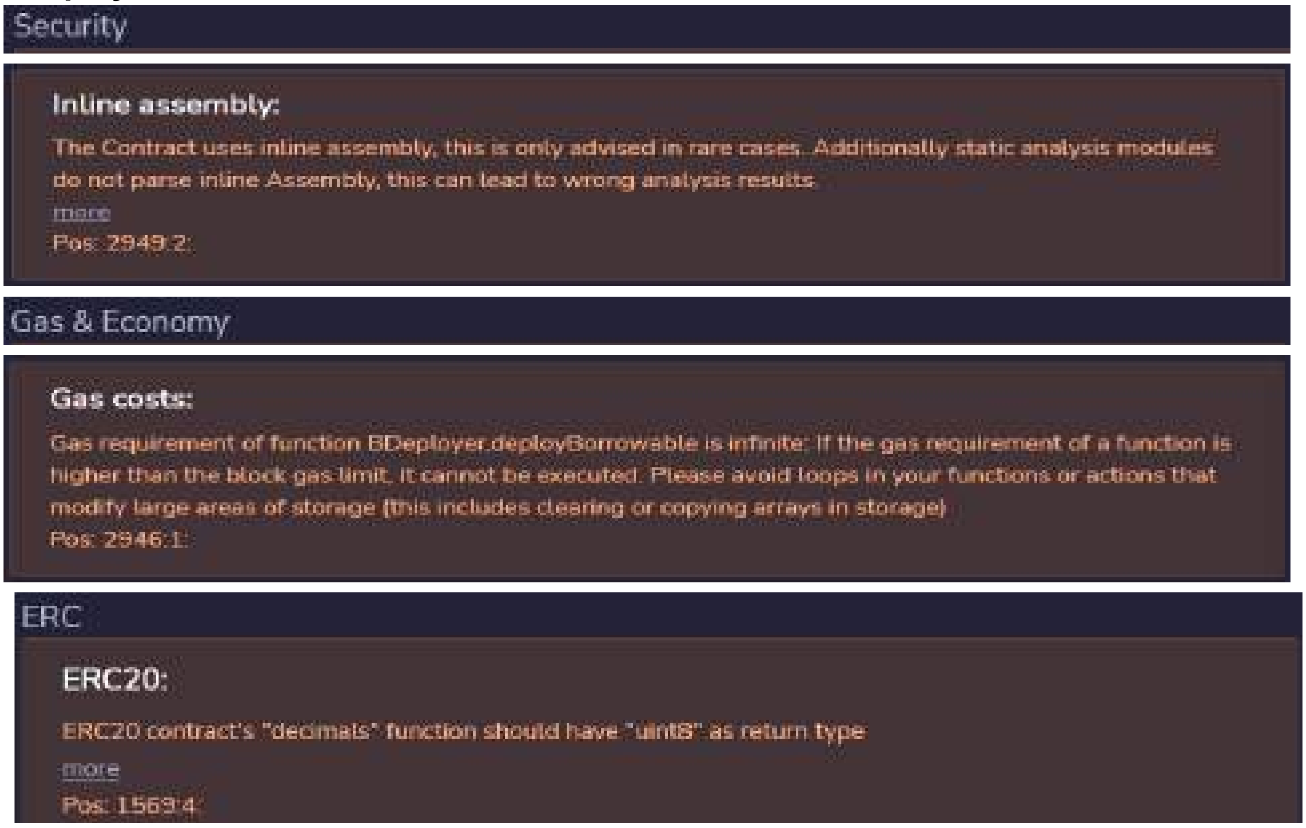
Gas requirement of function BAllowance.borrowPermit is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 625:4:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 616:12:

BDeployer.sol



Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 2949:2:

Gas & Economy

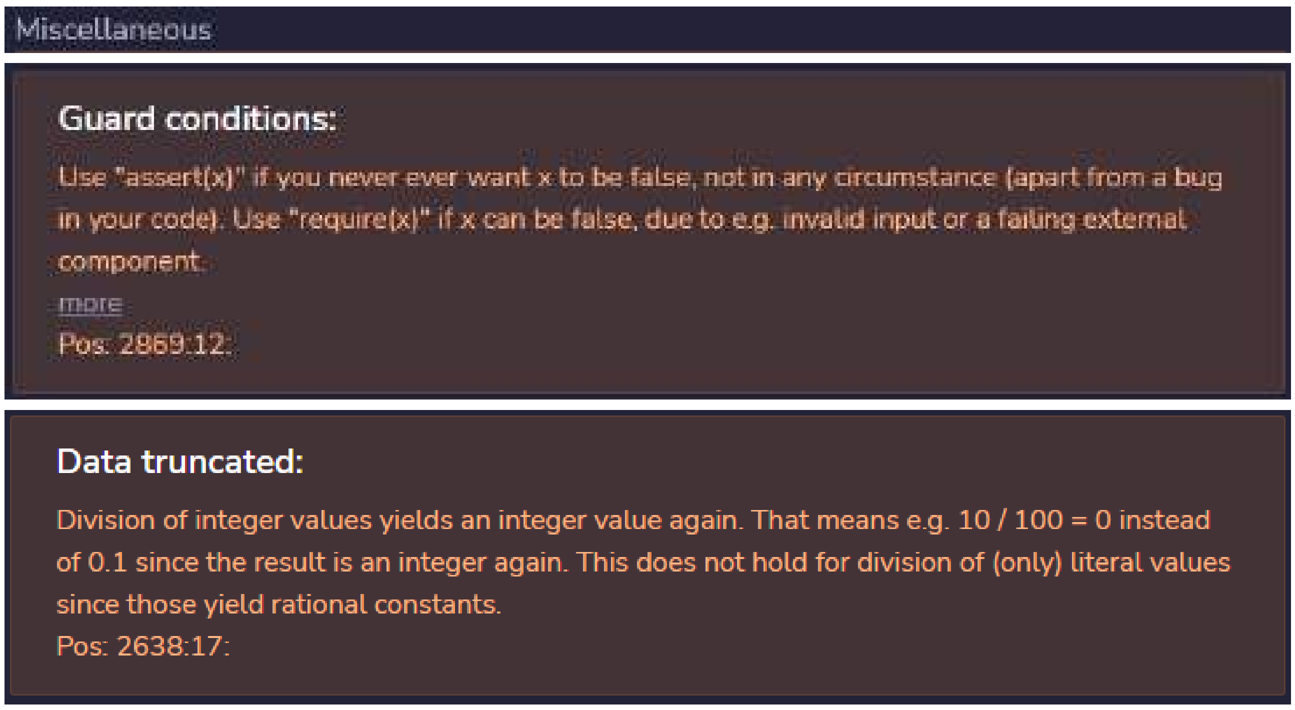
Gas costs:

Gas requirement of function BDeployer.deployBorrowable is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2946:1:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 1563:4:



Miscellaneous

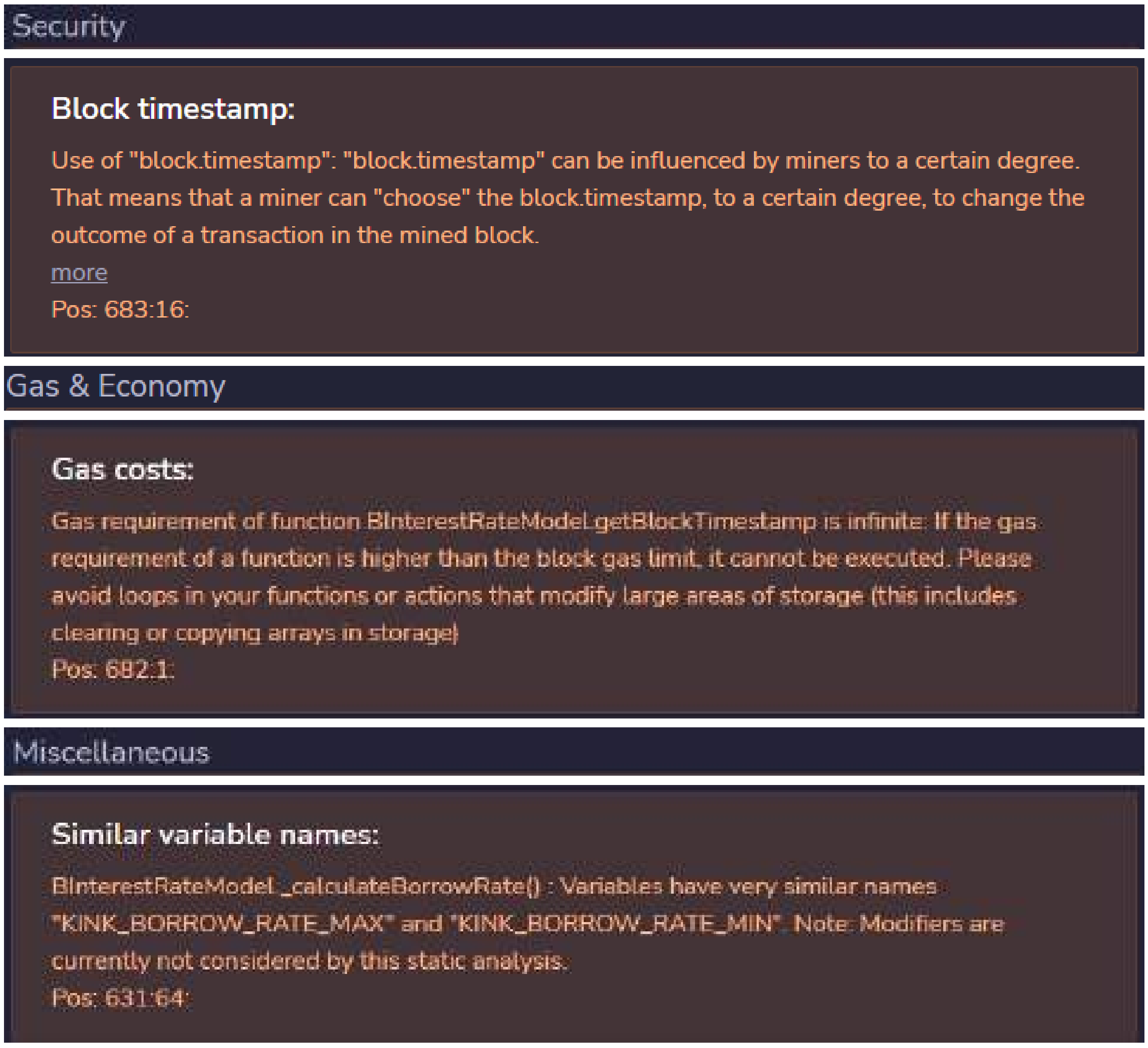
Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2869:12:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 2638:17:

BInterestRateModel.sol



Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 683:16:

Gas & Economy

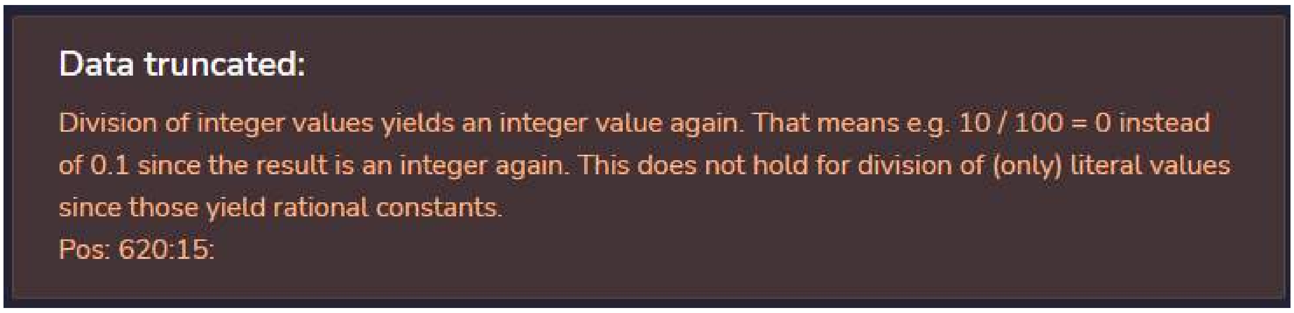
Gas costs:

Gas requirement of function BInterestRateModel.getBlockTimestamp is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 682:1:

Miscellaneous

Similar variable names:

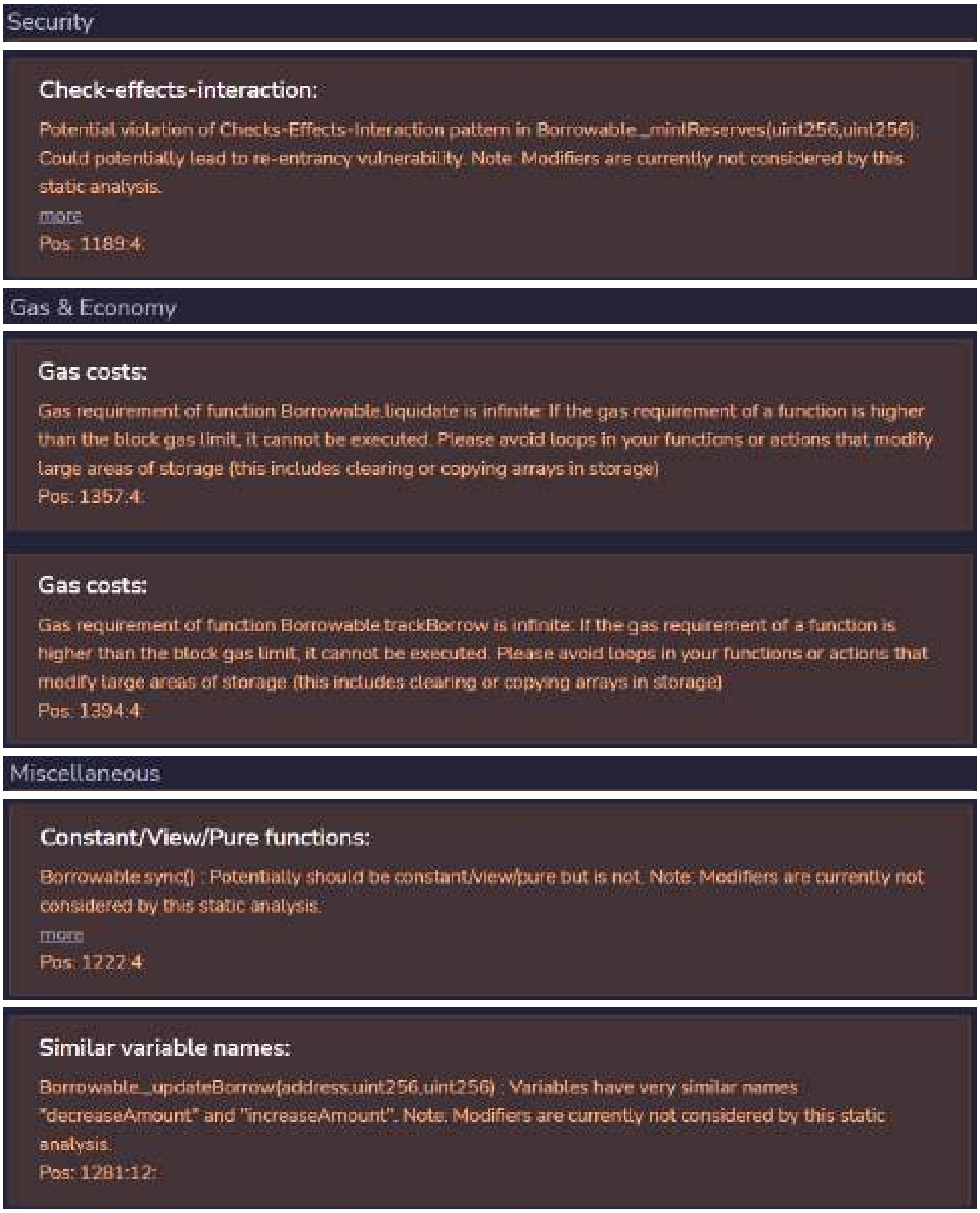
BInterestRateModel._calculateBorrowRate() : Variables have very similar names "KINK_BORROW_RATE_MAX" and "KINK_BORROW_RATE_MIN". Note: Modifiers are currently not considered by this static analysis.
Pos: 631:64:


Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 620:15:

Borrowable.sol


Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Borrowable._mintReserves(uint256,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1189:4:

Gas & Economy

Gas costs:

Gas requirement of function Borrowable.liquidate is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 1357:4:

Gas costs:

Gas requirement of function Borrowable.trackBorrow is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 1394:4:

Miscellaneous

Constant/View/Pure functions:

Borrowable.sync() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

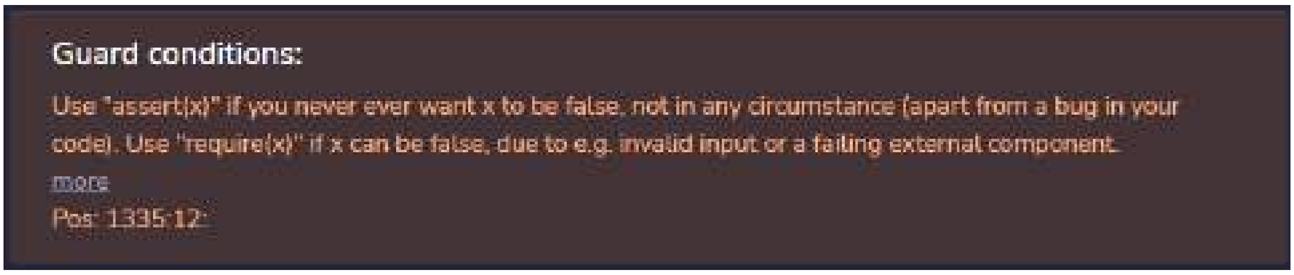
more

Pos: 1222:4:

Similar variable names:

Borrowable._updateBorrow(address,uint256,uint256) : Variables have very similar names "decreaseAmount" and "increaseAmount". Note: Modifiers are currently not considered by this static analysis.

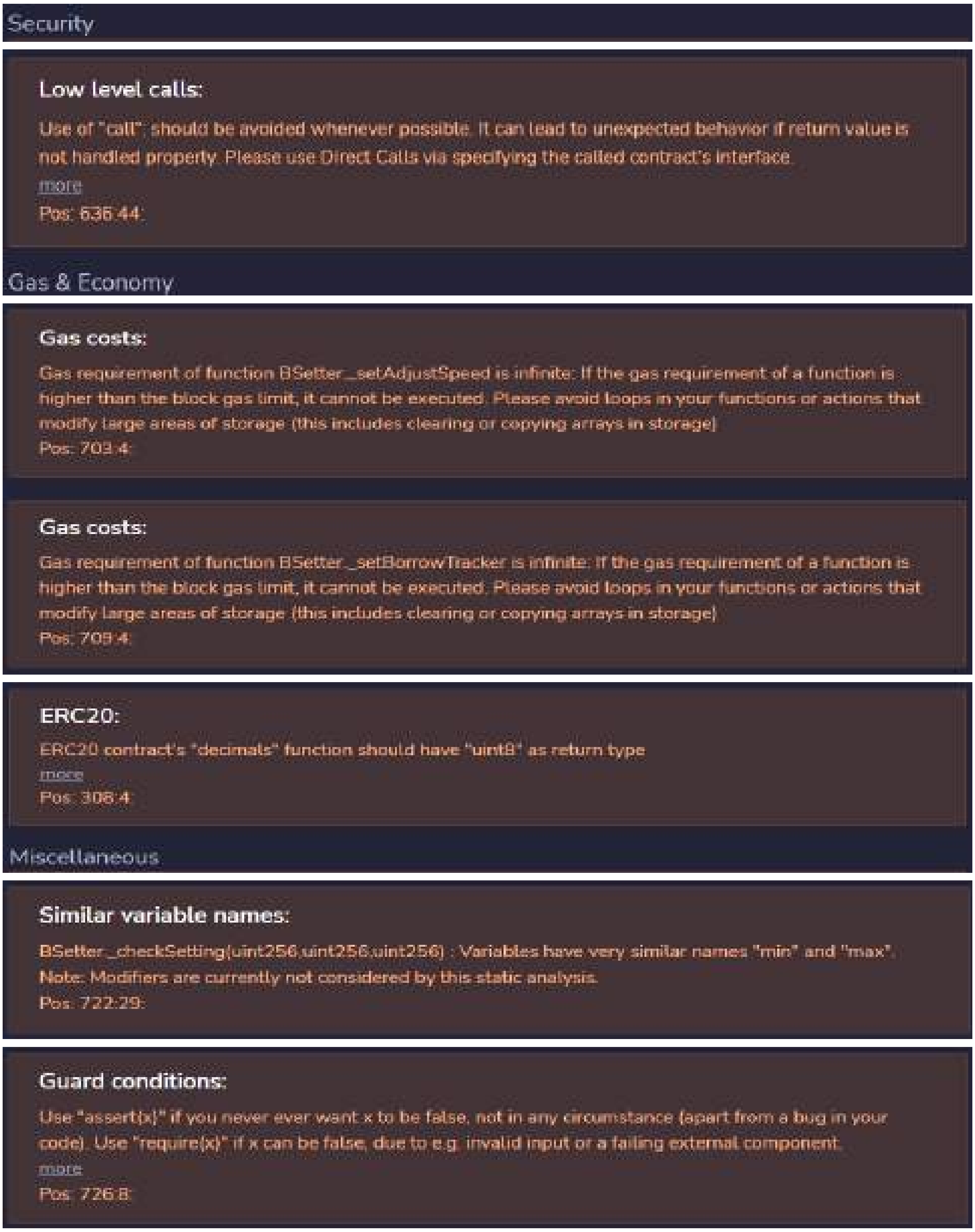
Pos: 1281:12:



Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1335:12:

BSetter.sol



Security

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 636:44:

Gas & Economy

Gas costs:

Gas requirement of function BSetter._setAdjustSpeed is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 703:4:

Gas costs:

Gas requirement of function BSetter._setBorrowTracker is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 709:4:

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 308:4:

Miscellaneous

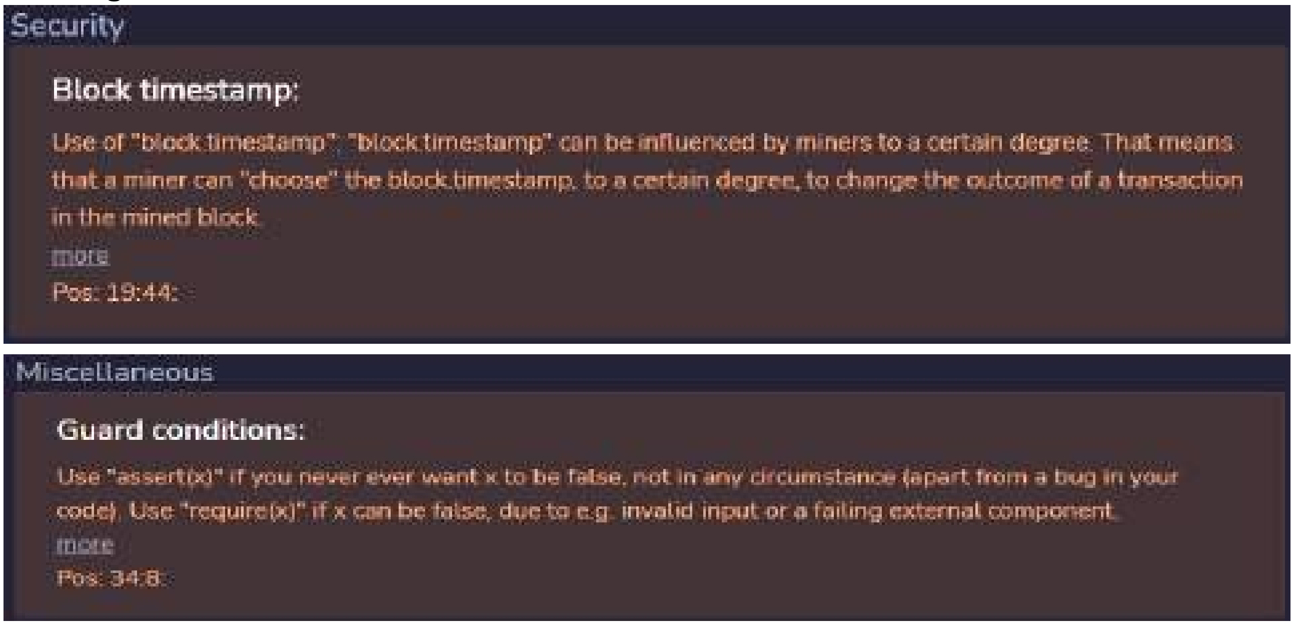
Similar variable names:

BSetter._checkSetting(uint256,uint256,uint256) : Variables have very similar names "min" and "max". Note: Modifiers are currently not considered by this static analysis.
Pos: 722:29:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 726:8:

BStorage.sol



Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 19:44:

Miscellaneous

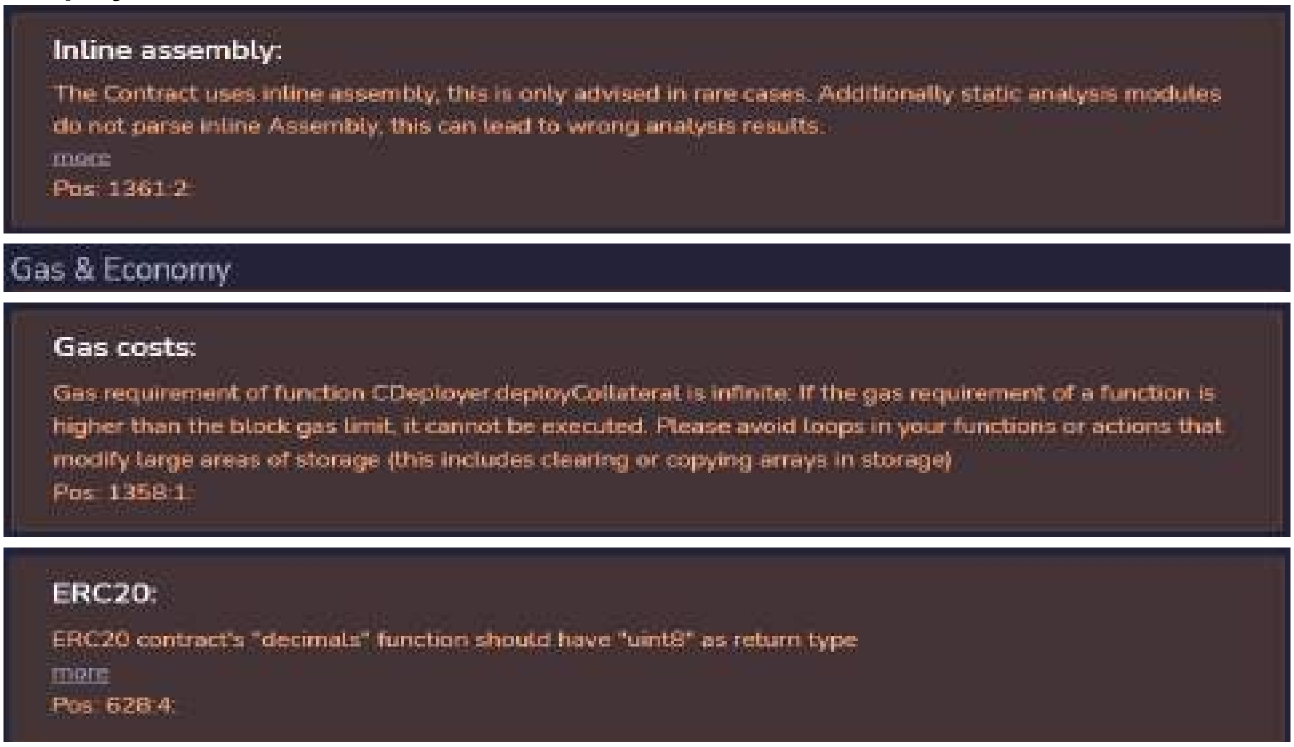
Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 34:8:

CDeployer.sol



Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 1361:2:

Gas & Economy

Gas costs:

Gas requirement of function CDeployer.deployCollateral is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 1358:1:

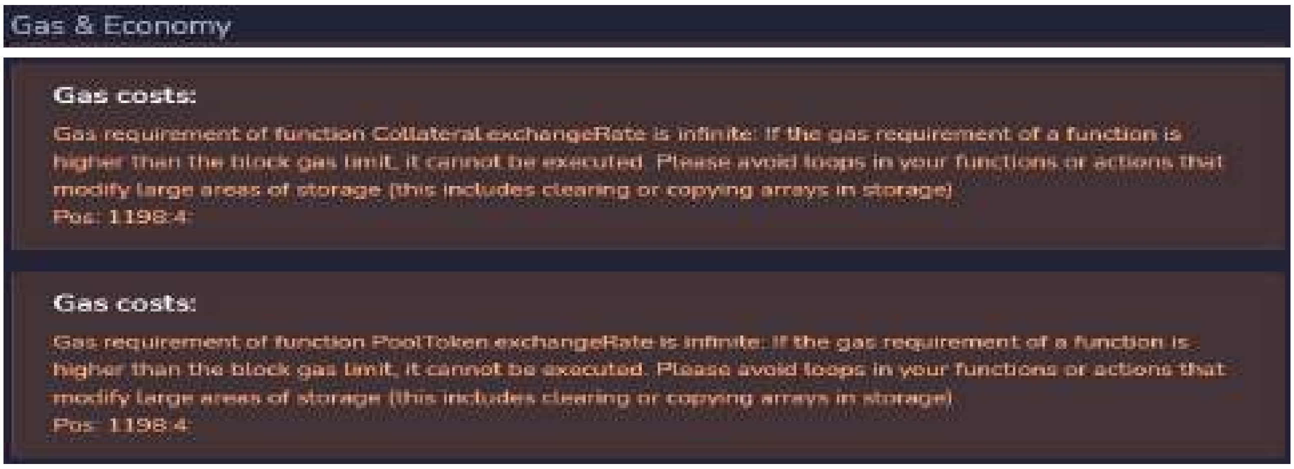
ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 628:4:

Collateral.sol



Gas & Economy

Gas costs:

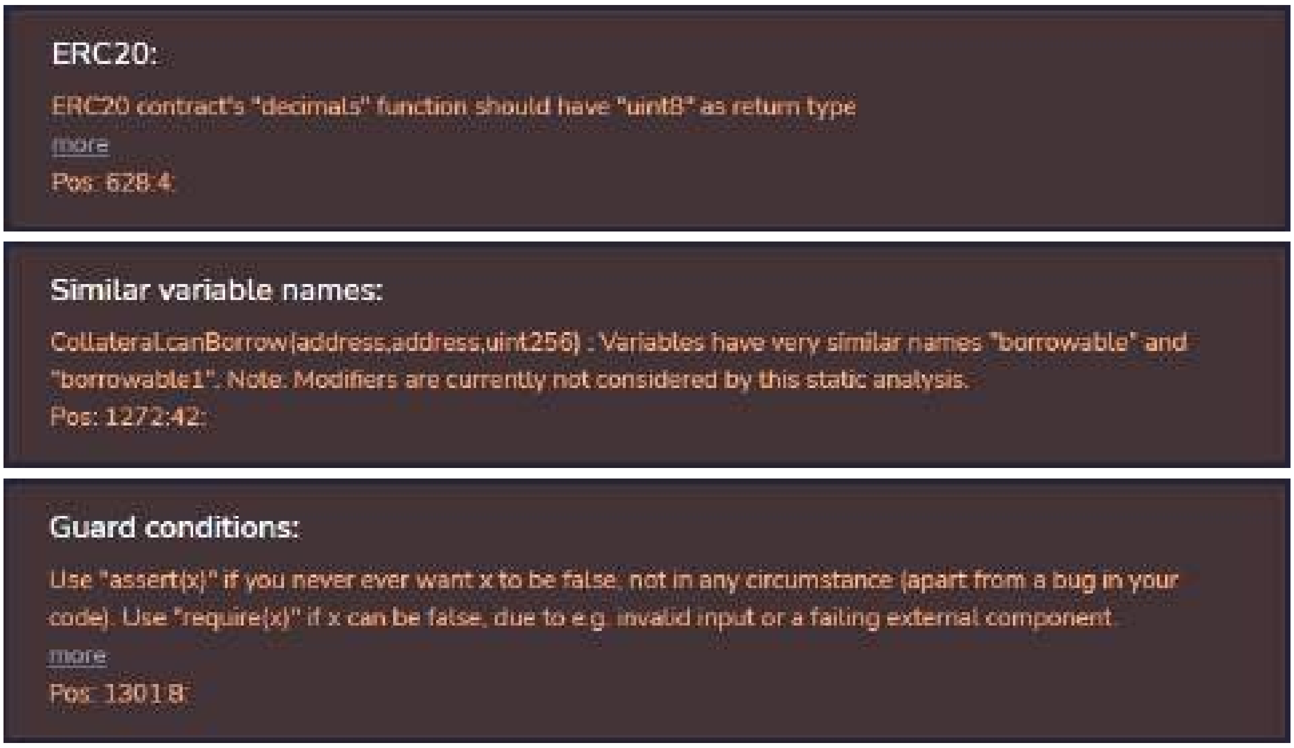
Gas requirement of function Collateral.exchangeRate is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 1198:4:

Gas costs:

Gas requirement of function PoolToken.exchangeRate is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 1198:4:



ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 628:4:

Similar variable names:

Collateral.canBorrow(address,address,uint256) : Variables have very similar names "borrowable" and "borrowable1". Note: Modifiers are currently not considered by this static analysis.

Pos: 1272:42:

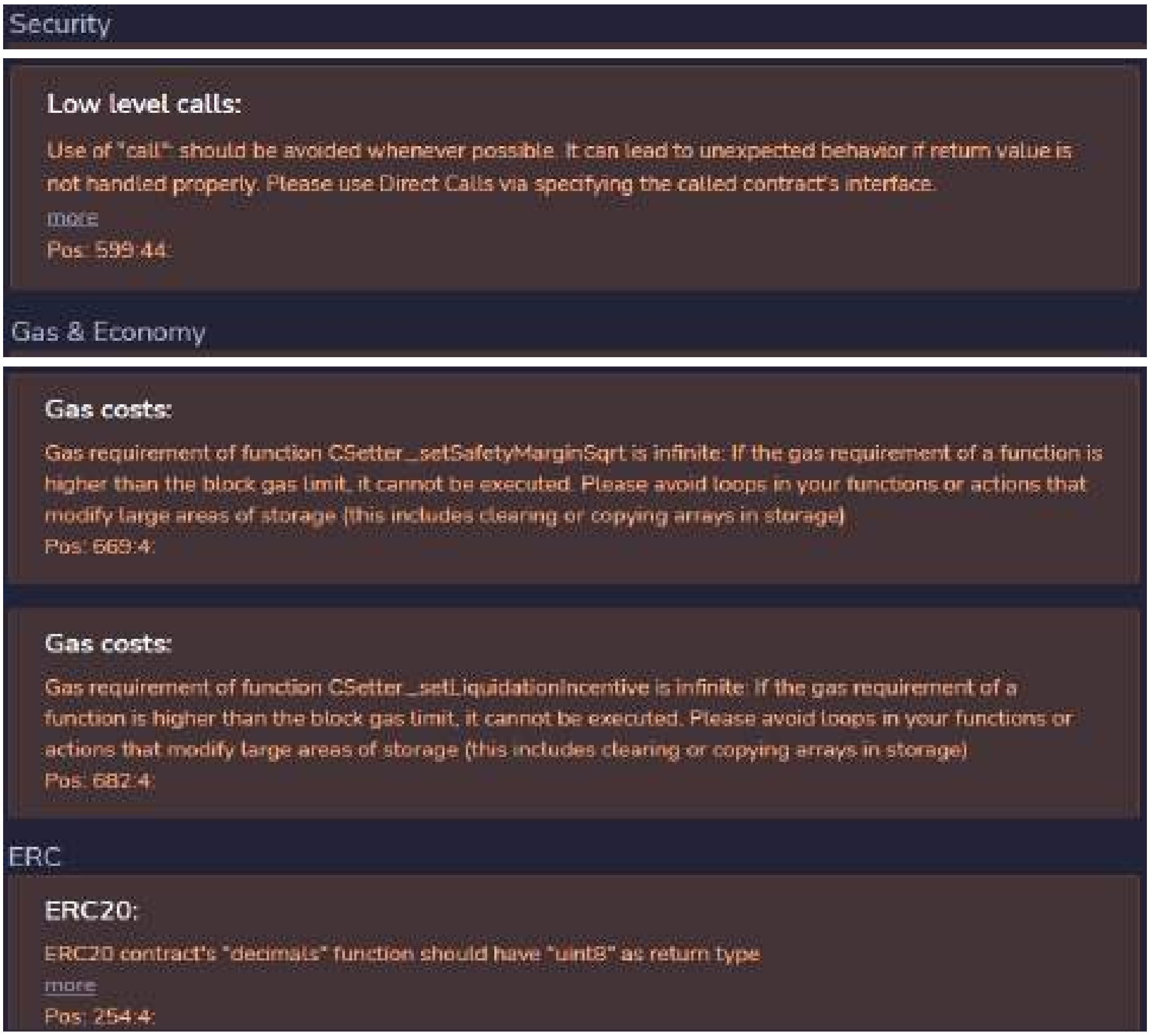
Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 1301:8:

CSetter.sol



Security

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

more

Pos: 599:44:

Gas & Economy

Gas costs:

Gas requirement of function CSetter._setSafetyMarginSqrt is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 669:4:

Gas costs:

Gas requirement of function CSetter._setLiquidationIncentive is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 682:4:

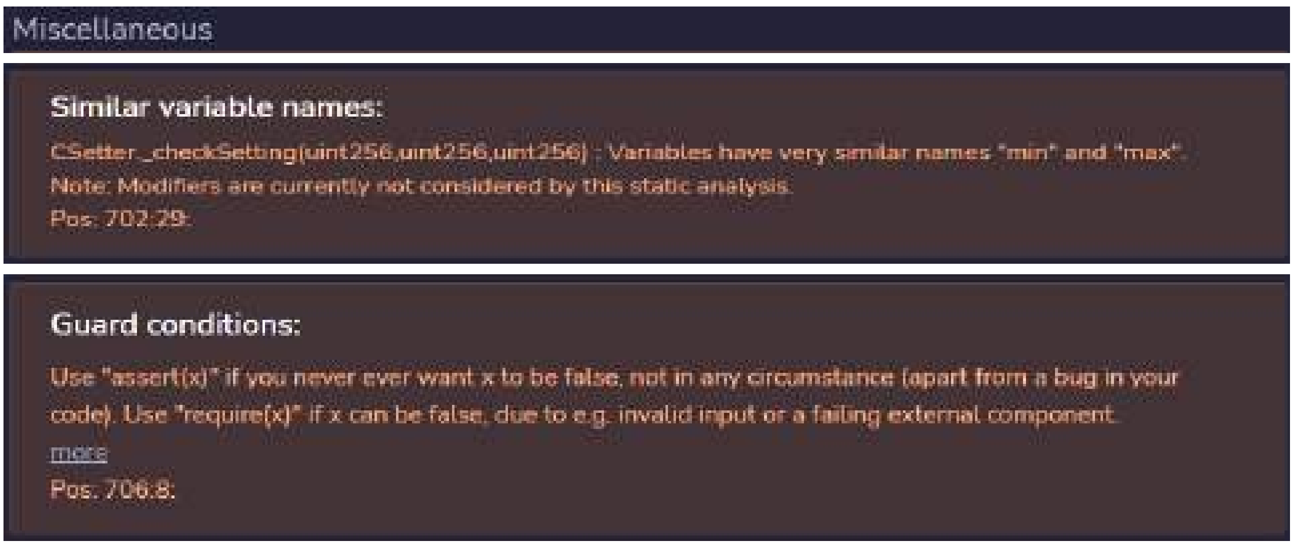
ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 254:4:



Miscellaneous

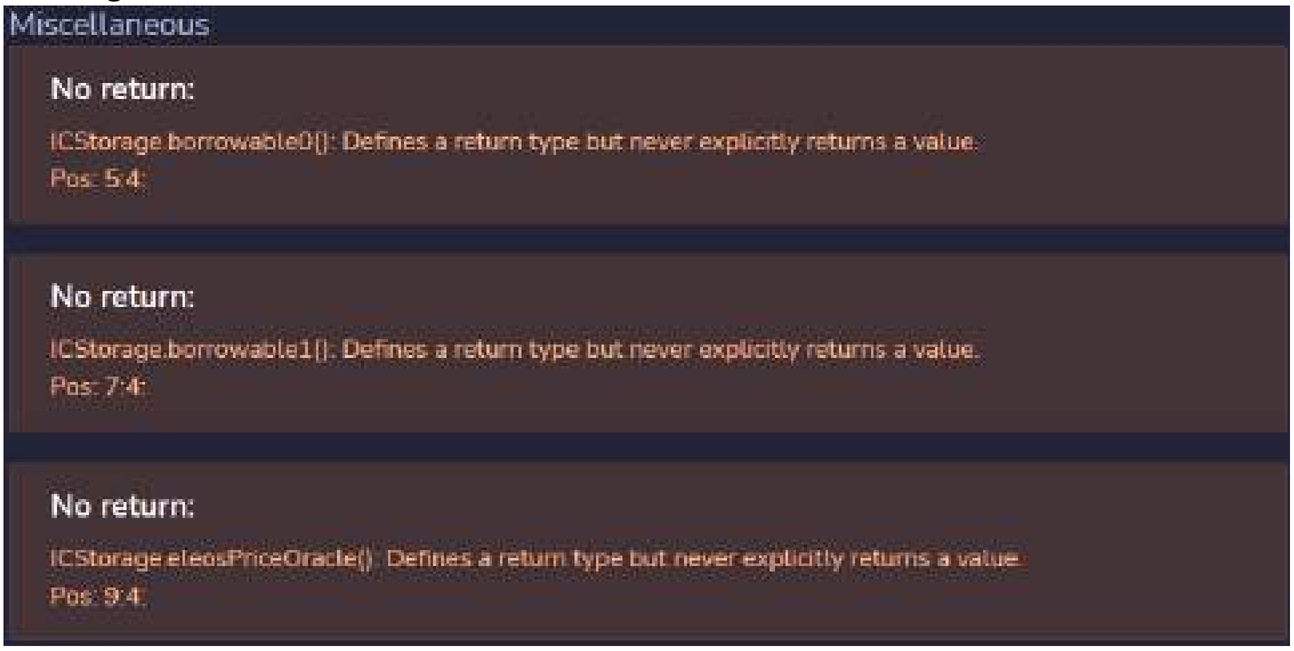
Similar variable names:

CSetter._checkSetting(uint256,uint256,uint256) : Variables have very similar names "min" and "max". Note: Modifiers are currently not considered by this static analysis.
Pos: 702:29:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 706:8:

CStorage.sol



Miscellaneous

No return:

ICStorage.borrowable0(): Defines a return type but never explicitly returns a value.
Pos: 5:4:

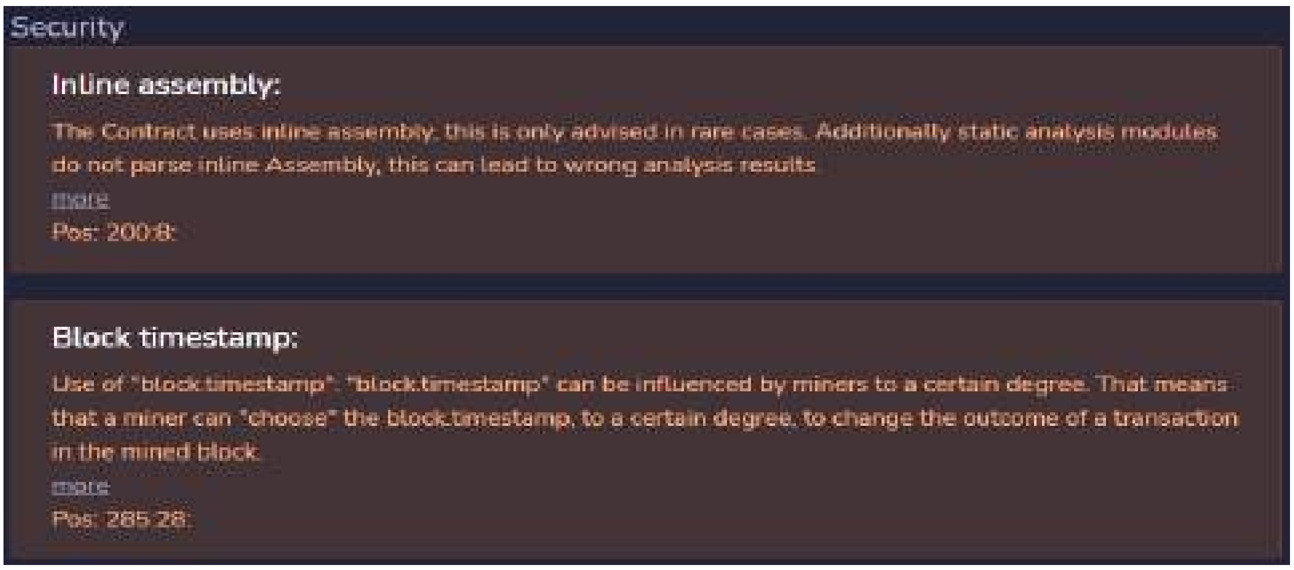
No return:

ICStorage.borrowable1(): Defines a return type but never explicitly returns a value.
Pos: 7:4:

No return:

ICStorage.eleosPriceOracle(): Defines a return type but never explicitly returns a value.
Pos: 9:4:

EleosERC20.sol



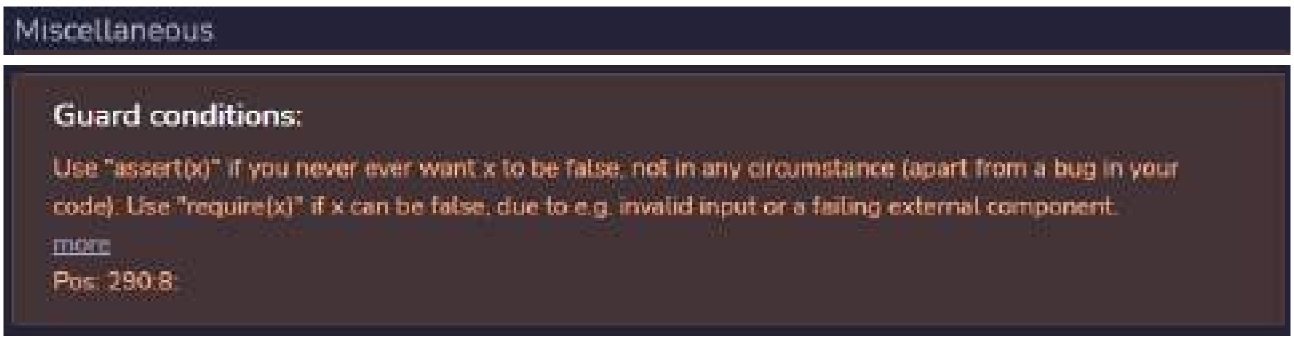
Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 200:8:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 285:28:

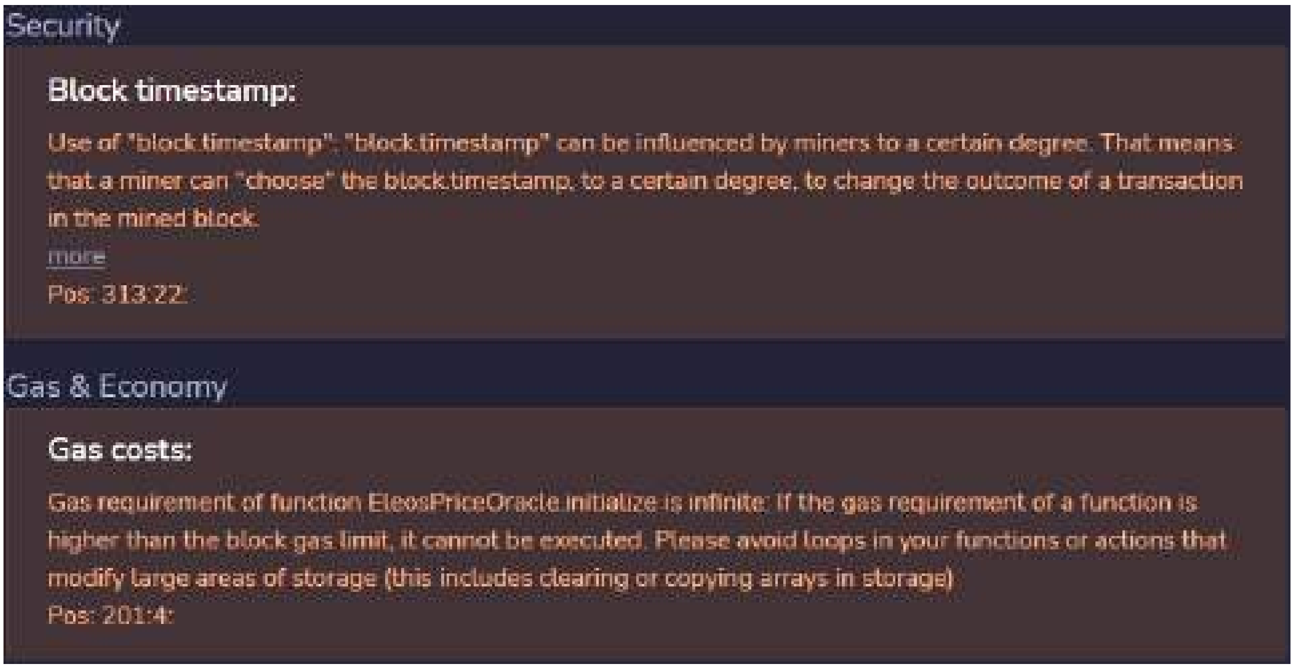


Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 290:8:

EleosPriceOracle.sol



Security

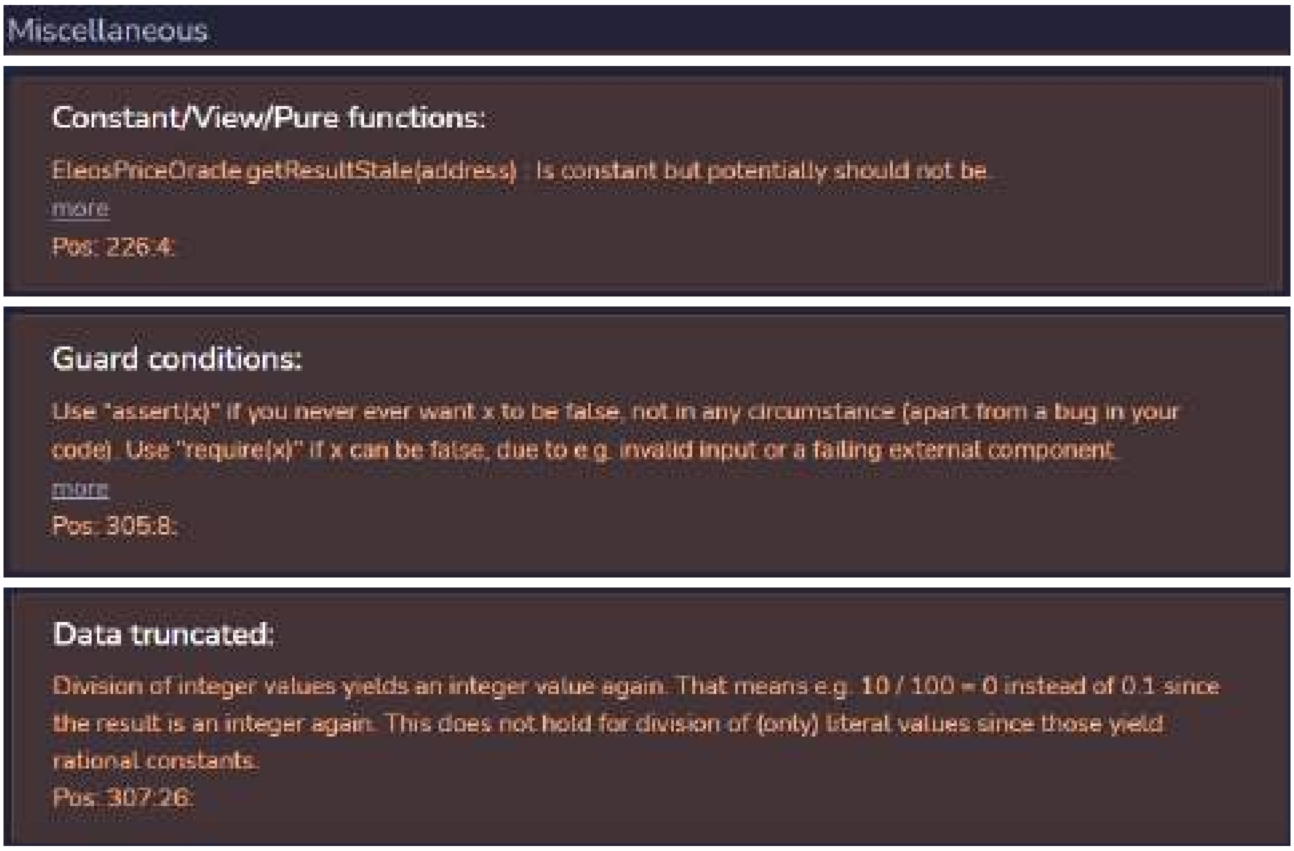
Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 313:22:

Gas & Economy

Gas costs:

Gas requirement of function EleosPriceOracle.initialize is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 201:4:



Miscellaneous

Constant/View/Pure functions:

EleosPriceOracle.getResultStale(address) : Is constant but potentially should not be.
more
Pos: 226:4:

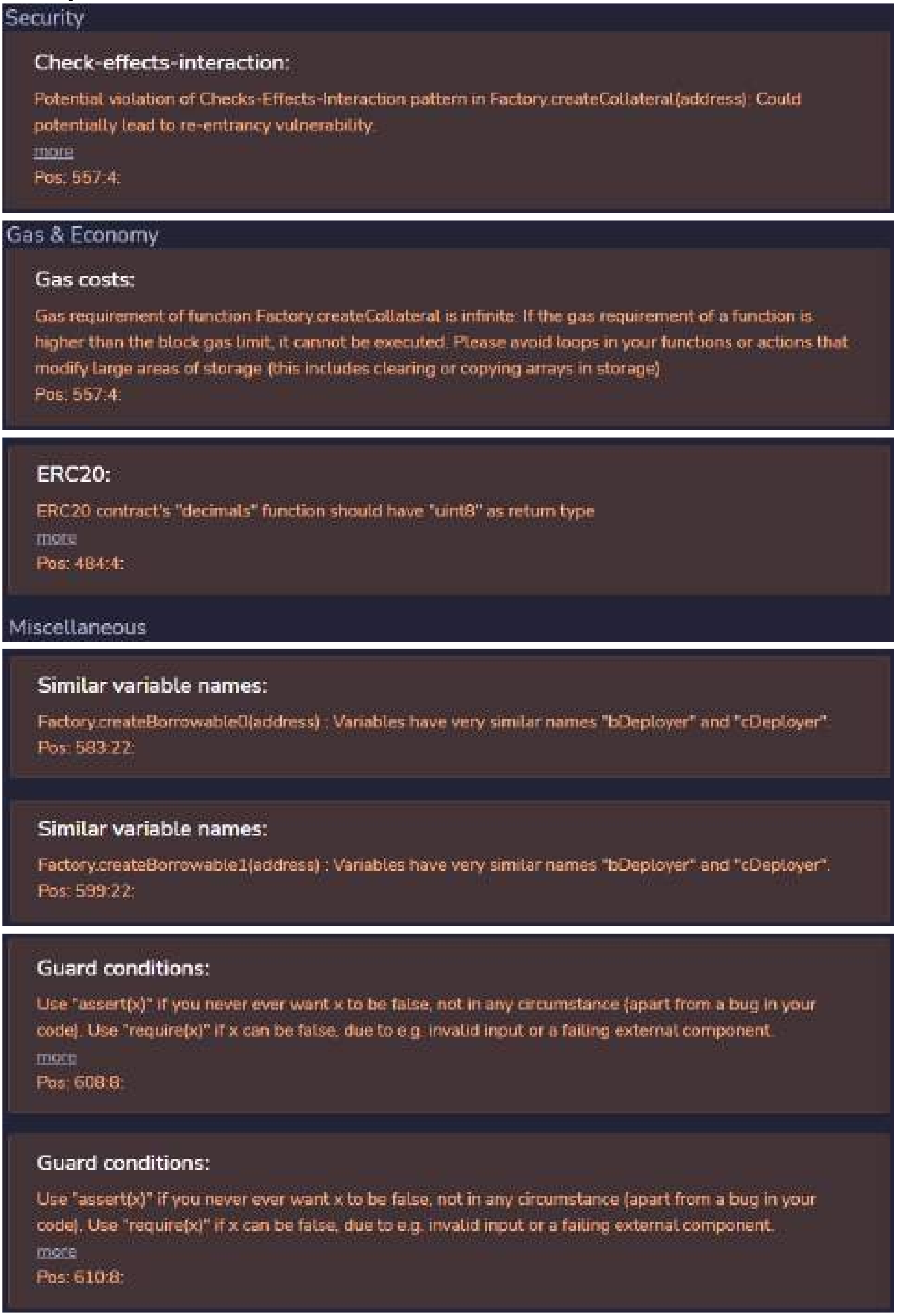
Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 305:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 307:26:

Factory.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Factory.createCollateral(address): Could potentially lead to re-entrancy vulnerability.

more

Pos: 557:4:

Gas & Economy

Gas costs:

Gas requirement of function Factory.createCollateral is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 557:4:

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 484:4:

Miscellaneous

Similar variable names:

Factory.createBorrowable0(address) : Variables have very similar names "bDeployer" and "cDeployer".

Pos: 583:22:

Similar variable names:

Factory.createBorrowable1(address) : Variables have very similar names "bDeployer" and "cDeployer".

Pos: 599:22:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 608:8:

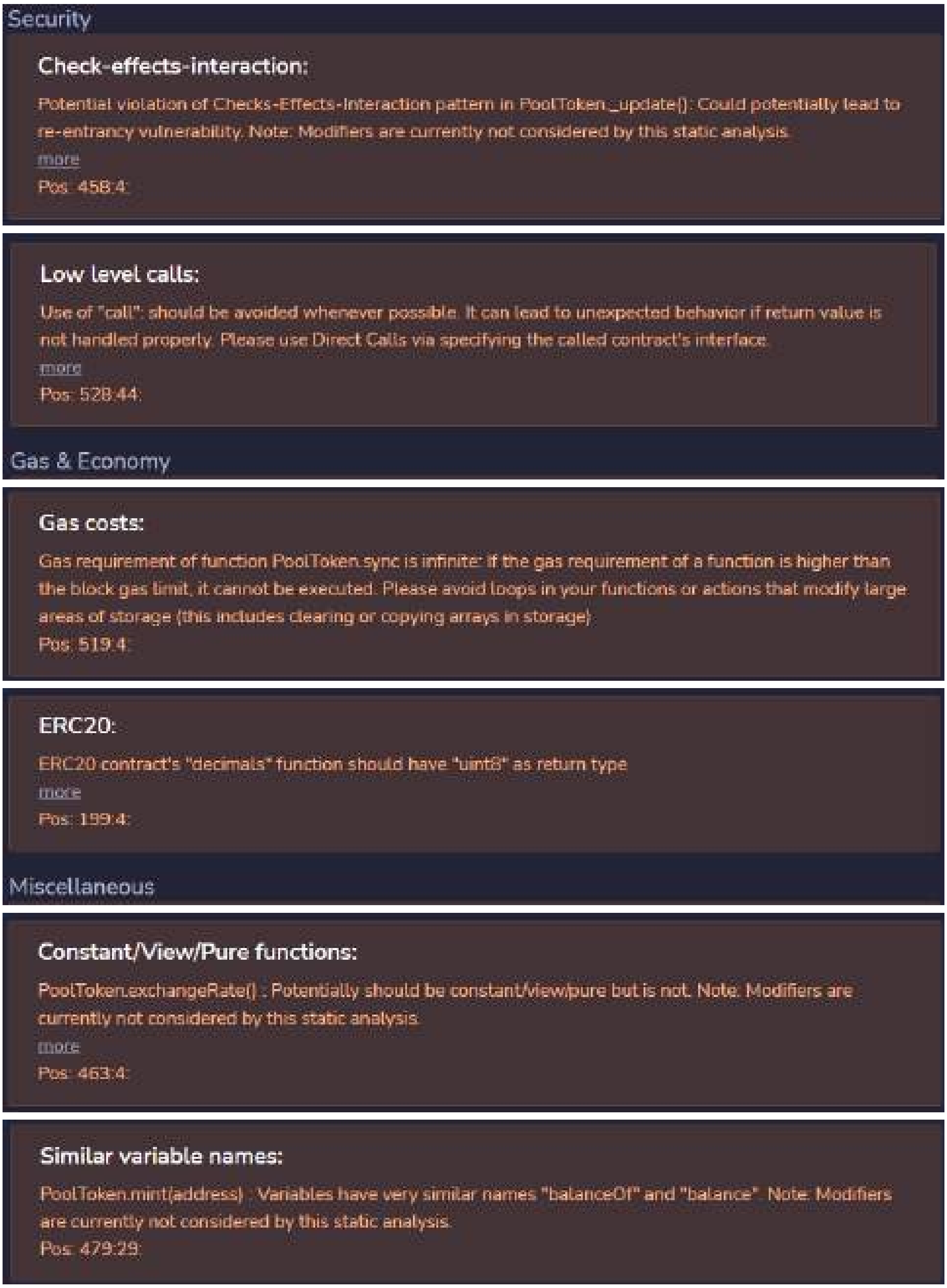
Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 610:8:

PoolToken.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in PoolToken._update(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 458:4:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 528:44:

Gas & Economy

Gas costs:

Gas requirement of function PoolToken.sync is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 519:4:

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 199:4:

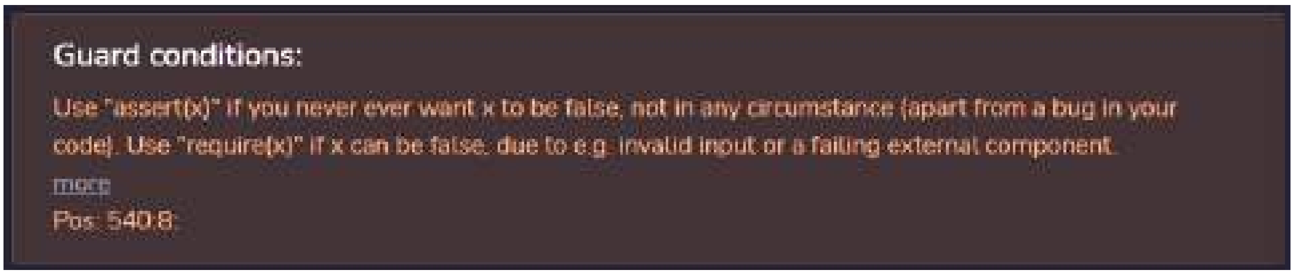
Miscellaneous

Constant/View/Pure functions:

PoolToken.exchangeRate() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 463:4:

Similar variable names:

PoolToken.mint(address) : Variables have very similar names "balanceOf" and "balance". Note: Modifiers are currently not considered by this static analysis.
Pos: 479:29:

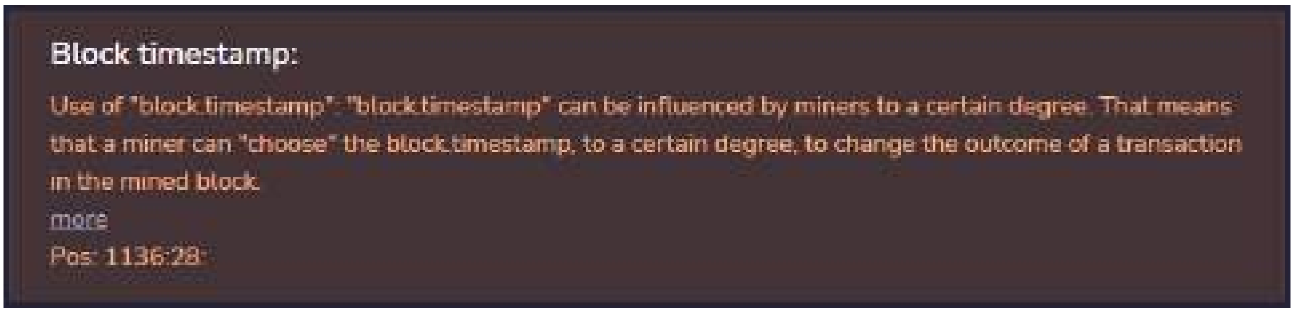

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 540:8:

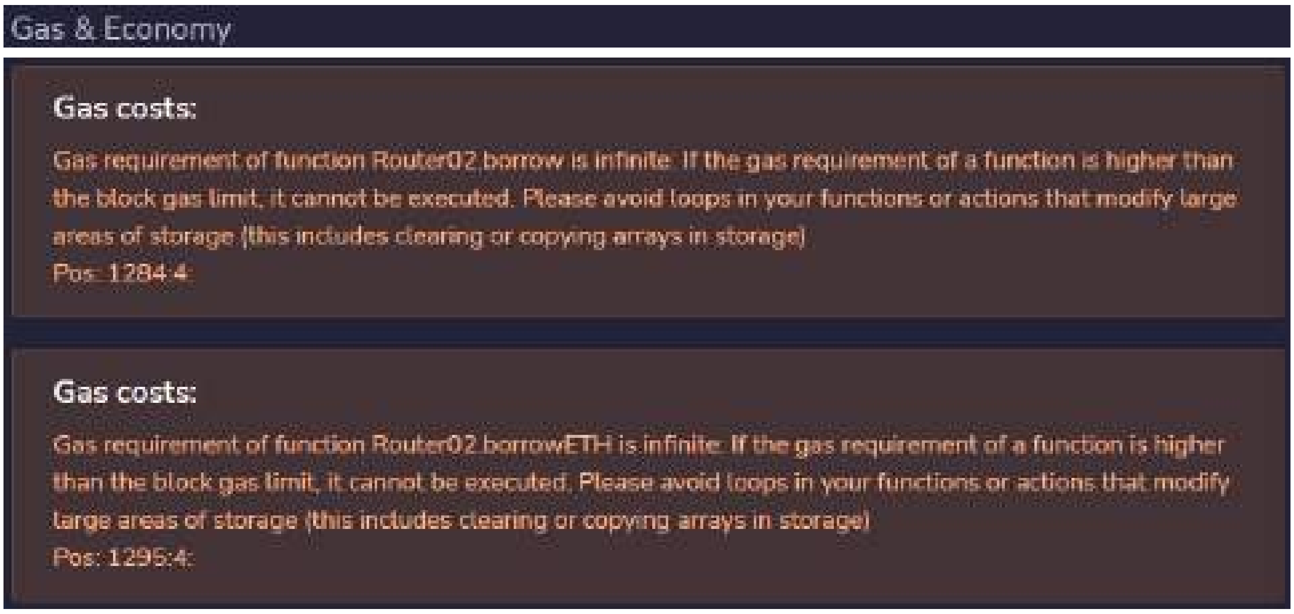
Router02.sol


Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 1136:28:


Gas & Economy

Gas costs:

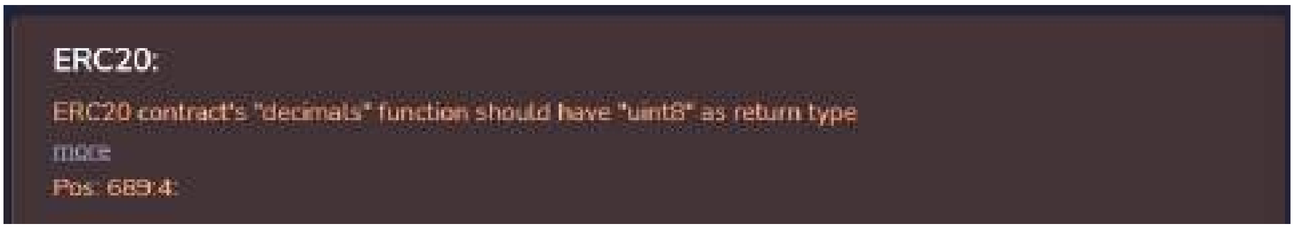
Gas requirement of function Router02.borrow is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 1284:4:

Gas costs:

Gas requirement of function Router02.borrowETH is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

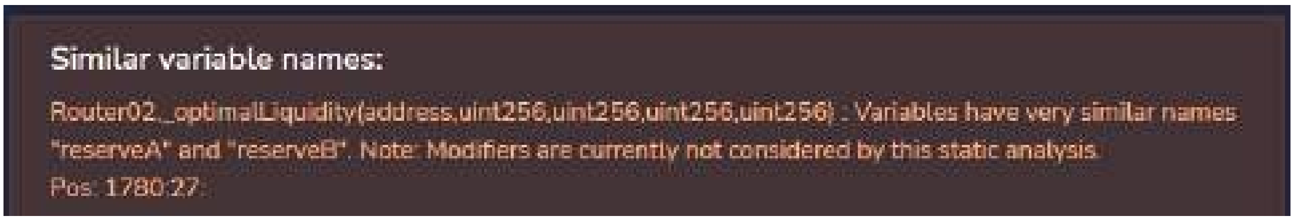
Pos: 1295:4:


ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

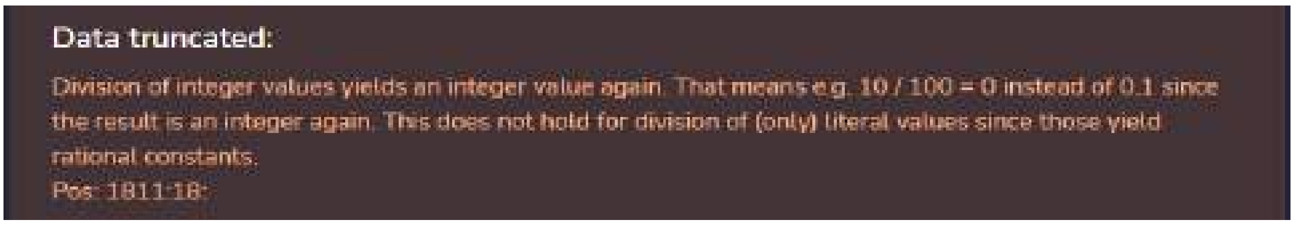
more

Pos: 689:4:


Similar variable names:

Router02._optimalLiquidity(address,uint256,uint256,uint256,uint256) : Variables have very similar names "reserveA" and "reserveB". Note: Modifiers are currently not considered by this static analysis.

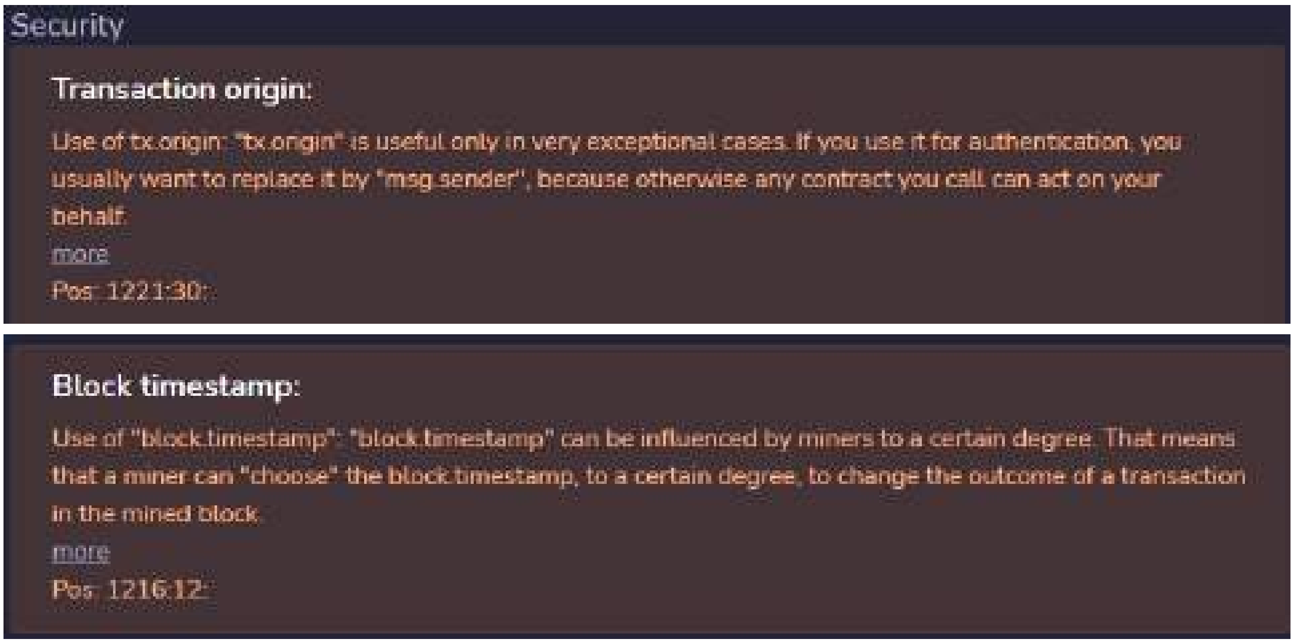
Pos: 1780:27:


Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 1811:18:

VaultToken.sol



Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.

more

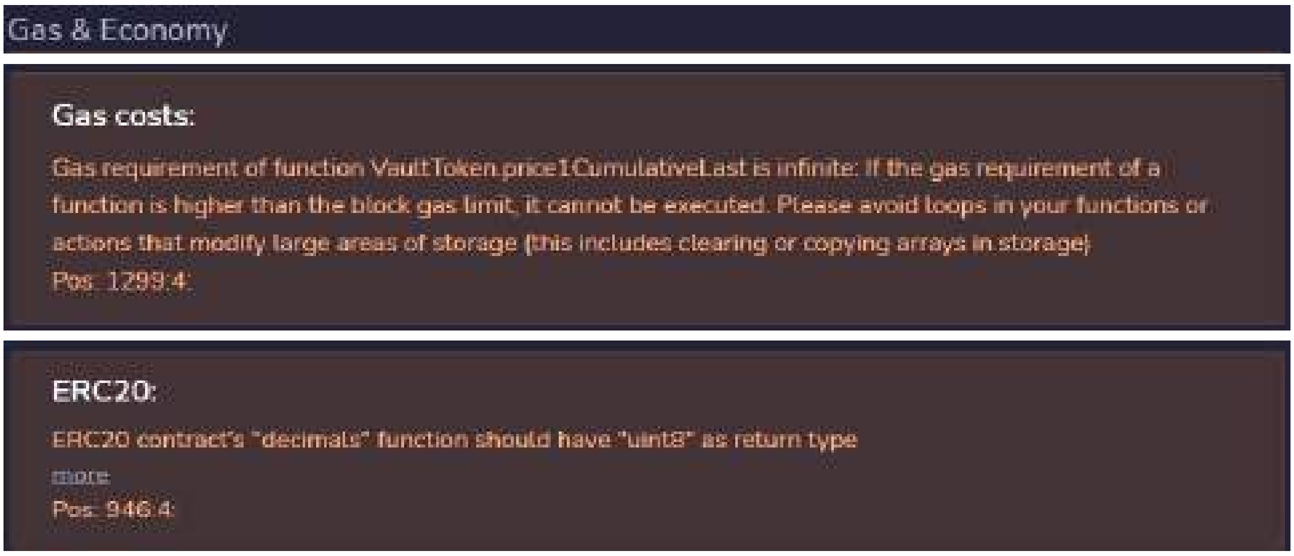
Pos: 1221:30:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 1216:12:



Gas & Economy

Gas costs:

Gas requirement of function VaultToken.price1CumulativeLast is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

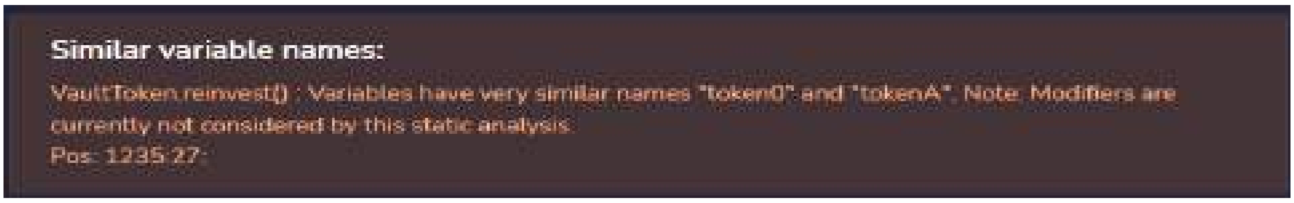
Pos: 1299:4:

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

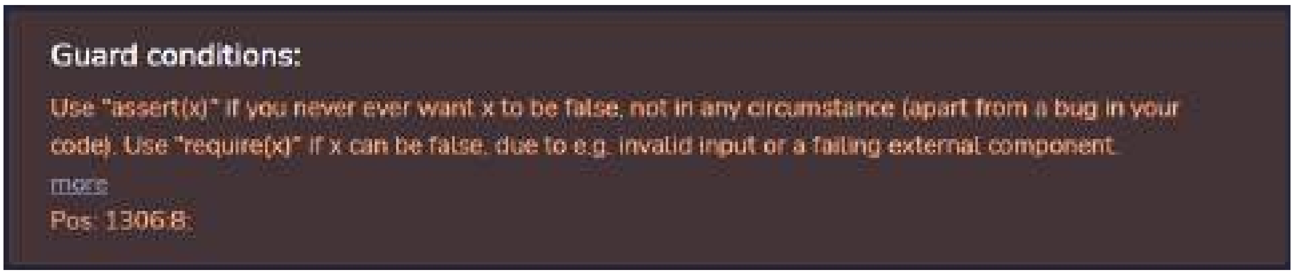
Pos: 946:4:



Similar variable names:

VaultToken.reinvest() : Variables have very similar names "token0" and "tokenA". Note: Modifiers are currently not considered by this static analysis.

Pos: 1235:27:

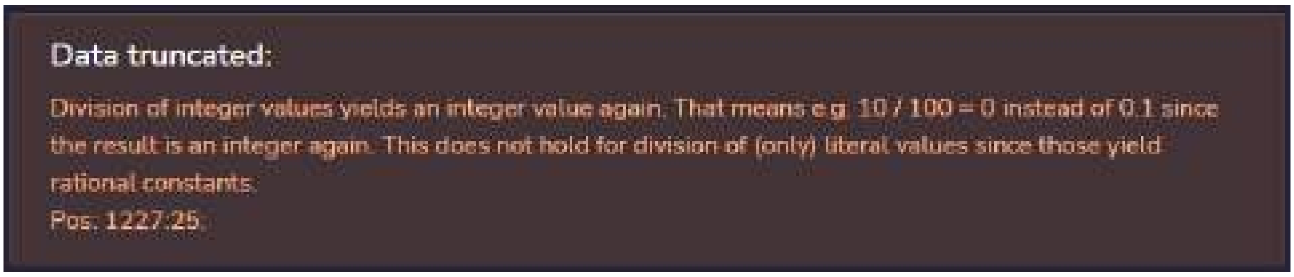


Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

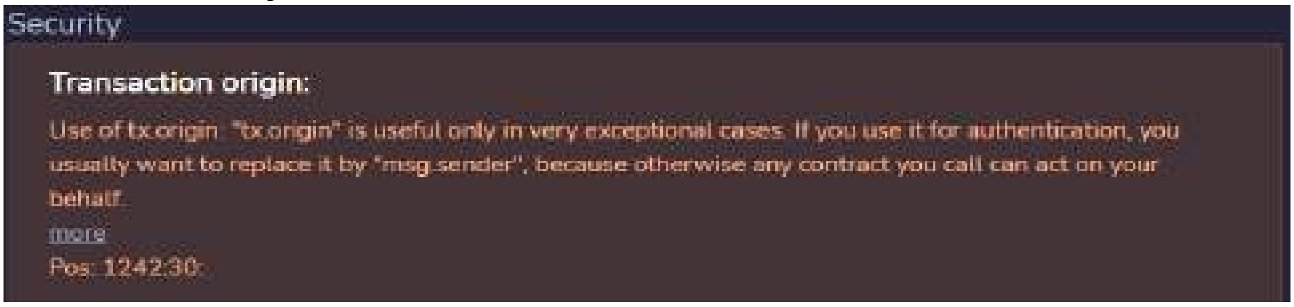
more

Pos: 1306:8:


Data truncated:

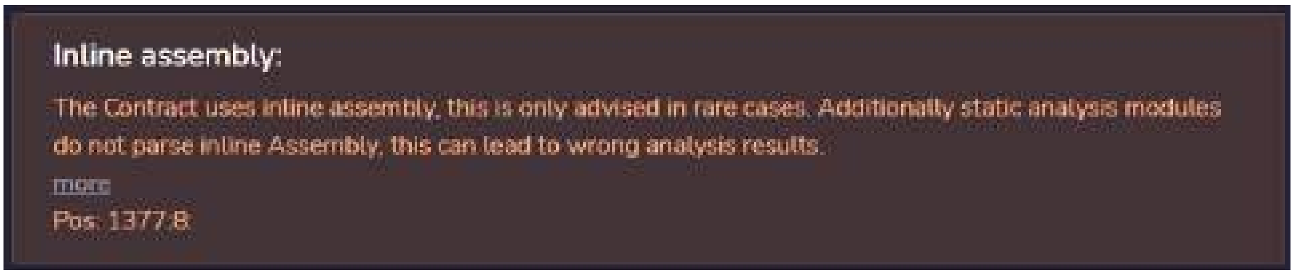
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1227:25:

VaultTokenFactory.sol

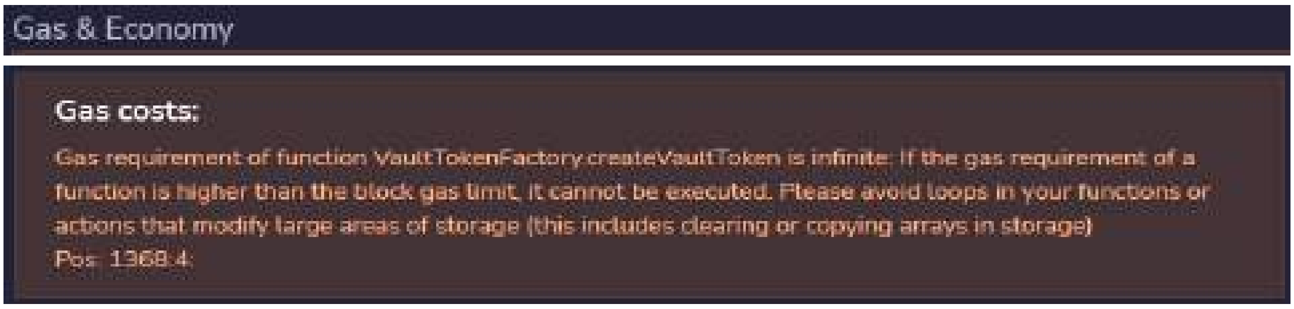

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 1242:30:

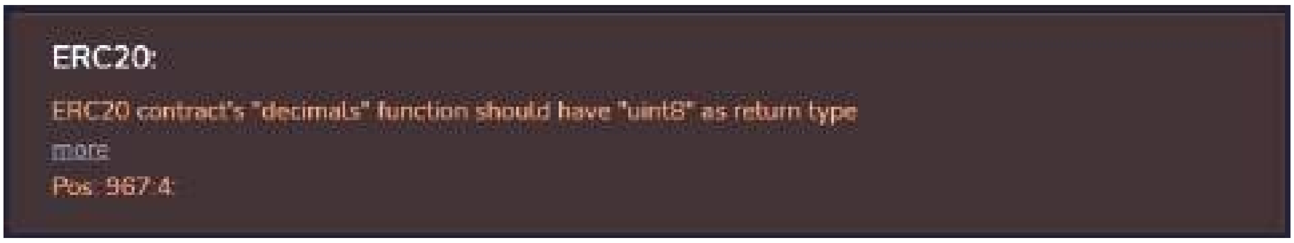

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 1377:8:

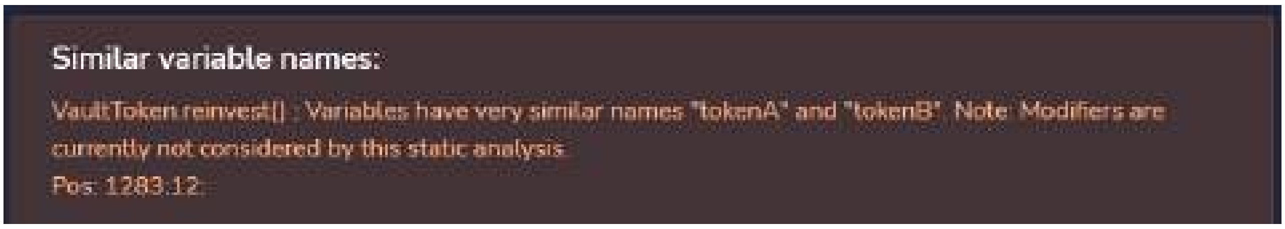

Gas & Economy

Gas costs:

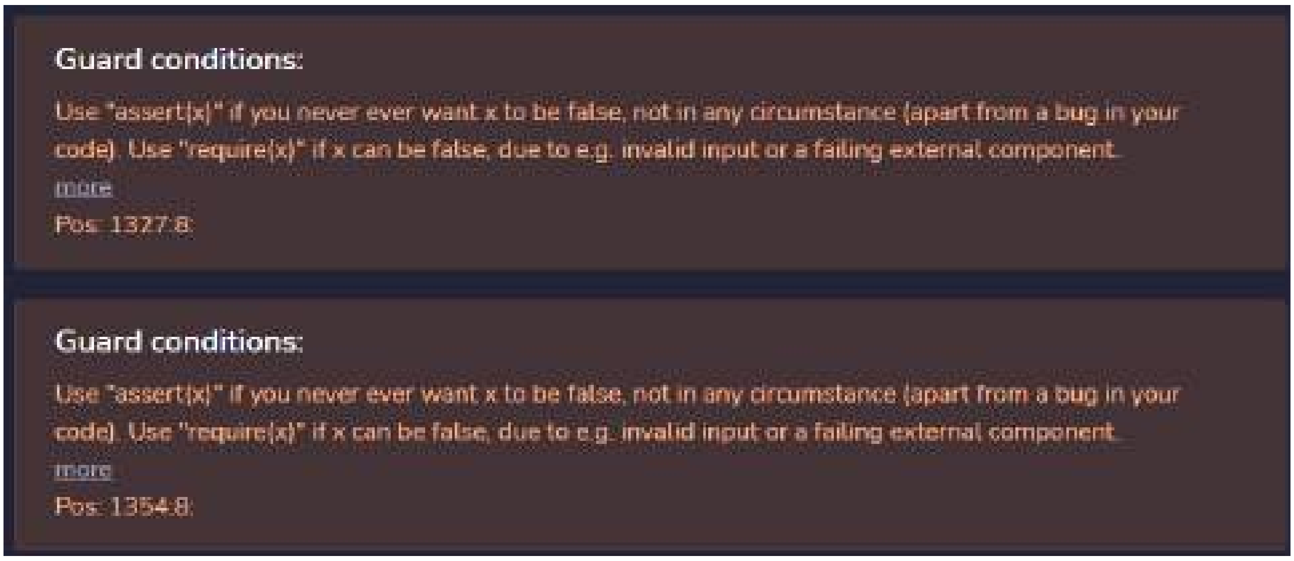
Gas requirement of function VaultTokenFactory.createVaultToken is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1368:4:


ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 967:4:


Similar variable names:

VaultToken.reinvest() : Variables have very similar names "tokenA" and "tokenB". Note: Modifiers are currently not considered by this static analysis.
Pos: 1283:12:



Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

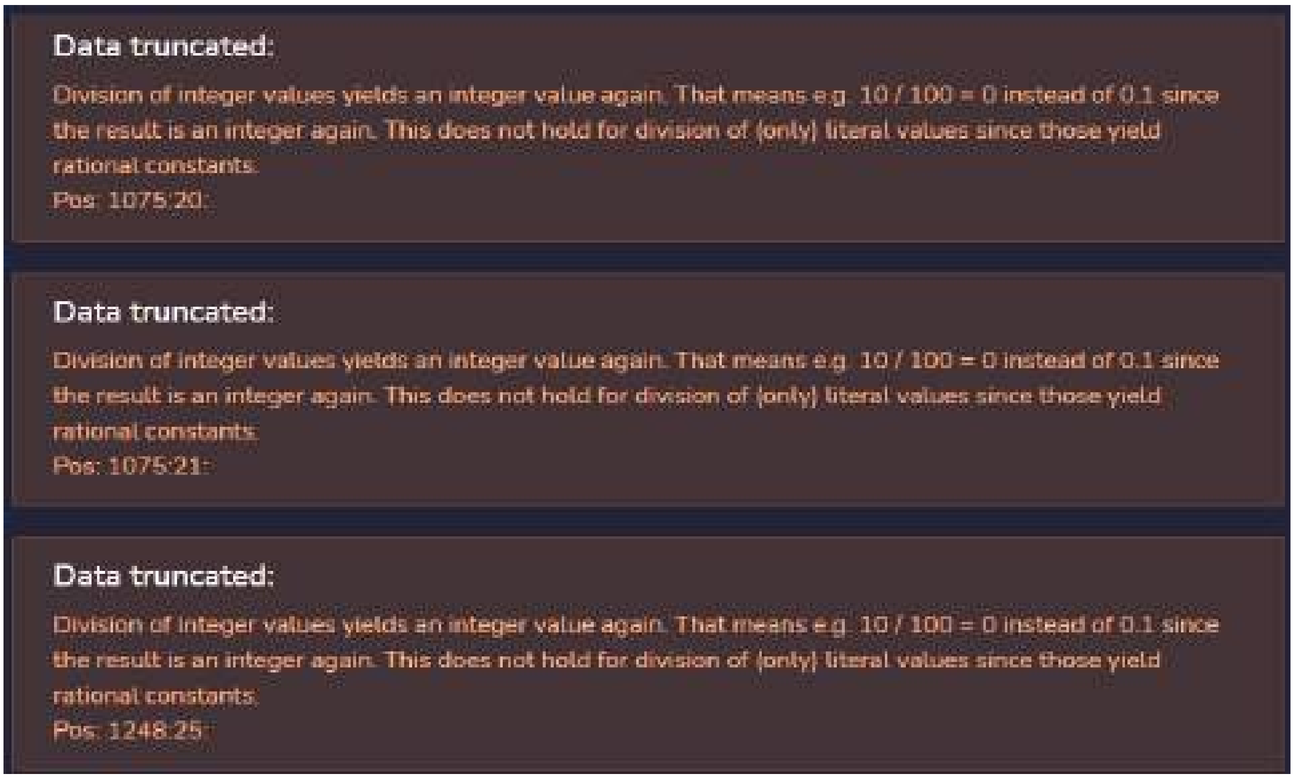
Pos: 1327:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 1354:8:



Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 1075:20:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 1075:21:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 1248:25:

Solhint Linter

BAllowance.sol

```
BAllowance.sol:3:1: Error: Compiler version =0.8.4 does not satisfy
the r semver requirement
BAllowance.sol:19:45: Error: Avoid to make time-based decisions in
your business logic
BAllowance.sol:290:5: Error: Function name must be in mixedCase
BAllowance.sol:315:20: Error: Variable name must be in mixedCase
BAllowance.sol:325:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
BAllowance.sol:325:19: Error: Code contains empty blocks
BAllowance.sol:331:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
BAllowance.sol:416:29: Error: Avoid to make time-based decisions in
your business logic
BAllowance.sol:473:31: Error: Constant name must be in capitalized
SNAKE_CASE
BAllowance.sol:550:58: Error: Code contains empty blocks
BAllowance.sol:559:45: Error: Avoid using low level calls.
```

BDeployer.sol

```
BDeployer.sol:2602:12: Error: Parse error: missing ';' at '{'
BDeployer.sol:2658:12: Error: Parse error: missing ';' at '{'
```

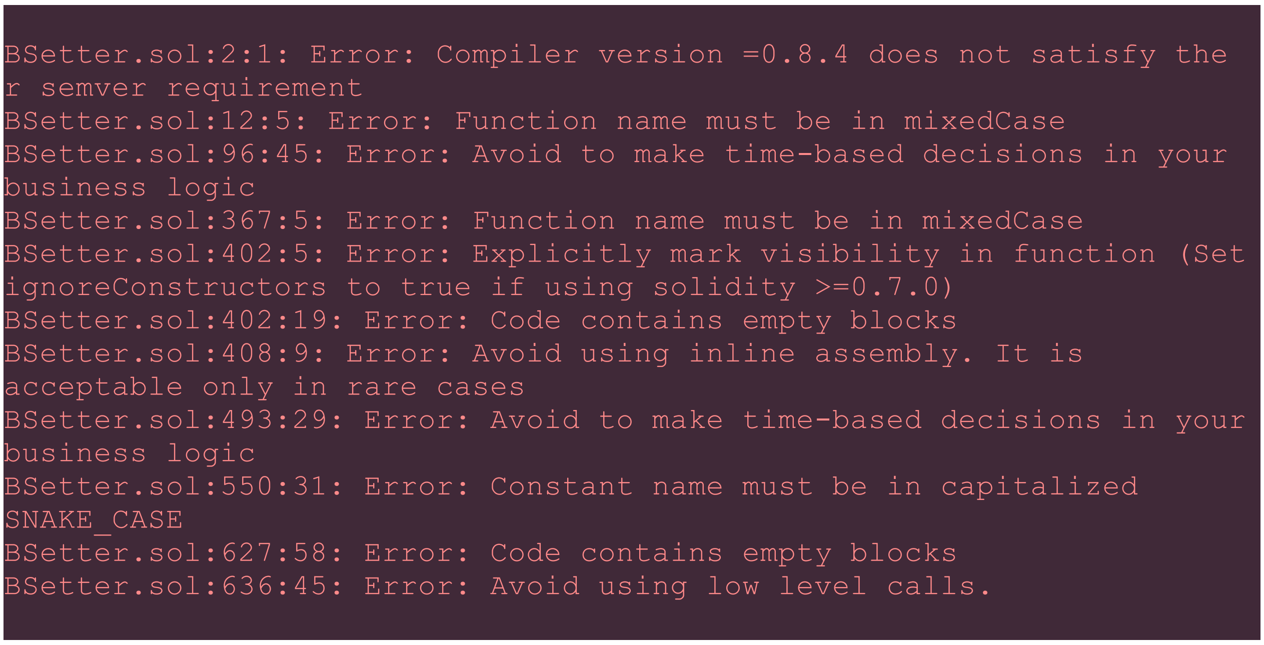
BInterestRateModel.sol

```
BInterestRateModel.sol:611:12: Error: Parse error: missing ';' at '{'
BInterestRateModel.sol:667:12: Error: Parse error: missing ';' at '{'
```

Borrowable.sol

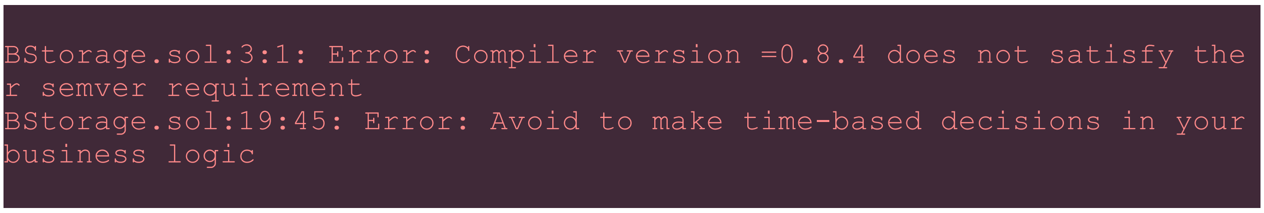
```
Borrowable.sol:1072:12: Error: Parse error: missing ';' at '{'
Borrowable.sol:1128:12: Error: Parse error: missing ';' at '{'
```

BSetter.sol



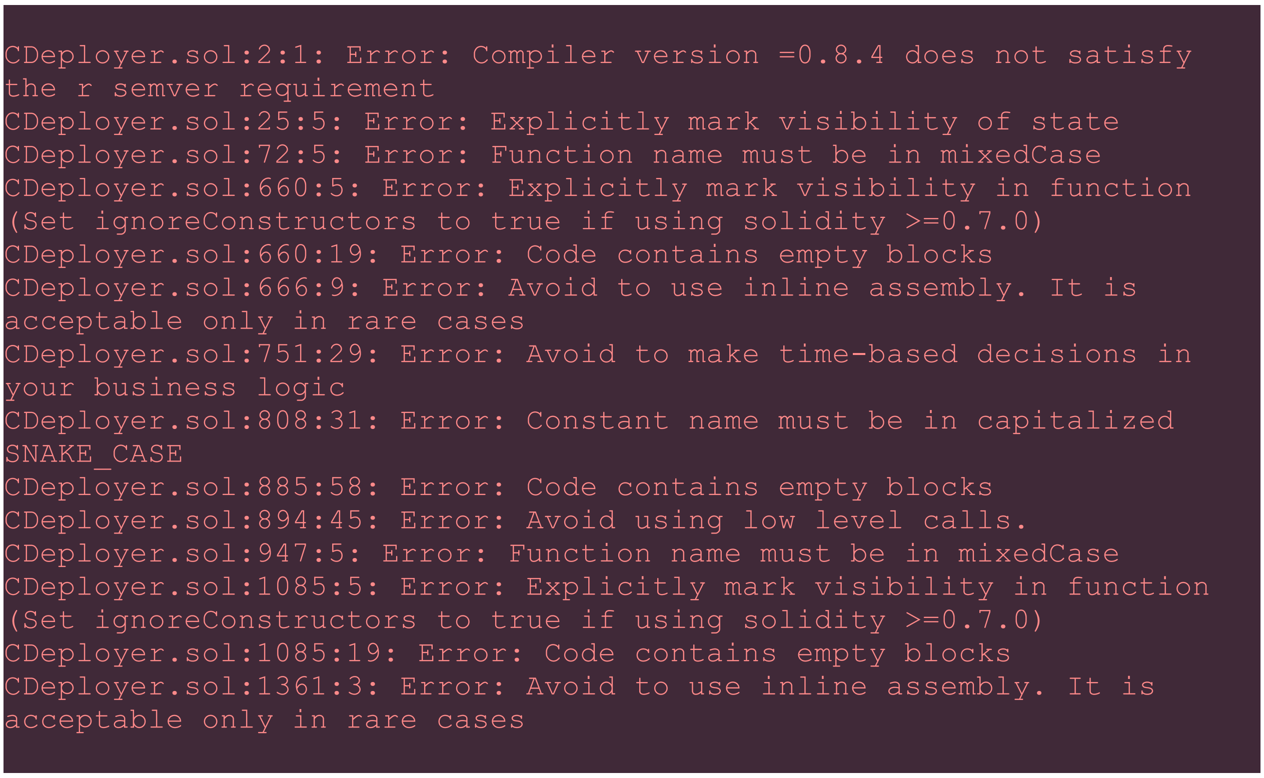
```
BSetter.sol:2:1: Error: Compiler version =0.8.4 does not satisfy the
r semver requirement
BSetter.sol:12:5: Error: Function name must be in mixedCase
BSetter.sol:96:45: Error: Avoid to make time-based decisions in your
business logic
BSetter.sol:367:5: Error: Function name must be in mixedCase
BSetter.sol:402:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
BSetter.sol:402:19: Error: Code contains empty blocks
BSetter.sol:408:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
BSetter.sol:493:29: Error: Avoid to make time-based decisions in your
business logic
BSetter.sol:550:31: Error: Constant name must be in capitalized
SNAKE_CASE
BSetter.sol:627:58: Error: Code contains empty blocks
BSetter.sol:636:45: Error: Avoid using low level calls.
```

BStorage.sol



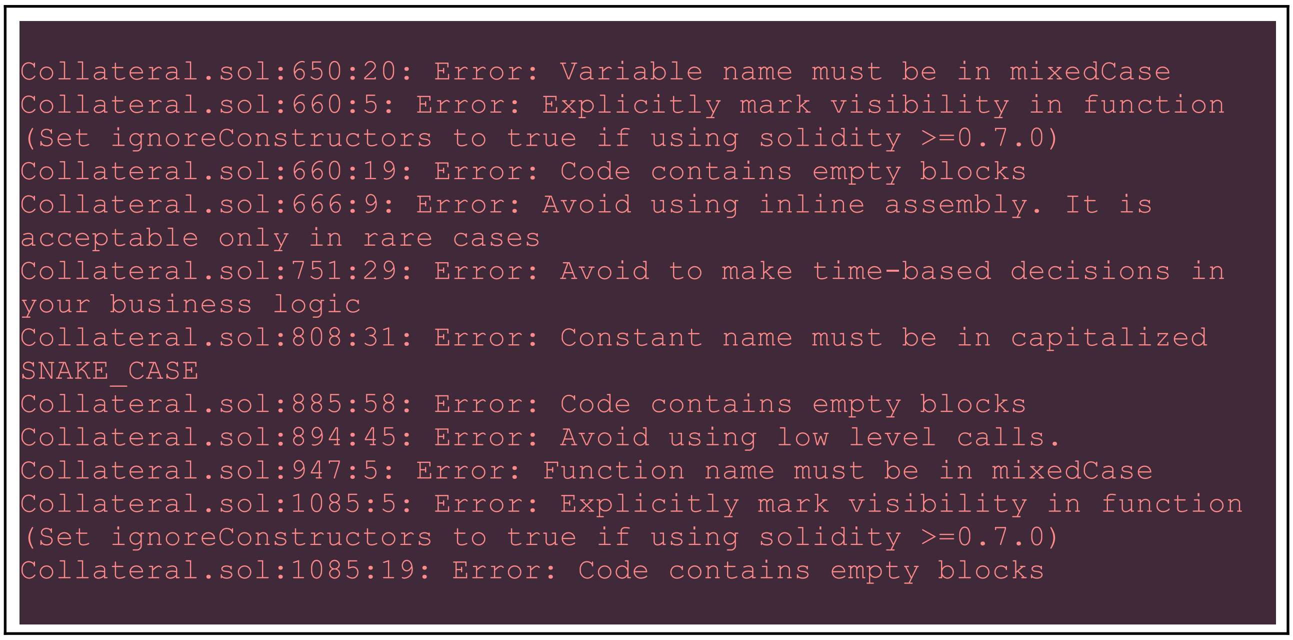
```
BStorage.sol:3:1: Error: Compiler version =0.8.4 does not satisfy the
r semver requirement
BStorage.sol:19:45: Error: Avoid to make time-based decisions in your
business logic
```

CDeployer.sol



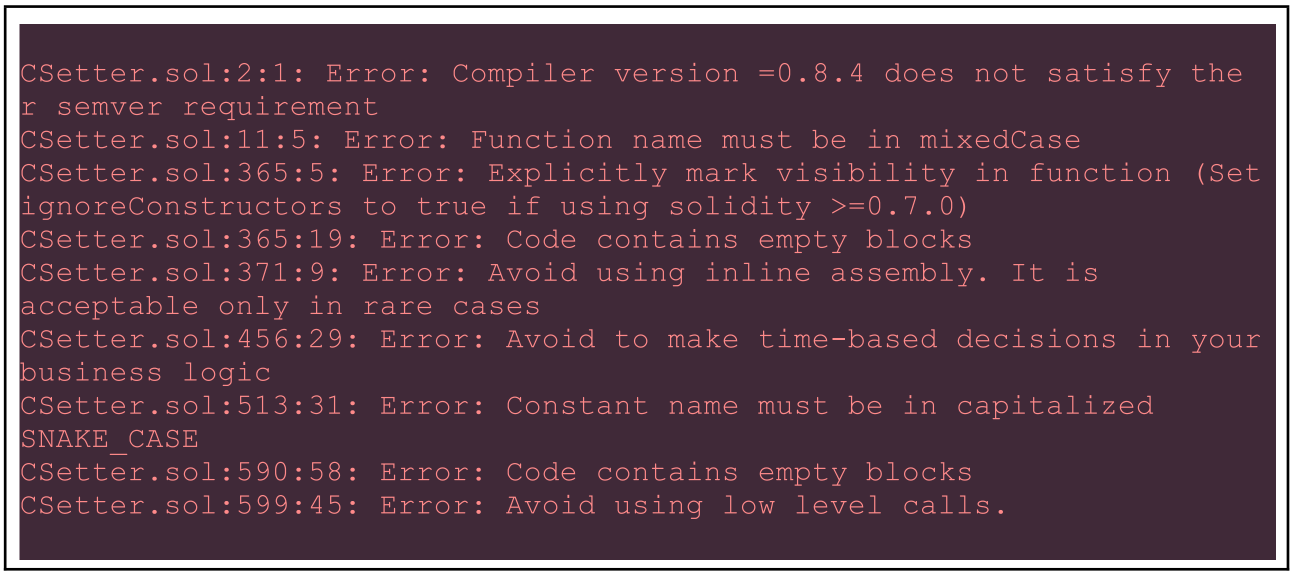
```
CDeployer.sol:2:1: Error: Compiler version =0.8.4 does not satisfy
the r semver requirement
CDeployer.sol:25:5: Error: Explicitly mark visibility of state
CDeployer.sol:72:5: Error: Function name must be in mixedCase
CDeployer.sol:660:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
CDeployer.sol:660:19: Error: Code contains empty blocks
CDeployer.sol:666:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
CDeployer.sol:751:29: Error: Avoid to make time-based decisions in
your business logic
CDeployer.sol:808:31: Error: Constant name must be in capitalized
SNAKE_CASE
CDeployer.sol:885:58: Error: Code contains empty blocks
CDeployer.sol:894:45: Error: Avoid using low level calls.
CDeployer.sol:947:5: Error: Function name must be in mixedCase
CDeployer.sol:1085:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
CDeployer.sol:1085:19: Error: Code contains empty blocks
CDeployer.sol:1361:3: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
```

Collateral.sol



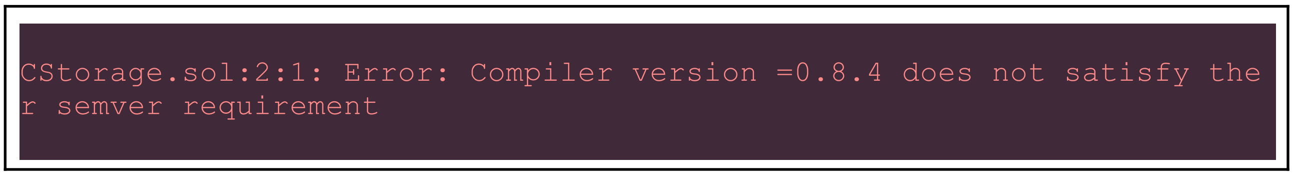
```
Collateral.sol:650:20: Error: Variable name must be in mixedCase
Collateral.sol:660:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
Collateral.sol:660:19: Error: Code contains empty blocks
Collateral.sol:666:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Collateral.sol:751:29: Error: Avoid to make time-based decisions in
your business logic
Collateral.sol:808:31: Error: Constant name must be in capitalized
SNAKE_CASE
Collateral.sol:885:58: Error: Code contains empty blocks
Collateral.sol:894:45: Error: Avoid using low level calls.
Collateral.sol:947:5: Error: Function name must be in mixedCase
Collateral.sol:1085:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
Collateral.sol:1085:19: Error: Code contains empty blocks
```

CSetter.sol



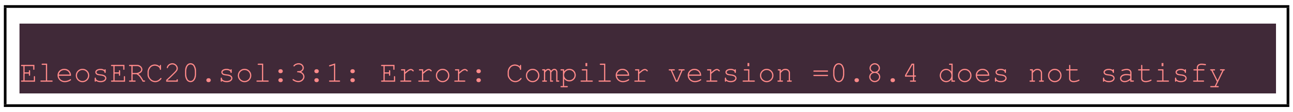
```
CSetter.sol:2:1: Error: Compiler version =0.8.4 does not satisfy the
r semver requirement
CSetter.sol:11:5: Error: Function name must be in mixedCase
CSetter.sol:365:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
CSetter.sol:365:19: Error: Code contains empty blocks
CSetter.sol:371:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
CSetter.sol:456:29: Error: Avoid to make time-based decisions in your
business logic
CSetter.sol:513:31: Error: Constant name must be in capitalized
SNAKE_CASE
CSetter.sol:590:58: Error: Code contains empty blocks
CSetter.sol:599:45: Error: Avoid using low level calls.
```

CStorage.sol

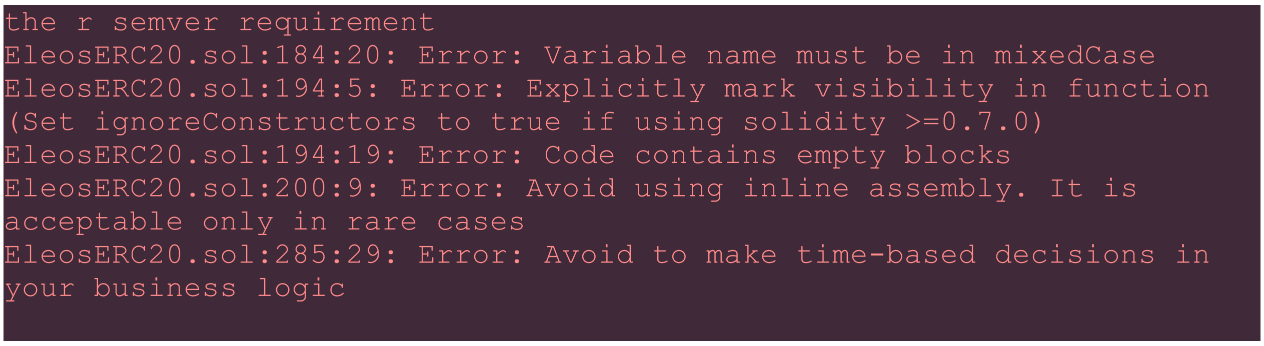


```
CStorage.sol:2:1: Error: Compiler version =0.8.4 does not satisfy the
r semver requirement
```

EleosERC20.sol

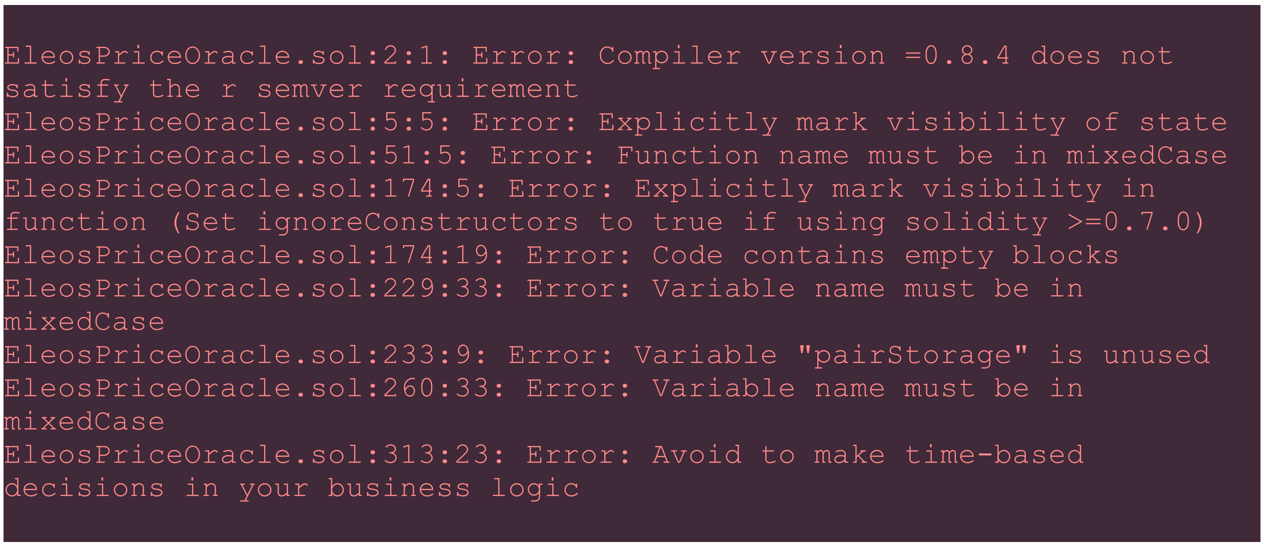


```
EleosERC20.sol:3:1: Error: Compiler version =0.8.4 does not satisfy
```



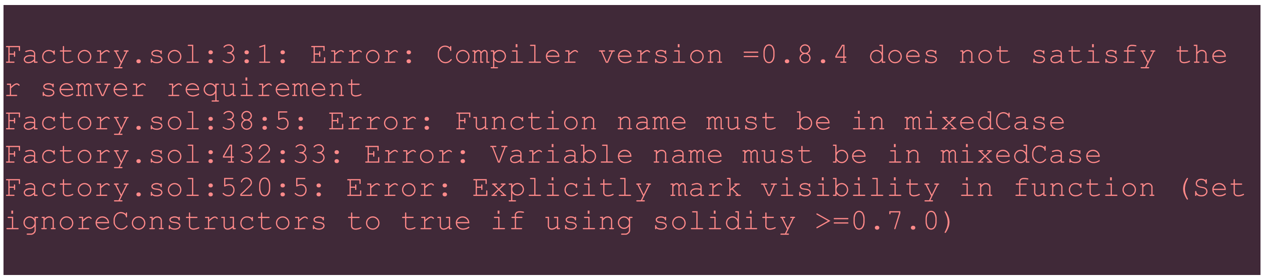
```
the r semver requirement
EleosERC20.sol:184:20: Error: Variable name must be in mixedCase
EleosERC20.sol:194:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
EleosERC20.sol:194:19: Error: Code contains empty blocks
EleosERC20.sol:200:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
EleosERC20.sol:285:29: Error: Avoid to make time-based decisions in
your business logic
```

EleosPriceOracle.sol



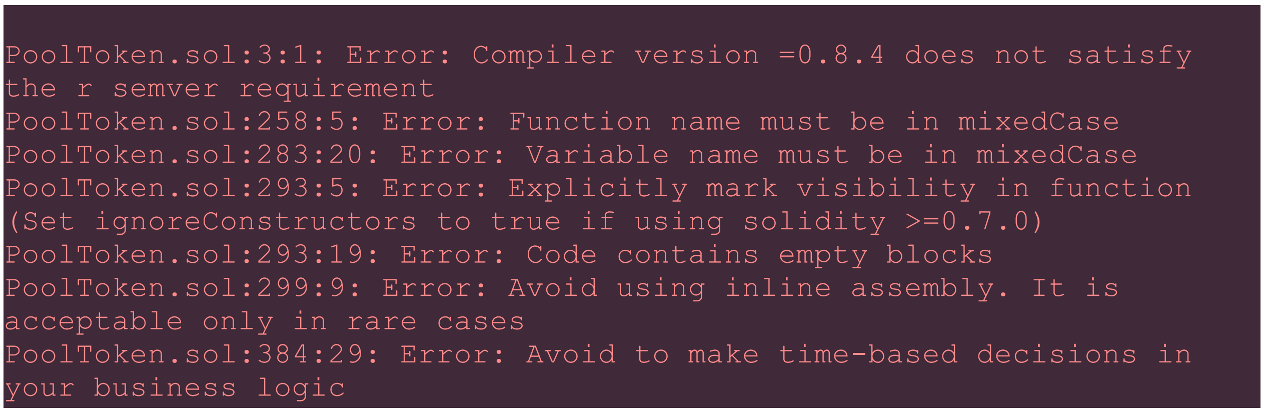
```
EleosPriceOracle.sol:2:1: Error: Compiler version =0.8.4 does not
satisfy the r semver requirement
EleosPriceOracle.sol:5:5: Error: Explicitly mark visibility of state
EleosPriceOracle.sol:51:5: Error: Function name must be in mixedCase
EleosPriceOracle.sol:174:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
EleosPriceOracle.sol:174:19: Error: Code contains empty blocks
EleosPriceOracle.sol:229:33: Error: Variable name must be in
mixedCase
EleosPriceOracle.sol:233:9: Error: Variable "pairStorage" is unused
EleosPriceOracle.sol:260:33: Error: Variable name must be in
mixedCase
EleosPriceOracle.sol:313:23: Error: Avoid to make time-based
decisions in your business logic
```

Factory.sol

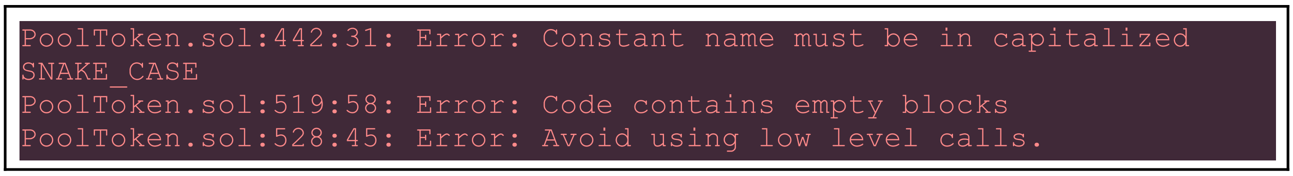


```
Factory.sol:3:1: Error: Compiler version =0.8.4 does not satisfy the
r semver requirement
Factory.sol:38:5: Error: Function name must be in mixedCase
Factory.sol:432:33: Error: Variable name must be in mixedCase
Factory.sol:520:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
```

PoolToken.sol

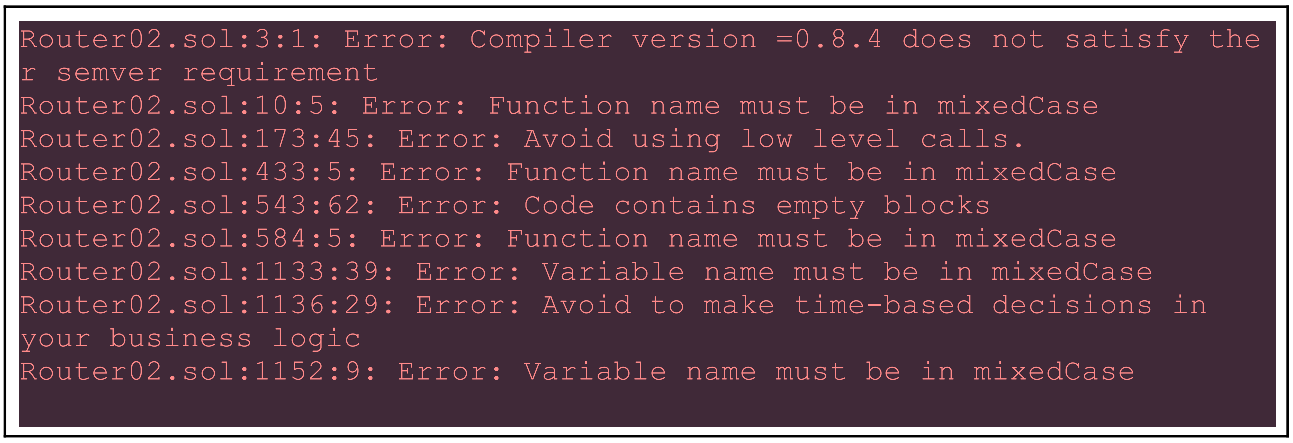


```
PoolToken.sol:3:1: Error: Compiler version =0.8.4 does not satisfy
the r semver requirement
PoolToken.sol:258:5: Error: Function name must be in mixedCase
PoolToken.sol:283:20: Error: Variable name must be in mixedCase
PoolToken.sol:293:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
PoolToken.sol:293:19: Error: Code contains empty blocks
PoolToken.sol:299:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
PoolToken.sol:384:29: Error: Avoid to make time-based decisions in
your business logic
```



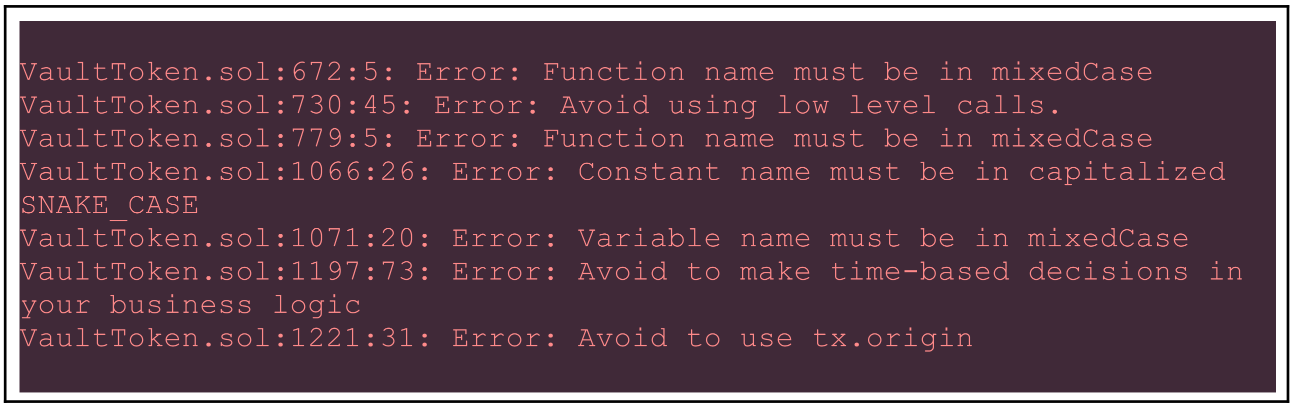
```
PoolToken.sol:442:31: Error: Constant name must be in capitalized
SNAKE_CASE
PoolToken.sol:519:58: Error: Code contains empty blocks
PoolToken.sol:528:45: Error: Avoid using low level calls.
```

Router02.sol



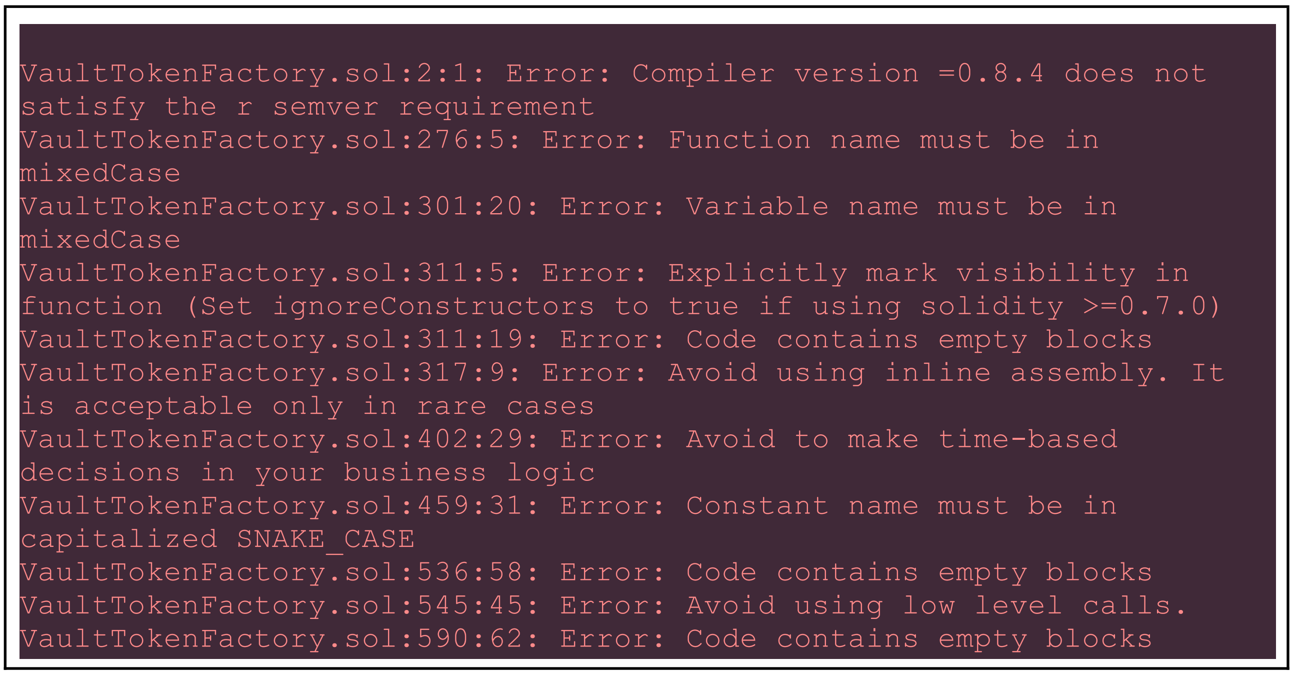
```
Router02.sol:3:1: Error: Compiler version =0.8.4 does not satisfy the
r semver requirement
Router02.sol:10:5: Error: Function name must be in mixedCase
Router02.sol:173:45: Error: Avoid using low level calls.
Router02.sol:433:5: Error: Function name must be in mixedCase
Router02.sol:543:62: Error: Code contains empty blocks
Router02.sol:584:5: Error: Function name must be in mixedCase
Router02.sol:1133:39: Error: Variable name must be in mixedCase
Router02.sol:1136:29: Error: Avoid to make time-based decisions in
your business logic
Router02.sol:1152:9: Error: Variable name must be in mixedCase
```

VaultToken.sol

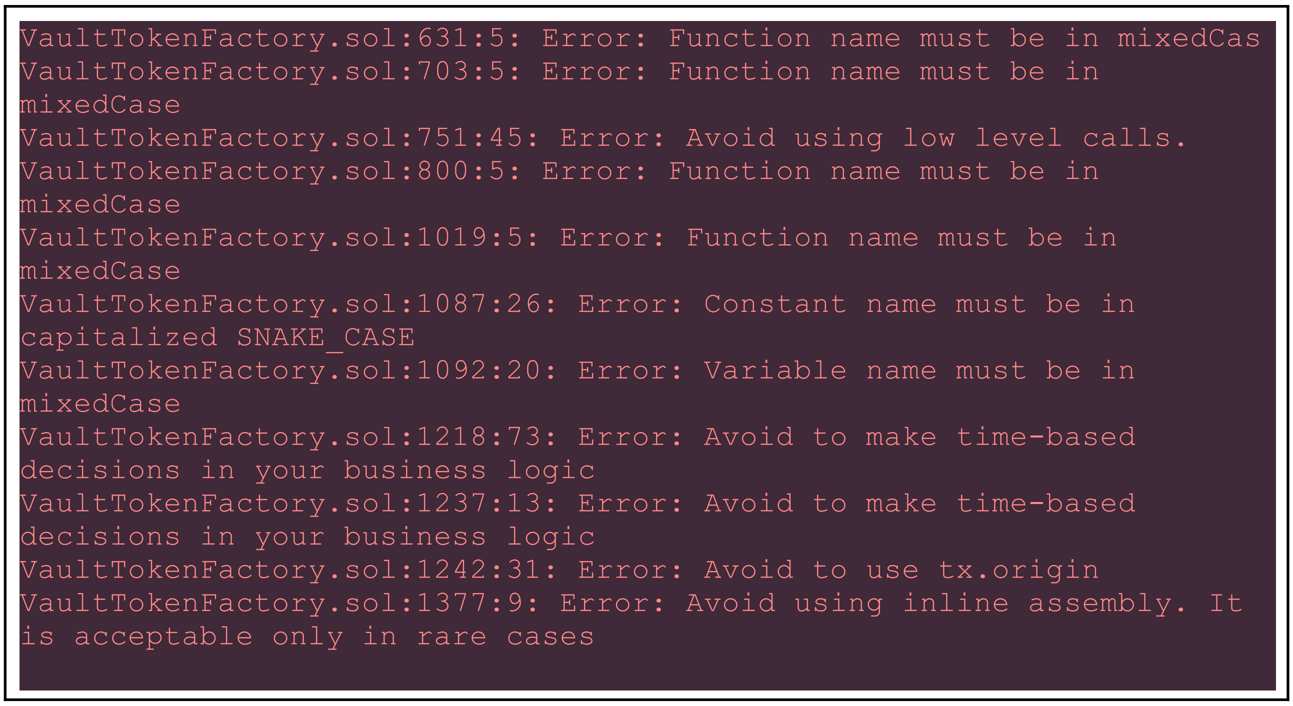


```
VaultToken.sol:672:5: Error: Function name must be in mixedCase
VaultToken.sol:730:45: Error: Avoid using low level calls.
VaultToken.sol:779:5: Error: Function name must be in mixedCase
VaultToken.sol:1066:26: Error: Constant name must be in capitalized
SNAKE_CASE
VaultToken.sol:1071:20: Error: Variable name must be in mixedCase
VaultToken.sol:1197:73: Error: Avoid to make time-based decisions in
your business logic
VaultToken.sol:1221:31: Error: Avoid to use tx.origin
```

VaultTokenFactory.sol



```
VaultTokenFactory.sol:2:1: Error: Compiler version =0.8.4 does not
satisfy the r semver requirement
VaultTokenFactory.sol:276:5: Error: Function name must be in
mixedCase
VaultTokenFactory.sol:301:20: Error: Variable name must be in
mixedCase
VaultTokenFactory.sol:311:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
VaultTokenFactory.sol:311:19: Error: Code contains empty blocks
VaultTokenFactory.sol:317:9: Error: Avoid using inline assembly. It
is acceptable only in rare cases
VaultTokenFactory.sol:402:29: Error: Avoid to make time-based
decisions in your business logic
VaultTokenFactory.sol:459:31: Error: Constant name must be in
capitalized SNAKE_CASE
VaultTokenFactory.sol:536:58: Error: Code contains empty blocks
VaultTokenFactory.sol:545:45: Error: Avoid using low level calls.
VaultTokenFactory.sol:590:62: Error: Code contains empty blocks
```


```
VaultTokenFactory.sol:631:5: Error: Function name must be in mixedCas
VaultTokenFactory.sol:703:5: Error: Function name must be in
mixedCase
VaultTokenFactory.sol:751:45: Error: Avoid using low level calls.
VaultTokenFactory.sol:800:5: Error: Function name must be in
mixedCase
VaultTokenFactory.sol:1019:5: Error: Function name must be in
mixedCase
VaultTokenFactory.sol:1087:26: Error: Constant name must be in
capitalized SNAKE_CASE
VaultTokenFactory.sol:1092:20: Error: Variable name must be in
mixedCase
VaultTokenFactory.sol:1218:73: Error: Avoid to make time-based
decisions in your business logic
VaultTokenFactory.sol:1237:13: Error: Avoid to make time-based
decisions in your business logic
VaultTokenFactory.sol:1242:31: Error: Avoid to use tx.origin
VaultTokenFactory.sol:1377:9: Error: Avoid using inline assembly. It
is acceptable only in rare cases
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.