Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Gamut Exchange
Website:	gamut.exchange
Platform:	Ethereum
Language:	Solidity
Date:	October 5th, 2022

Table of contents

Introduction

EtherAuthority was contracted by Gamut Exchange protocol to perform the Security audit of the Gamut Exchange 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 October 5th, 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

Gamut Exchange is an algorithmic market maker(AMM) and decentralized financial(Defi) product which has functionalities like create pool, join or exit the pool, swap, etc.

Audit scope

Name	**Code Review and Security Analysis Report for Gamut Exchange Protocol Smart Contracts**
Platform	**Ethereum / Solidity**
File 1	HedgeFactory.sol
File 1 MD5 Hash	790FCC49EB81D3E736A2A7E12C1EEBD9
File 2	HedgePoolToken.sol
File 2 MD5 Hash	EC765FAECC4B069970C966B904FDC18C
File 3	Pool.sol
File 3 MD5 Hash	97678C48930E15C40ED54EC510837758
File 4	ProtocolFeesCollector.sol
File 4 MD5 Hash	E41DF03AB5B8E44C206C1DA6CE24BB54
File 5	Router.sol
File 5 MD5 Hash	6B77495E30DD3FC4E8F4723AE968A7FB
File 6	Vault.sol

File 6 MD5 Hash	4CF72EB24A323FB6650C352DA9788B0B
File 7	WeightedMath.sol
File 7 MD5 Hash	B035975E0BCE7A006D1FD295B9298D23
File 8	WETH9.sol
File 8 MD5 Hash	A109DC8265B78411CAD4525A5E7AE73F
Audit Date	October 5th, 2022
Revision Date	October 14th, 2022

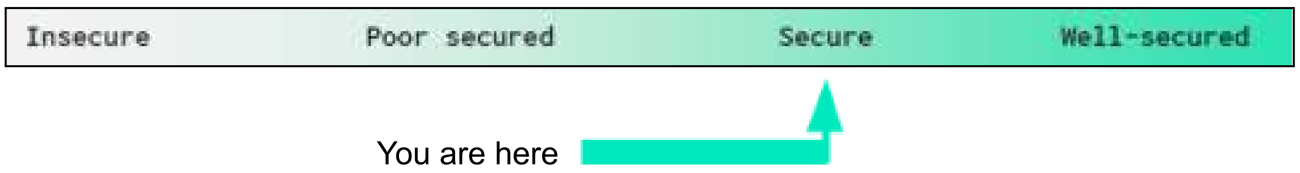
Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 HedgeFactory.sol ● HedgeFactory can create a new pool. ● HedgeFactory can set the protocol fee collector.	**YES, This is valid.**
File 2 HedgePoolToken.sol ● Name: Hedge Pool Token ● Symbol: HT	**YES, This is valid.**
File 3 Pool.sol ● Swap Fee Percentage: 0.0001% ● Maximum Swap Fee Percentage: 10% ● Minimum Weight: 20% ● Pool owners can set a swap fee. ● Pool owners can set the balances of Pool's tokens and update the lastChangeBlock.	**YES, This is valid.**
File 4 ProtocolFeesCollector.sol ● Maximum Protocol Swap Fee Percentage: 50% ● Owner can withdraw collected fees. ● Owner can set a swap fee percentage.	**YES, This is valid.**
File 5 Router.sol ● Owner can set up a hedge factory. ● Routers can join the pool. ● Routers can exit pool.	**YES, This is valid.**
File 6 Vault.sol ● Vault has functions like: _receiveAsset, _sendAsset, _handleRemainingEth, etc.	**YES, This is valid.**
File 7 WeightedMath.sol ● One: 18 ● WeightedMath has functions like:	**YES, This is valid.**

_calculateInvariant, _calcOutGivenIn, etc.	
File 8 WETH9.sol ● Name: Wrapped Ether ● Symbol: WETH ● Decimals: 18	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. Also, these contracts do contain owner control, which does not 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.
These issues are fixed/resolved in the revised smart contract code.

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	Passed
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 8 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Gamut Exchange 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 Gamut Exchange Protocol.

The Gamut Exchange team has provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

All code parts are not well commented on smart contracts.

Documentation

We were given a Gamut Exchange 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. but the logic is straightforward. So it is 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.

Another source of information was its website https://gamut.exchange which provided rich information about the project architecture.

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

HedgeFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getRouter	read	Passed	No Issue
3	setProtocolFeeCollector	external	Passed	No Issue
4	getProtocolFeesCollector	read	Passed	No Issue
5	_getProtocolSwapFeePercentage	external	Passed	No Issue
6	allPoolsLength	external	Passed	No Issue
7	create	external	Passed	No Issue
8	owner	read	Passed	No Issue
9	onlyOwner	modifier	Passed	No Issue
10	renounceOwnership	write	access only Owner	No Issue
11	transferOwnership	write	access only Owner	No Issue
12	_transferOwnership	internal	Passed	No Issue

HedgePoolToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	name	read	Passed	No Issue
3	symbol	read	Passed	No Issue
4	decimals	read	Passed	No Issue
5	totalSupply	read	Passed	No Issue
6	balanceOf	read	Passed	No Issue
7	transfer	write	Passed	No Issue
8	allowance	read	Passed	No Issue
9	approve	write	Passed	No Issue
10	transferFrom	write	Passed	No Issue
11	increaseAllowance	write	Passed	No Issue
12	decreaseAllowance	write	Passed	No Issue
13	_transfer	internal	Passed	No Issue
14	_mint	internal	Passed	No Issue
15	_burn	internal	Passed	No Issue
16	_approve	internal	Passed	No Issue
17	_spendAllowance	internal	Passed	No Issue
18	_beforeTokenTransfer	internal	Passed	No Issue
19	_afterTokenTransfer	internal	Passed	No Issue

Pool.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyRouter	modifier	Passed	No Issue
3	getRouter	read	Passed	No Issue
4	getSwapFeePercentage	read	Passed	No Issue
5	getWeights	external	Passed	No Issue
6	_weights	read	Passed	No Issue
7	_weights	read	Passed	No Issue
8	getWeightsAndScalingFactors	read	Passed	No Issue
9	getPoolTokensAndBalances	external	Passed	No Issue
10	getPoolBalancesAndChangeBlock	read	Passed	No Issue
11	setSwapFeePercentage	external	Passed	No Issue
12	_setSwapFeePercentage	write	Passed	No Issue
13	setPoolBalancesAndLastChangeBlock	external	access only Router	No Issue
14	onSwap	write	access only Router	No Issue
15	_onVirtualSwap	write	Passed	No Issue
16	_calcPoolAndProtocolSwapFee	read	Passed	No Issue
17	_calcSwapOut	read	Passed	No Issue
18	_updateWeights	write	Passed	No Issue
19	onJoinPool	external	access only Router	No Issue
20	_onInitializePool	read	Passed	No Issue
21	_onJoinPool	write	Passed	No Issue
22	_joinExactTokensInForHPTOut	write	Passed	No Issue
23	_unEqualJoin	write	Passed	No Issue
24	_joinTokenInForHPTOut	write	Passed	No Issue
25	_calculateHptOut	read	Passed	No Issue
26	_doVirtualSwap	write	Passed	No Issue
27	onExitPool	external	access only Router	No Issue
28	_onExitPool	write	Passed	No Issue
29	_doExit	write	Passed	No Issue
30	_computeScalingFactor	read	Passed	No Issue
31	_scalingFactor	read	Passed	No Issue
32	_upscale	write	Passed	No Issue
33	_upscaleArray	read	Passed	No Issue
34	_downscaleDown	write	Passed	No Issue
35	_downscaleDownArray	read	Passed	No Issue
36	_downscaleUp	write	Passed	No Issue
37	_downscaleUpArray	read	Passed	No Issue
38	_calculateInvariant	internal	Passed	No Issue
39	_calcOutGivenIn	internal	Passed	No Issue
40	_calculateNewWeights	internal	Passed	No Issue
41	_calcHptOutGivenExactTokensIn	internal	Passed	No Issue
42	_calculateVirtualSwapAmountIn	internal	Passed	No Issue
43	_calculateNextIterationAmountIn	internal	Passed	No Issue
44	_calcTokensOutGivenExactHptIn	internal	Passed	No Issue

ProtocolFeesCollector.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	withdrawCollectedFees	external	Passed	No Issue
8	setSwapFeePercentage	external	Passed	No Issue
9	getProtocolSwapFeePercentage	external	Passed	No Issue
10	getCollectedFeeAmounts	external	Passed	No Issue

Router.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	setHedgeFactory	external	Passed	No Issue
8	_toPoolBalanceChange	write	Passed	No Issue
9	_toPoolBalanceChange	write	Passed	No Issue
10	joinPool	external	Passed	No Issue
11	exitPool	external	Passed	No Issue
12	_joinOrExit	write	Passed	No Issue
13	_callPoolBalanceChange	write	Passed	No Issue
14	_processJoinPoolTransfers	write	Passed	No Issue
15	_processExitPoolTransfers	write	Passed	No Issue
16	swap	external	Passed	No Issue
17	_swapWithPool	write	Passed	No Issue
18	_processPoolSwapRequest	write	Passed	No Issue
19	_callPoolOnSwapHook	write	Passed	No Issue
20	batchSwap	external	Passed	No Issue
21	_swapWithPools	write	Passed	No Issue
22	_validateTokensAndGetBalances	read	Passed	No Issue
23	_unsafeCastToInt256	write	Passed	No Issue
24	_receiveAsset	internal	Passed	No Issue
25	_sendAsset	internal	Passed	No Issue
26	_handleRemainingEth	internal	Passed	No Issue

27	_payFeeAmount	internal	Passed	No Issue
28	receive	external	Passed	No Issue
29	nonReentrant	modifier	Passed	No Issue
30	_enterNonReentrant	write	Passed	No Issue
31	_exitNonReentrant	write	Passed	No Issue

Vault.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_receiveAsset	internal	Passed	No Issue
3	_sendAsset	internal	Passed	No Issue
4	_handleRemainingEth	internal	Passed	No Issue
5	_payFeeAmount	internal	Passed	No Issue
6	receive	external	Passed	No Issue

WeightedMath.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_calculateInvariant	internal	Passed	No Issue
3	_calcOutGivenIn	internal	Passed	No Issue
4	_calculateNewWeights	internal	Passed	No Issue
5	_calcHptOutGivenExactTokensIn	internal	Passed	No Issue
6	_calculateVirtualSwapAmountIn	internal	Passed	No Issue
7	_calculateNextIterationAmountIn	internal	Passed	No Issue
8	_calcTokensOutGivenExactHptIn	internal	Passed	No Issue

WETH9.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	receive	external	Passed	No Issue
3	deposit	write	Passed	No Issue
4	withdraw	write	Passed	No Issue
5	totalSupply	read	Passed	No Issue
6	approve	write	Passed	No Issue
7	transfer	write	Passed	No Issue
8	transferFrom	write	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 loss
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) Function input parameters lack of check:

Variable validation is not performed in below functions:

Router.sol

- joinPool = recipient
- exitPool = sender

HedgeFactory.sol

- create = tokenA, tokenB

Resolution: We advise to put validation: integer type variables should be greater than 0 and address type variables should not be address(0).

Status: This issue is fixed

Very Low / Informational / Best practices:

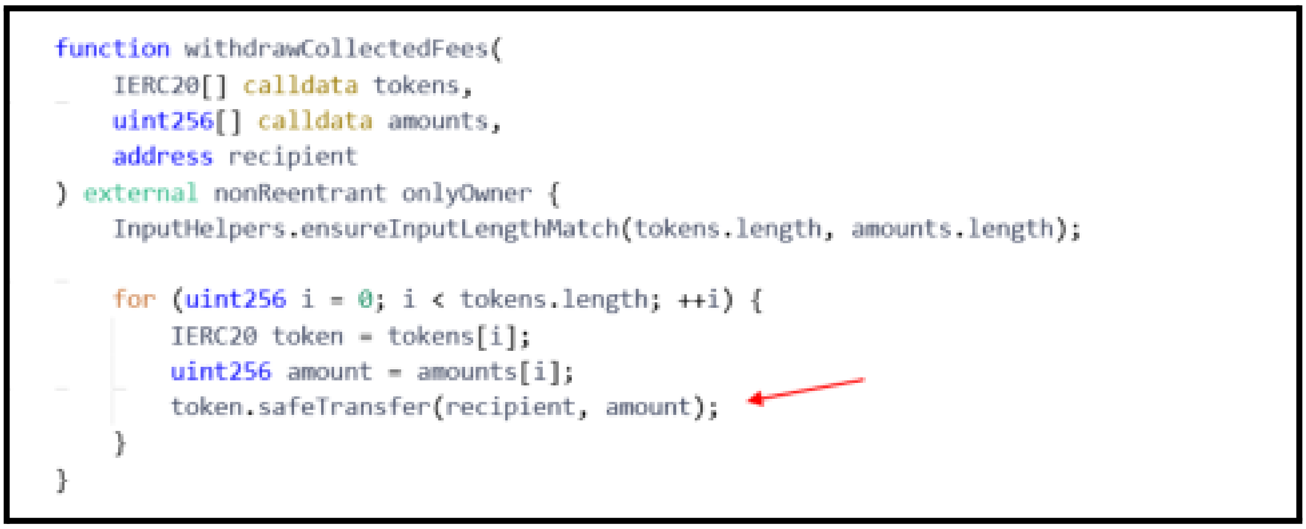
(1) Infinite loop: **ProtocolFeesCollector.sol**

In the withdrawCollectedFees function, tokens for loop do not have an upper length limit, which costs more gas.

Resolution: Upper bound should have a certain limit for loops.

Status: This issue is fixed

(2) Owner can drain all ERC20 tokens: **ProtocolFeesCollector.sol**



```
function withdrawCollectedFees(
    IERC20[] calldata tokens,
    uint256[] calldata amounts,
    address recipient
) external nonReentrant onlyOwner {
    InputHelpers.ensureInputLengthMatch(tokens.length, amounts.length);

    for (uint256 i = 0; i < tokens.length; ++i) {
        IERC20 token = tokens[i];
        uint256 amount = amounts[i];
        token.safeTransfer(recipient, amount);
    }
}
```

Using the withdrawCollectedFees function, the owner can drain tokens from the contract.

Resolution: Owner should confirm this feature.

Status: This issue is fixed

(3) Critical operation lacks event log:

Missing event log for:

Router.sol

- setHedgeFactory

HedgeFactory.sol

- setProtocolFeeCollector

ProtocolFeesCollector.sol

- withdrawCollectedFees
- setSwapFeePercentage

Resolution: Write an event log for listed events.

Status: This issue is fixed

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- setProtocolFeeCollector: HedgeFactory owner can set a new protocol fee collector address.
- setSwapFeePercentage: Pool owner can set swap fee percentage value.
- setPoolBalancesAndLastChangeBlock: Pool router owner can set the balances of Pool's tokens and update the lastChangeBlock.
- onSwap: Pool router owner can swap hooks.
- onJoinPool: Pool router owner can join pool.
- onExitPool: Pool router owner can exit pool hook.
- withdrawCollectedFees: ProtocolFeesCollector owner can withdraw collected fees.
- setSwapFeePercentage: ProtocolFeesCollector owner can set swap fee percentage value.
- setHedgeFactory: Router owner can set hedge factory value.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We have not observed any major issues in smart contracts. **So smart contracts are ready for 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 **"Secure".**

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.

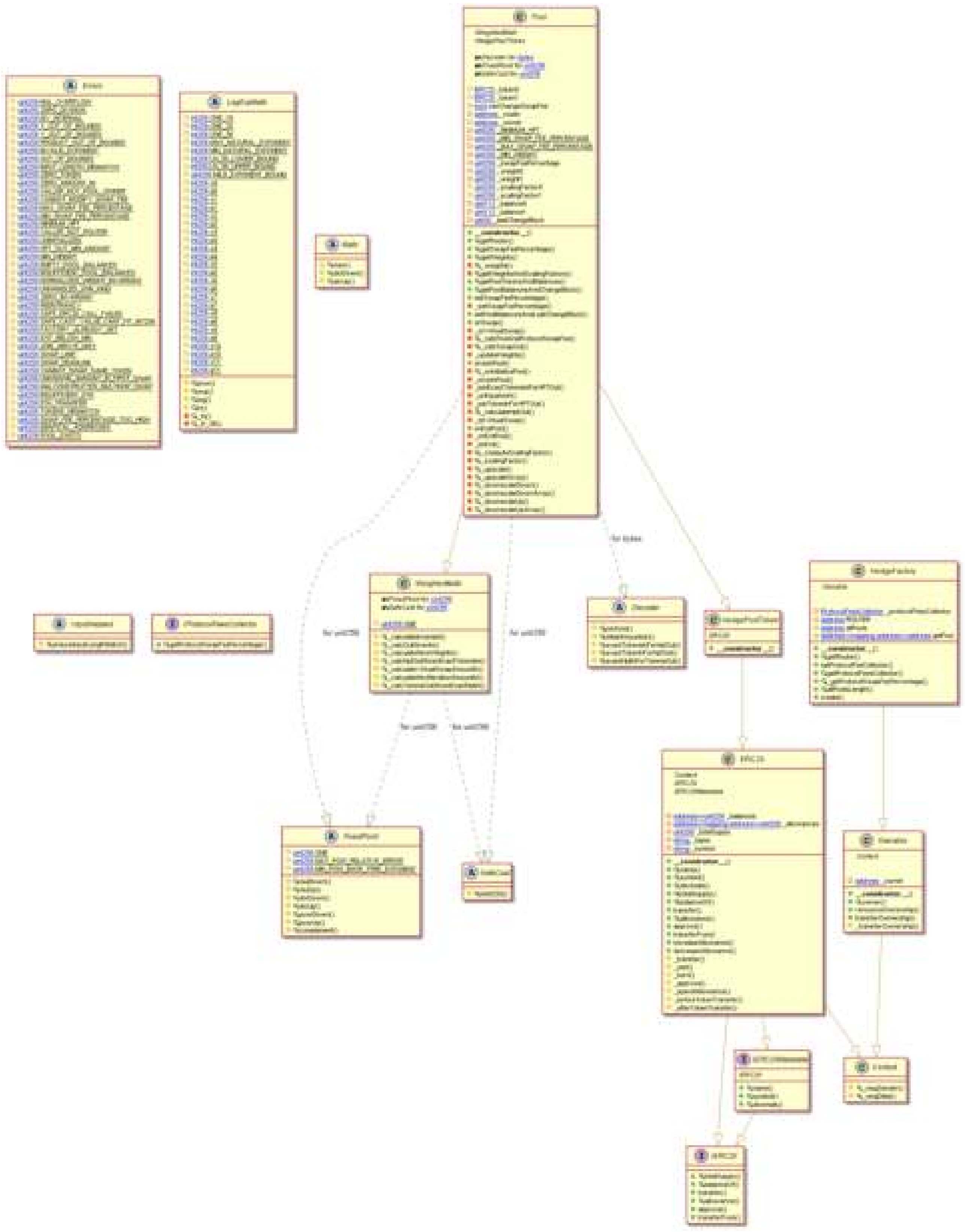
Technical Disclaimer

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.

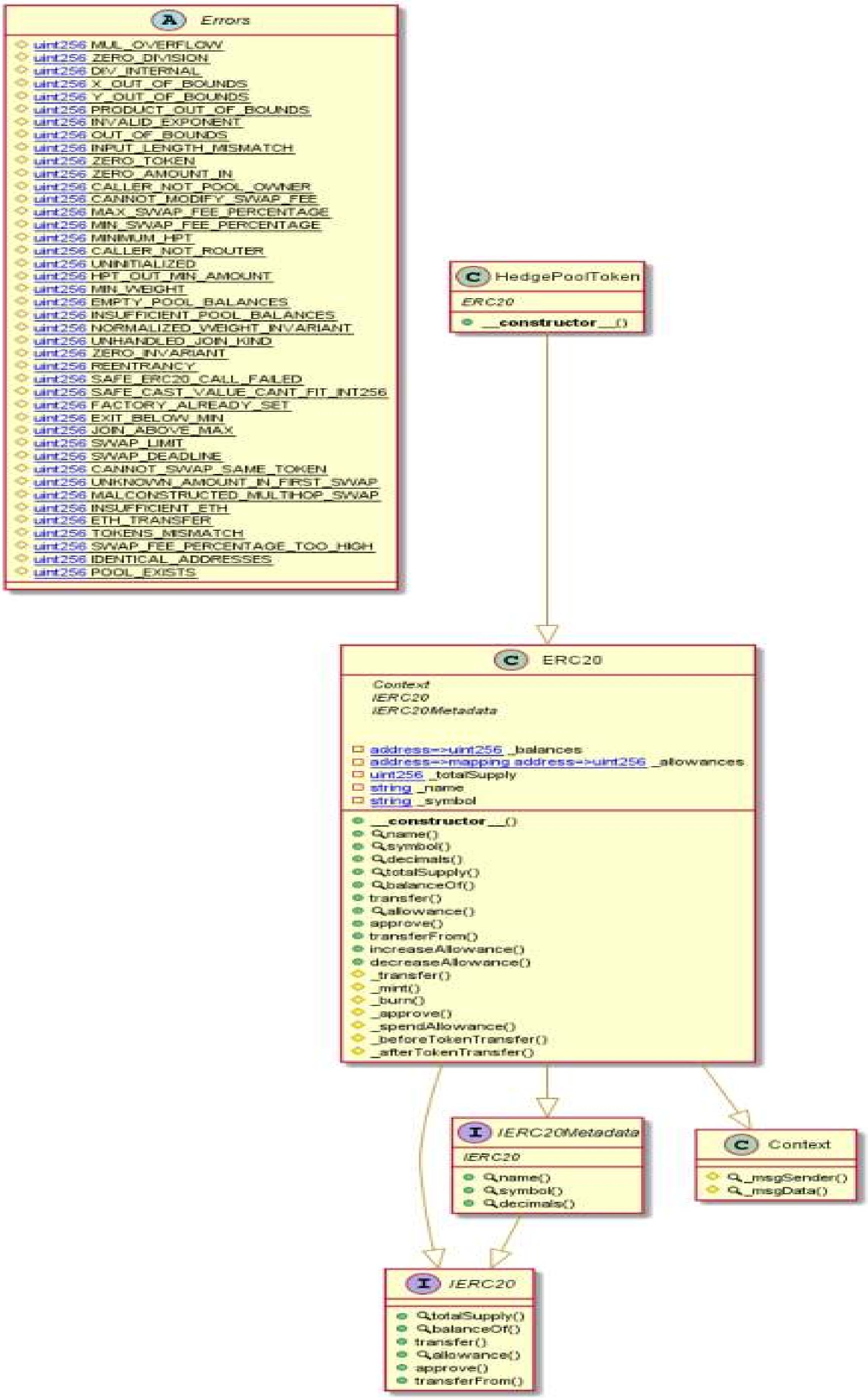
Appendix

Code Flow Diagram - Gamut Exchange Protocol

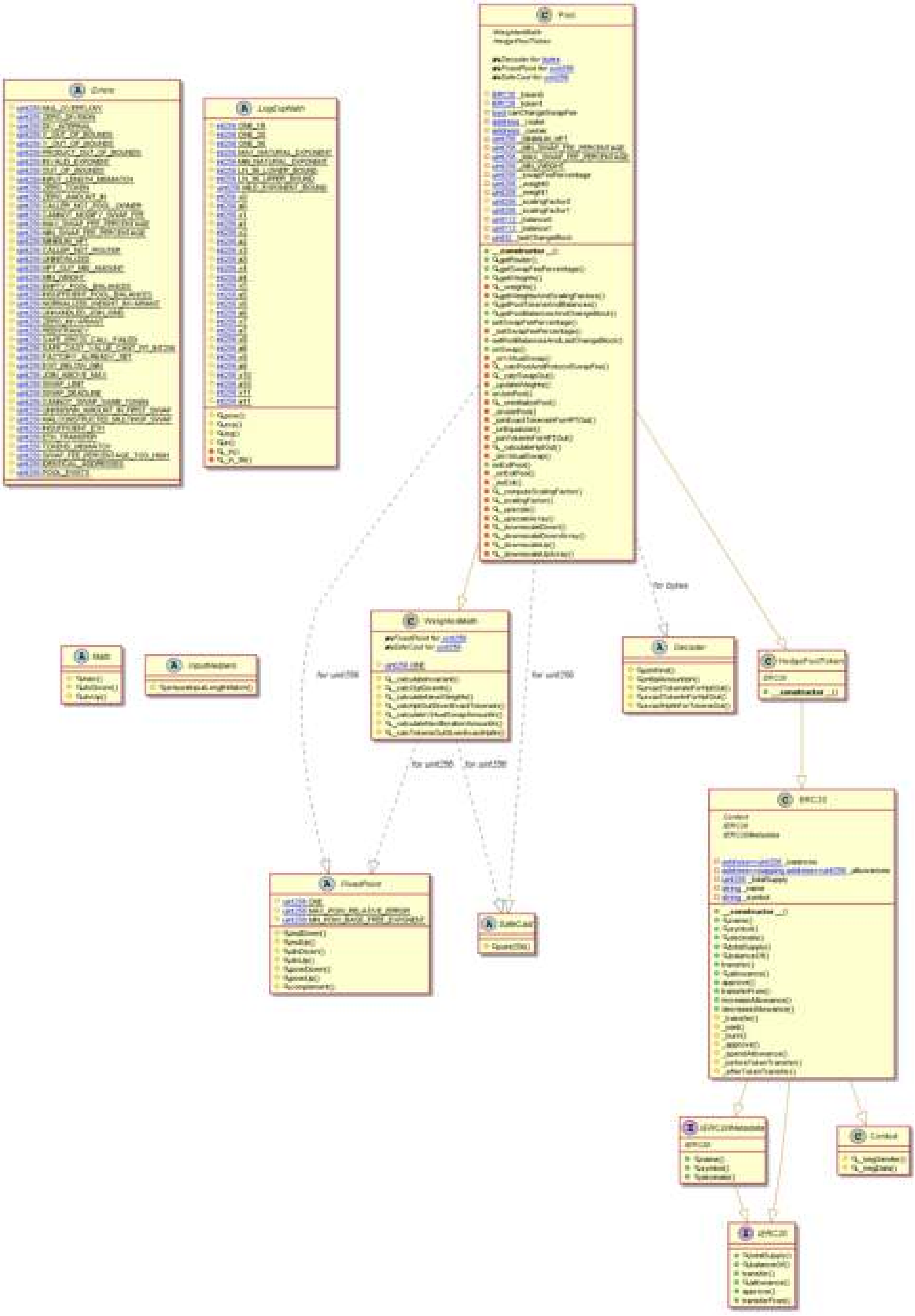
HedgeFactory Diagram



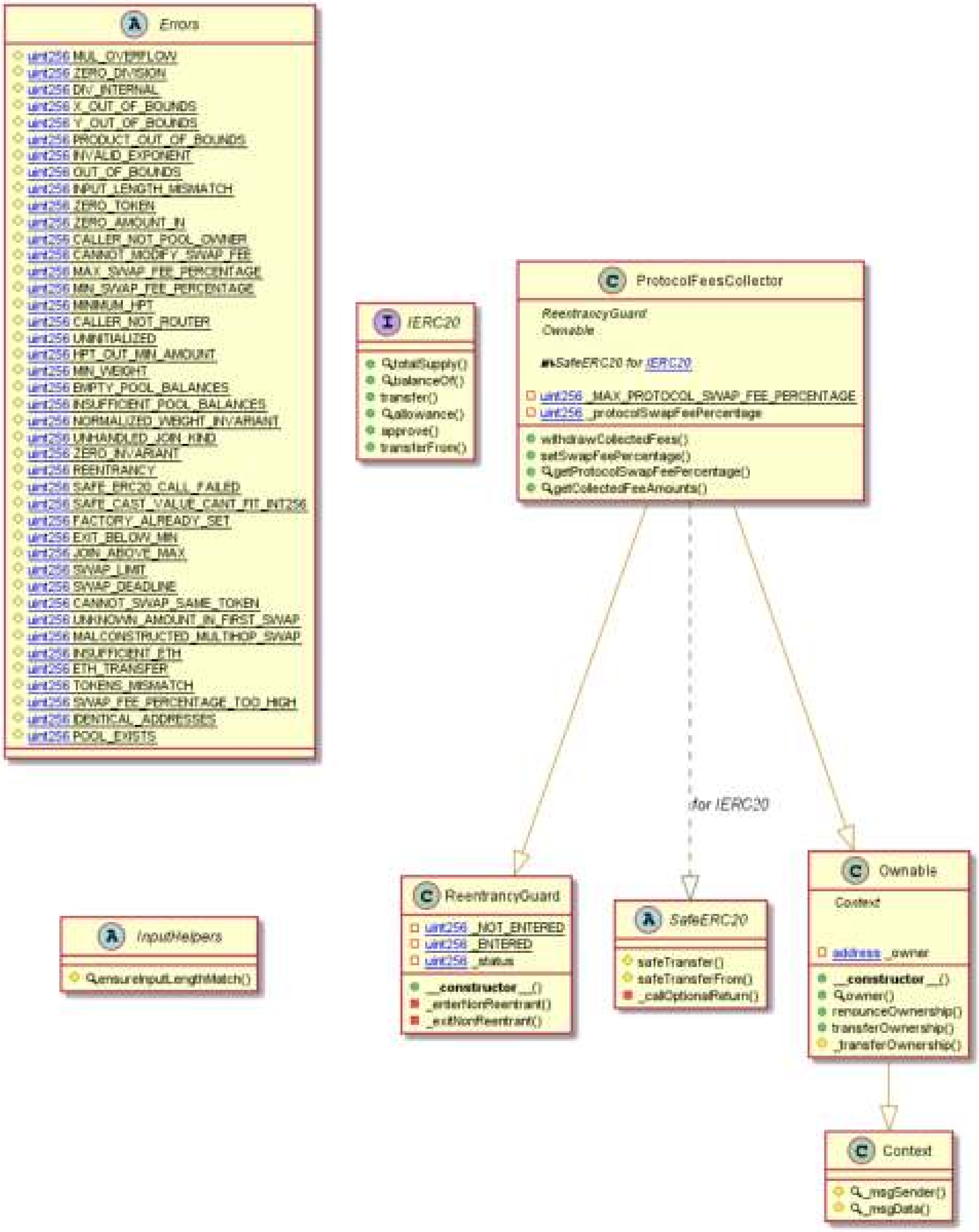
HedgePoolToken Diagram



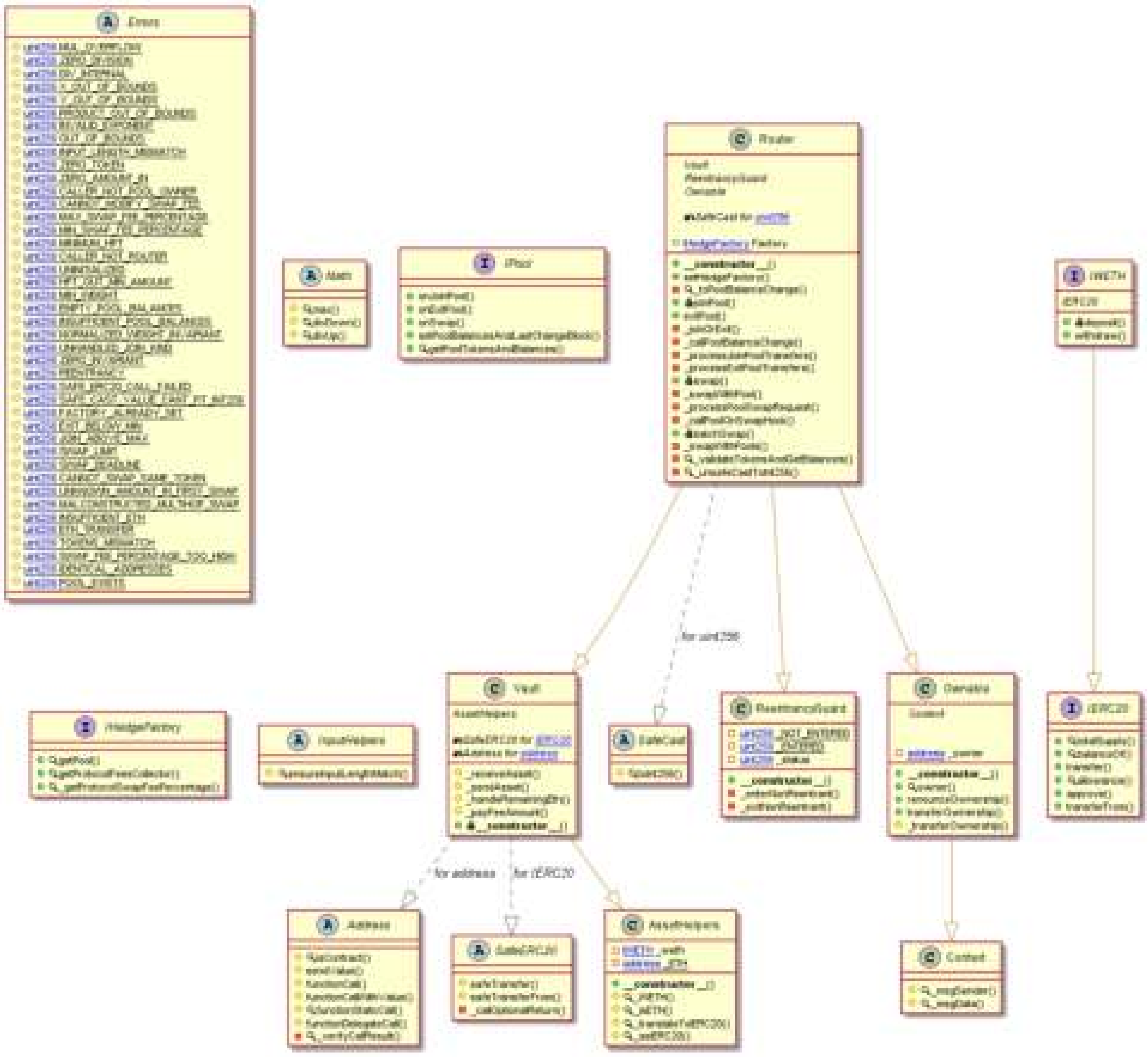
Pool Diagram



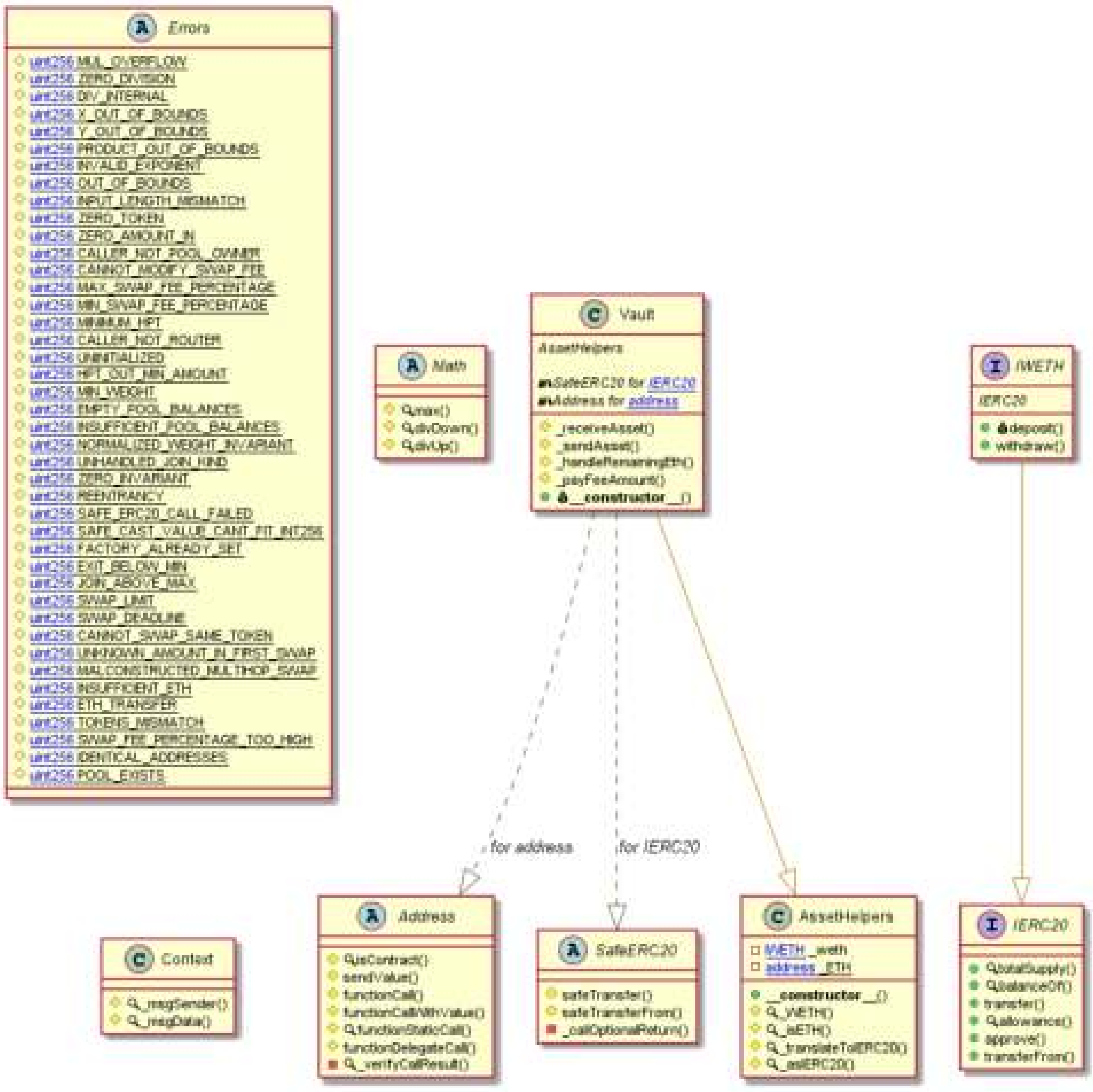
ProtocolFeesCollector Diagram



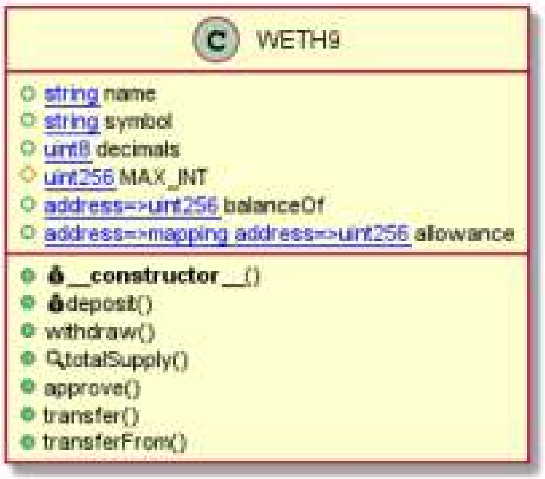
Router Diagram



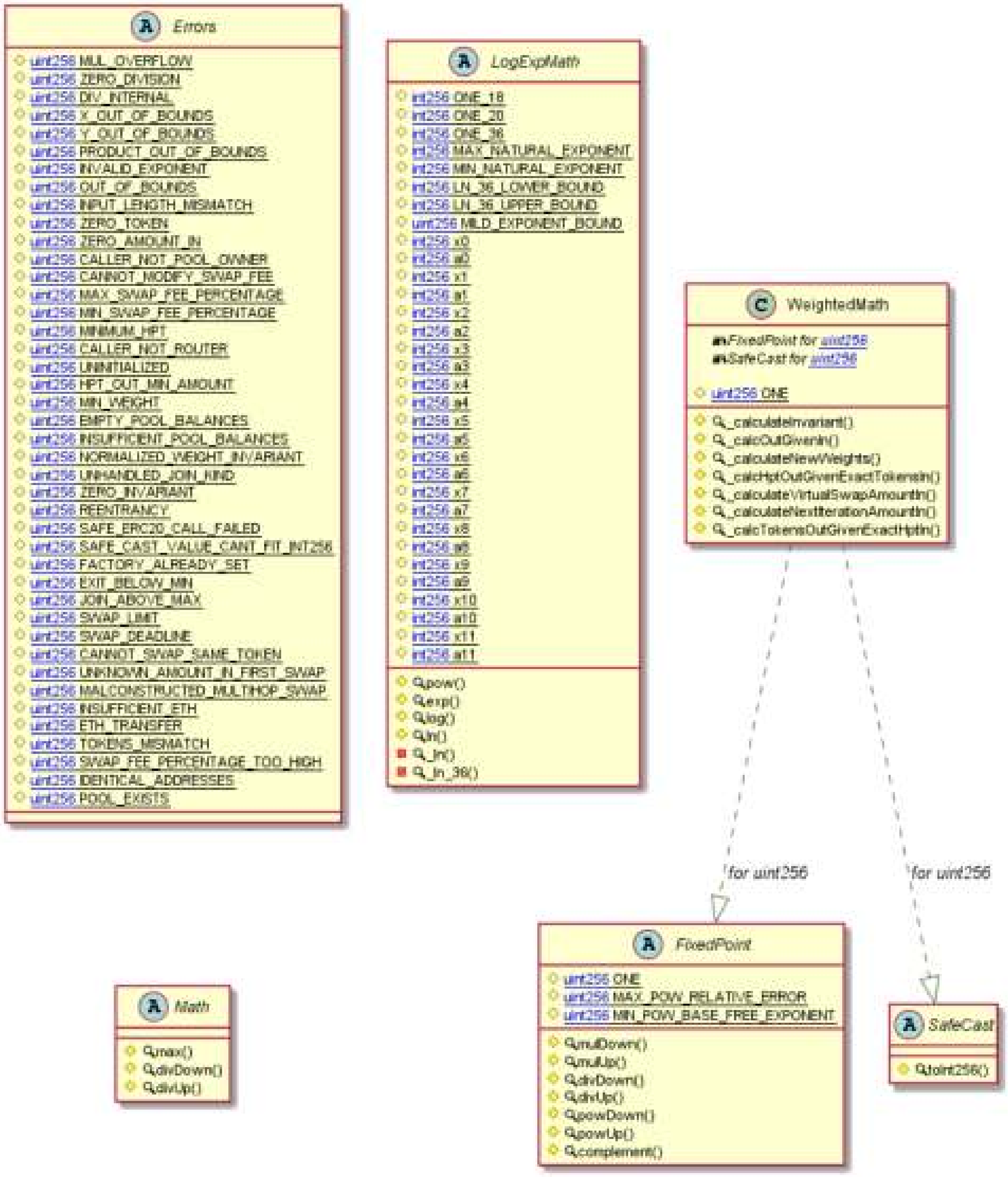
Vault Diagram



WETH9 Diagram

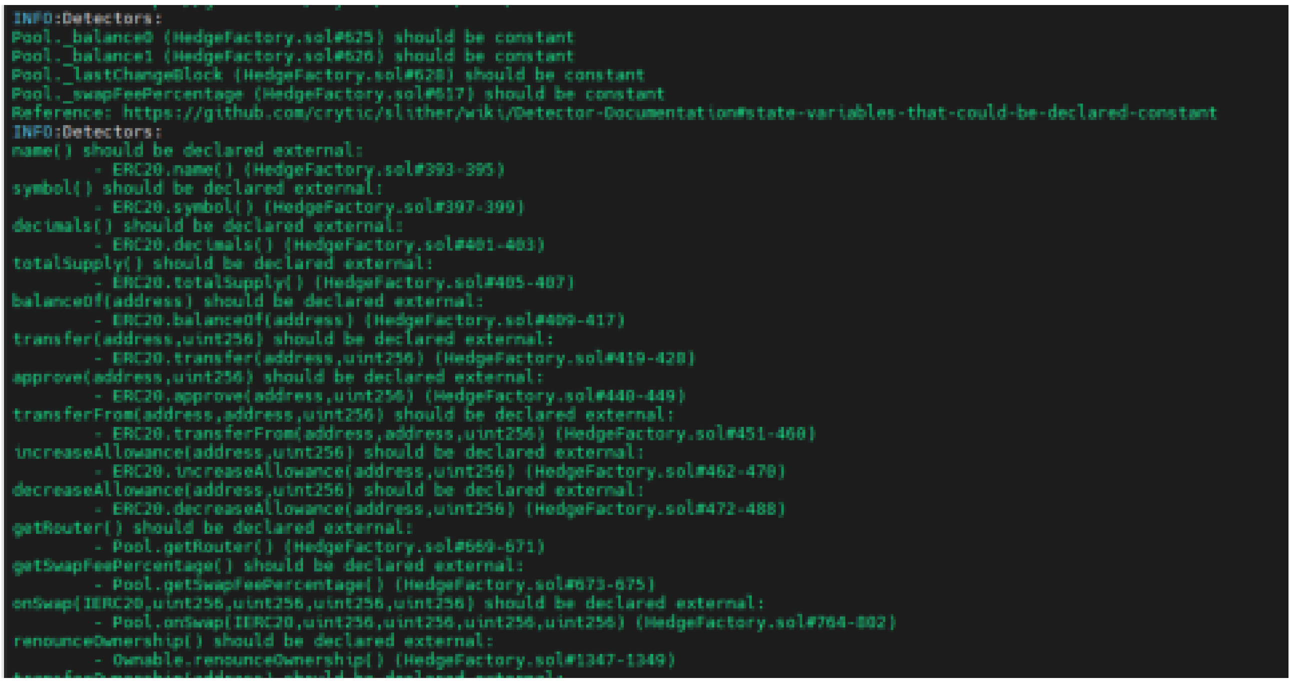


WeightedMath Diagram

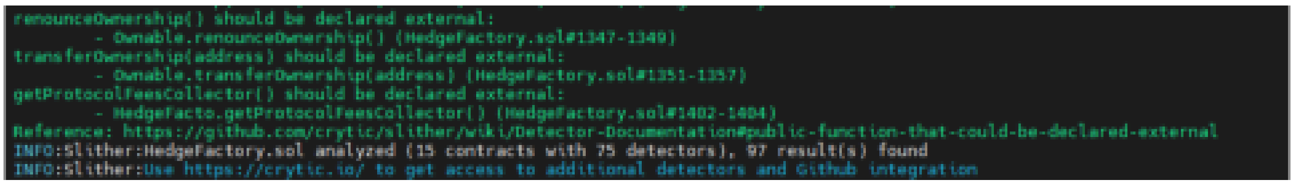


Slither Results Log

Slither log >> HedgeFactory.sol

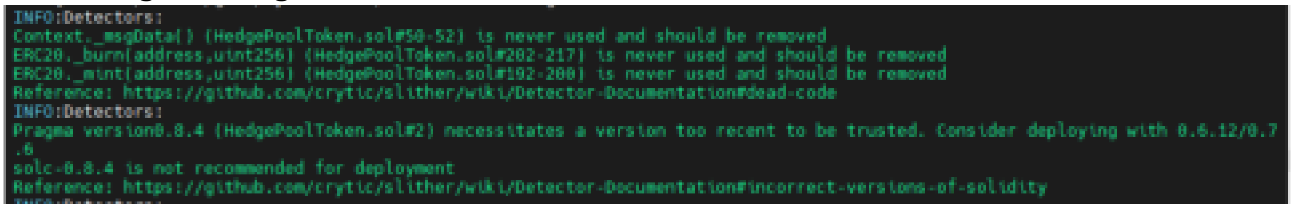


```
INFO:Detectors:
Pool._balance0 (HedgeFactory.sol#625) should be constant
Pool._balance1 (HedgeFactory.sol#626) should be constant
Pool._lastChangeBlock (HedgeFactory.sol#628) should be constant
Pool._swapFeePercentage (HedgeFactory.sol#617) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (HedgeFactory.sol#393-395)
symbol() should be declared external:
        - ERC20.symbol() (HedgeFactory.sol#397-399)
decimals() should be declared external:
        - ERC20.decimals() (HedgeFactory.sol#401-403)
totalSupply() should be declared external:
        - ERC20.totalSupply() (HedgeFactory.sol#405-407)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (HedgeFactory.sol#409-417)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (HedgeFactory.sol#419-428)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (HedgeFactory.sol#440-449)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (HedgeFactory.sol#451-460)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (HedgeFactory.sol#462-470)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (HedgeFactory.sol#472-488)
getRouter() should be declared external:
        - Pool.getRouter() (HedgeFactory.sol#669-671)
getSwapFeePercentage() should be declared external:
        - Pool.getSwapFeePercentage() (HedgeFactory.sol#673-675)
onSwap(IERC20,uint256,uint256,uint256,uint256) should be declared external:
        - Pool.onSwap(IERC20,uint256,uint256,uint256,uint256) (HedgeFactory.sol#764-802)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (HedgeFactory.sol#1347-1349)
```

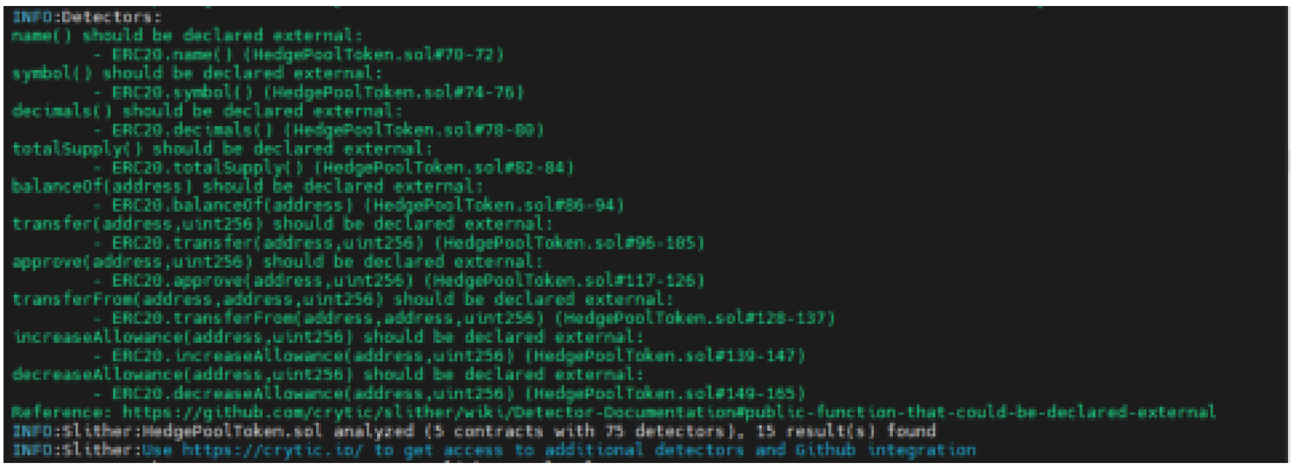


```
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (HedgeFactory.sol#1347-1349)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (HedgeFactory.sol#1351-1357)
getProtocolFeesCollector() should be declared external:
        - HedgeFacto.getProtocolFeesCollector() (HedgeFactory.sol#1402-1404)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:HedgeFactory.sol analyzed (15 contracts with 75 detectors), 97 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> HedgePoolToken.sol

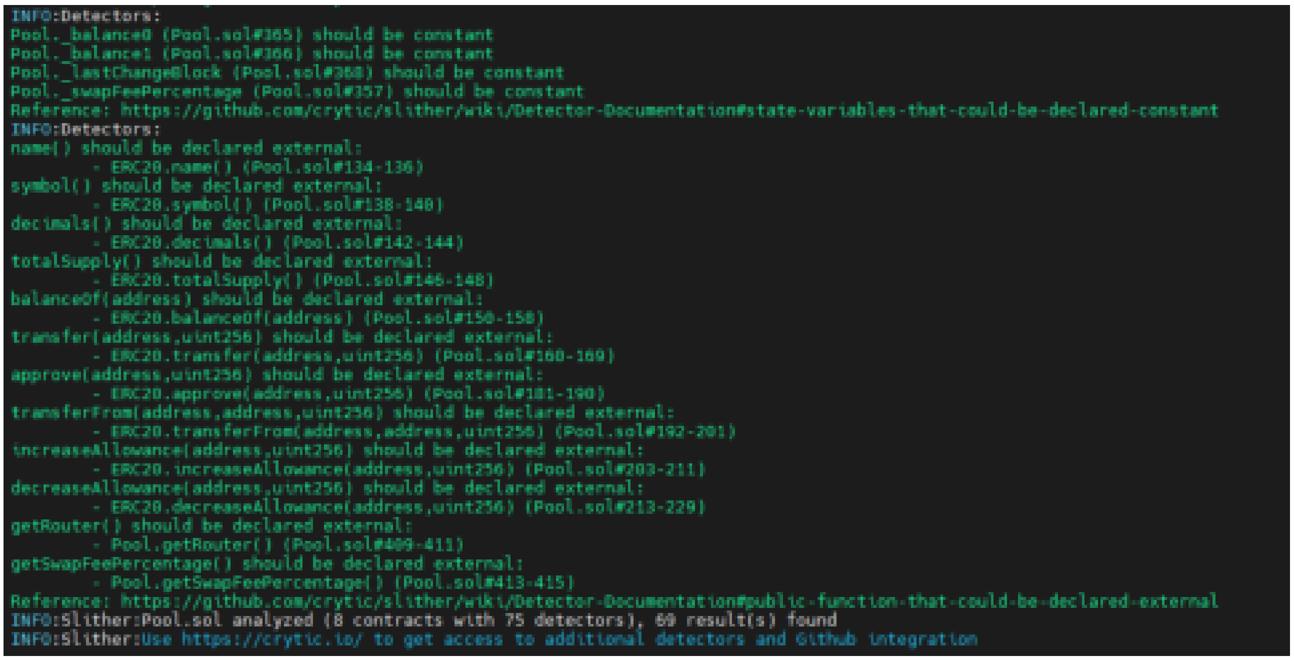


```
INFO:Detectors:
Context._msgData() (HedgePoolToken.sol#50-52) is never used and should be removed
ERC20._burn(address,uint256) (HedgePoolToken.sol#202-217) is never used and should be removed
ERC20._mint(address,uint256) (HedgePoolToken.sol#192-200) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (HedgePoolToken.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
```



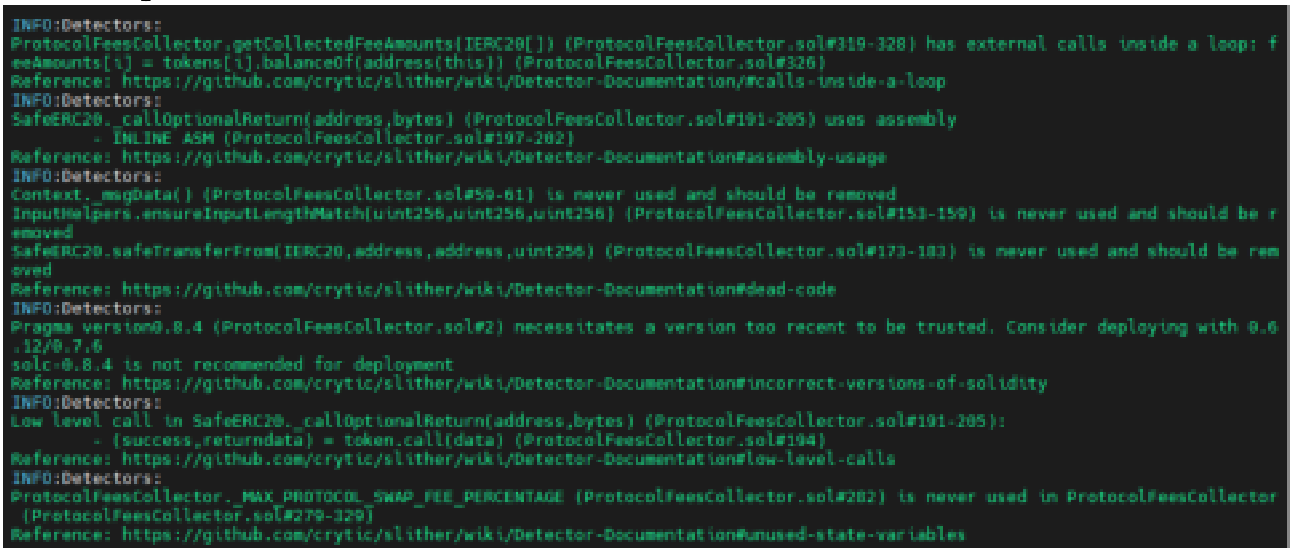
```
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (HedgePoolToken.sol#70-72)
symbol() should be declared external:
        - ERC20.symbol() (HedgePoolToken.sol#74-76)
decimals() should be declared external:
        - ERC20.decimals() (HedgePoolToken.sol#78-80)
totalSupply() should be declared external:
        - ERC20.totalSupply() (HedgePoolToken.sol#82-84)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (HedgePoolToken.sol#86-94)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (HedgePoolToken.sol#96-105)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (HedgePoolToken.sol#117-126)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (HedgePoolToken.sol#128-137)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (HedgePoolToken.sol#139-147)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (HedgePoolToken.sol#149-165)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:HedgePoolToken.sol analyzed (5 contracts with 75 detectors), 15 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Pool.sol

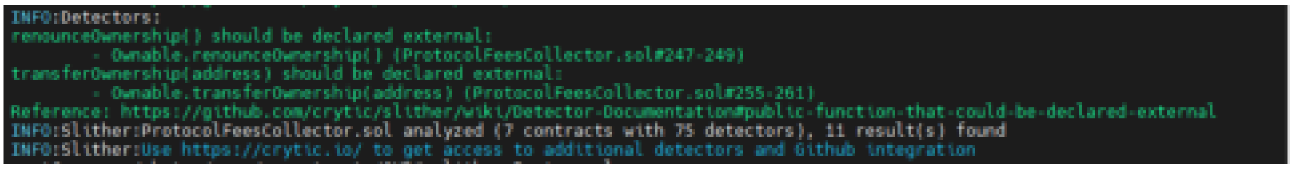


```
INFO:Detectors:
Pool._balance0 (Pool.sol#365) should be constant
Pool._balance1 (Pool.sol#366) should be constant
Pool._lastChangeBlock (Pool.sol#368) should be constant
Pool._swapFeePercentage (Pool.sol#357) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (Pool.sol#134-136)
symbol() should be declared external:
        - ERC20.symbol() (Pool.sol#138-140)
decimals() should be declared external:
        - ERC20.decimals() (Pool.sol#142-144)
totalSupply() should be declared external:
        - ERC20.totalSupply() (Pool.sol#146-148)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (Pool.sol#150-158)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (Pool.sol#160-169)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (Pool.sol#181-190)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (Pool.sol#192-201)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (Pool.sol#203-211)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (Pool.sol#213-229)
getRouter() should be declared external:
        - Pool.getRouter() (Pool.sol#409-411)
getSwapFeePercentage() should be declared external:
        - Pool.getSwapFeePercentage() (Pool.sol#413-415)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Pool.sol analyzed (8 contracts with 75 detectors), 69 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ProtocolFeesCollector.sol

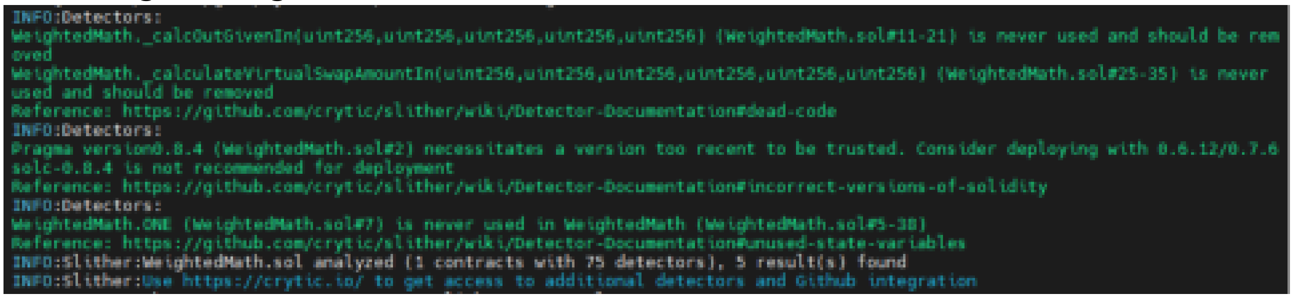


```
INFO:Detectors:
ProtocolFeesCollector.getCollectedFeeAmounts(IERC20[]) (ProtocolFeesCollector.sol#319-328) has external calls inside a loop: f
eeAmounts[i] = tokens[i].balanceOf(address(this)) (ProtocolFeesCollector.sol#326)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
SafeERC20._callOptionalReturn(address,bytes) (ProtocolFeesCollector.sol#191-205) uses assembly
        - INLINE ASM (ProtocolFeesCollector.sol#197-202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Context._msgData() (ProtocolFeesCollector.sol#59-61) is never used and should be removed
InputHelpers.ensureInputLengthMatch(uint256,uint256,uint256) (ProtocolFeesCollector.sol#153-159) is never used and should be r
emoved
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (ProtocolFeesCollector.sol#173-183) is never used and should be rem
oved
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (ProtocolFeesCollector.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 SafeERC20._callOptionalReturn(address,bytes) (ProtocolFeesCollector.sol#191-205):
        - (success,returndata) = token.call(data) (ProtocolFeesCollector.sol#194)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
ProtocolFeesCollector._MAX_PROTOCOL_SWAP_FEE_PERCENTAGE (ProtocolFeesCollector.sol#282) is never used in ProtocolFeesCollector
 (ProtocolFeesCollector.sol#279-329)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
```



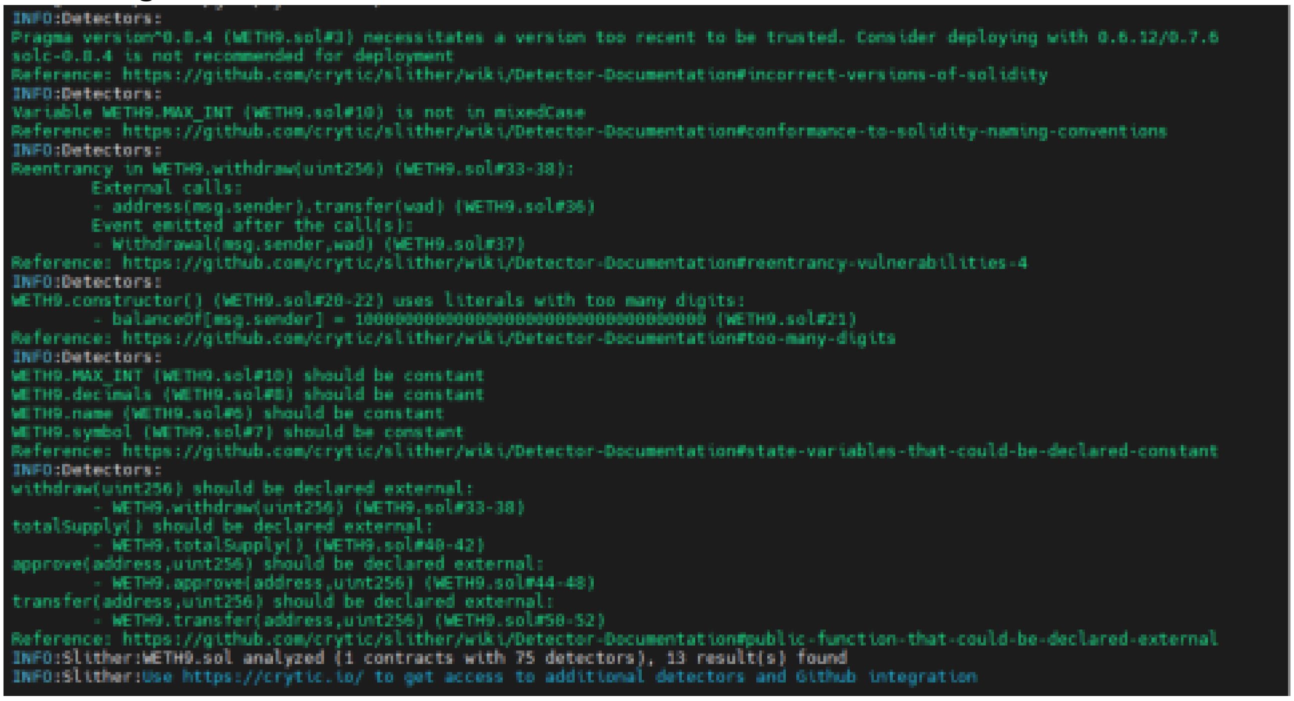
```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (ProtocolFeesCollector.sol#247-249)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (ProtocolFeesCollector.sol#255-261)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ProtocolFeesCollector.sol analyzed (7 contracts with 75 detectors), 11 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> WeightedMath.sol



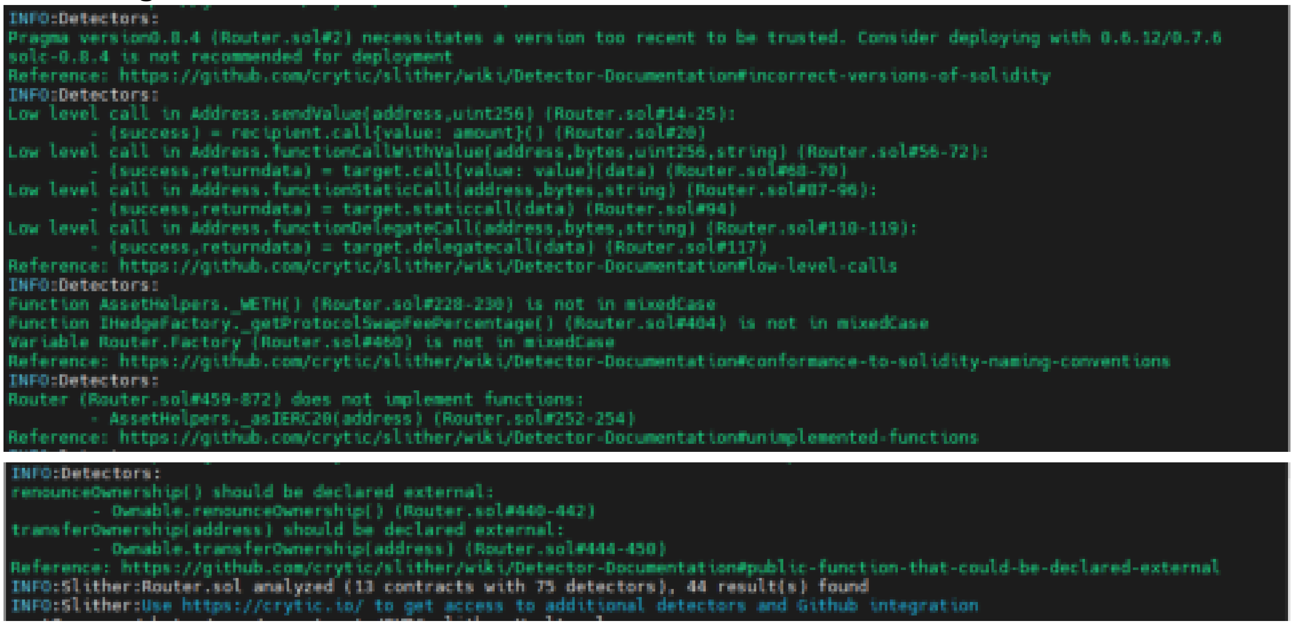
```
INFO:Detectors:
WeightedMath._calcOutGivenIn(uint256,uint256,uint256,uint256,uint256) (WeightedMath.sol#11-21) is never used and should be rem
oved
WeightedMath._calculateVirtualSwapAmountIn(uint256,uint256,uint256,uint256,uint256,uint256) (WeightedMath.sol#25-35) is never
used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (WeightedMath.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:
WeightedMath.ONE (WeightedMath.sol#7) is never used in WeightedMath (WeightedMath.sol#5-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Slither:WeightedMath.sol analyzed (1 contracts with 75 detectors), 5 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> WETH9.sol



```
INFO:Detectors:
Pragma version^0.8.4 (WETH9.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:
Variable WETH9.MAX_INT (WETH9.sol#10) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in WETH9.withdraw(uint256) (WETH9.sol#33-38):
        External calls:
        - address(msg.sender).transfer(wad) (WETH9.sol#36)
        Event emitted after the call(s):
        - Withdrawal(msg.sender,wad) (WETH9.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
WETH9.constructor() (WETH9.sol#20-22) uses literals with too many digits:
        - balanceOf[msg.sender] = 10000000000000000000000000000000000 (WETH9.sol#21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
WETH9.MAX_INT (WETH9.sol#10) should be constant
WETH9.decimals (WETH9.sol#8) should be constant
WETH9.name (WETH9.sol#6) should be constant
WETH9.symbol (WETH9.sol#7) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
withdraw(uint256) should be declared external:
        - WETH9.withdraw(uint256) (WETH9.sol#33-38)
totalSupply() should be declared external:
        - WETH9.totalSupply() (WETH9.sol#40-42)
approve(address,uint256) should be declared external:
        - WETH9.approve(address,uint256) (WETH9.sol#44-48)
transfer(address,uint256) should be declared external:
        - WETH9.transfer(address,uint256) (WETH9.sol#50-52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:WETH9.sol analyzed (1 contracts with 75 detectors), 13 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

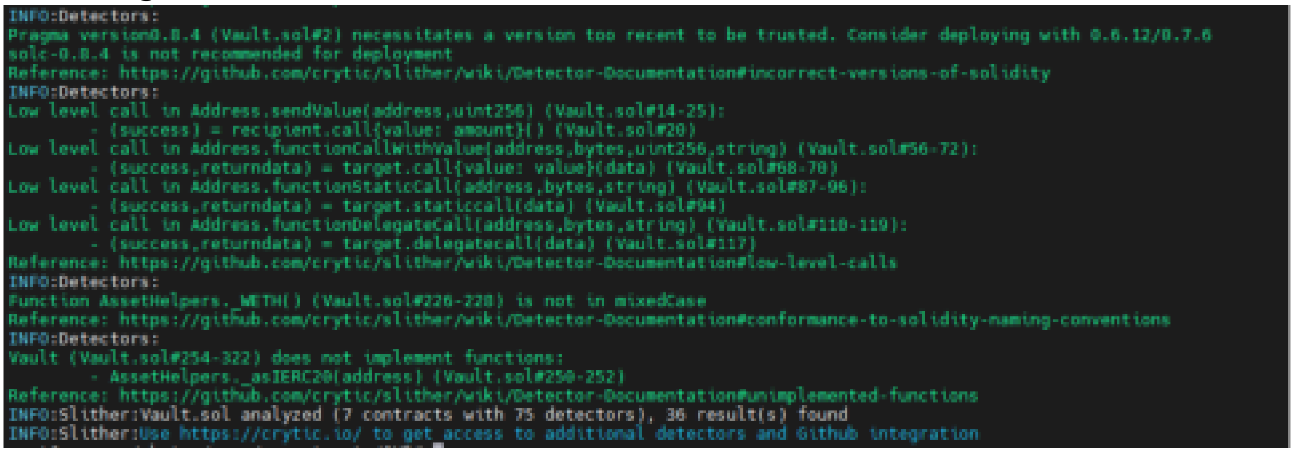
Slither log >> Router.sol



```
INFO:Detectors:
Pragma version0.8.4 (Router.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 Address.sendValue(address,uint256) (Router.sol#14-25):
        - (success) = recipient.call{value: amount}() (Router.sol#20)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Router.sol#56-72):
        - (success,returndata) = target.call{value: value}(data) (Router.sol#68-70)
Low level call in Address.functionStaticCall(address,bytes,string) (Router.sol#87-96):
        - (success,returndata) = target.staticcall(data) (Router.sol#94)
Low level call in Address.functionDelegateCall(address,bytes,string) (Router.sol#110-119):
        - (success,returndata) = target.delegatecall(data) (Router.sol#117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function AssetHelpers._WETH() (Router.sol#228-230) is not in mixedCase
Function IHedgeFactory._getProtocolSwapFeePercentage() (Router.sol#404) is not in mixedCase
Variable Router.Factory (Router.sol#460) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Router (Router.sol#459-872) does not implement functions:
        - AssetHelpers._asIERC20(address) (Router.sol#252-254)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
```

```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Router.sol#440-442)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Router.sol#444-450)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Router.sol analyzed (13 contracts with 75 detectors), 44 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

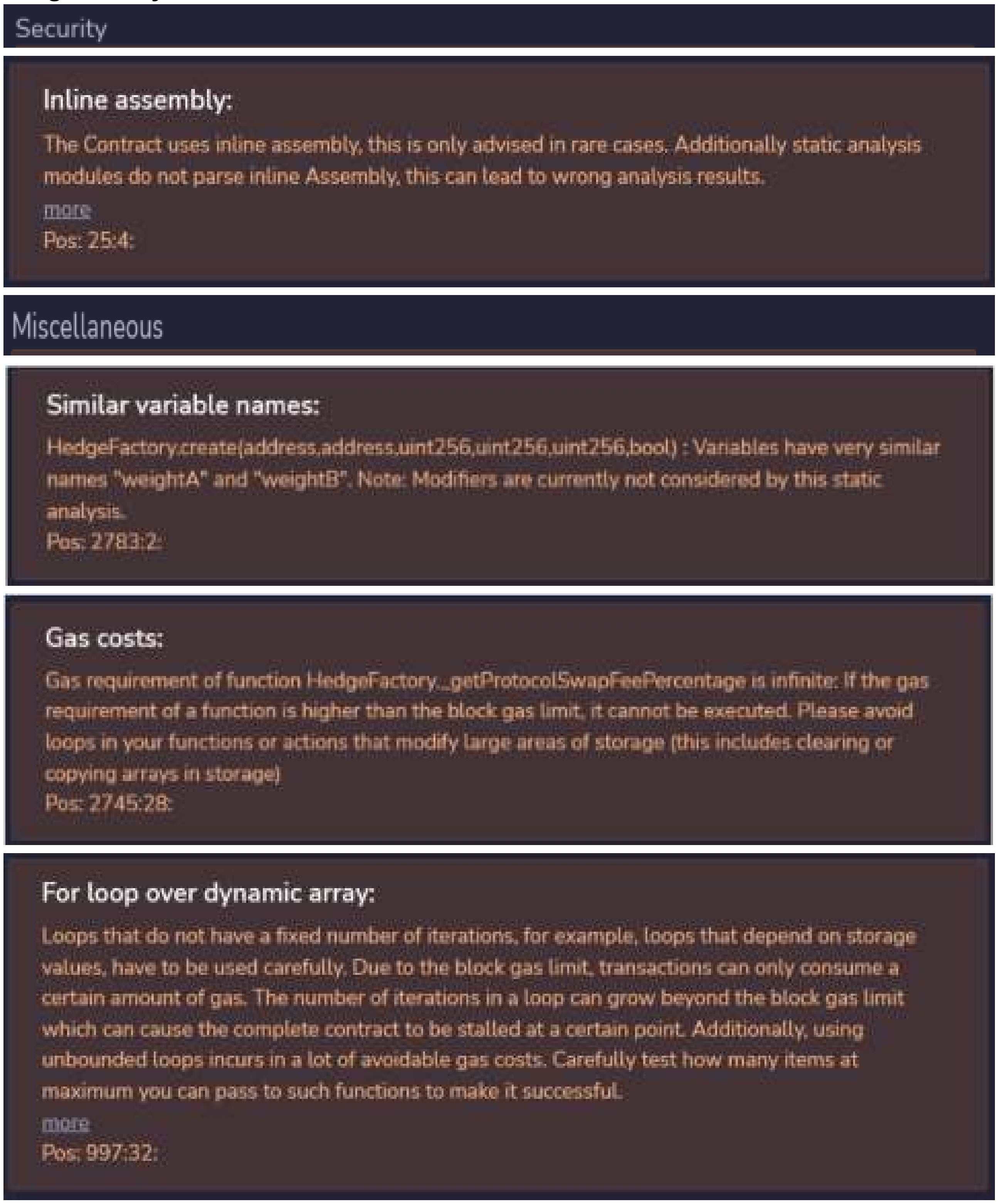
Slither log >> Vault.sol



```
INFO:Detectors:
Pragma version0.8.4 (Vault.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 Address.sendValue(address,uint256) (Vault.sol#14-25):
        - (success) = recipient.call{value: amount}() (Vault.sol#20)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Vault.sol#56-72):
        - (success,returndata) = target.call{value: value}(data) (Vault.sol#68-70)
Low level call in Address.functionStaticCall(address,bytes,string) (Vault.sol#87-96):
        - (success,returndata) = target.staticcall(data) (Vault.sol#94)
Low level call in Address.functionDelegateCall(address,bytes,string) (Vault.sol#110-119):
        - (success,returndata) = target.delegatecall(data) (Vault.sol#117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function AssetHelpers._WETH() (Vault.sol#226-228) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Vault (Vault.sol#254-322) does not implement functions:
        - AssetHelpers._asIERC20(address) (Vault.sol#250-252)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Slither:Vault.sol analyzed (7 contracts with 75 detectors), 36 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

HedgeFactory.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: 25:4:

Miscellaneous

Similar variable names:

HedgeFactory.create(address,address,uint256,uint256,uint256,bool) : Variables have very similar names "weightA" and "weightB". Note: Modifiers are currently not considered by this static analysis.

Pos: 2783:2:

Gas costs:

Gas requirement of function HedgeFactory._getProtocolSwapFeePercentage 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: 2745:28:

For loop over dynamic array:

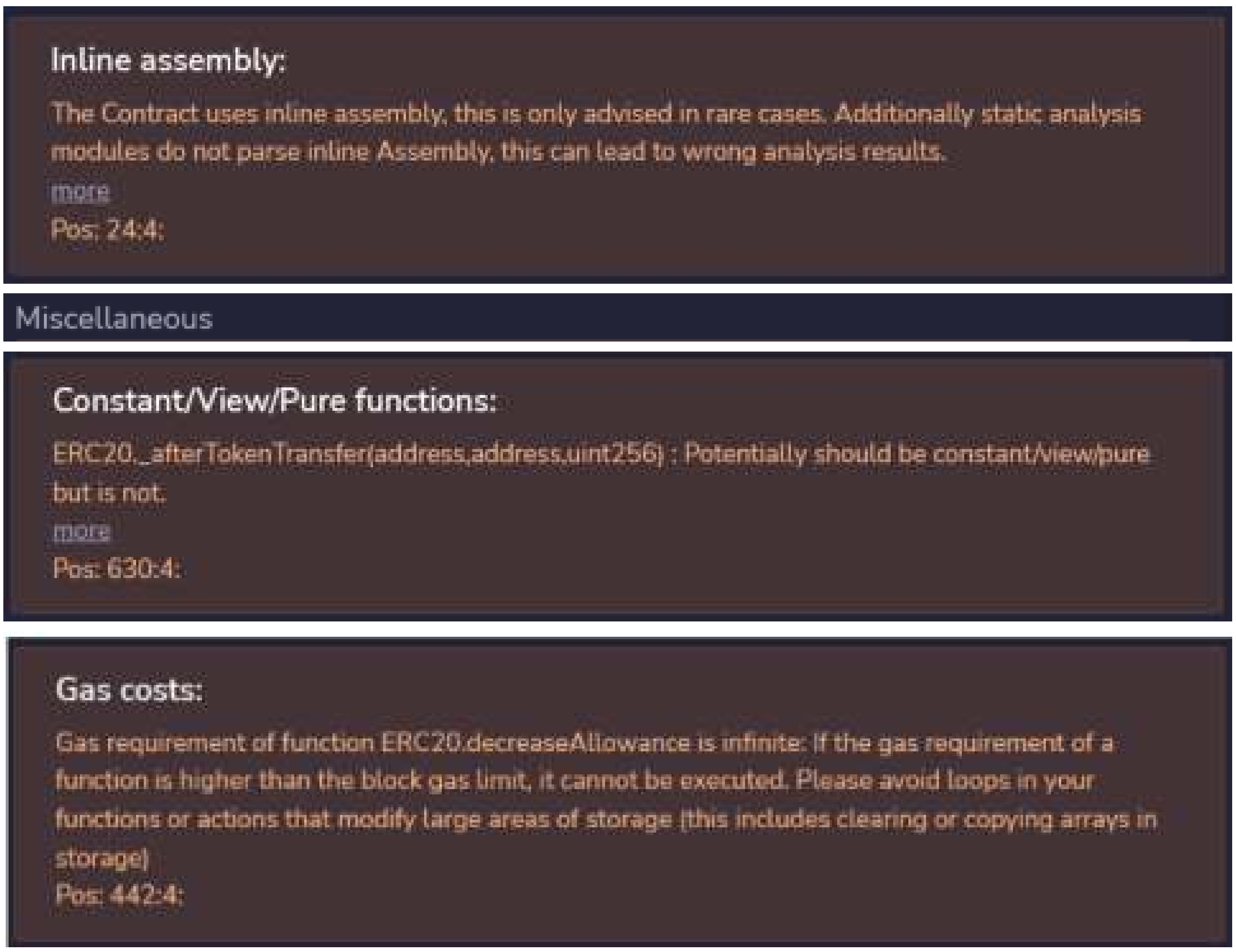
Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 997:32:

HedgePoolToken.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: 24:4:

Miscellaneous

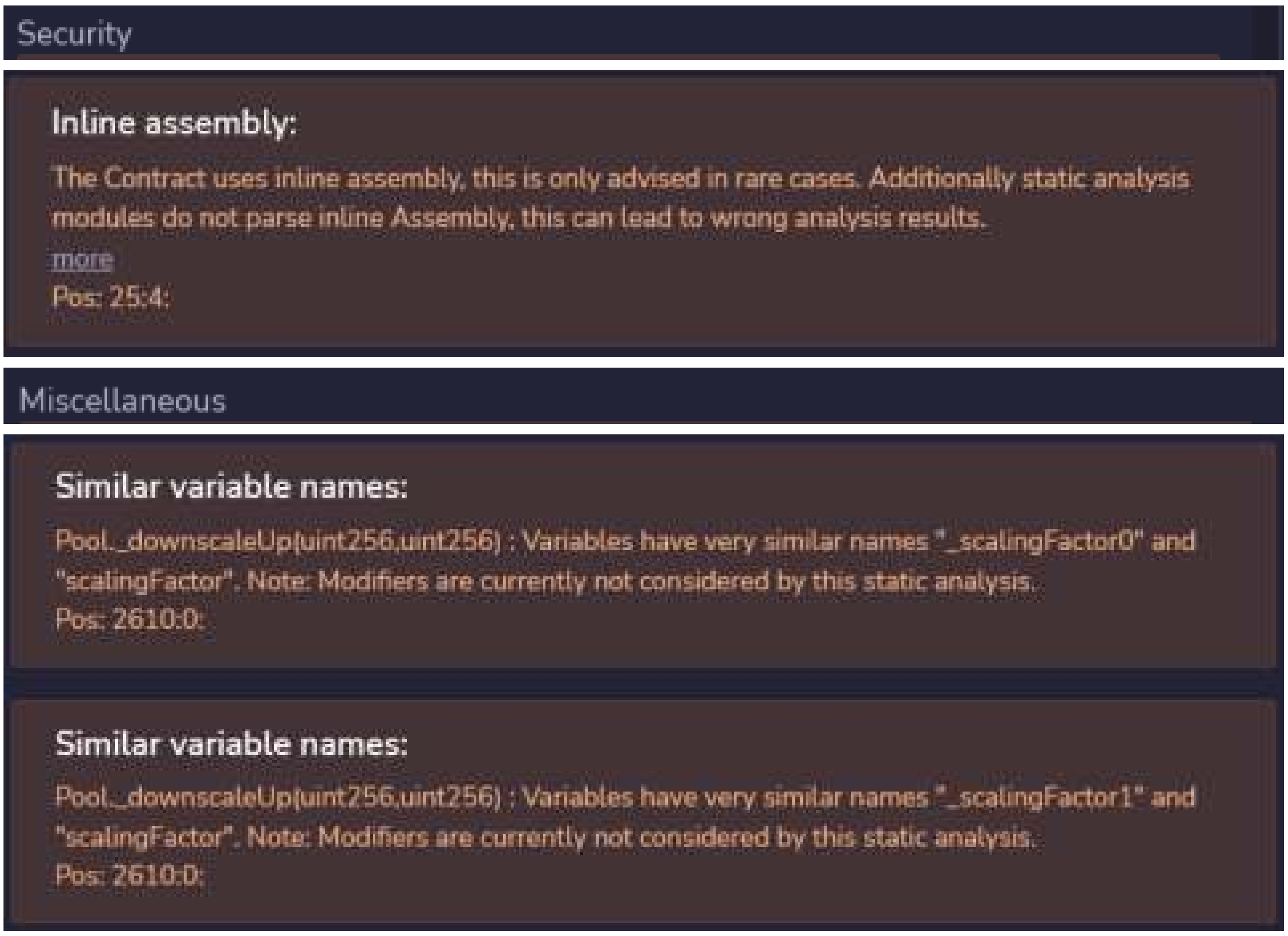
Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 630:4:

Gas costs:

Gas requirement of function ERC20.decreaseAllowance 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: 442:4:

Pool.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: 25:4:

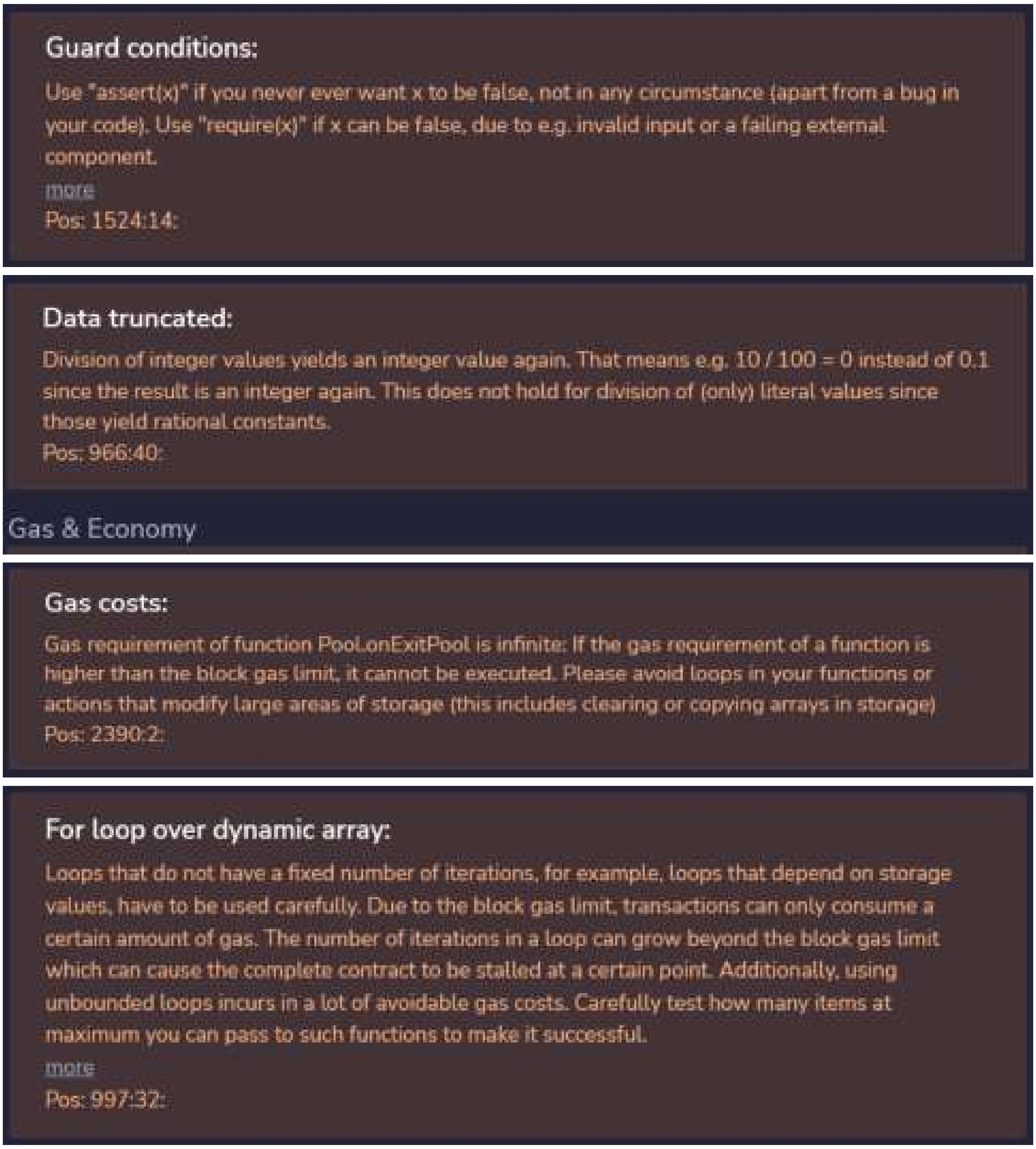
Miscellaneous

Similar variable names:

Pool._downscaleUp(uint256,uint256) : Variables have very similar names "_scalingFactor0" and "scalingFactor". Note: Modifiers are currently not considered by this static analysis.
Pos: 2610:0:

Similar variable names:

Pool._downscaleUp(uint256,uint256) : Variables have very similar names "_scalingFactor1" and "scalingFactor". Note: Modifiers are currently not considered by this static analysis.
Pos: 2610:0:


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: 1524:14:

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: 966:40:

Gas & Economy

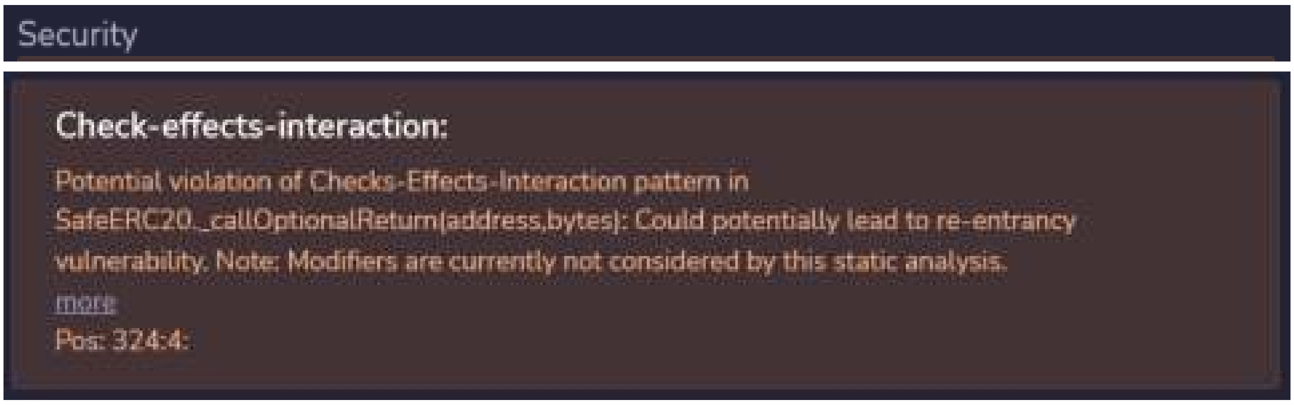
Gas costs:

Gas requirement of function Pool.onExitPool 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: 2390:2:

For loop over dynamic array:

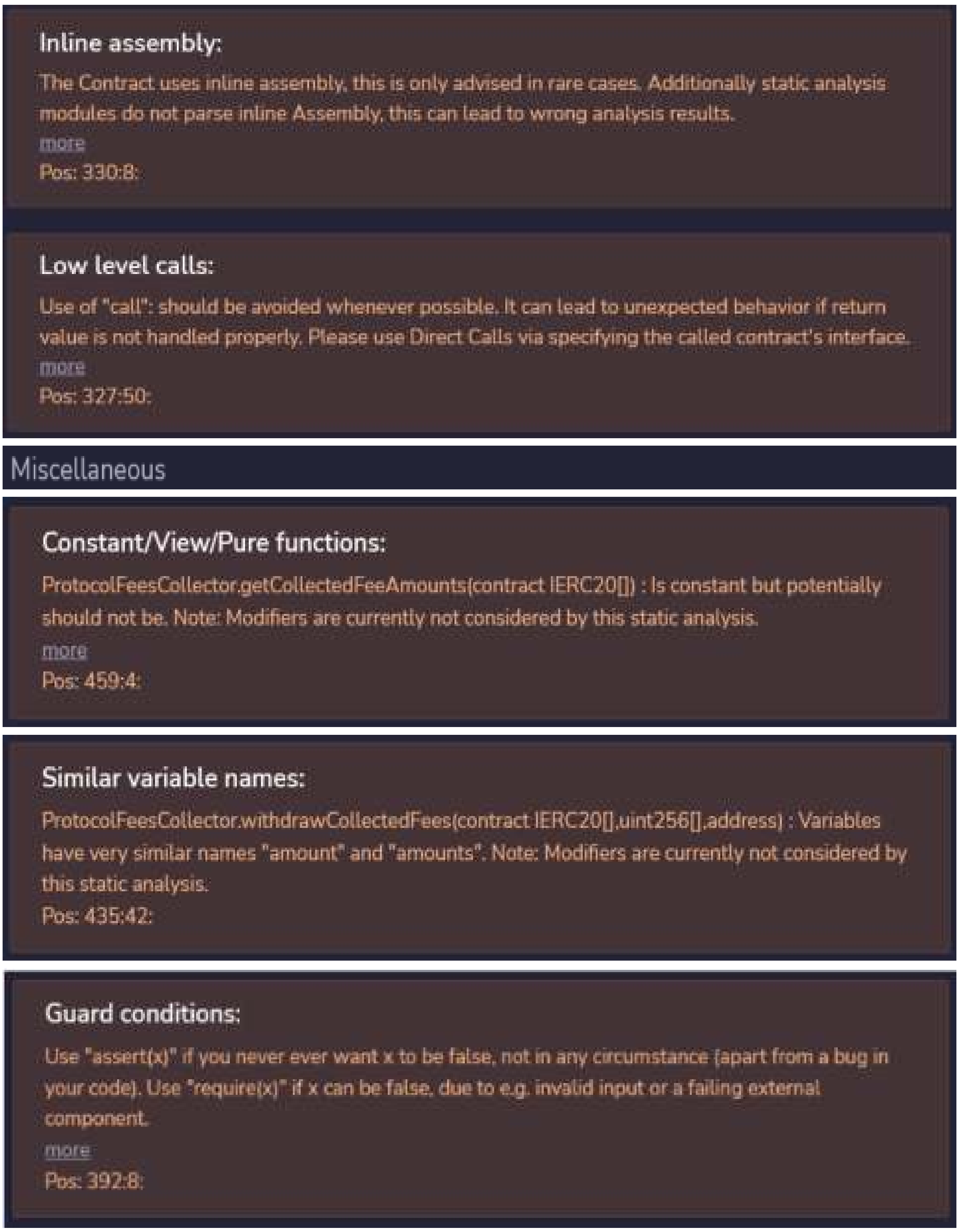
Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 997:32:

ProtocolFeesCollector.sol


Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20._callOptionalReturn(address,bytes): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 324:4:



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: 330:8:

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: 327:50:

Miscellaneous

Constant/View/Pure functions:

ProtocolFeesCollector.getCollectedFeeAmounts(contract IERC20[]) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 459:4:

Similar variable names:

ProtocolFeesCollector.withdrawCollectedFees(contract IERC20[],uint256[],address) : Variables have very similar names "amount" and "amounts". Note: Modifiers are currently not considered by this static analysis.

Pos: 435:42:

Guard conditions:

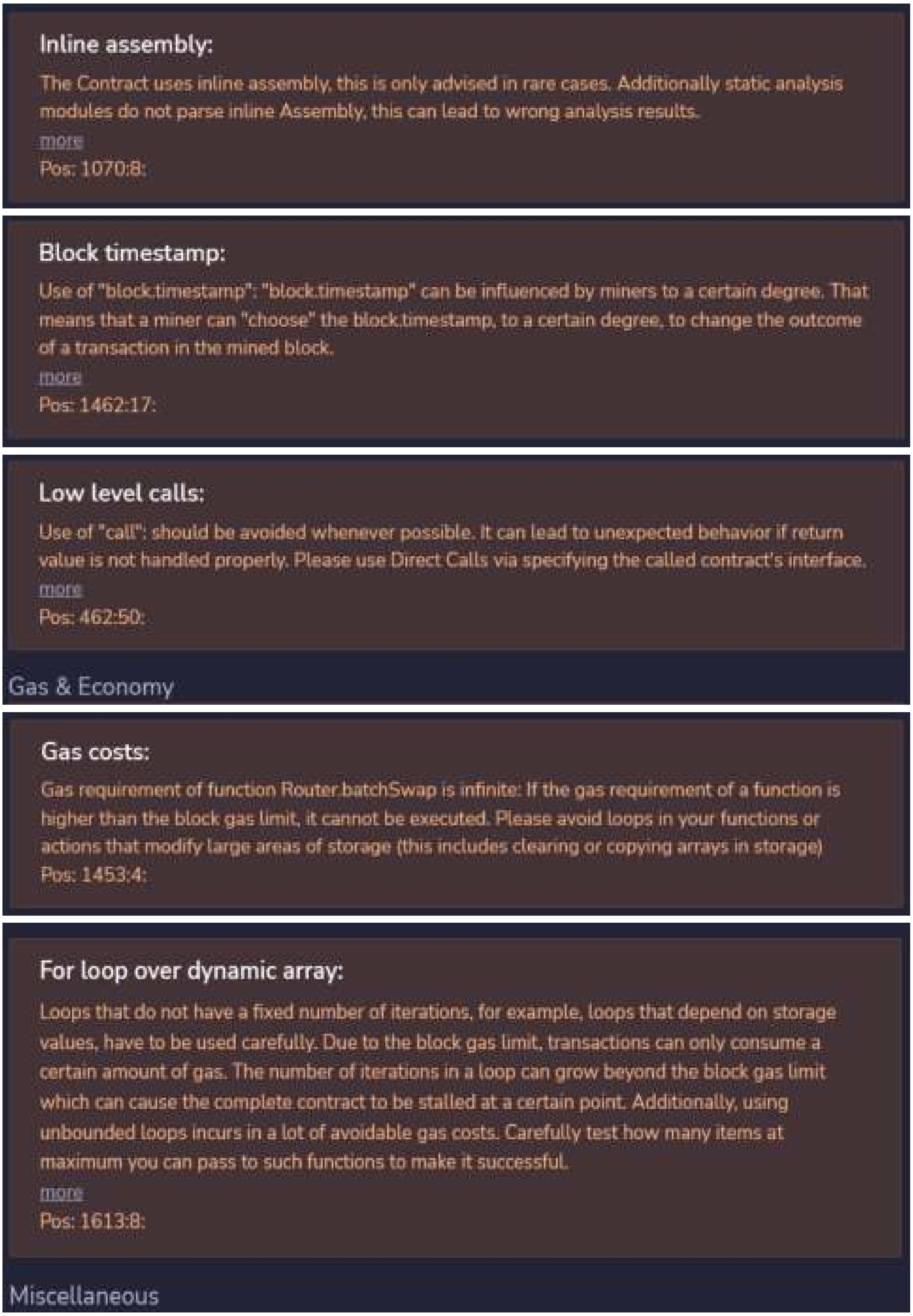
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 392:8:

Router.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: 1070:8:

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: 1462:17:

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: 462:50:

Gas & Economy

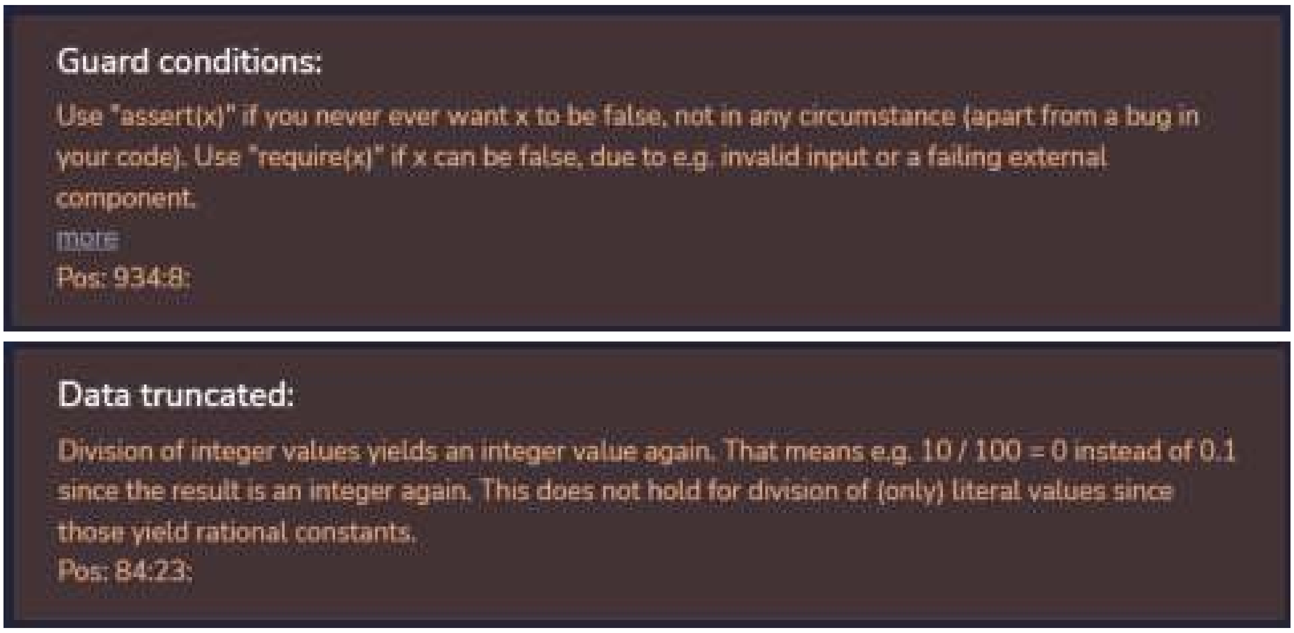
Gas costs:

Gas requirement of function Router.batchSwap 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: 1453:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1613:8:

Miscellaneous

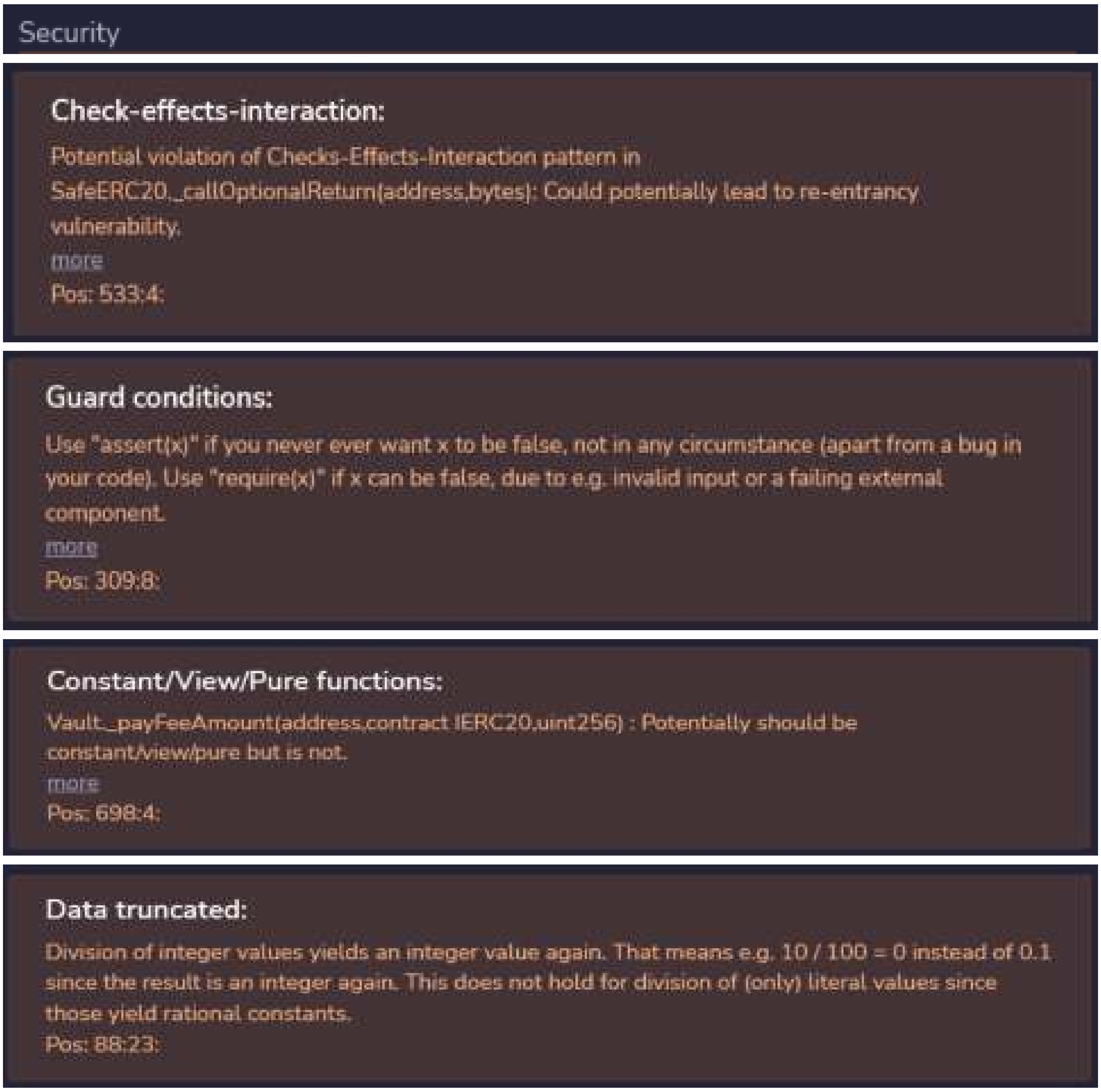

Guard conditions:

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: 934:8:

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: 84:23:

Vault.sol


Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20._callOptionalReturn(address,bytes): Could potentially lead to re-entrancy vulnerability.
more
Pos: 533:4:

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: 309:8:

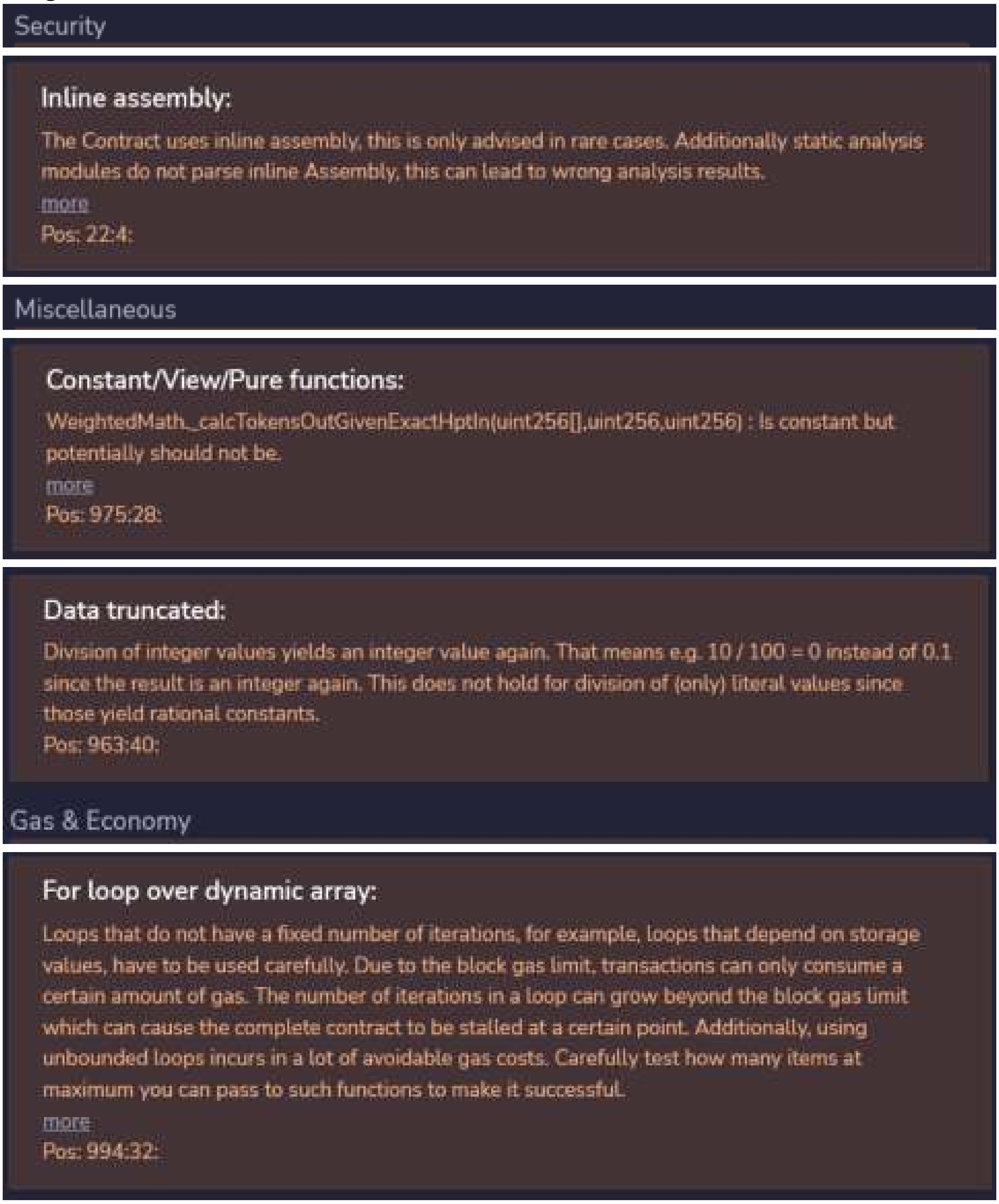
Constant/View/Pure functions:

Vault._payFeeAmount(address,contract IERC20,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 698:4:

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: 88:23:

WeightedMath.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: 22:4:

Miscellaneous

Constant/View/Pure functions:

WeightedMath._calcTokensOutGivenExactHptIn(uint256[],uint256,uint256) : Is constant but potentially should not be.

more

Pos: 975:28:

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: 963:40:

Gas & Economy

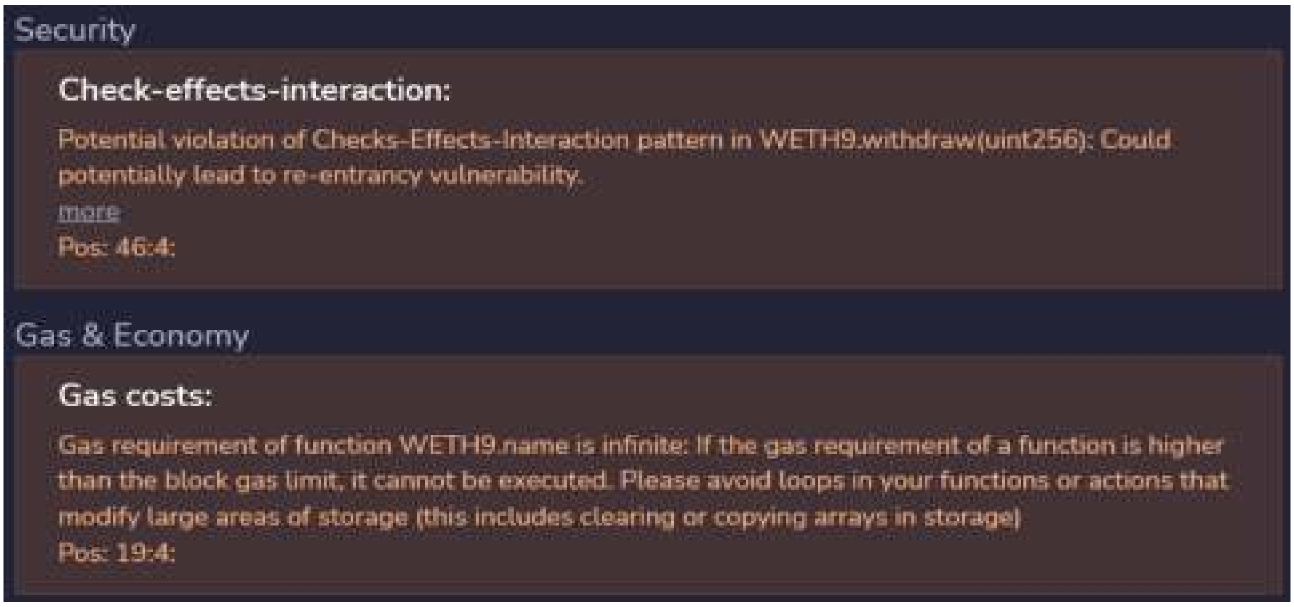
For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 994:32:

WETH9.sol



Security

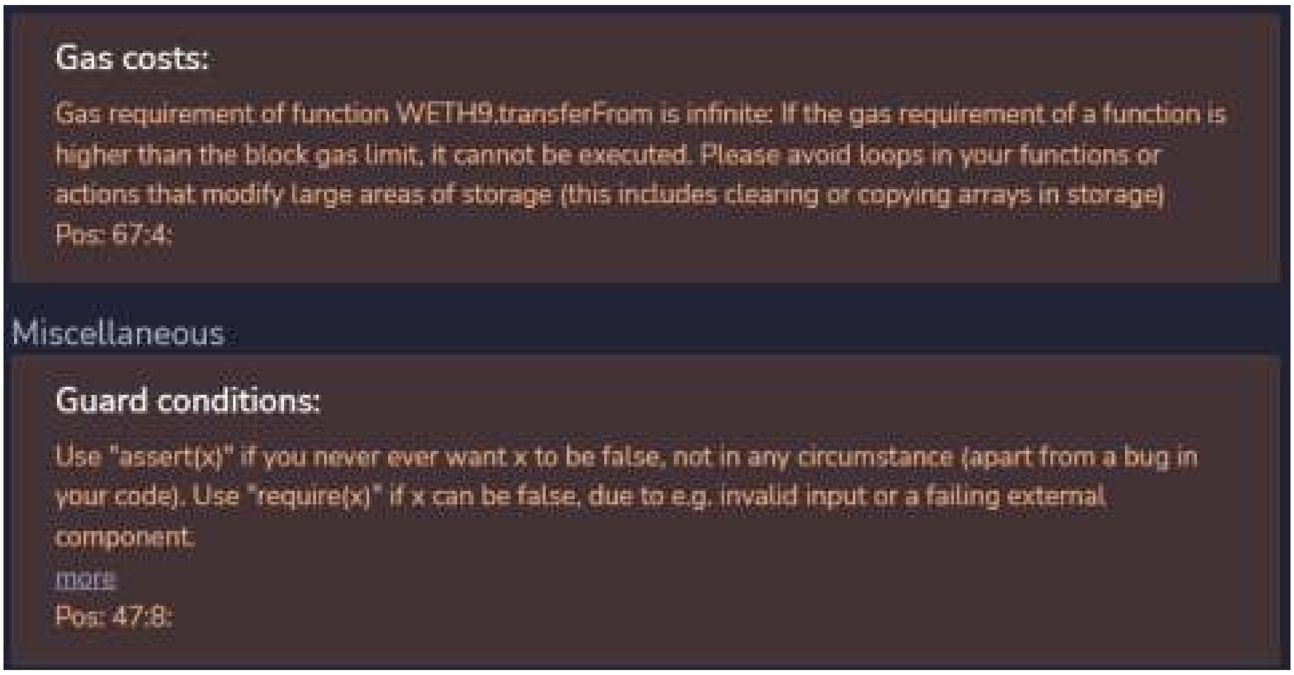
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in WETH9.withdraw(uint256): Could potentially lead to re-entrancy vulnerability.
more
Pos: 46:4:

Gas & Economy

Gas costs:

Gas requirement of function WETH9.name 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: 19:4:



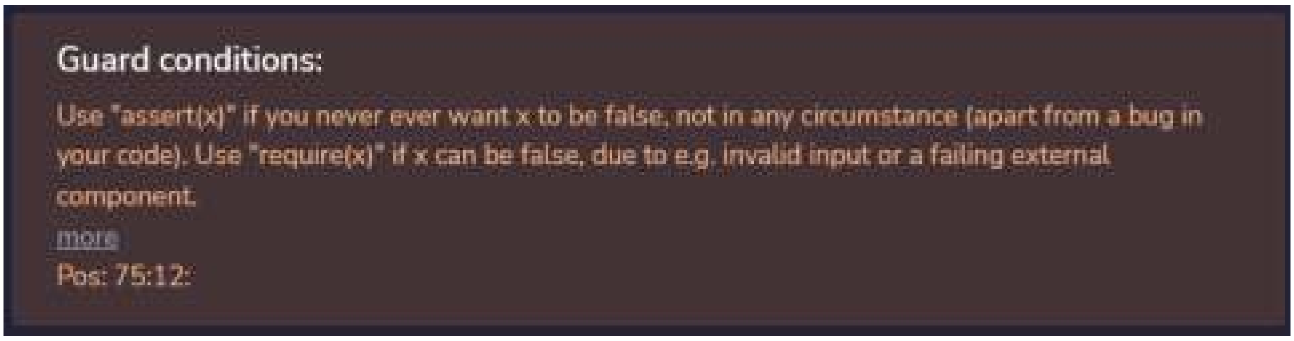
Gas costs:

Gas requirement of function WETH9.transferFrom 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: 67:4:

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: 47:8:



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: 75:12:

Solhint Linter

HedgeFactory.sol

```
HedgeFactory.sol:1390:18: Error: Parse error: missing ';' at '{'
HedgeFactory.sol:1426:18: Error: Parse error: missing ';' at '{'
HedgeFactory.sol:1473:18: Error: Parse error: missing ';' at '{'
HedgeFactory.sol:1527:22: Error: Parse error: missing ';' at '{'
```

HedgePoolToken.sol

```
HedgePoolToken.sol:453:18: Error: Parse error: missing ';' at '{'
HedgePoolToken.sol:489:18: Error: Parse error: missing ';' at '{'
HedgePoolToken.sol:536:18: Error: Parse error: missing ';' at '{'
HedgePoolToken.sol:590:22: Error: Parse error: missing ';' at '{'
```

Pool.sol

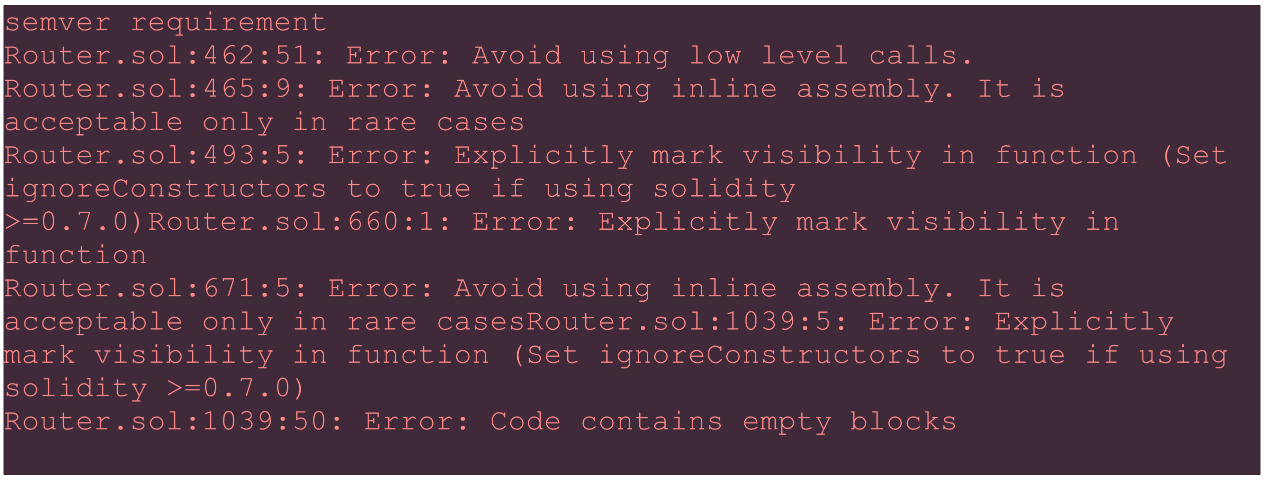
```
Pool.sol:1390:18: Error: Parse error: missing ';' at '{'
Pool.sol:1426:18: Error: Parse error: missing ';' at '{'
Pool.sol:1473:18: Error: Parse error: missing ';' at '{'
Pool.sol:1527:22: Error: Parse error: missing ';' at '{'
```

ProtocolFeesCollector.sol

```
ProtocolFeesCollector.sol:2:1: Error: Compiler version 0.8.11 does
not satisfy the r semver requirementProtocolFeesCollector.sol:23:5:
Error: Avoid using inline assembly. It is acceptable only in rare
cases
ProtocolFeesCollector.sol:154:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
ProtocolFeesCollector.sol:327:51: Error: Avoid using low level calls.
ProtocolFeesCollector.sol:330:9: Error: Avoid using inline assembly.
It is acceptable only in rare cases
ProtocolFeesCollector.sol:357:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
```

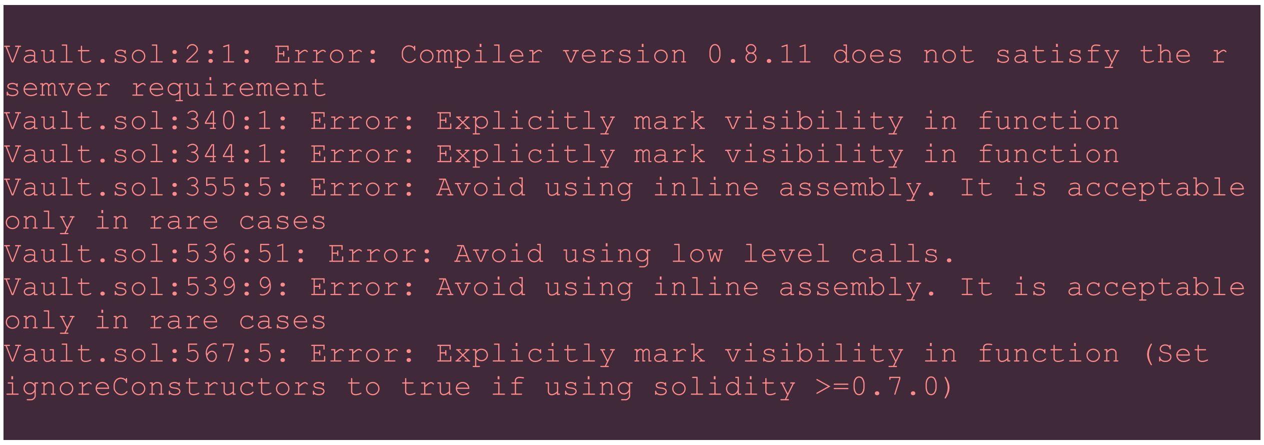
Router.sol

```
Router.sol:2:1: Error: Compiler version 0.8.11 does not satisfy the r
```



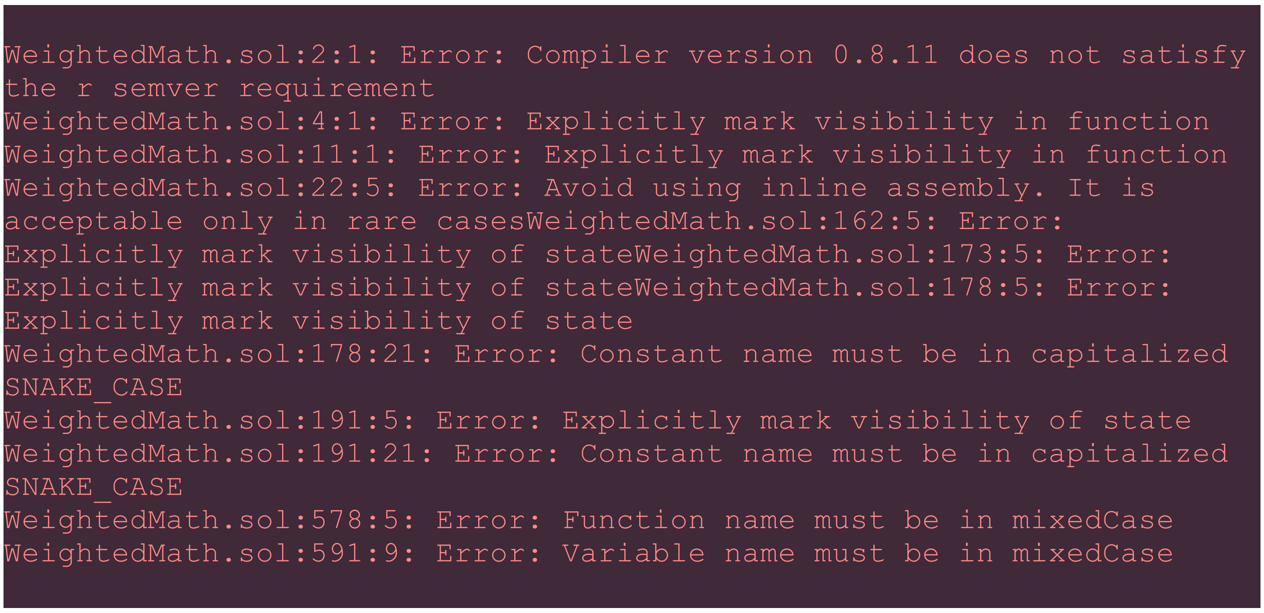
```
semver requirement
Router.sol:462:51: Error: Avoid using low level calls.
Router.sol:465:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Router.sol:493:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity
>=0.7.0)Router.sol:660:1: Error: Explicitly mark visibility in
function
Router.sol:671:5: Error: Avoid using inline assembly. It is
acceptable only in rare casesRouter.sol:1039:5: Error: Explicitly
mark visibility in function (Set ignoreConstructors to true if using
solidity >=0.7.0)
Router.sol:1039:50: Error: Code contains empty blocks
```

Vault.sol



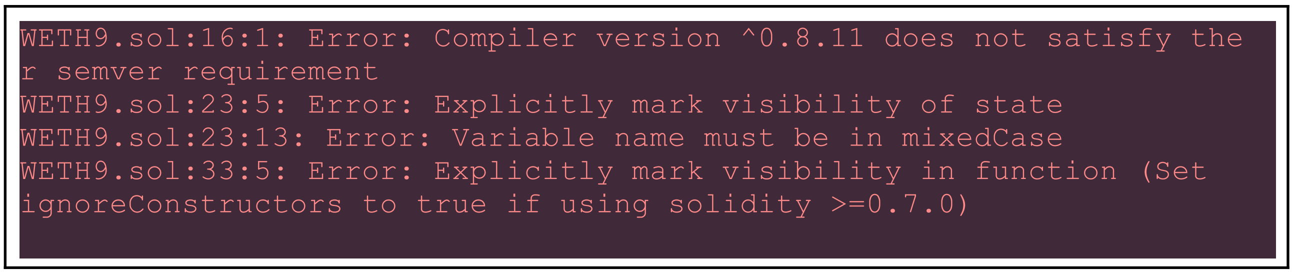
```
Vault.sol:2:1: Error: Compiler version 0.8.11 does not satisfy the r
semver requirement
Vault.sol:340:1: Error: Explicitly mark visibility in function
Vault.sol:344:1: Error: Explicitly mark visibility in function
Vault.sol:355:5: Error: Avoid using inline assembly. It is acceptable
only in rare cases
Vault.sol:536:51: Error: Avoid using low level calls.
Vault.sol:539:9: Error: Avoid using inline assembly. It is acceptable
only in rare cases
Vault.sol:567:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
```

WeightedMath.sol



```
WeightedMath.sol:2:1: Error: Compiler version 0.8.11 does not satisfy
the r semver requirement
WeightedMath.sol:4:1: Error: Explicitly mark visibility in function
WeightedMath.sol:11:1: Error: Explicitly mark visibility in function
WeightedMath.sol:22:5: Error: Avoid using inline assembly. It is
acceptable only in rare casesWeightedMath.sol:162:5: Error:
Explicitly mark visibility of stateWeightedMath.sol:173:5: Error:
Explicitly mark visibility of stateWeightedMath.sol:178:5: Error:
Explicitly mark visibility of state
WeightedMath.sol:178:21: Error: Constant name must be in capitalized
SNAKE_CASE
WeightedMath.sol:191:5: Error: Explicitly mark visibility of state
WeightedMath.sol:191:21: Error: Constant name must be in capitalized
SNAKE_CASE
WeightedMath.sol:578:5: Error: Function name must be in mixedCase
WeightedMath.sol:591:9: Error: Variable name must be in mixedCase
```

WETH9.sol



```
WETH9.sol:16:1: Error: Compiler version ^0.8.11 does not satisfy the
r semver requirement
WETH9.sol:23:5: Error: Explicitly mark visibility of state
WETH9.sol:23:13: Error: Variable name must be in mixedCase
WETH9.sol:33:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
```

Software analysis result:

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