EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	AGS Finance Protocol
Website:	https://ags.finance
Platform:	Astar Network
Language:	Solidity
Date:	May 10th, 2022

Table of contents

Introduction

EtherAuthority was contracted by the AGS Finance team to perform the Security audit of the AGS Finance 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 May 10th, 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 AGS Finance Contracts have functions like safeAgsTransfer, delegateBySig, getPriorVotes, withdrawAll, harvest, setTreasury, safeAgsTransfer, emergencyWithdraw, leaveStaking, createPair, allPairsLength, getAmountsOut, quote, etc.

Audit scope

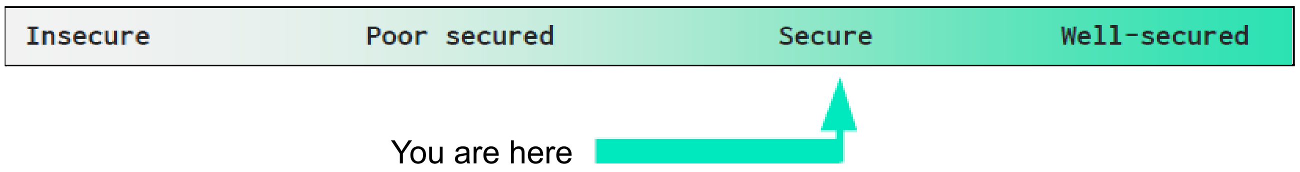
Name	**Code Review and Security Analysis Report for AGS Finance Protocol Smart Contracts**
Platform	**Astar / Solidity**
File 1	AgsRouter.sol
File 1 MD5 Hash	DBD0DDCA78C5BFFC84C384345AE1C7C3
File 2	AgsFactory.sol
File 2 MD5 Hash	599A3543BF3EC943FA16361FC705DB5C
File 3	MasterGrimace.sol
File 3 MD5 Hash	25880AD1D7252F826EF6E997EDF04399
File 4	AgsVault.sol
File 4 MD5 Hash	1635ED38C5F796EE544465813D0C3EC2
File 5	SyrupBar.sol
File 5 MD5 Hash	2CBD197D511776808B4A945A2CAC62E3
Audit Date	May 10th, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 AgsRouter.sol ● AgsRouter has functions like: receive, addLiquidity, removeLiquidity, swapTokensForExactTokens,etc.	**YES, This is valid.**
File 2 AgsFactory.sol ● AgsFactory has functions like: allPairsLength, createPair, setFeeTo, setFeeToSetter.	**YES, This is valid.**
File 3 MasterGrimace.sol ● NFT Boost Rate: 1% ● Bonus Multiplier: 1	**YES, This is valid. Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**
File 4 AgsVault.sol ● Maximum Performance Fee: 5% ● Maximum Call Fee: 1% ● Maximum Withdraw Fee: 1% ● Maximum Withdraw Fee Period: 3 Days ● Performance Fee: 2% ● Call Fee: 0.25% ● Withdraw Fee: 0.1% ● Withdraw Fee Period: 3 Days	**YES, This is valid. Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**
File 5 SyrupBar.sol ● Name: SyrupBar Token ● Symbol: SYRUP	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. This token contract does contain owner control, which does not make it 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 3 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	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Moderated
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	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	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Moderated
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 5 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 AGS Finance 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 AGS Finance Protocol.

The AGS Finance 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 well commented on smart contracts.

Documentation

We were given an AGS Finance Protocol smart contract code in the form of a blockscout astar weblink. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **well** commented. 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 official website https://ags.finance which provided rich information about the project architecture and tokenomics.

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

AgsRouter.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	ensure	modifier	Passed	No Issue
3	receive	external	Passed	No Issue
4	_addLiquidity	internal	Passed	No Issue
5	addLiquidity	external	Passed	No Issue
6	addLiquidityETH	external	Passed	No Issue
7	removeLiquidity	write	Passed	No Issue
8	removeLiquidityETH	write	Passed	No Issue
9	removeLiquidityWithPermit	external	Passed	No Issue
10	removeLiquidityETHWithPermit	external	Passed	No Issue
11	removeLiquidityETHSupportingFeeOnTransferTokens	external	Passed	No Issue
12	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	external	Passed	No Issue
13	_swap	internal	Passed	No Issue
14	swapExactTokensForTokens	external	Passed	No Issue
15	swapTokensForExactTokens	external	Passed	No Issue
16	swapExactETHForTokens	external	Passed	No Issue
17	swapTokensForExactETH	external	Passed	No Issue
18	swapExactTokensForETH	external	Passed	No Issue
19	swapETHForExactTokens	external	Passed	No Issue
20	_swapSupportingFeeOnTransferTokens	internal	Passed	No Issue
21	swapExactTokensForTokensSupportingFeeOnTransferTokens	internal	Passed	No Issue
22	swapExactETHForTokensSupportingFeeOnTransferTokens	external	Passed	No Issue
23	swapExactTokensForETHSupportingFeeOnTransferTokens	external	Passed	No Issue
24	quote	write	Passed	No Issue
25	getAmountOut	write	Passed	No Issue
26	getAmountIn	write	Passed	No Issue
27	getAmountsOut	read	Passed	No Issue
28	getAmountsIn	read	Passed	No Issue

AgsFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	allPairsLength	external	Passed	No Issue
3	createPair	external	Passed	No Issue
4	setFeeTo	external	Passed	No Issue
5	setFeeToSetter	external	Passed	No Issue

MasterGrimace.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	onlyWhitelisted	modifier	Passed	No Issue
8	isWhitelist	read	Passed	No Issue
9	setWhitelist	external	access only Owner	No Issue
10	disableWhitelist	external	access only Owner	No Issue
11	nonDuplicated	modifier	Passed	No Issue
12	nonContract	modifier	Passed	No Issue
13	getBoost	read	Passed	No Issue
14	getSlots	read	Passed	No Issue
15	getTokenIds	read	Passed	No Issue
16	poolLength	external	Passed	No Issue
17	getMultiplier	write	Passed	No Issue
18	pendingAgs	external	Passed	No Issue
19	add	write	Critical operation lacks event log, Function input parameters lack of check	Refer Audit Findings
20	set	write	Critical operation lacks event log	Refer Audit Findings
21	updateStakingPool	internal	Passed	No Issue
22	depositNFT	write	Passed	No Issue
23	withdrawNFT	write	Passed	No Issue
24	massUpdatePools	write	Critical operation lacks event log, Infinite loop	No Issue

25	updatePool	write	Critical operation lacks event log	Refer Audit Findings
26	deposit	write	Passed	No Issue
27	withdraw	write	Passed	No Issue
28	enterStaking	write	Passed	No Issue
29	leaveStaking	write	Passed	No Issue
30	emergencyWithdraw	write	Passed	No Issue
31	safeAgsTransfer	internal	Passed	No Issue
32	updateEmissionRate	write	access only Owner	No Issue
33	setNftController	write	Function input parameters lack of check	Refer Audit Findings
34	setNftBoostRate	write	access only Owner	No Issue
35	flipWhitelistAll	write	access only Owner	No Issue
36	setEnableNFTBoost	external	access only Owner	No Issue
37	dev	write	Function input parameters lack of check	Refer Audit Findings
38	setStartBlock	external	access only Owner	No Issue

AgsVault.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	paused	read	Passed	No Issue
8	whenNotPaused	modifier	Passed	No Issue
9	whenPaused	modifier	Passed	No Issue
10	_pause	internal	Passed	No Issue
11	_unpause	internal	Passed	No Issue
12	onlyAdmin	modifier	Passed	No Issue
13	notContract	modifier	Passed	No Issue
14	deposit	external	Passed	No Issue
15	withdrawAll	external	Passed	No Issue
16	harvest	external	Passed	No Issue
17	setAdmin	external	access only Owner	No Issue
18	setTreasury	external	access only Owner	No Issue
19	setPerformanceFee	external	access only Admin	No Issue
20	setCallFee	external	access only Admin	No Issue
21	setWithdrawFee	external	access only Admin	No Issue
22	setWithdrawFeePeriod	external	access only Admin	No Issue
23	emergencyWithdraw	external	access only Admin	No Issue

24	inCaseTokensGetStuck	external	access only Admin	No Issue
25	pause	external	access only Admin	No Issue
26	unpause	external	access only Admin	No Issue
27	calculateHarvestCakeRewards	external	Passed	No Issue
28	calculateTotalPendingCakeRewards	external	Passed	No Issue
29	getPricePerFullShare	external	Passed	No Issue
30	withdraw	write	Passed	No Issue
31	available	read	Passed	No Issue
32	balanceOf	read	Passed	No Issue
33	_earn	internal	Passed	No Issue
34	_isContract	internal	Passed	No Issue

SyrupBar.sol

Functions

Sl.	**Functions**	**Type**	**Observation**	**Conclusion**
1	constructor	write	Passed	No Issue
2	getOwner	external	Passed	No Issue
3	name	write	Passed	No Issue
4	decimals	write	Passed	No Issue
5	symbol	write	Passed	No Issue
6	totalSupply	write	Passed	No Issue
7	balanceOf	write	Passed	No Issue
8	transfer	write	Passed	No Issue
9	allowance	write	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	mint	write	access only Owner	No Issue
15	_transfer	internal	Passed	No Issue
16	_mint	internal	Passed	No Issue
17	_burn	internal	Passed	No Issue
18	_approve	internal	Passed	No Issue
19	_burnFrom	internal	Passed	No Issue
20	mint	external	Unlimited minting	Refer Audit Findings
21	burn	external	access only Owner	No Issue
22	safeAgsTransfer	external	access only Owner	No Issue
23	delegates	external	Passed	No Issue
24	delegate	external	Passed	No Issue
25	delegateBySig	external	Passed	No Issue
26	getCurrentVotes	external	Passed	No Issue
27	getPriorVotes	external	Infinite Loop	Refer Audit Findings

28	_delegate	internal	Passed	No Issue
29	_moveDelegates	internal	Passed	No Issue
30	_writeCheckpoint	internal	Passed	No Issue
31	safe32	internal	Passed	No Issue
32	getChainId	internal	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) Infinite Loop:

SyrupBar.sol

In the getPriorVotes function, if the upper value is too high than lower, then it will consume a lot of gas. It may possibly hit the block gas limit.

Resolution: The nCheckpoints should be kept limited, so it does not execute a lot of code blocks.

MasterGrimace.sol

In below functions ,for loops do not have pid length limit , which costs more gas : massUpdatePools.

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

(2) Critical operation lacks event log: **MasterGrimace.sol**

Missing event log for:

- add
- set
- updatePool
- depositNFT
- withdrawNFT

Resolution: Write an event log for listed events.

(3) Function input parameters lack of check: **MasterGrimace.sol**

Variable validation is not performed in below functions:

- add
- setNftController
- dev

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

Very Low / Informational / Best practices:

(1) Unlimited minting: **SyrupBar.sol**

Owner can mint unlimited tokens.

Resolution: We suggest putting a minting limit.

(2) Solidity version: **SyrupBar.sol, MasterGrimace.sol, AgsVault.sol**

Using the latest solidity will prevent any compiler-level bugs.

Resolution: We suggest using the latest solidity version.

(3) Immutable variables:

These variable values are set in the constructor & will be unchanged.

SyrupBar.sol

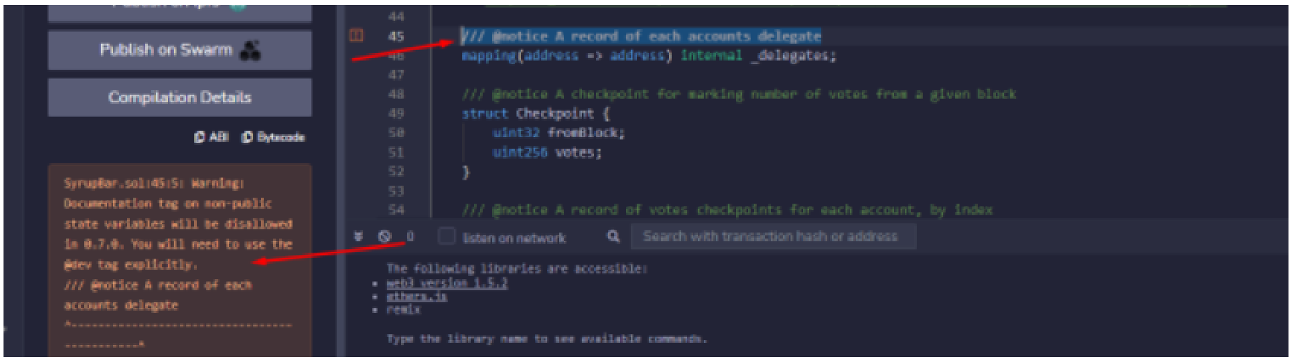
- ags

MasterGrimace.sol

- agsToken
- syrup

Resolution: We suggest setting these variables as immutable.

(4) Other Programming Issue: **SyrupBar.sol**



```
SyrupBar.sol:45:5: Warning:
Documentation tag on non-public
state variables will be disallowed
in 0.7.0. You will need to use the
@dev tag explicitly.
/// @notice A record of each
accounts delegate
^----------------------------------
-----------^
```

```
/// @notice A record of each accounts delegate
mapping(address => address) internal _delegates;

/// @notice A checkpoint for marking number of votes from a given block
struct Checkpoint {
    uint32 fromBlock;
    uint256 votes;
}

/// @notice A record of votes checkpoints for each account, by index
```

```
The following libraries are accessible:
• web3 version 1.5.2
• ethers.js
• remix

Type the library name to see available commands.
```

Warning: Documentation tag on non-public state variables will be disallowed in 0.7.0. You will need to use the @dev tag explicitly.

Resolution: We suggest replacing /// @notice with /// @dev.

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:

- mint: SyrupBar owner can create `_amount` token to `_to` Must only be called by the owner (MasterChef).
- burn: SyrupBar owner can burn token from account.
- safeAgsTransfer: SyrupBar owner can safe ags transfer function, just in case if rounding error causes pool to not have enough AGSs.
- setAdmin: AgsVault owner can set admin address.
- setTreasury: AgsVault owner can set treasury address.
- setPerformanceFee: AgsVault admin can set performance fees.
- setCallFee: AgsVault admin can set call fees.
- setWithdrawFee: AgsVault admin can set withdrawal fees.
- setWithdrawFeePeriod: AgsVault admin can set withdrawal fee period.
- emergencyWithdraw: AgsVault admin can withdraw unexpected tokens sent to the Cake Vault.
- pause: AgsVault admin can trigger a stopped state.
- unpause: AgsVault admin can return to normal state.
- add: MasterGrimace owner can add a new lp to the pool.
- set: MasterGrimace owner can update the given pool's AGS allocation point and deposit fee.
- updateEmissionRate: MasterGrimace owner can update emission rate.
- setNftController: MasterGrimace owner can set NFT controller address.
- setNftBoostRate: MasterGrimace owner can set NFT boostrate.
- flipWhitelistAll: MasterGrimace owner can flip whitelist all.
- setEnableNFTBoost: MasterGrimace owner can enable NFT Boost status.
- dev: MasterGrimace owner can update dev address by the previous dev.
- setStartBlock: MasterGrimace owner can set start block 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 files. And we have used all possible tests based on given objects as files. We have not observed any major issues in the smart contracts. **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.

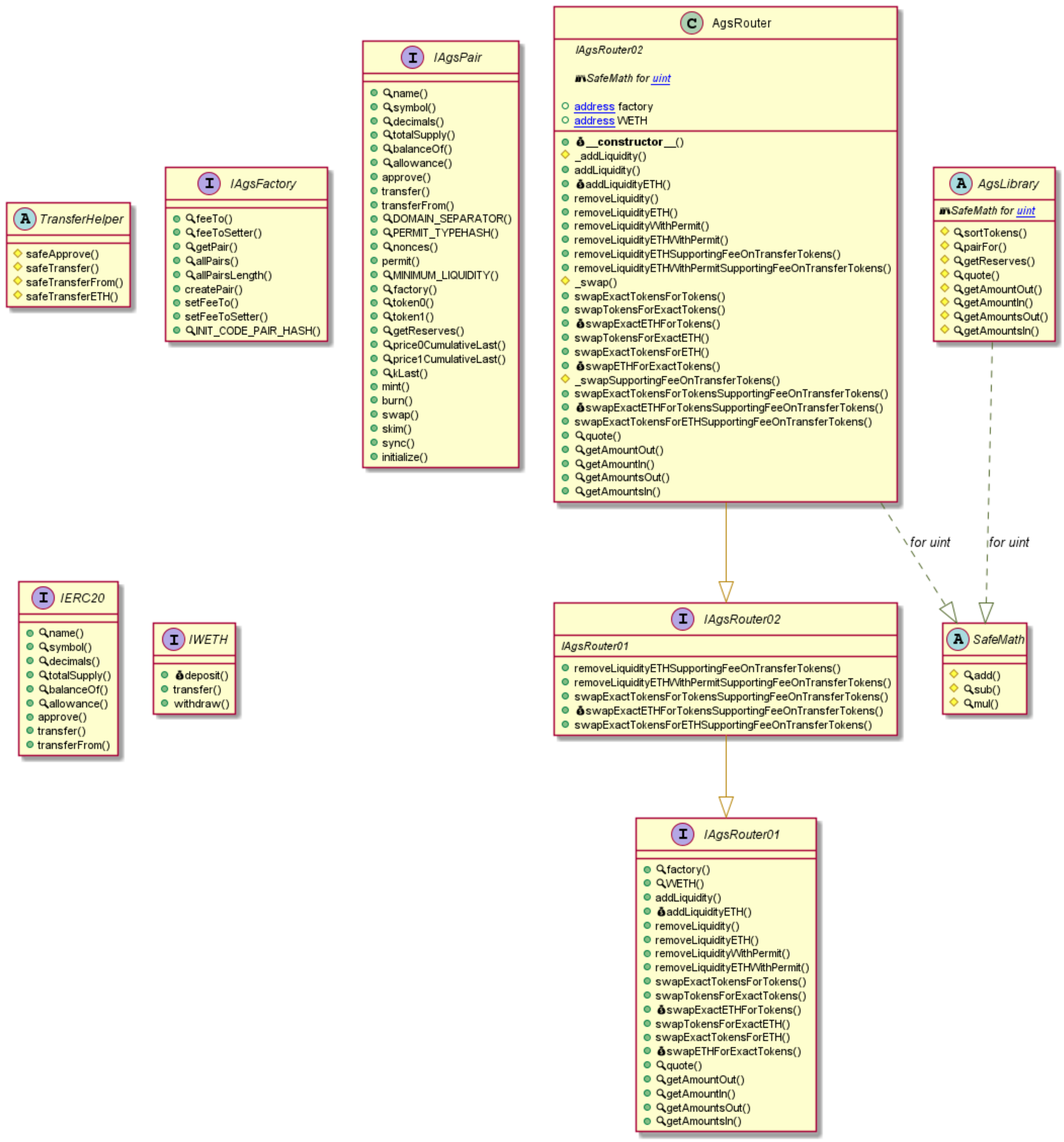
Technical Disclaimer

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.

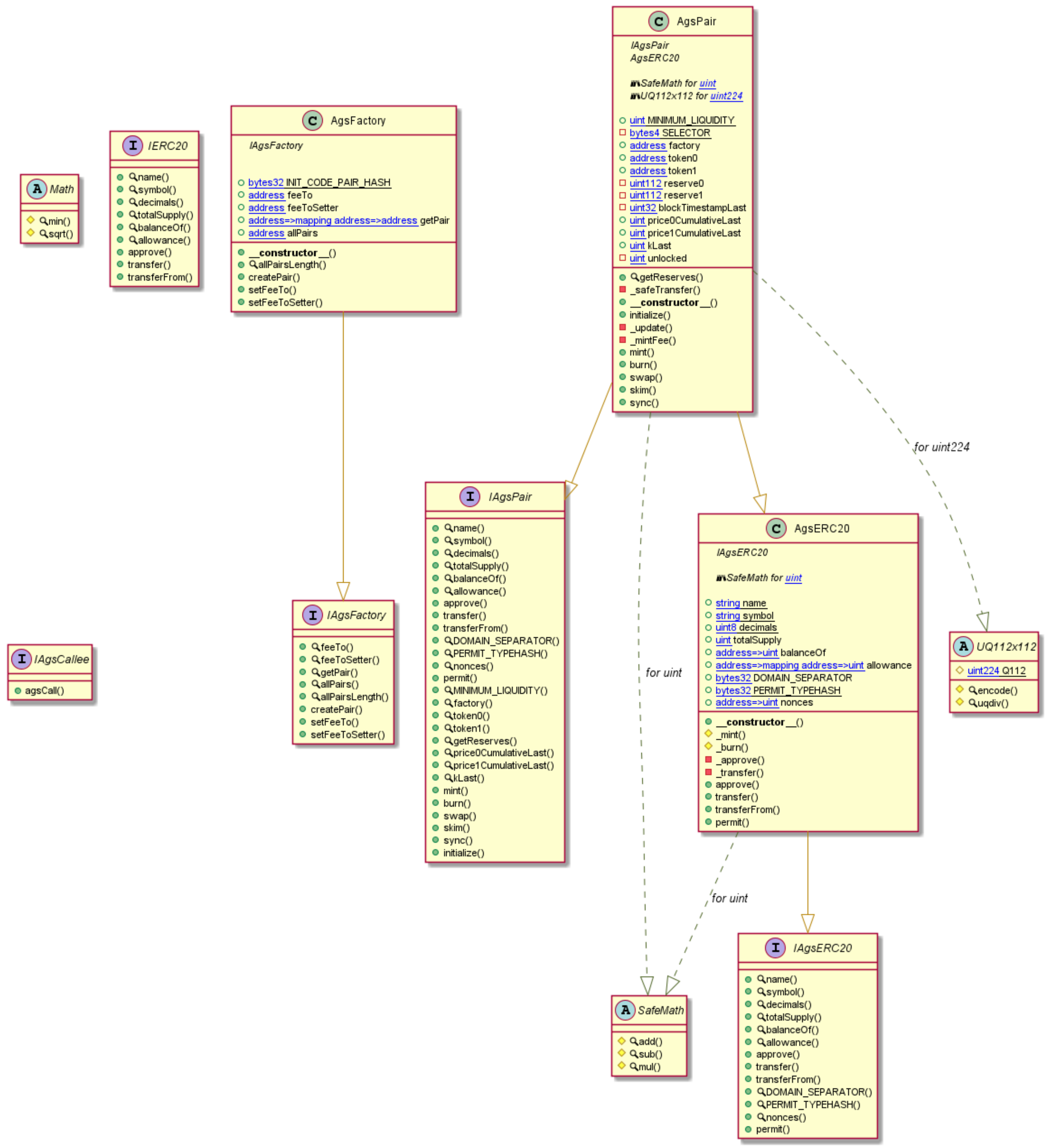
Appendix

Code Flow Diagram - AGS Finance Protocol

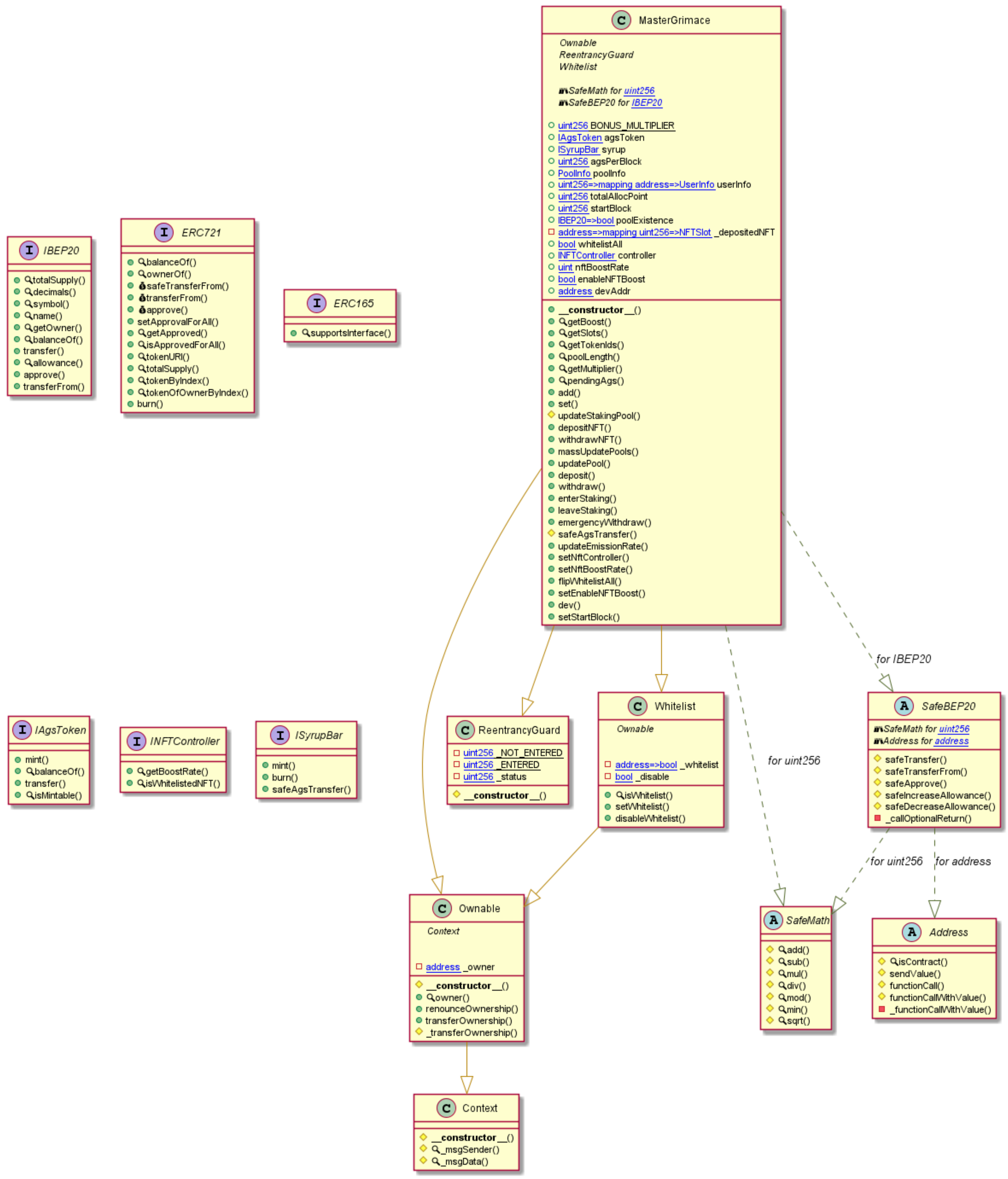
AgsRouter Diagram



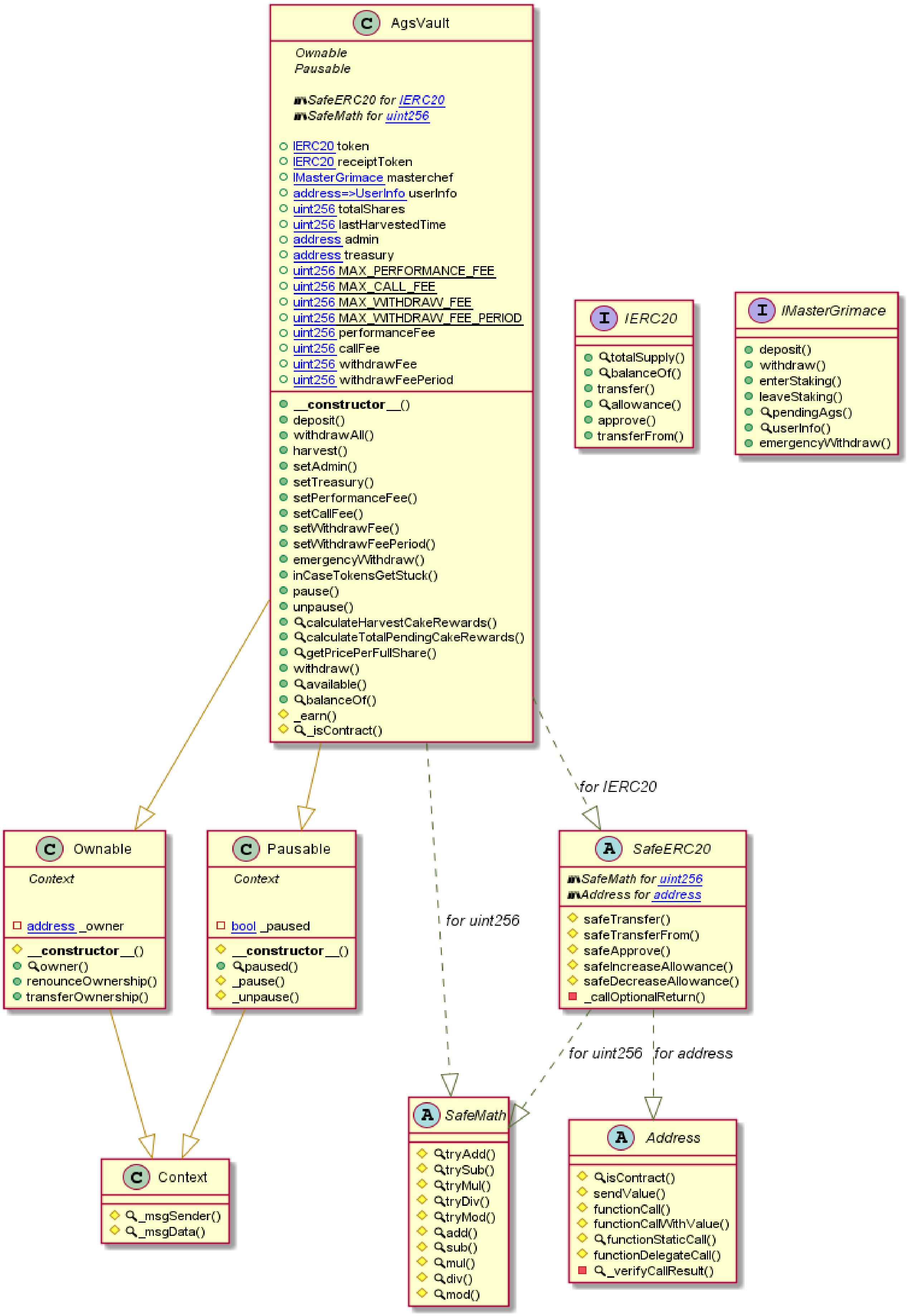
AgsFactory Diagram



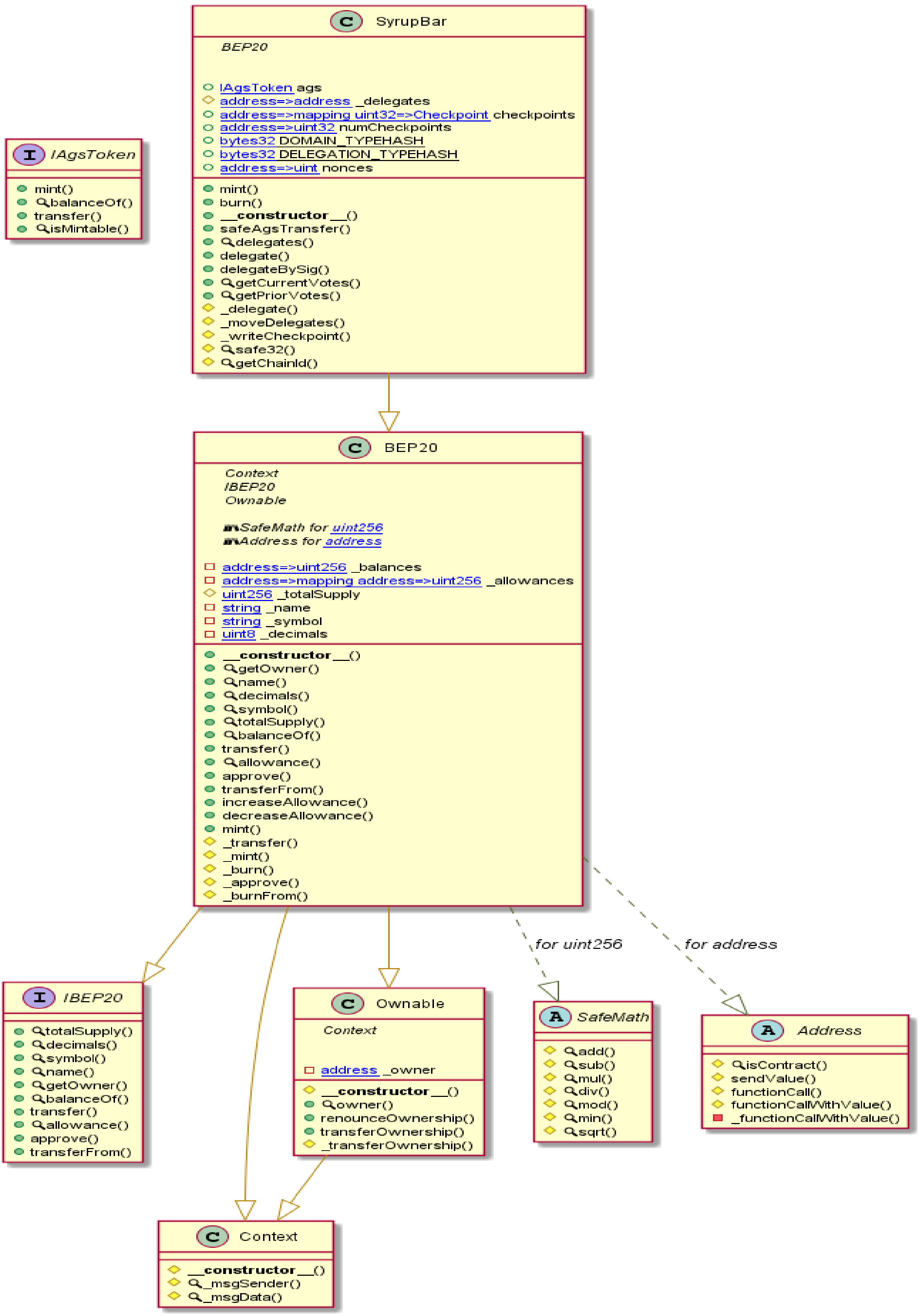
MasterGrimace Diagram



AgsVault Diagram

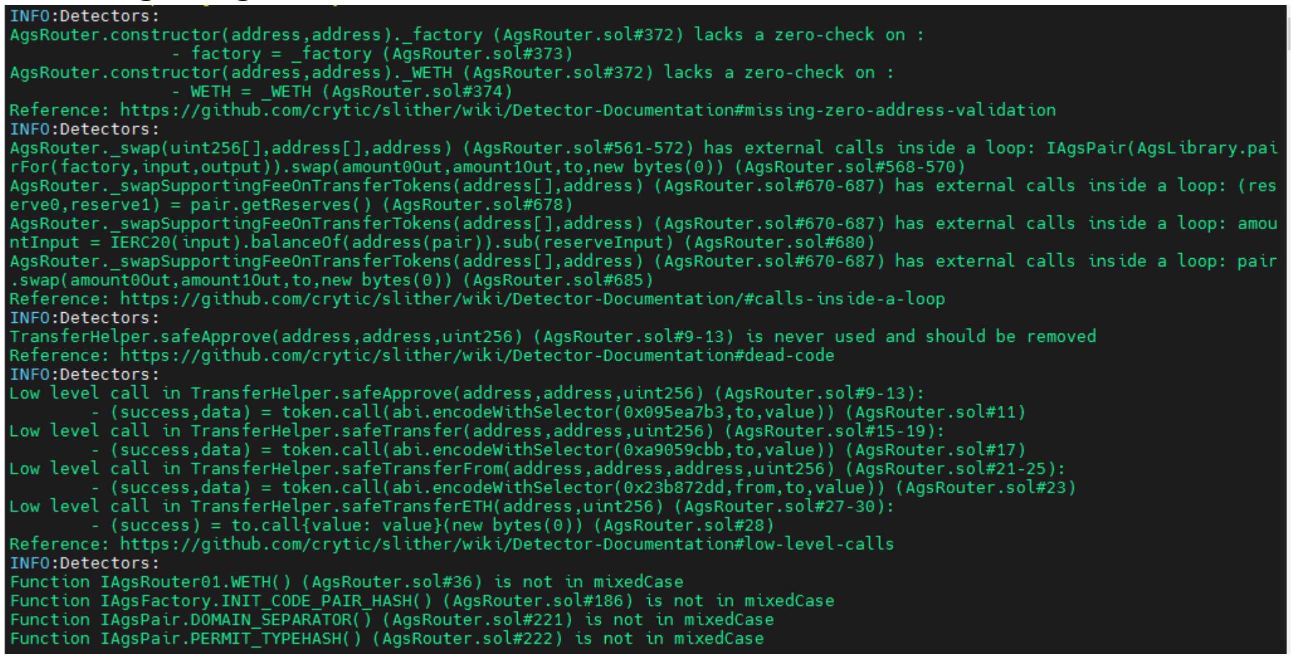


SyrupBar Diagram

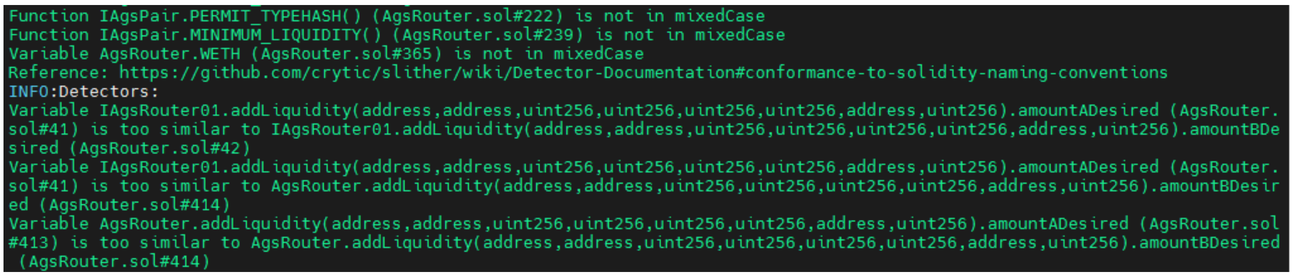


Slither Results Log

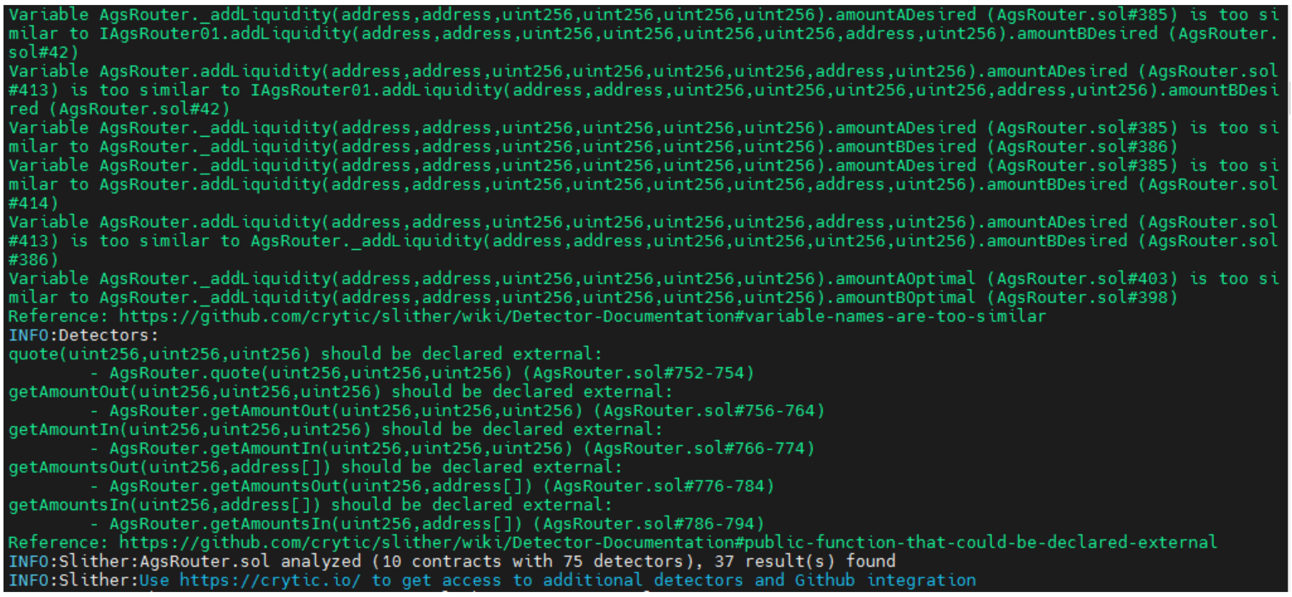
Slither log >> AgsRouter.sol



```
INFO:Detectors:
AgsRouter.constructor(address,address)._factory (AgsRouter.sol#372) lacks a zero-check on :
                - factory = _factory (AgsRouter.sol#373)
AgsRouter.constructor(address,address)._WETH (AgsRouter.sol#372) lacks a zero-check on :
                - WETH = _WETH (AgsRouter.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
AgsRouter._swap(uint256[],address[],address) (AgsRouter.sol#561-572) has external calls inside a loop: IAgsPair(AgsLibrary.pai
rFor(factory,input,output)).swap(amount0Out,amount1Out,to,new bytes(0)) (AgsRouter.sol#568-570)
AgsRouter._swapSupportingFeeOnTransferTokens(address[],address) (AgsRouter.sol#670-687) has external calls inside a loop: (res
erve0,reserve1) = pair.getReserves() (AgsRouter.sol#678)
AgsRouter._swapSupportingFeeOnTransferTokens(address[],address) (AgsRouter.sol#670-687) has external calls inside a loop: amou
ntInput = IERC20(input).balanceOf(address(pair)).sub(reserveInput) (AgsRouter.sol#680)
AgsRouter._swapSupportingFeeOnTransferTokens(address[],address) (AgsRouter.sol#670-687) has external calls inside a loop: pair
.swap(amount0Out,amount1Out,to,new bytes(0)) (AgsRouter.sol#685)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
TransferHelper.safeApprove(address,address,uint256) (AgsRouter.sol#9-13) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in TransferHelper.safeApprove(address,address,uint256) (AgsRouter.sol#9-13):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (AgsRouter.sol#11)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (AgsRouter.sol#15-19):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (AgsRouter.sol#17)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (AgsRouter.sol#21-25):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (AgsRouter.sol#23)
Low level call in TransferHelper.safeTransferETH(address,uint256) (AgsRouter.sol#27-30):
        - (success) = to.call{value: value}(new bytes(0)) (AgsRouter.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IAgsRouter01.WETH() (AgsRouter.sol#36) is not in mixedCase
Function IAgsFactory.INIT_CODE_PAIR_HASH() (AgsRouter.sol#186) is not in mixedCase
Function IAgsPair.DOMAIN_SEPARATOR() (AgsRouter.sol#221) is not in mixedCase
Function IAgsPair.PERMIT_TYPEHASH() (AgsRouter.sol#222) is not in mixedCase
```

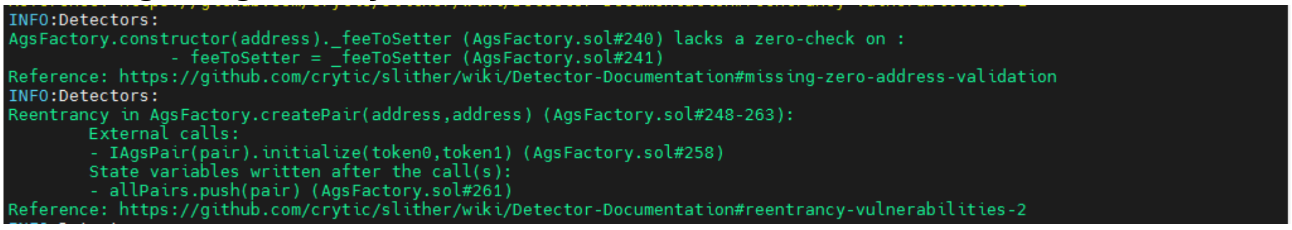


```
Function IAgsPair.PERMIT_TYPEHASH() (AgsRouter.sol#222) is not in mixedCase
Function IAgsPair.MINIMUM_LIQUIDITY() (AgsRouter.sol#239) is not in mixedCase
Variable AgsRouter.WETH (AgsRouter.sol#365) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IAgsRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (AgsRouter.
sol#41) is too similar to IAgsRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDe
sired (AgsRouter.sol#42)
Variable IAgsRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (AgsRouter.
sol#41) is too similar to AgsRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesir
ed (AgsRouter.sol#414)
Variable AgsRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (AgsRouter.sol
#413) is too similar to AgsRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired
 (AgsRouter.sol#414)
```

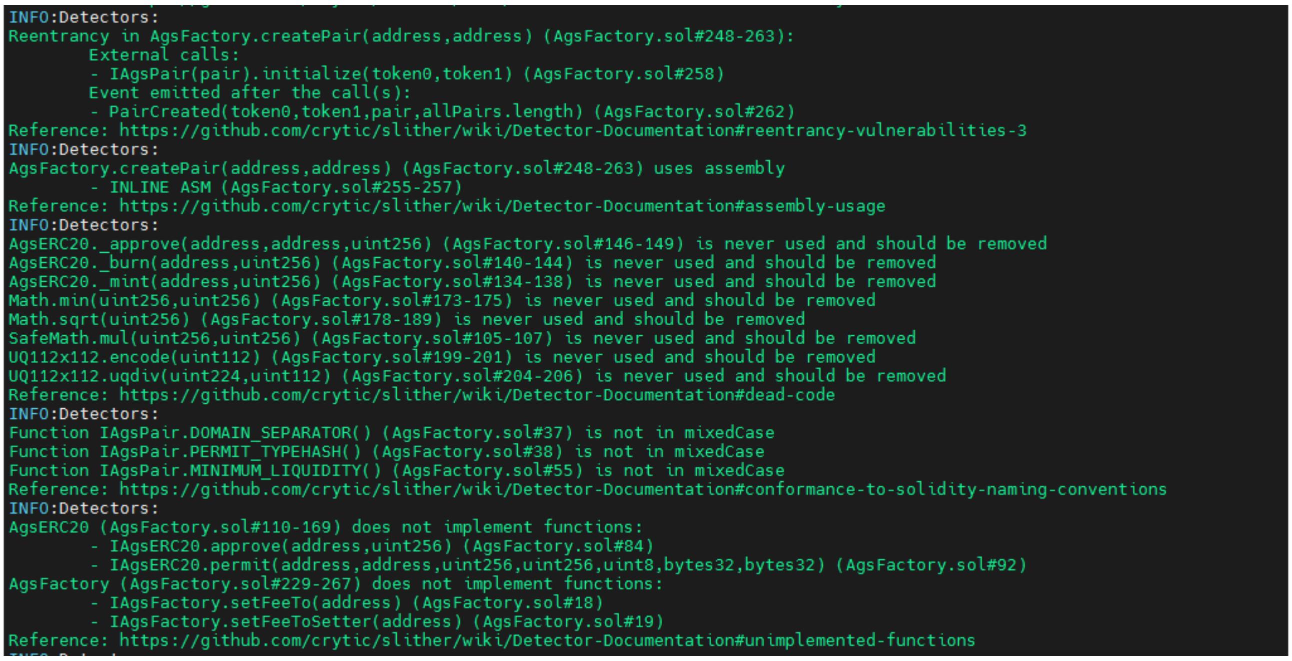


```
Variable AgsRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (AgsRouter.sol#385) is too si
milar to IAgsRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (AgsRouter.
sol#42)
Variable AgsRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (AgsRouter.sol
#413) is too similar to IAgsRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesi
red (AgsRouter.sol#42)
Variable AgsRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (AgsRouter.sol#385) is too si
milar to AgsRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (AgsRouter.sol#386)
Variable AgsRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (AgsRouter.sol#385) is too si
milar to AgsRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (AgsRouter.sol
#414)
Variable AgsRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (AgsRouter.sol
#413) is too similar to AgsRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (AgsRouter.sol
#386)
Variable AgsRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (AgsRouter.sol#403) is too si
milar to AgsRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (AgsRouter.sol#398)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
quote(uint256,uint256,uint256) should be declared external:
        - AgsRouter.quote(uint256,uint256,uint256) (AgsRouter.sol#752-754)
getAmountOut(uint256,uint256,uint256) should be declared external:
        - AgsRouter.getAmountOut(uint256,uint256,uint256) (AgsRouter.sol#756-764)
getAmountIn(uint256,uint256,uint256) should be declared external:
        - AgsRouter.getAmountIn(uint256,uint256,uint256) (AgsRouter.sol#766-774)
getAmountsOut(uint256,address[]) should be declared external:
        - AgsRouter.getAmountsOut(uint256,address[]) (AgsRouter.sol#776-784)
getAmountsIn(uint256,address[]) should be declared external:
        - AgsRouter.getAmountsIn(uint256,address[]) (AgsRouter.sol#786-794)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:AgsRouter.sol analyzed (10 contracts with 75 detectors), 37 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

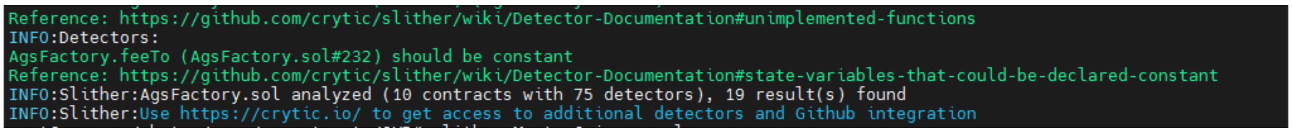
Slither log >> AgsFactory.sol



```
INFO:Detectors:
AgsFactory.constructor(address)._feeToSetter (AgsFactory.sol#240) lacks a zero-check on :
                - feeToSetter = _feeToSetter (AgsFactory.sol#241)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in AgsFactory.createPair(address,address) (AgsFactory.sol#248-263):
        External calls:
        - IAgsPair(pair).initialize(token0,token1) (AgsFactory.sol#258)
        State variables written after the call(s):
        - allPairs.push(pair) (AgsFactory.sol#261)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

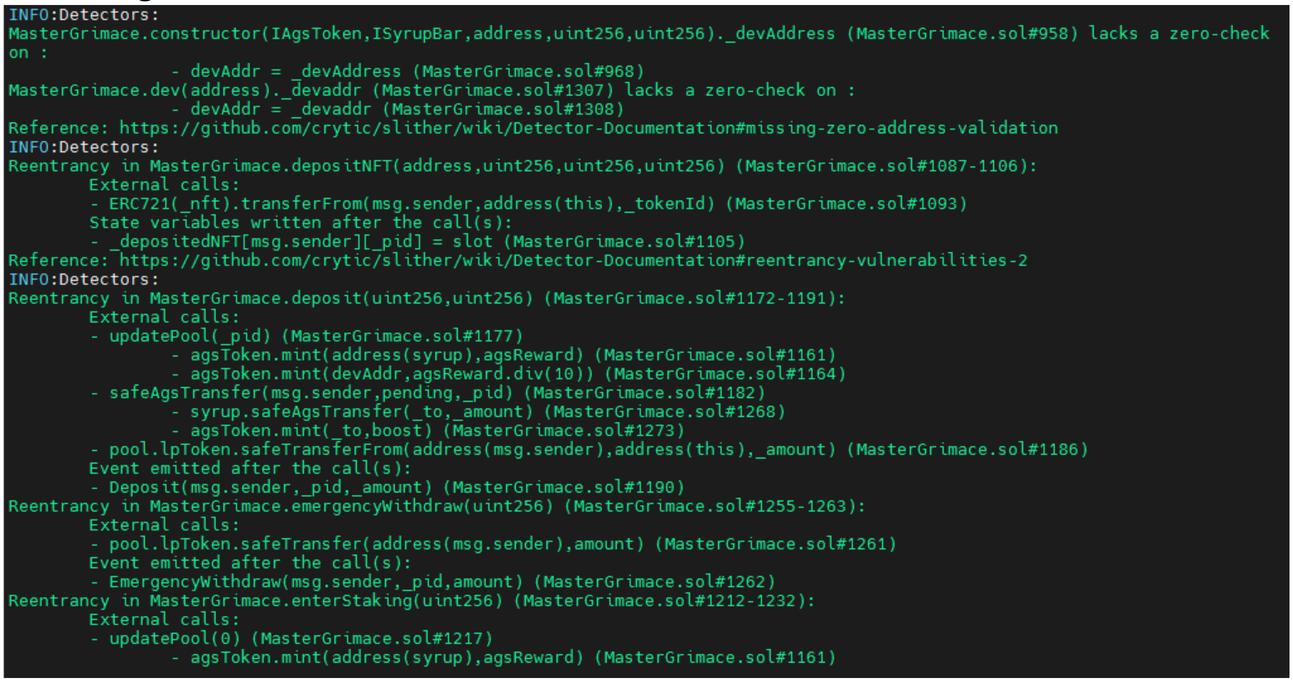


```
INFO:Detectors:
Reentrancy in AgsFactory.createPair(address,address) (AgsFactory.sol#248-263):
        External calls:
        - IAgsPair(pair).initialize(token0,token1) (AgsFactory.sol#258)
        Event emitted after the call(s):
        - PairCreated(token0,token1,pair,allPairs.length) (AgsFactory.sol#262)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
AgsFactory.createPair(address,address) (AgsFactory.sol#248-263) uses assembly
        - INLINE ASM (AgsFactory.sol#255-257)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AgsERC20._approve(address,address,uint256) (AgsFactory.sol#146-149) is never used and should be removed
AgsERC20._burn(address,uint256) (AgsFactory.sol#140-144) is never used and should be removed
AgsERC20._mint(address,uint256) (AgsFactory.sol#134-138) is never used and should be removed
Math.min(uint256,uint256) (AgsFactory.sol#173-175) is never used and should be removed
Math.sqrt(uint256) (AgsFactory.sol#178-189) is never used and should be removed
SafeMath.mul(uint256,uint256) (AgsFactory.sol#105-107) is never used and should be removed
UQ112x112.encode(uint112) (AgsFactory.sol#199-201) is never used and should be removed
UQ112x112.uqdiv(uint224,uint112) (AgsFactory.sol#204-206) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Function IAgsPair.DOMAIN_SEPARATOR() (AgsFactory.sol#37) is not in mixedCase
Function IAgsPair.PERMIT_TYPEHASH() (AgsFactory.sol#38) is not in mixedCase
Function IAgsPair.MINIMUM_LIQUIDITY() (AgsFactory.sol#55) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
AgsERC20 (AgsFactory.sol#110-169) does not implement functions:
        - IAgsERC20.approve(address,uint256) (AgsFactory.sol#84)
        - IAgsERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (AgsFactory.sol#92)
AgsFactory (AgsFactory.sol#229-267) does not implement functions:
        - IAgsFactory.setFeeTo(address) (AgsFactory.sol#18)
        - IAgsFactory.setFeeToSetter(address) (AgsFactory.sol#19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
```

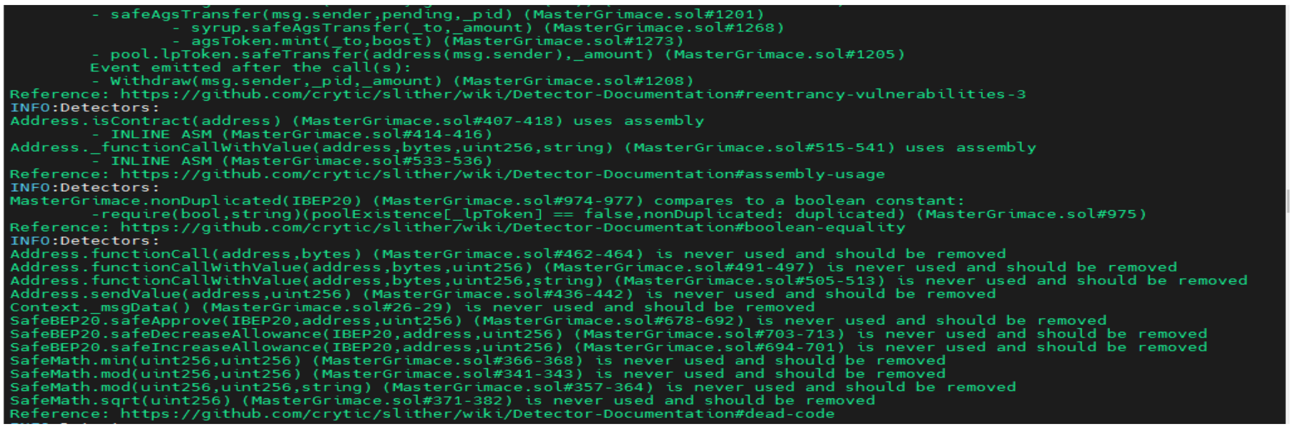


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
AgsFactory.feeTo (AgsFactory.sol#232) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:AgsFactory.sol analyzed (10 contracts with 75 detectors), 19 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

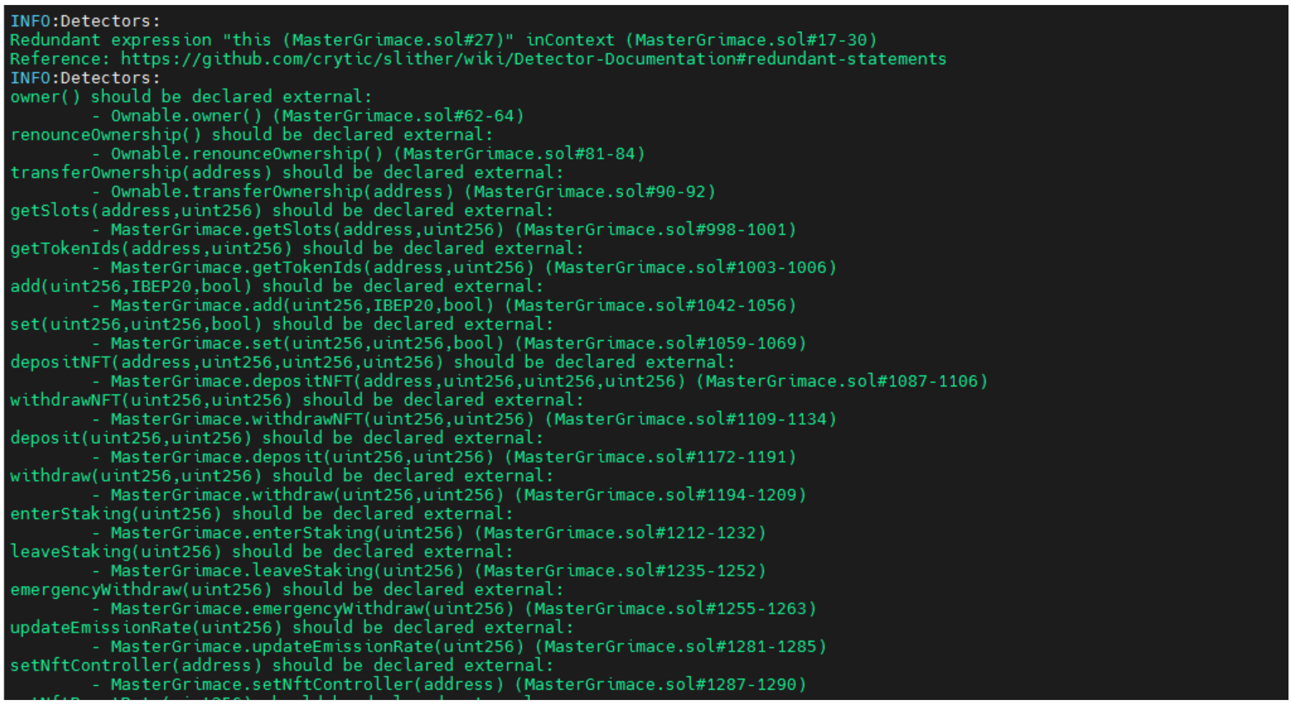
Slither log >> MasterGrimace.sol



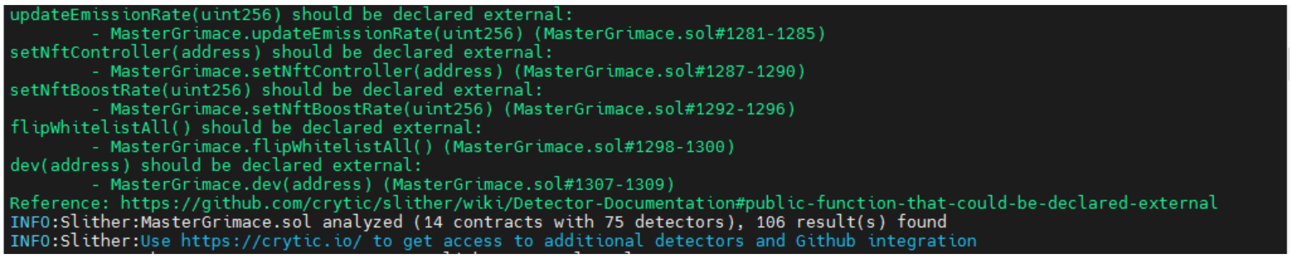
```
INFO:Detectors:
MasterGrimace.constructor(IAgsToken,ISyrupBar,address,uint256,uint256)._devAddress (MasterGrimace.sol#958) lacks a zero-check
on :
                - devAddr = _devAddress (MasterGrimace.sol#968)
MasterGrimace.dev(address)._devaddr (MasterGrimace.sol#1307) lacks a zero-check on :
                - devAddr = _devaddr (MasterGrimace.sol#1308)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in MasterGrimace.depositNFT(address,uint256,uint256,uint256) (MasterGrimace.sol#1087-1106):
        External calls:
        - ERC721(_nft).transferFrom(msg.sender,address(this),_tokenId) (MasterGrimace.sol#1093)
        State variables written after the call(s):
        - _depositedNFT[msg.sender][_pid] = slot (MasterGrimace.sol#1105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in MasterGrimace.deposit(uint256,uint256) (MasterGrimace.sol#1172-1191):
        External calls:
        - updatePool(_pid) (MasterGrimace.sol#1177)
                - agsToken.mint(address(syrup),agsReward) (MasterGrimace.sol#1161)
                - agsToken.mint(devAddr,agsReward.div(10)) (MasterGrimace.sol#1164)
        - safeAgsTransfer(msg.sender,pending,_pid) (MasterGrimace.sol#1182)
                - syrup.safeAgsTransfer(_to,_amount) (MasterGrimace.sol#1268)
                - agsToken.mint(_to,boost) (MasterGrimace.sol#1273)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterGrimace.sol#1186)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (MasterGrimace.sol#1190)
Reentrancy in MasterGrimace.emergencyWithdraw(uint256) (MasterGrimace.sol#1255-1263):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),amount) (MasterGrimace.sol#1261)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,amount) (MasterGrimace.sol#1262)
Reentrancy in MasterGrimace.enterStaking(uint256) (MasterGrimace.sol#1212-1232):
        External calls:
        - updatePool(0) (MasterGrimace.sol#1217)
                - agsToken.mint(address(syrup),agsReward) (MasterGrimace.sol#1161)
```



```
        - safeAgsTransfer(msg.sender,pending,_pid) (MasterGrimace.sol#1201)
                - syrup.safeAgsTransfer(_to,_amount) (MasterGrimace.sol#1268)
                - agsToken.mint(_to,boost) (MasterGrimace.sol#1273)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterGrimace.sol#1205)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_amount) (MasterGrimace.sol#1208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (MasterGrimace.sol#407-418) uses assembly
        - INLINE ASM (MasterGrimace.sol#414-416)
Address._functionCallWithValue(address,bytes,uint256,string) (MasterGrimace.sol#515-541) uses assembly
        - INLINE ASM (MasterGrimace.sol#533-536)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
MasterGrimace.nonDuplicated(IBEP20) (MasterGrimace.sol#974-977) compares to a boolean constant:
        -require(bool,string)(poolExistence[_lpToken] == false,nonDuplicated: duplicated) (MasterGrimace.sol#975)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address.functionCall(address,bytes) (MasterGrimace.sol#462-464) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (MasterGrimace.sol#491-497) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (MasterGrimace.sol#505-513) is never used and should be removed
Address.sendValue(address,uint256) (MasterGrimace.sol#436-442) is never used and should be removed
Context._msgData() (MasterGrimace.sol#26-29) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (MasterGrimace.sol#678-692) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (MasterGrimace.sol#703-713) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (MasterGrimace.sol#694-701) is never used and should be removed
SafeMath.min(uint256,uint256) (MasterGrimace.sol#366-368) is never used and should be removed
SafeMath.mod(uint256,uint256) (MasterGrimace.sol#341-343) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (MasterGrimace.sol#357-364) is never used and should be removed
SafeMath.sqrt(uint256) (MasterGrimace.sol#371-382) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

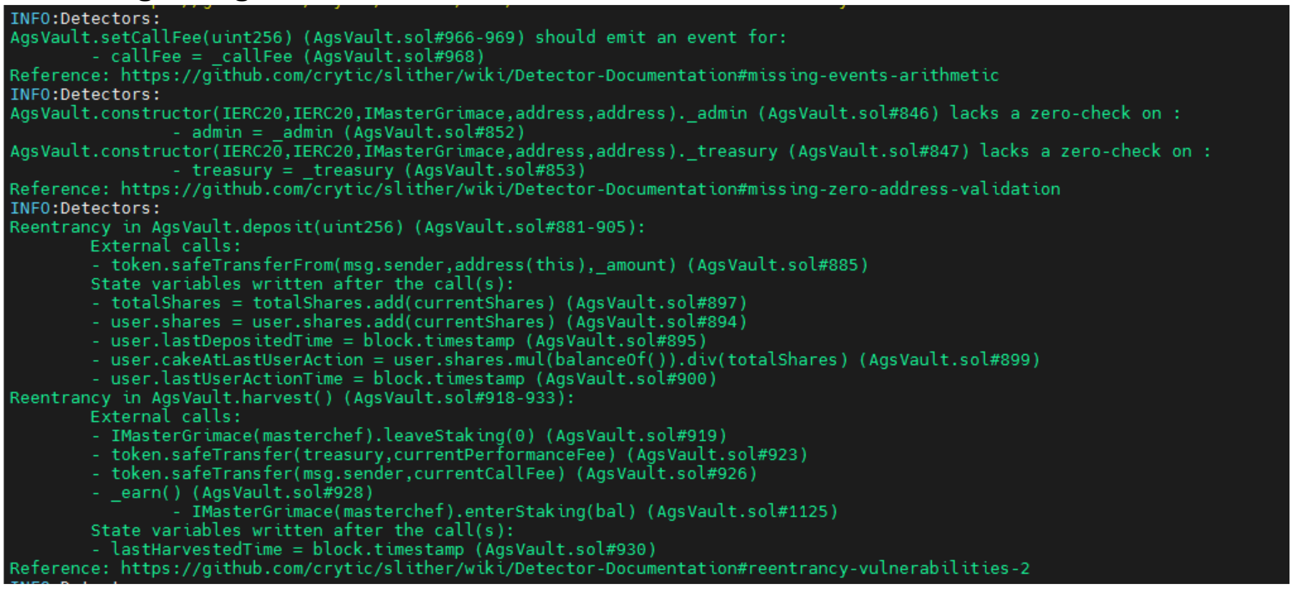


```
INFO:Detectors:
Redundant expression "this (MasterGrimace.sol#27)" inContext (MasterGrimace.sol#17-30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (MasterGrimace.sol#62-64)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MasterGrimace.sol#81-84)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MasterGrimace.sol#90-92)
getSlots(address,uint256) should be declared external:
        - MasterGrimace.getSlots(address,uint256) (MasterGrimace.sol#998-1001)
getTokenIds(address,uint256) should be declared external:
        - MasterGrimace.getTokenIds(address,uint256) (MasterGrimace.sol#1003-1006)
add(uint256,IBEP20,bool) should be declared external:
        - MasterGrimace.add(uint256,IBEP20,bool) (MasterGrimace.sol#1042-1056)
set(uint256,uint256,bool) should be declared external:
        - MasterGrimace.set(uint256,uint256,bool) (MasterGrimace.sol#1059-1069)
depositNFT(address,uint256,uint256) should be declared external:
        - MasterGrimace.depositNFT(address,uint256,uint256,uint256) (MasterGrimace.sol#1087-1106)
withdrawNFT(uint256,uint256) should be declared external:
        - MasterGrimace.withdrawNFT(uint256,uint256) (MasterGrimace.sol#1109-1134)
deposit(uint256,uint256) should be declared external:
        - MasterGrimace.deposit(uint256,uint256) (MasterGrimace.sol#1172-1191)
withdraw(uint256,uint256) should be declared external:
        - MasterGrimace.withdraw(uint256,uint256) (MasterGrimace.sol#1194-1209)
enterStaking(uint256) should be declared external:
        - MasterGrimace.enterStaking(uint256) (MasterGrimace.sol#1212-1232)
leaveStaking(uint256) should be declared external:
        - MasterGrimace.leaveStaking(uint256) (MasterGrimace.sol#1235-1252)
emergencyWithdraw(uint256) should be declared external:
        - MasterGrimace.emergencyWithdraw(uint256) (MasterGrimace.sol#1255-1263)
updateEmissionRate(uint256) should be declared external:
        - MasterGrimace.updateEmissionRate(uint256) (MasterGrimace.sol#1281-1285)
setNftController(address) should be declared external:
        - MasterGrimace.setNftController(address) (MasterGrimace.sol#1287-1290)
```

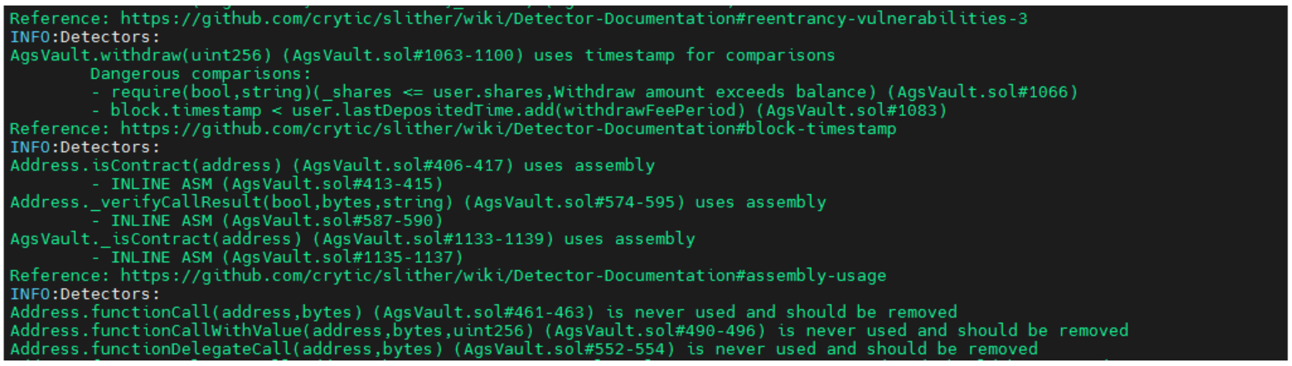


```
updateEmissionRate(uint256) should be declared external:
        - MasterGrimace.updateEmissionRate(uint256) (MasterGrimace.sol#1281-1285)
setNftController(address) should be declared external:
        - MasterGrimace.setNftController(address) (MasterGrimace.sol#1287-1290)
setNftBoostRate(uint256) should be declared external:
        - MasterGrimace.setNftBoostRate(uint256) (MasterGrimace.sol#1292-1296)
flipWhitelistAll() should be declared external:
        - MasterGrimace.flipWhitelistAll() (MasterGrimace.sol#1298-1300)
dev(address) should be declared external:
        - MasterGrimace.dev(address) (MasterGrimace.sol#1307-1309)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MasterGrimace.sol analyzed (14 contracts with 75 detectors), 106 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

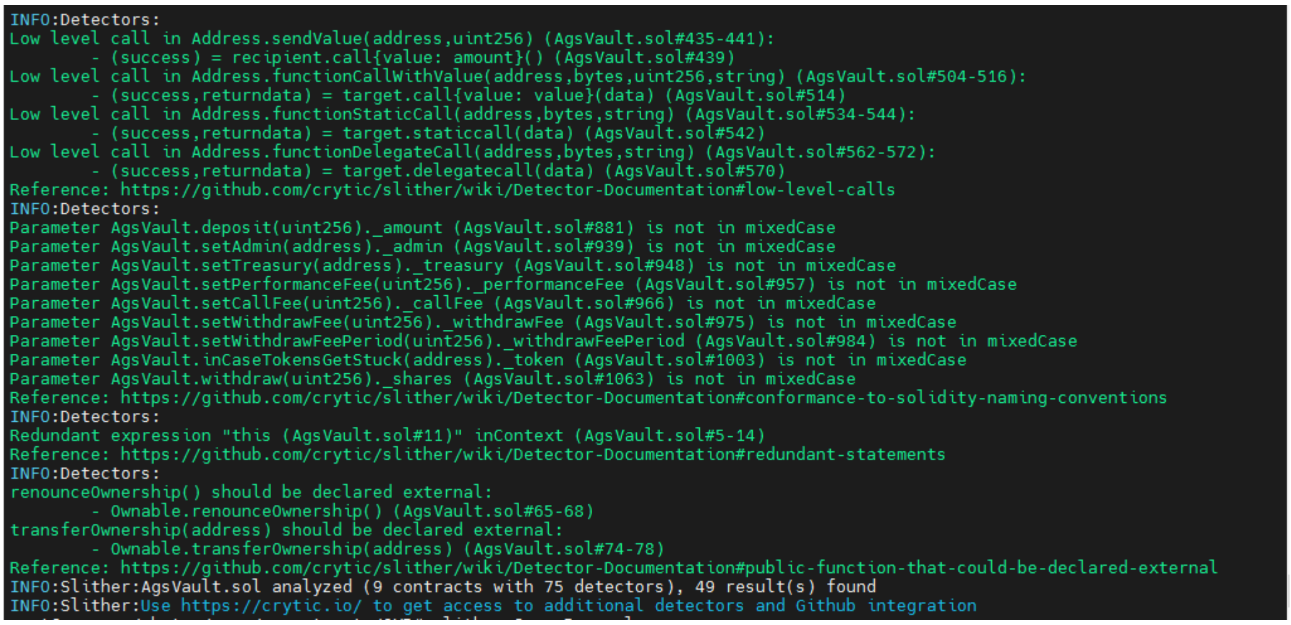
Slither log >> AgsVault.sol



```
INFO:Detectors:
AgsVault.setCallFee(uint256) (AgsVault.sol#966-969) should emit an event for:
        - callFee = _callFee (AgsVault.sol#968)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
AgsVault.constructor(IERC20,IERC20,IMasterGrimace,address,address)._admin (AgsVault.sol#846) lacks a zero-check on :
                - admin = _admin (AgsVault.sol#852)
AgsVault.constructor(IERC20,IERC20,IMasterGrimace,address,address)._treasury (AgsVault.sol#847) lacks a zero-check on :
                - treasury = _treasury (AgsVault.sol#853)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in AgsVault.deposit(uint256) (AgsVault.sol#881-905):
        External calls:
        - token.safeTransferFrom(msg.sender,address(this),_amount) (AgsVault.sol#885)
        State variables written after the call(s):
        - totalShares = totalShares.add(currentShares) (AgsVault.sol#897)
        - user.shares = user.shares.add(currentShares) (AgsVault.sol#894)
        - user.lastDepositedTime = block.timestamp (AgsVault.sol#895)
        - user.cakeAtLastUserAction = user.shares.mul(balanceOf()).div(totalShares) (AgsVault.sol#899)
        - user.lastUserActionTime = block.timestamp (AgsVault.sol#900)
Reentrancy in AgsVault.harvest() (AgsVault.sol#918-933):
        External calls:
        - IMasterGrimace(masterchef).leaveStaking(0) (AgsVault.sol#919)
        - token.safeTransfer(treasury,currentPerformanceFee) (AgsVault.sol#923)
        - token.safeTransfer(msg.sender,currentCallFee) (AgsVault.sol#926)
        - _earn() (AgsVault.sol#928)
                - IMasterGrimace(masterchef).enterStaking(bal) (AgsVault.sol#1125)
        State variables written after the call(s):
        - lastHarvestedTime = block.timestamp (AgsVault.sol#930)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

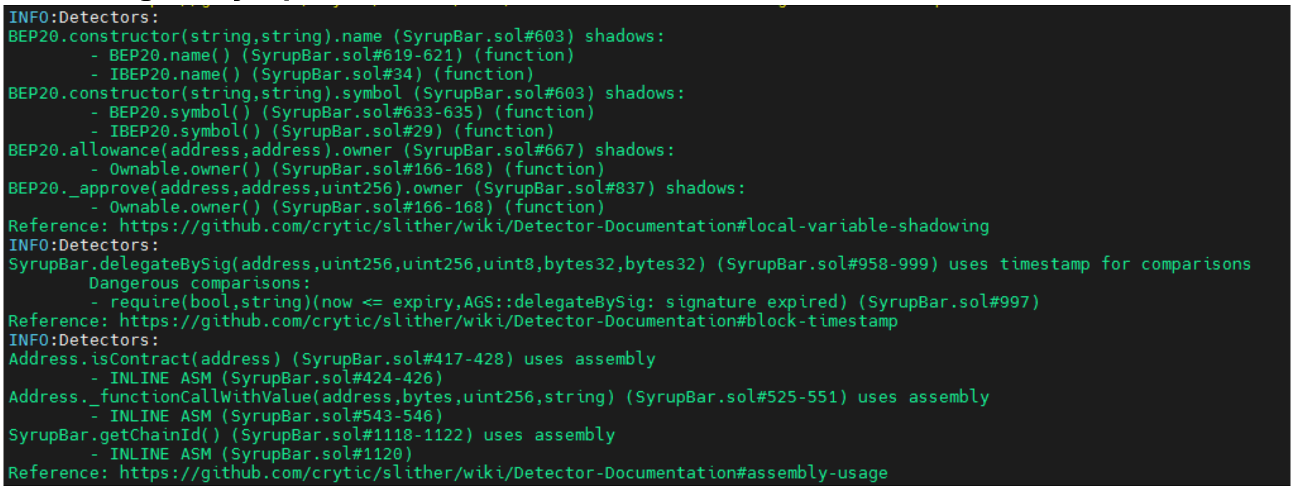


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
AgsVault.withdraw(uint256) (AgsVault.sol#1063-1100) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_shares <= user.shares,Withdraw amount exceeds balance) (AgsVault.sol#1066)
        - block.timestamp < user.lastDepositedTime.add(withdrawFeePeriod) (AgsVault.sol#1083)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (AgsVault.sol#406-417) uses assembly
        - INLINE ASM (AgsVault.sol#413-415)
Address._verifyCallResult(bool,bytes,string) (AgsVault.sol#574-595) uses assembly
        - INLINE ASM (AgsVault.sol#587-590)
AgsVault._isContract(address) (AgsVault.sol#1133-1139) uses assembly
        - INLINE ASM (AgsVault.sol#1135-1137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (AgsVault.sol#461-463) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (AgsVault.sol#490-496) is never used and should be removed
Address.functionDelegateCall(address,bytes) (AgsVault.sol#552-554) is never used and should be removed
```

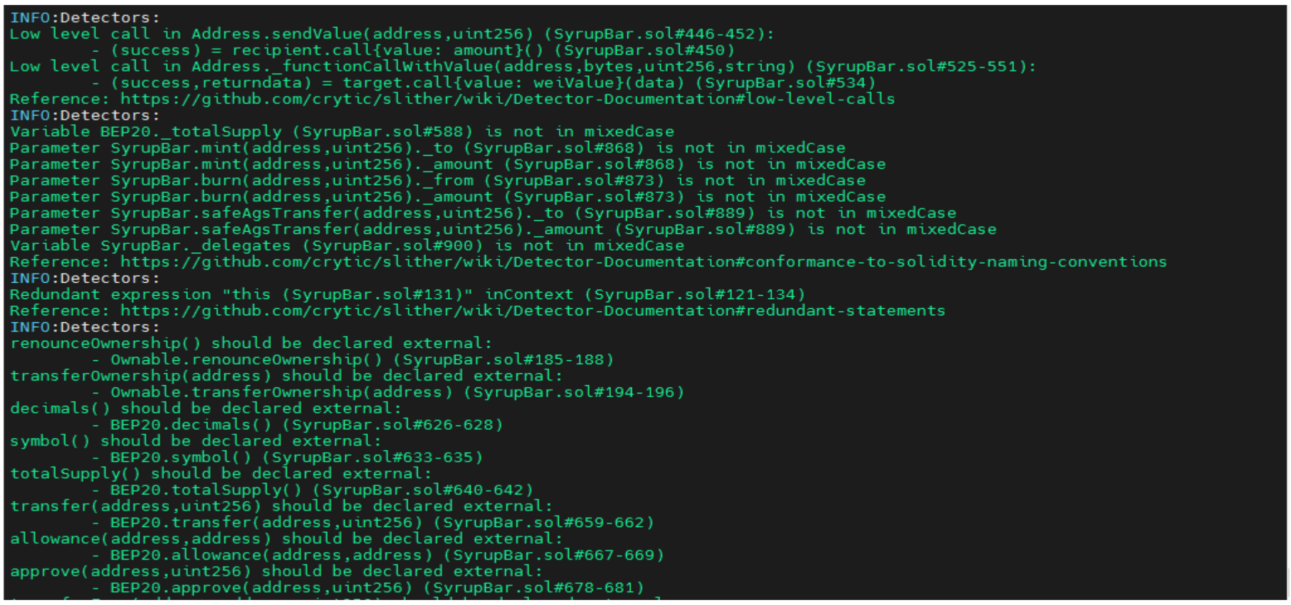


```
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (AgsVault.sol#435-441):
        - (success) = recipient.call{value: amount}() (AgsVault.sol#439)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (AgsVault.sol#504-516):
        - (success,returndata) = target.call{value: value}(data) (AgsVault.sol#514)
Low level call in Address.functionStaticCall(address,bytes,string) (AgsVault.sol#534-544):
        - (success,returndata) = target.staticcall(data) (AgsVault.sol#542)
Low level call in Address.functionDelegateCall(address,bytes,string) (AgsVault.sol#562-572):
        - (success,returndata) = target.delegatecall(data) (AgsVault.sol#570)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter AgsVault.deposit(uint256)._amount (AgsVault.sol#881) is not in mixedCase
Parameter AgsVault.setAdmin(address)._admin (AgsVault.sol#939) is not in mixedCase
Parameter AgsVault.setTreasury(address)._treasury (AgsVault.sol#948) is not in mixedCase
Parameter AgsVault.setPerformanceFee(uint256)._performanceFee (AgsVault.sol#957) is not in mixedCase
Parameter AgsVault.setCallFee(uint256)._callFee (AgsVault.sol#966) is not in mixedCase
Parameter AgsVault.setWithdrawFee(uint256)._withdrawFee (AgsVault.sol#975) is not in mixedCase
Parameter AgsVault.setWithdrawFeePeriod(uint256)._withdrawFeePeriod (AgsVault.sol#984) is not in mixedCase
Parameter AgsVault.inCaseTokensGetStuck(address)._token (AgsVault.sol#1003) is not in mixedCase
Parameter AgsVault.withdraw(uint256)._shares (AgsVault.sol#1063) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (AgsVault.sol#11)" inContext (AgsVault.sol#5-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (AgsVault.sol#65-68)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (AgsVault.sol#74-78)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:AgsVault.sol analyzed (9 contracts with 75 detectors), 49 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

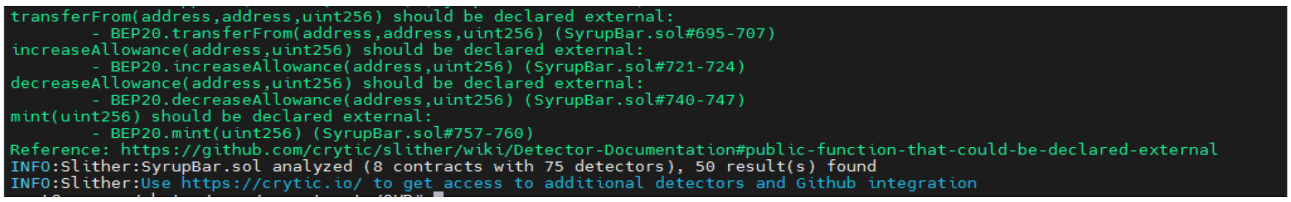
Slither log >> SyrupBar.sol



```
INFO:Detectors:
BEP20.constructor(string,string).name (SyrupBar.sol#603) shadows:
        - BEP20.name() (SyrupBar.sol#619-621) (function)
        - IBEP20.name() (SyrupBar.sol#34) (function)
BEP20.constructor(string,string).symbol (SyrupBar.sol#603) shadows:
        - BEP20.symbol() (SyrupBar.sol#633-635) (function)
        - IBEP20.symbol() (SyrupBar.sol#29) (function)
BEP20.allowance(address,address).owner (SyrupBar.sol#667) shadows:
        - Ownable.owner() (SyrupBar.sol#166-168) (function)
BEP20._approve(address,address,uint256).owner (SyrupBar.sol#837) shadows:
        - Ownable.owner() (SyrupBar.sol#166-168) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
SyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#958-999) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,AGS::delegateBySig: signature expired) (SyrupBar.sol#997)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (SyrupBar.sol#417-428) uses assembly
        - INLINE ASM (SyrupBar.sol#424-426)
Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#525-551) uses assembly
        - INLINE ASM (SyrupBar.sol#543-546)
SyrupBar.getChainId() (SyrupBar.sol#1118-1122) uses assembly
        - INLINE ASM (SyrupBar.sol#1120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```



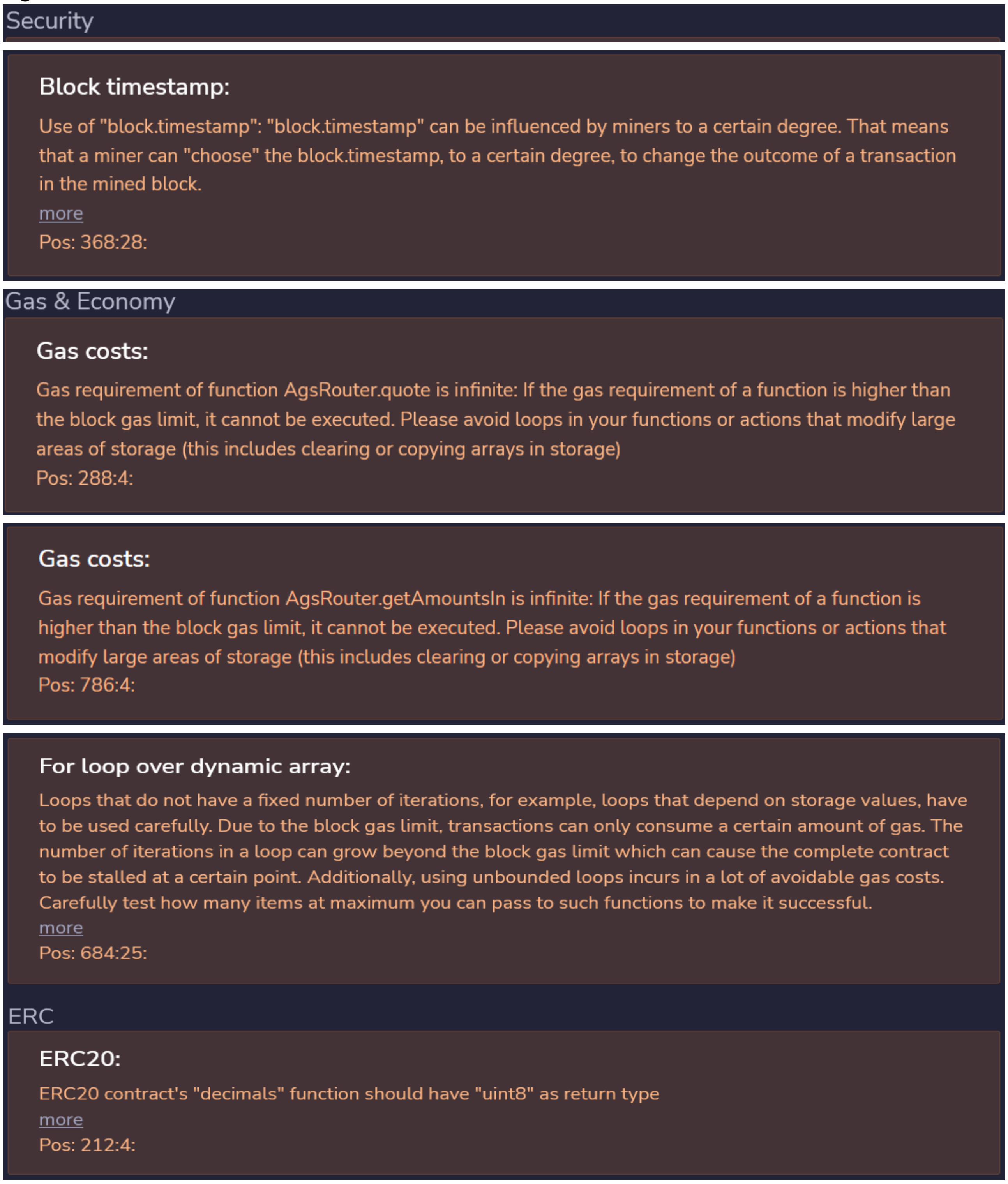
```
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (SyrupBar.sol#446-452):
        - (success) = recipient.call{value: amount}() (SyrupBar.sol#450)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#525-551):
        - (success,returndata) = target.call{value: weiValue}(data) (SyrupBar.sol#534)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable BEP20._totalSupply (SyrupBar.sol#588) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._to (SyrupBar.sol#868) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._amount (SyrupBar.sol#868) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._from (SyrupBar.sol#873) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._amount (SyrupBar.sol#873) is not in mixedCase
Parameter SyrupBar.safeAgsTransfer(address,uint256)._to (SyrupBar.sol#889) is not in mixedCase
Parameter SyrupBar.safeAgsTransfer(address,uint256)._amount (SyrupBar.sol#889) is not in mixedCase
Variable SyrupBar._delegates (SyrupBar.sol#900) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (SyrupBar.sol#131)" inContext (SyrupBar.sol#121-134)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SyrupBar.sol#185-188)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SyrupBar.sol#194-196)
decimals() should be declared external:
        - BEP20.decimals() (SyrupBar.sol#626-628)
symbol() should be declared external:
        - BEP20.symbol() (SyrupBar.sol#633-635)
totalSupply() should be declared external:
        - BEP20.totalSupply() (SyrupBar.sol#640-642)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (SyrupBar.sol#659-662)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (SyrupBar.sol#667-669)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (SyrupBar.sol#678-681)
```



```
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (SyrupBar.sol#695-707)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (SyrupBar.sol#721-724)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (SyrupBar.sol#740-747)
mint(uint256) should be declared external:
        - BEP20.mint(uint256) (SyrupBar.sol#757-760)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SyrupBar.sol analyzed (8 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

AgsRouter.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: 368:28:

Gas & Economy

Gas costs:

Gas requirement of function AgsRouter.quote 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: 288:4:

Gas costs:

Gas requirement of function AgsRouter.getAmountsIn 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: 786:4:

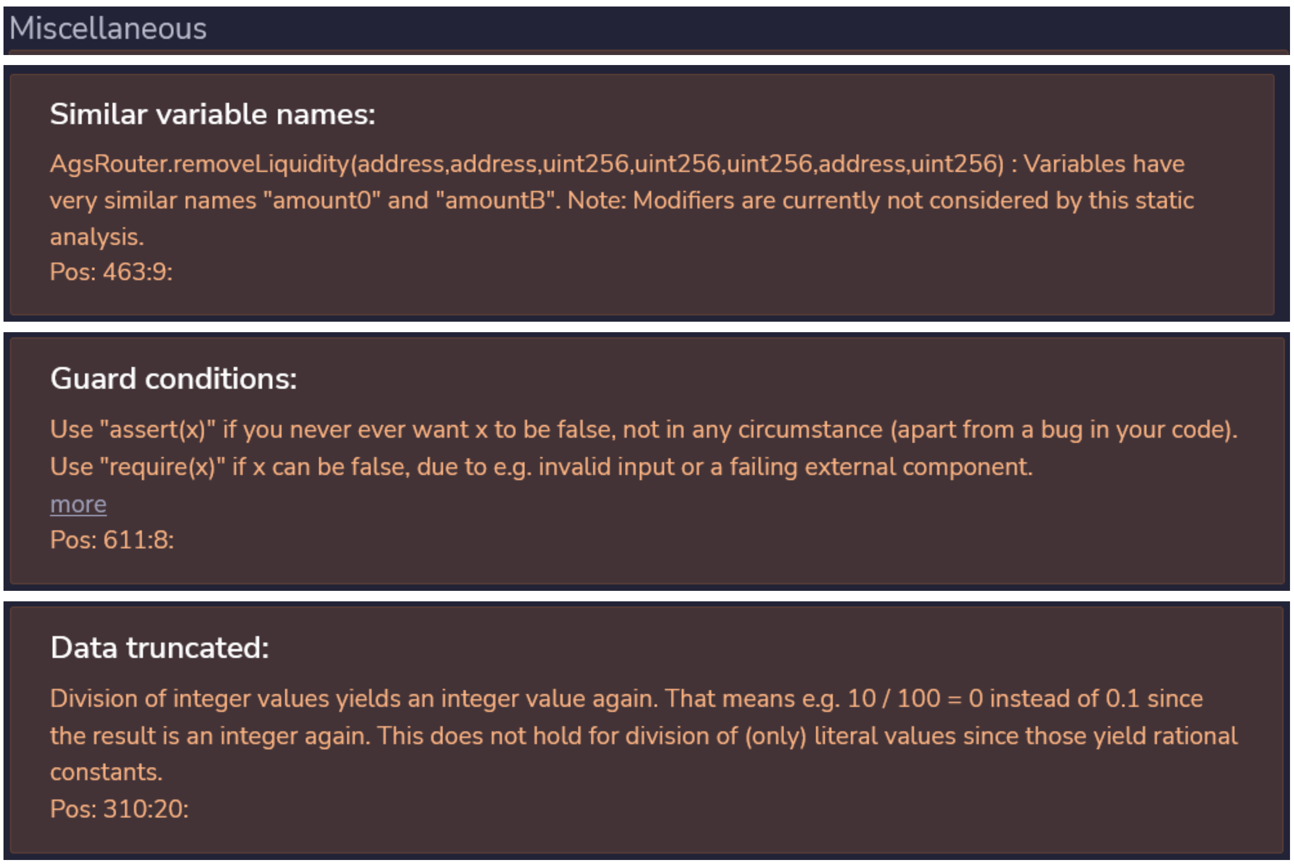
For loop over dynamic array:

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: 684:25:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 212:4:



Miscellaneous

Similar variable names:

AgsRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) : Variables have very similar names "amount0" and "amountB". Note: Modifiers are currently not considered by this static analysis.
Pos: 463:9:

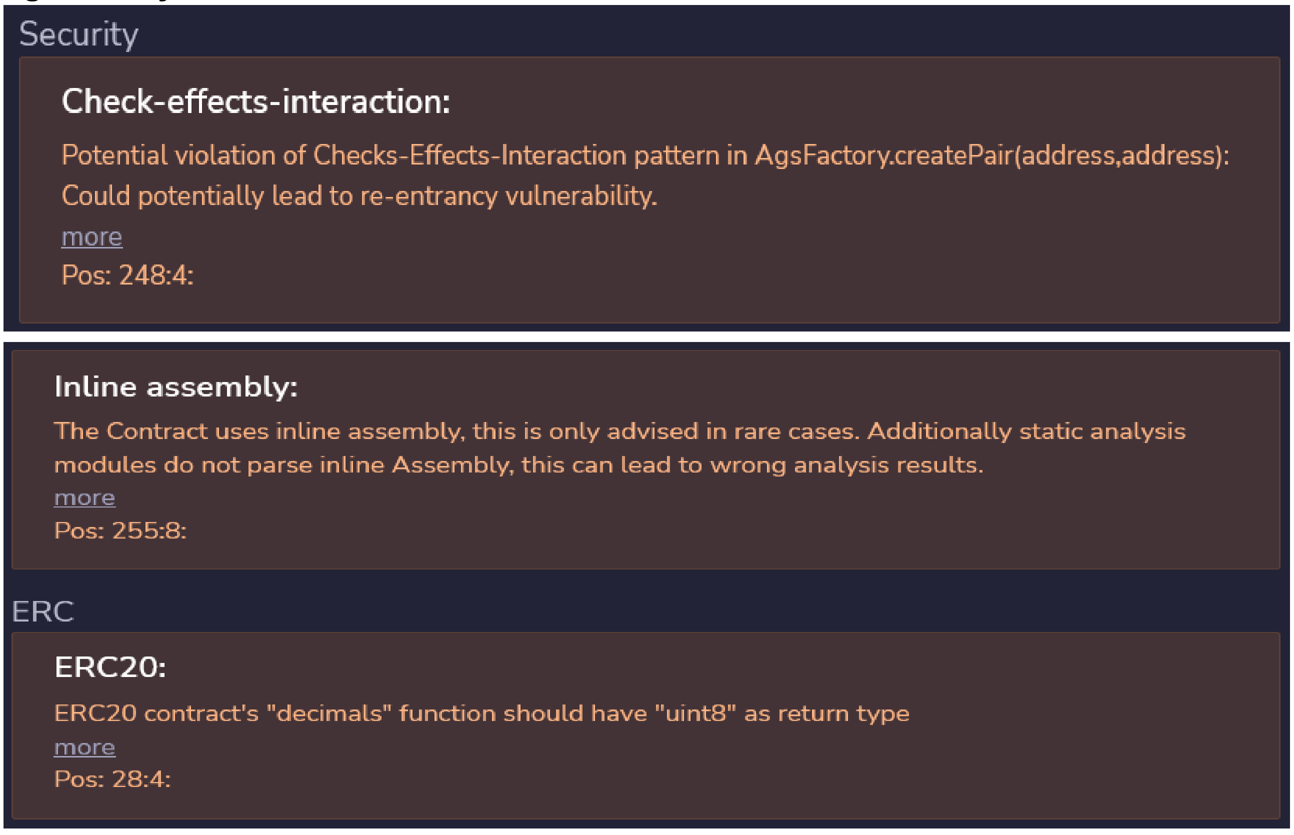
Guard conditions:

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: 611:8:

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: 310:20:

AgsFactory.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in AgsFactory.createPair(address,address): Could potentially lead to re-entrancy vulnerability.
more
Pos: 248:4:

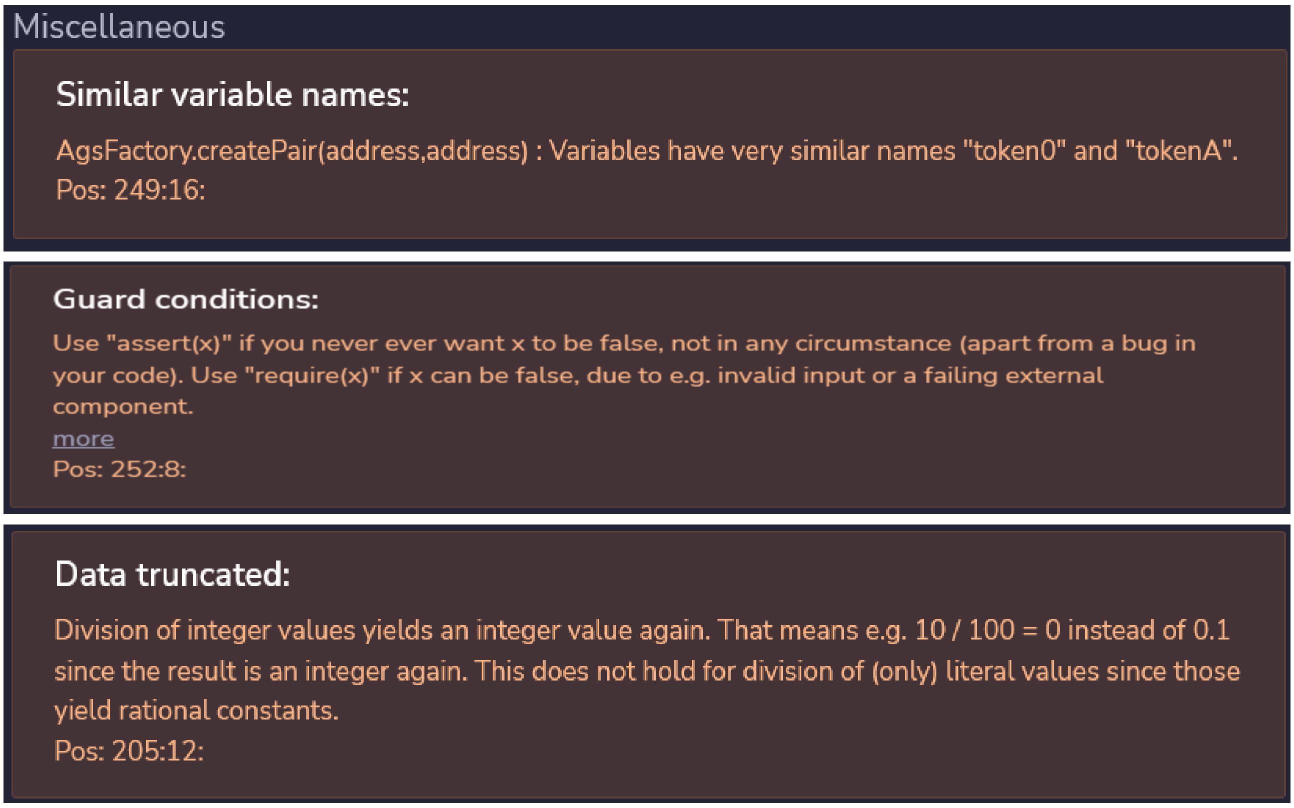
Inline assembly:

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: 255:8:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 28:4:


Miscellaneous

Similar variable names:

AgsFactory.createPair(address,address) : Variables have very similar names "token0" and "tokenA".
Pos: 249:16:

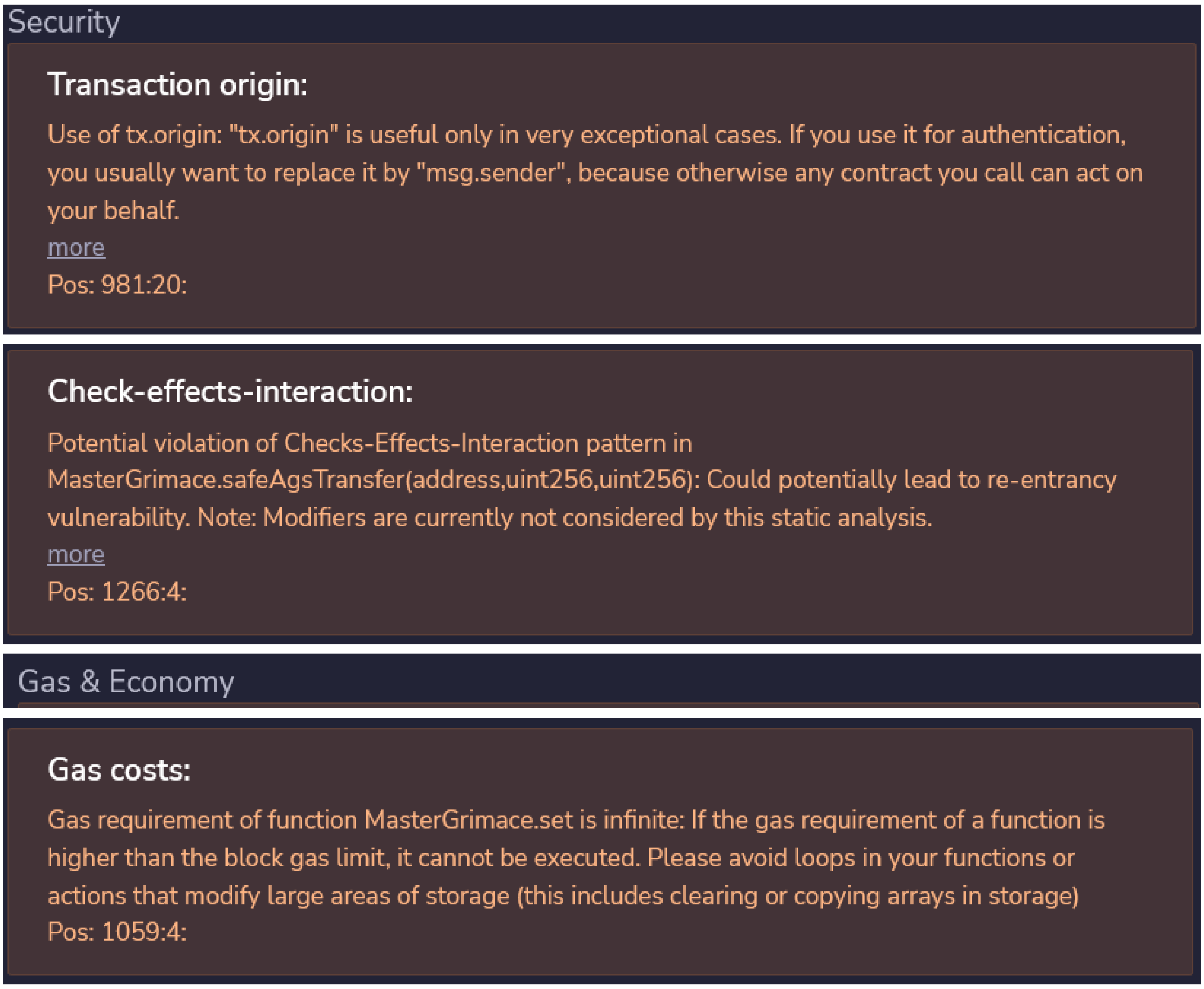
Guard conditions:

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: 252:8:

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: 205:12:

MasterGrimace.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: 981:20:

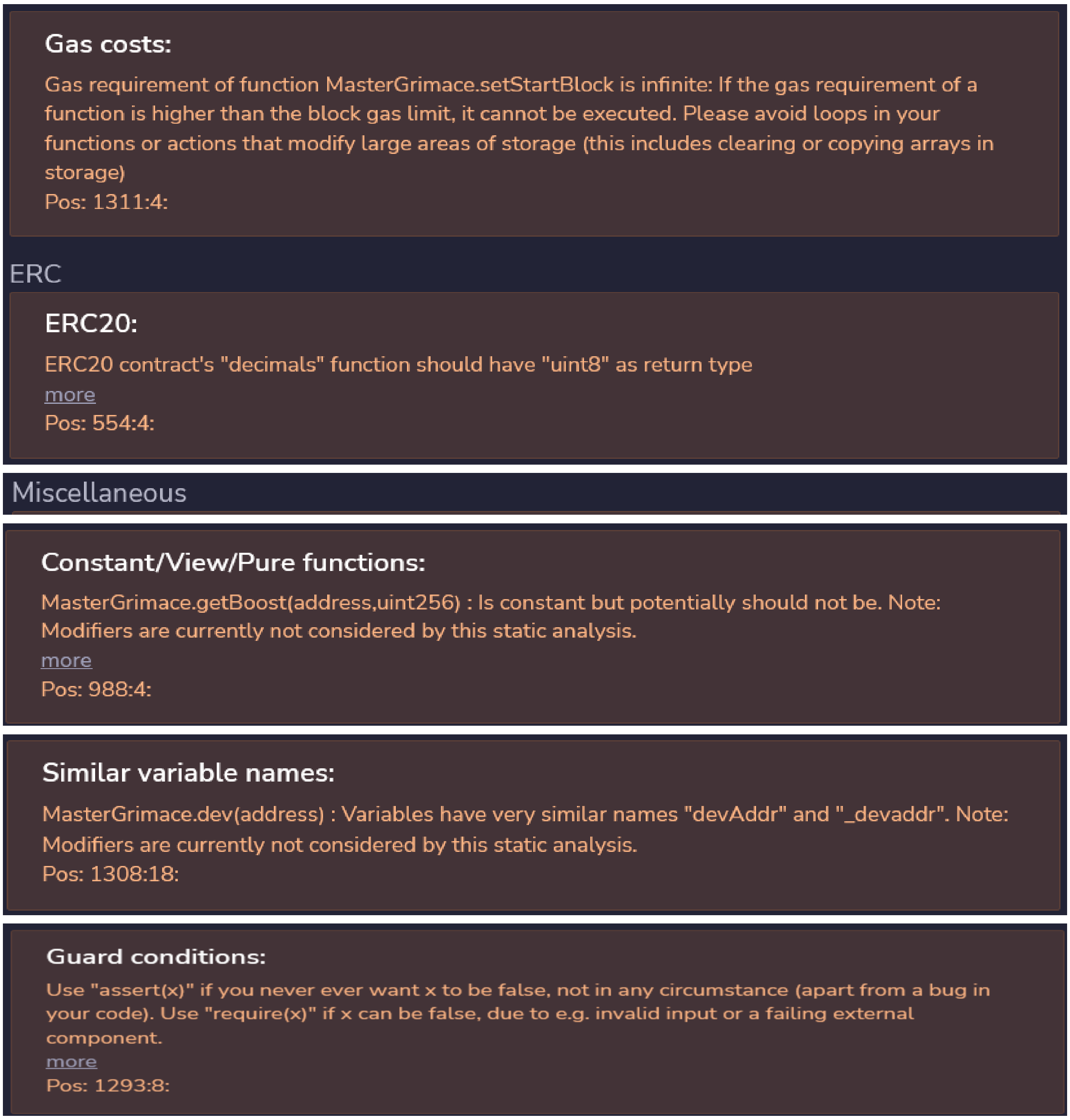
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in MasterGrimace.safeAgsTransfer(address,uint256,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1266:4:

Gas & Economy

Gas costs:

Gas requirement of function MasterGrimace.set 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: 1059:4:



Gas costs:

Gas requirement of function MasterGrimace.setStartBlock 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: 1311:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 554:4:

Miscellaneous

Constant/View/Pure functions:

MasterGrimace.getBoost(address,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 988:4:

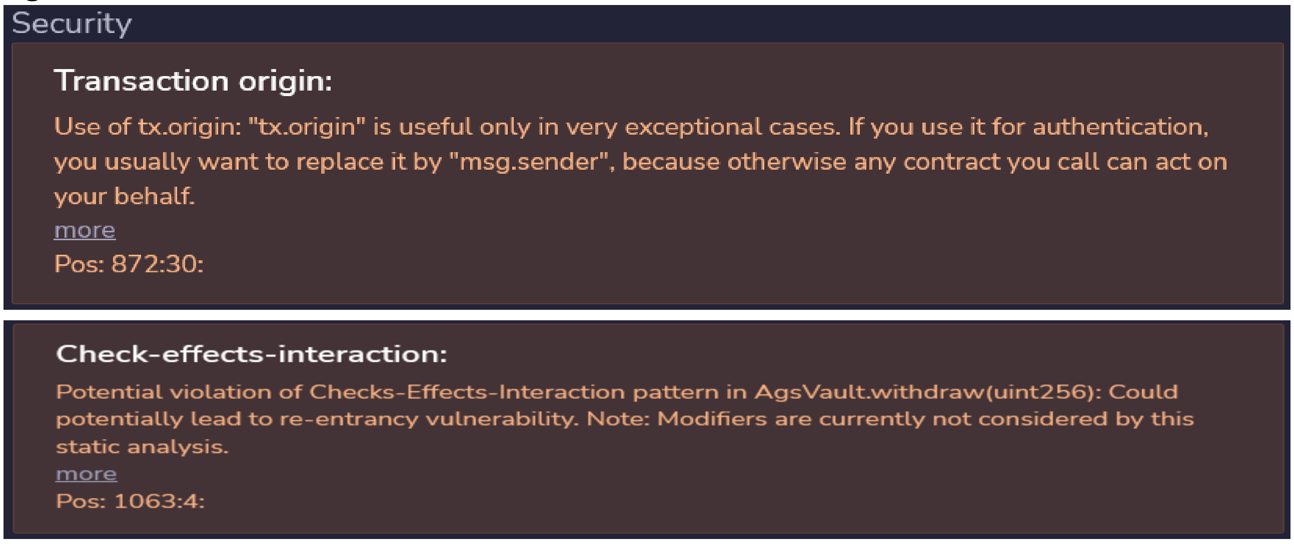
Similar variable names:

MasterGrimace.dev(address) : Variables have very similar names "devAddr" and "_devaddr". Note: Modifiers are currently not considered by this static analysis.
Pos: 1308:18:

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: 1293:8:

AgsVault.sol



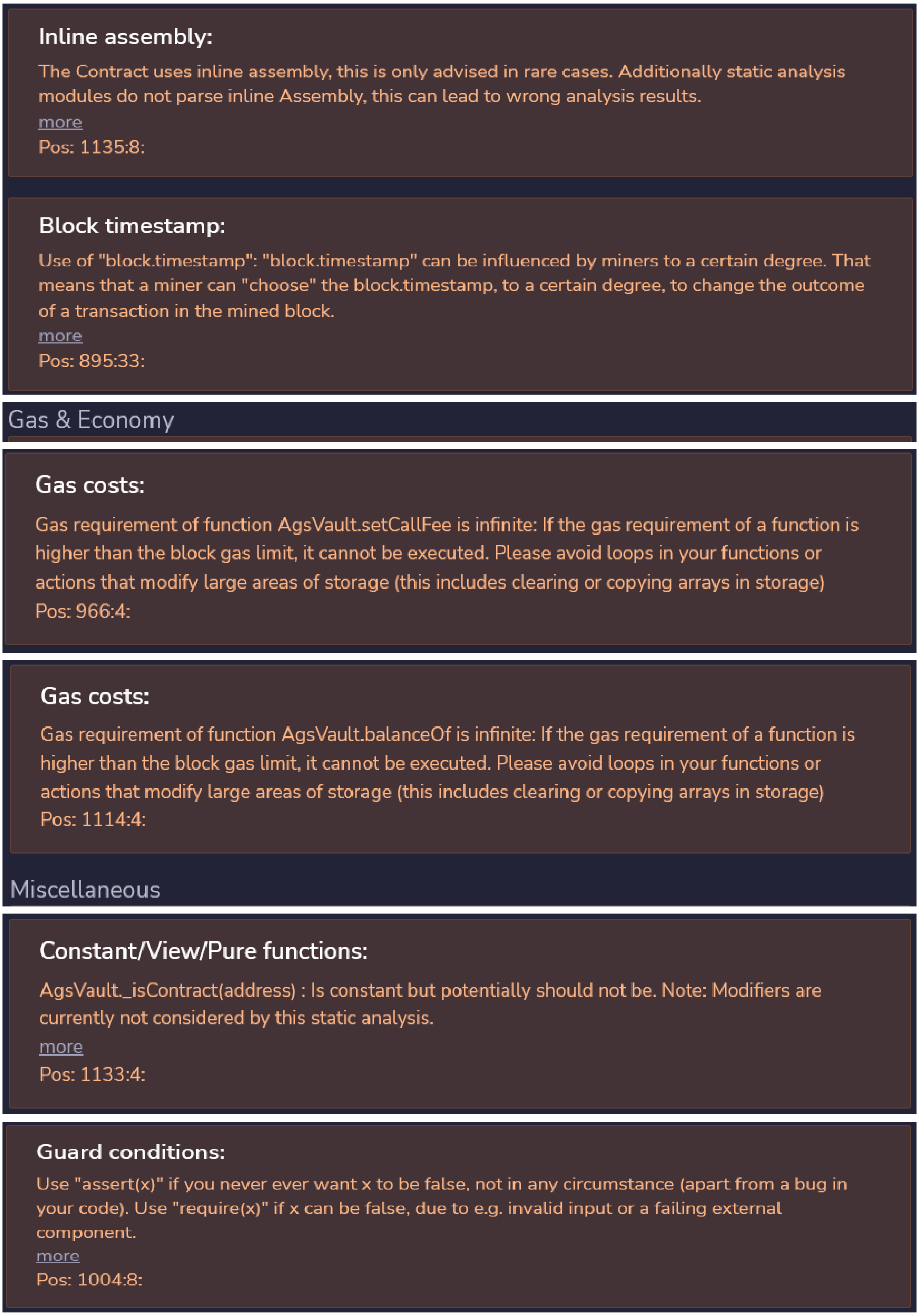
Security

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: 872:30:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in AgsVault.withdraw(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1063:4:



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: 1135:8:

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: 895:33:

Gas & Economy

Gas costs:

Gas requirement of function AgsVault.setCallFee 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: 966:4:

Gas costs:

Gas requirement of function AgsVault.balanceOf 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: 1114:4:

Miscellaneous

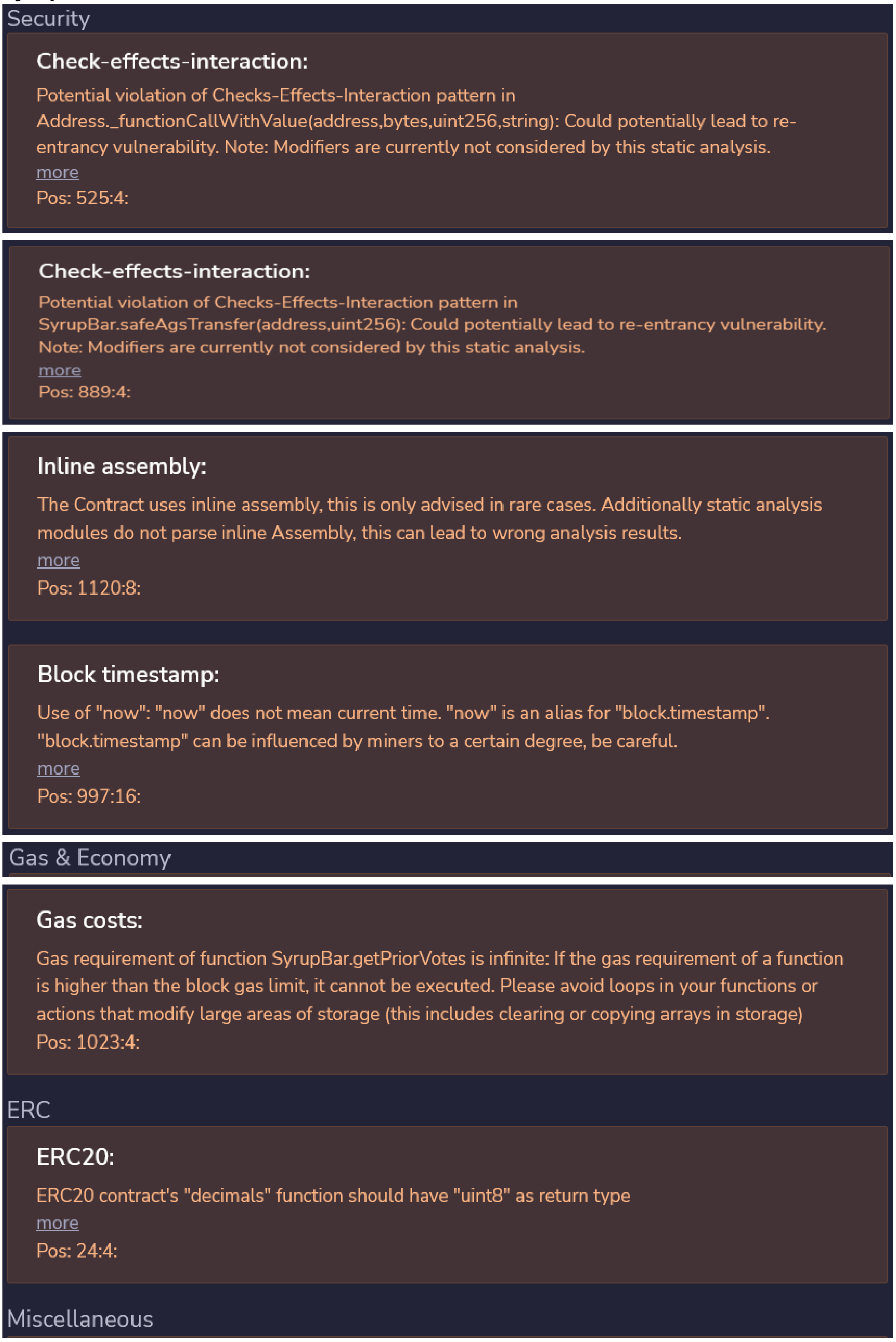
Constant/View/Pure functions:

AgsVault._isContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1133:4:

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: 1004:8:

SyrupBar.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 525:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SyrupBar.safeAgsTransfer(address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 889:4:

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: 1120:8:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp". "block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 997:16:

Gas & Economy

Gas costs:

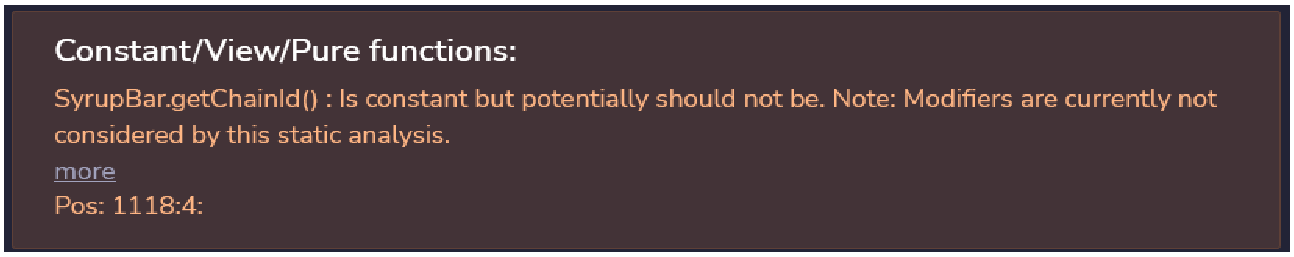
Gas requirement of function SyrupBar.getPriorVotes is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1023:4:

ERC

ERC20:

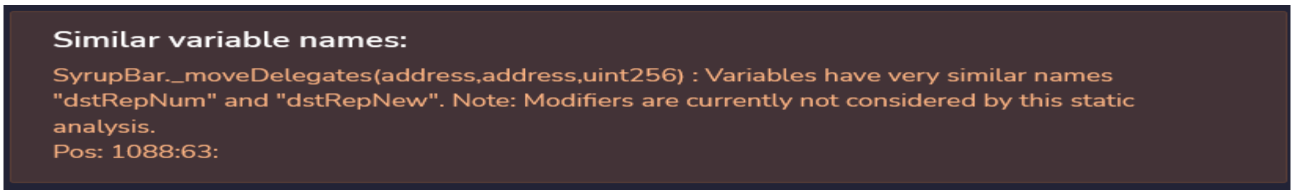
ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 24:4:

Miscellaneous



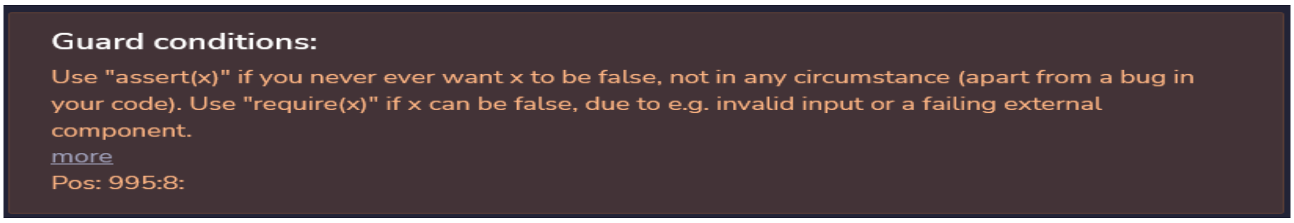
Constant/View/Pure functions:

SyrupBar.getChainId() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1118:4:



Similar variable names:

SyrupBar._moveDelegates(address,address,uint256) : Variables have very similar names "dstRepNum" and "dstRepNew". Note: Modifiers are currently not considered by this static analysis.
Pos: 1088:63:

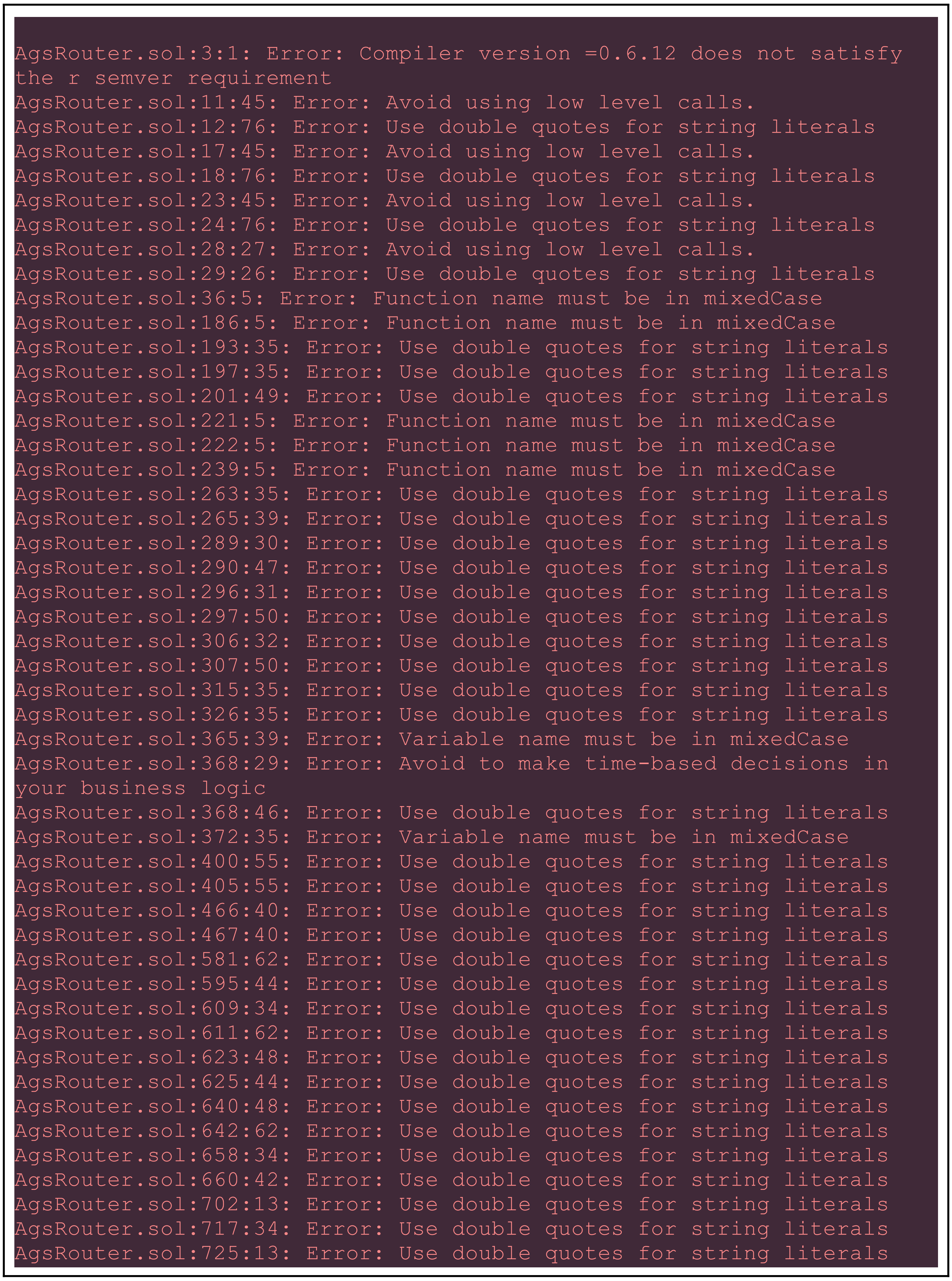


Guard conditions:

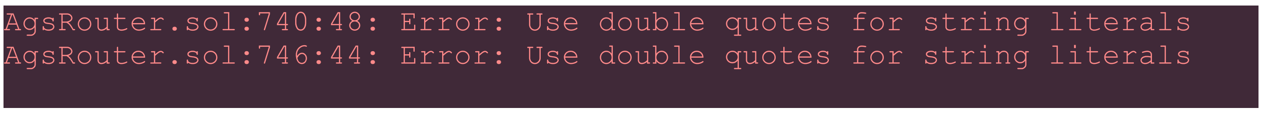
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 995:8:

Solhint Linter

AgsRouter.sol

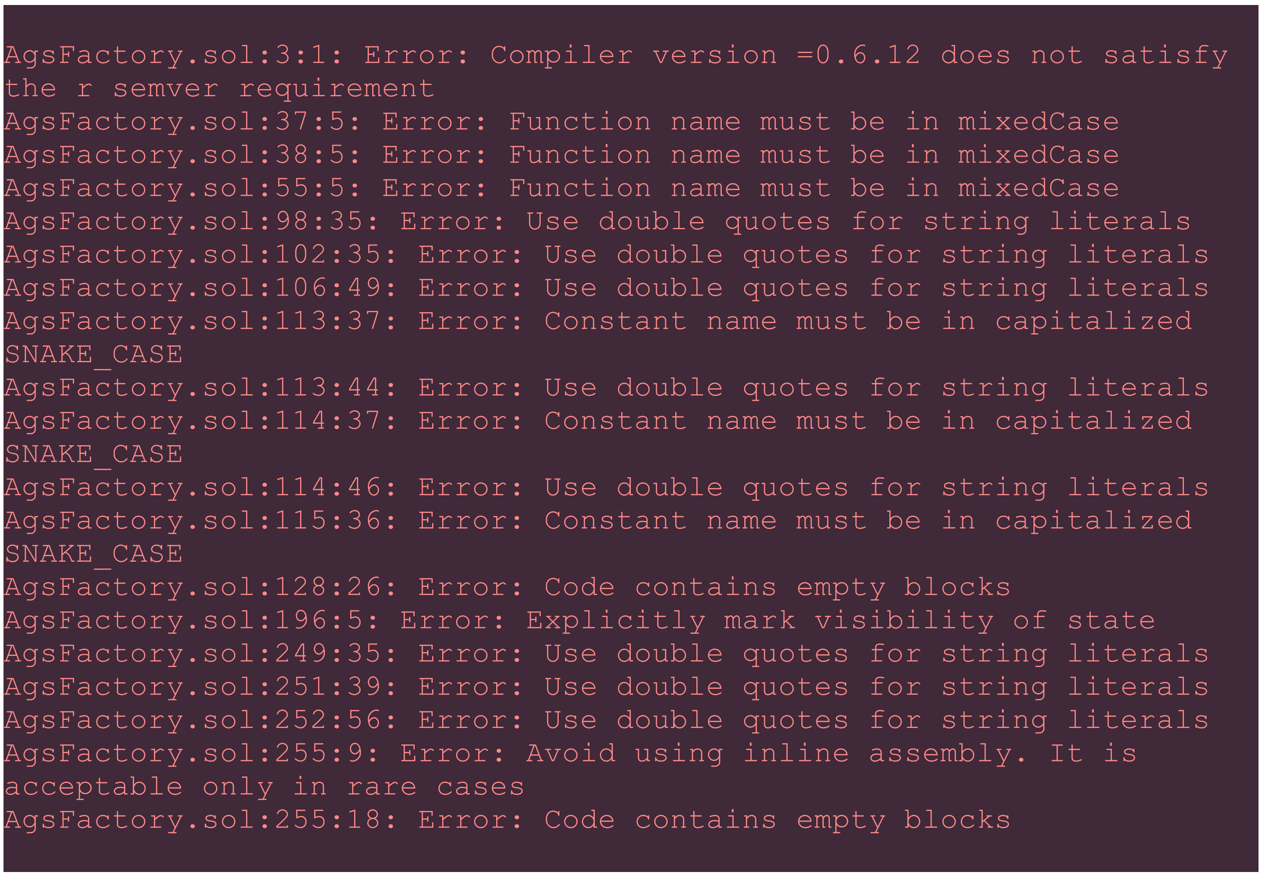


```
AgsRouter.sol:3:1: Error: Compiler version =0.6.12 does not satisfy
the r semver requirement
AgsRouter.sol:11:45: Error: Avoid using low level calls.
AgsRouter.sol:12:76: Error: Use double quotes for string literals
AgsRouter.sol:17:45: Error: Avoid using low level calls.
AgsRouter.sol:18:76: Error: Use double quotes for string literals
AgsRouter.sol:23:45: Error: Avoid using low level calls.
AgsRouter.sol:24:76: Error: Use double quotes for string literals
AgsRouter.sol:28:27: Error: Avoid using low level calls.
AgsRouter.sol:29:26: Error: Use double quotes for string literals
AgsRouter.sol:36:5: Error: Function name must be in mixedCase
AgsRouter.sol:186:5: Error: Function name must be in mixedCase
AgsRouter.sol:193:35: Error: Use double quotes for string literals
AgsRouter.sol:197:35: Error: Use double quotes for string literals
AgsRouter.sol:201:49: Error: Use double quotes for string literals
AgsRouter.sol:221:5: Error: Function name must be in mixedCase
AgsRouter.sol:222:5: Error: Function name must be in mixedCase
AgsRouter.sol:239:5: Error: Function name must be in mixedCase
AgsRouter.sol:263:35: Error: Use double quotes for string literals
AgsRouter.sol:265:39: Error: Use double quotes for string literals
AgsRouter.sol:289:30: Error: Use double quotes for string literals
AgsRouter.sol:290:47: Error: Use double quotes for string literals
AgsRouter.sol:296:31: Error: Use double quotes for string literals
AgsRouter.sol:297:50: Error: Use double quotes for string literals
AgsRouter.sol:306:32: Error: Use double quotes for string literals
AgsRouter.sol:307:50: Error: Use double quotes for string literals
AgsRouter.sol:315:35: Error: Use double quotes for string literals
AgsRouter.sol:326:35: Error: Use double quotes for string literals
AgsRouter.sol:365:39: Error: Variable name must be in mixedCase
AgsRouter.sol:368:29: Error: Avoid to make time-based decisions in
your business logic
AgsRouter.sol:368:46: Error: Use double quotes for string literals
AgsRouter.sol:372:35: Error: Variable name must be in mixedCase
AgsRouter.sol:400:55: Error: Use double quotes for string literals
AgsRouter.sol:405:55: Error: Use double quotes for string literals
AgsRouter.sol:466:40: Error: Use double quotes for string literals
AgsRouter.sol:467:40: Error: Use double quotes for string literals
AgsRouter.sol:581:62: Error: Use double quotes for string literals
AgsRouter.sol:595:44: Error: Use double quotes for string literals
AgsRouter.sol:609:34: Error: Use double quotes for string literals
AgsRouter.sol:611:62: Error: Use double quotes for string literals
AgsRouter.sol:623:48: Error: Use double quotes for string literals
AgsRouter.sol:625:44: Error: Use double quotes for string literals
AgsRouter.sol:640:48: Error: Use double quotes for string literals
AgsRouter.sol:642:62: Error: Use double quotes for string literals
AgsRouter.sol:658:34: Error: Use double quotes for string literals
AgsRouter.sol:660:42: Error: Use double quotes for string literals
AgsRouter.sol:702:13: Error: Use double quotes for string literals
AgsRouter.sol:717:34: Error: Use double quotes for string literals
AgsRouter.sol:725:13: Error: Use double quotes for string literals
```



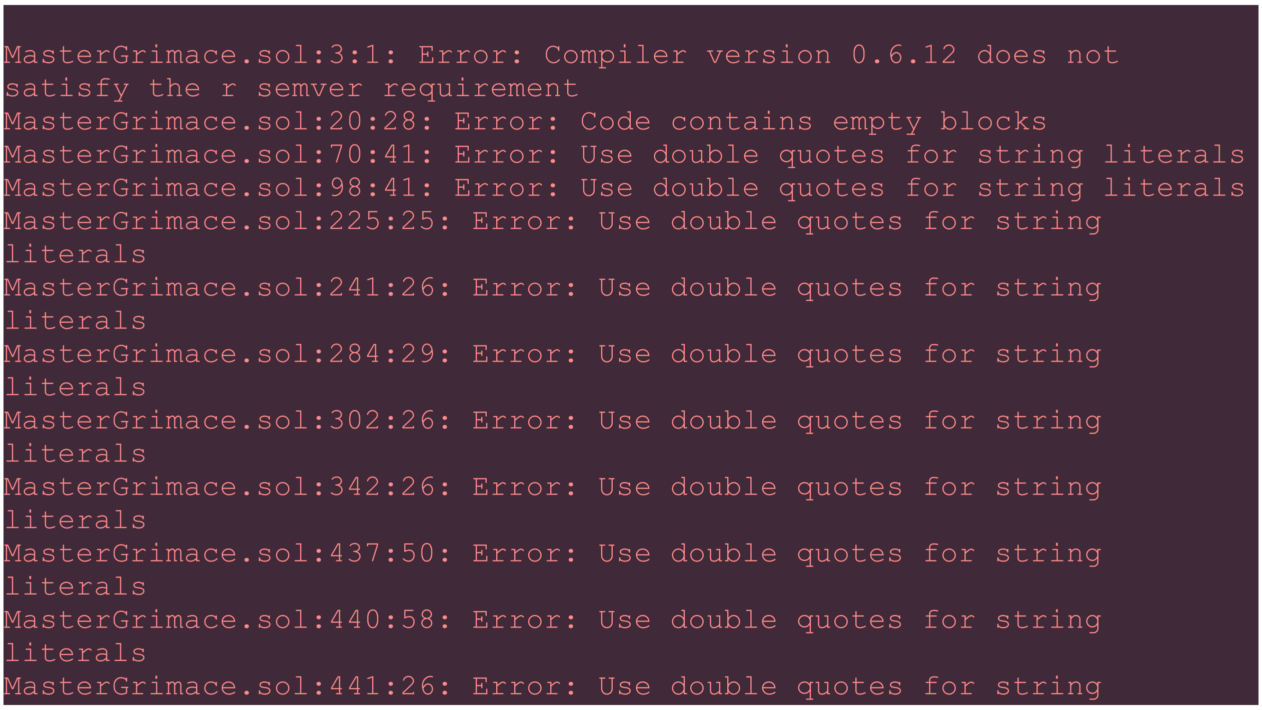
```
AgsRouter.sol:740:48: Error: Use double quotes for string literals
AgsRouter.sol:746:44: Error: Use double quotes for string literals
```

AgsFactory.sol

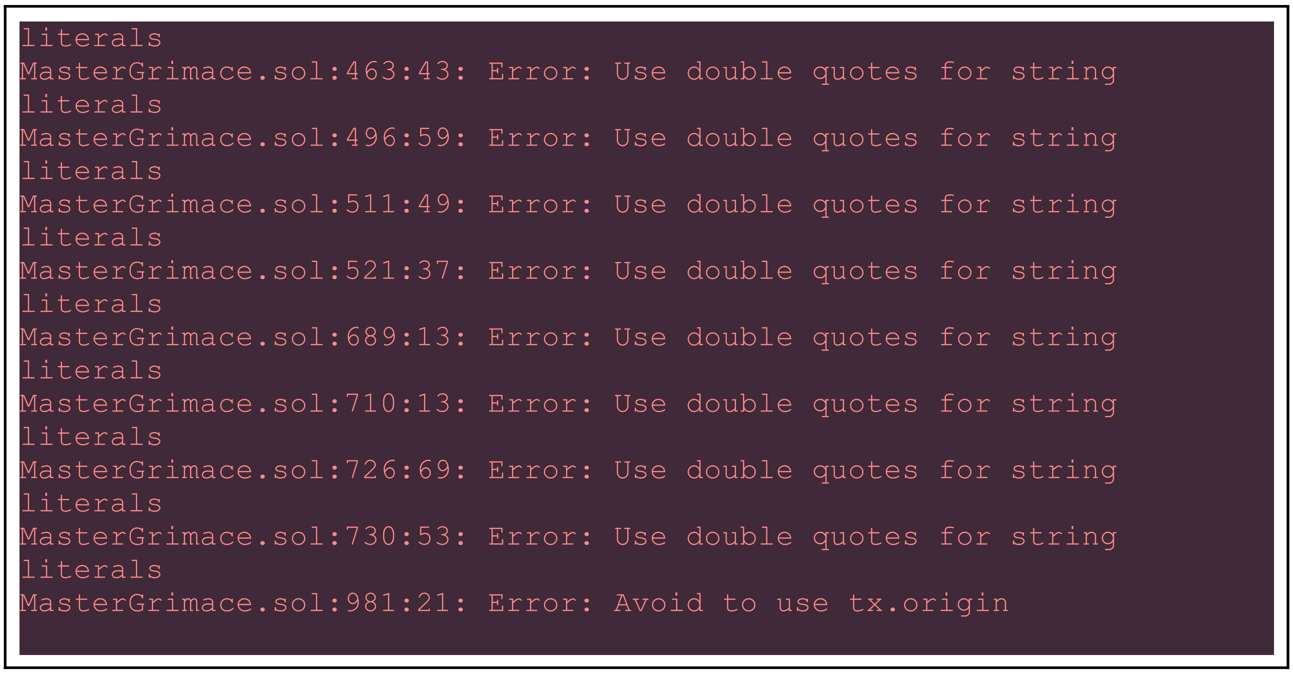


```
AgsFactory.sol:3:1: Error: Compiler version =0.6.12 does not satisfy
the r semver requirement
AgsFactory.sol:37:5: Error: Function name must be in mixedCase
AgsFactory.sol:38:5: Error: Function name must be in mixedCase
AgsFactory.sol:55:5: Error: Function name must be in mixedCase
AgsFactory.sol:98:35: Error: Use double quotes for string literals
AgsFactory.sol:102:35: Error: Use double quotes for string literals
AgsFactory.sol:106:49: Error: Use double quotes for string literals
AgsFactory.sol:113:37: Error: Constant name must be in capitalized
SNAKE_CASE
AgsFactory.sol:113:44: Error: Use double quotes for string literals
AgsFactory.sol:114:37: Error: Constant name must be in capitalized
SNAKE_CASE
AgsFactory.sol:114:46: Error: Use double quotes for string literals
AgsFactory.sol:115:36: Error: Constant name must be in capitalized
SNAKE_CASE
AgsFactory.sol:128:26: Error: Code contains empty blocks
AgsFactory.sol:196:5: Error: Explicitly mark visibility of state
AgsFactory.sol:249:35: Error: Use double quotes for string literals
AgsFactory.sol:251:39: Error: Use double quotes for string literals
AgsFactory.sol:252:56: Error: Use double quotes for string literals
AgsFactory.sol:255:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
AgsFactory.sol:255:18: Error: Code contains empty blocks
```

MasterGrimace.sol

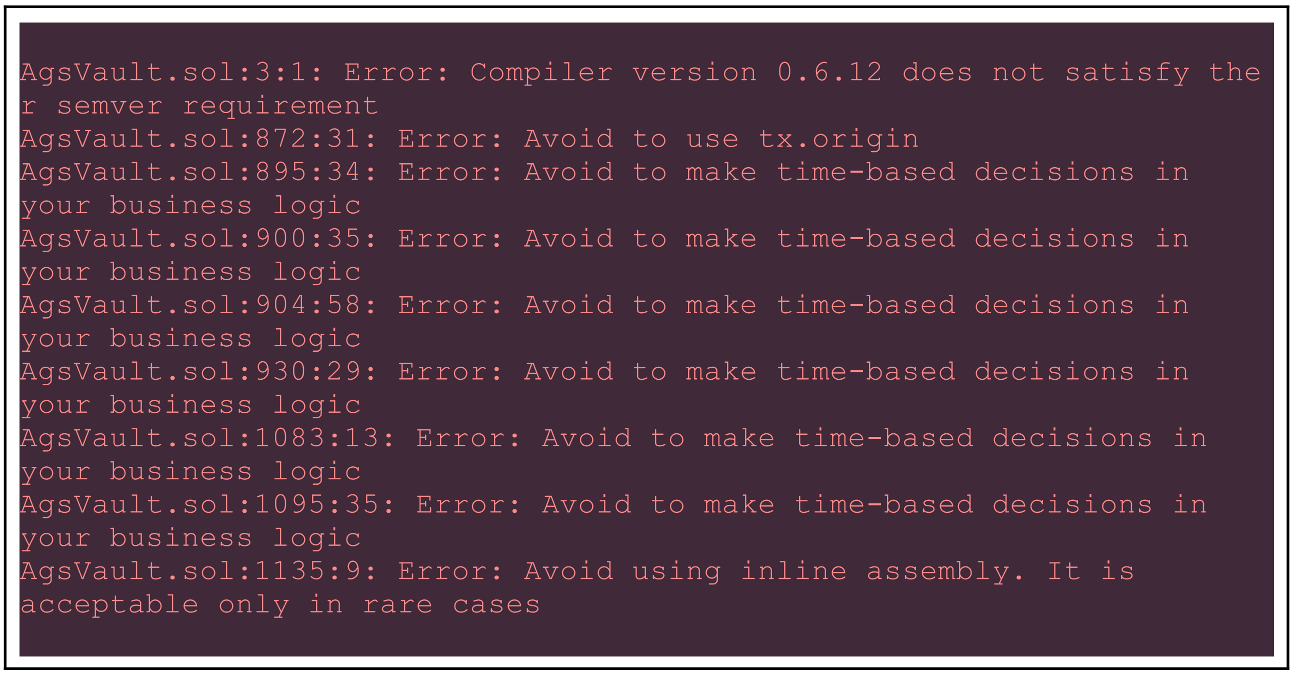


```
MasterGrimace.sol:3:1: Error: Compiler version 0.6.12 does not
satisfy the r semver requirement
MasterGrimace.sol:20:28: Error: Code contains empty blocks
MasterGrimace.sol:70:41: Error: Use double quotes for string literals
MasterGrimace.sol:98:41: Error: Use double quotes for string literals
MasterGrimace.sol:225:25: Error: Use double quotes for string
literals
MasterGrimace.sol:241:26: Error: Use double quotes for string
literals
MasterGrimace.sol:284:29: Error: Use double quotes for string
literals
MasterGrimace.sol:302:26: Error: Use double quotes for string
literals
MasterGrimace.sol:342:26: Error: Use double quotes for string
literals
MasterGrimace.sol:437:50: Error: Use double quotes for string
literals
MasterGrimace.sol:440:58: Error: Use double quotes for string
literals
MasterGrimace.sol:441:26: Error: Use double quotes for string
```



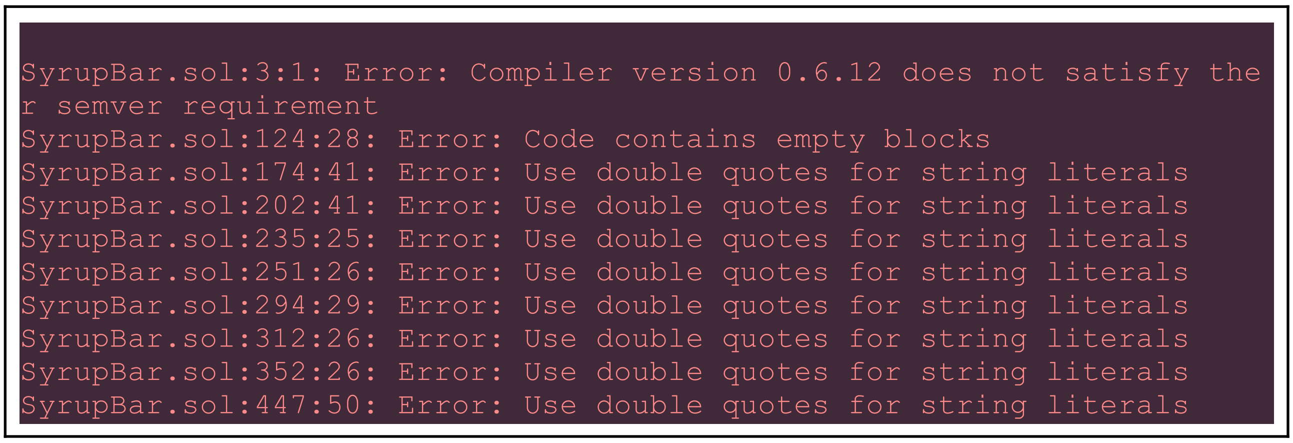
```
literals
MasterGrimace.sol:463:43: Error: Use double quotes for string
literals
MasterGrimace.sol:496:59: Error: Use double quotes for string
literals
MasterGrimace.sol:511:49: Error: Use double quotes for string
literals
MasterGrimace.sol:521:37: Error: Use double quotes for string
literals
MasterGrimace.sol:689:13: Error: Use double quotes for string
literals
MasterGrimace.sol:710:13: Error: Use double quotes for string
literals
MasterGrimace.sol:726:69: Error: Use double quotes for string
literals
MasterGrimace.sol:730:53: Error: Use double quotes for string
literals
MasterGrimace.sol:981:21: Error: Avoid to use tx.origin
```

AgsVault.sol

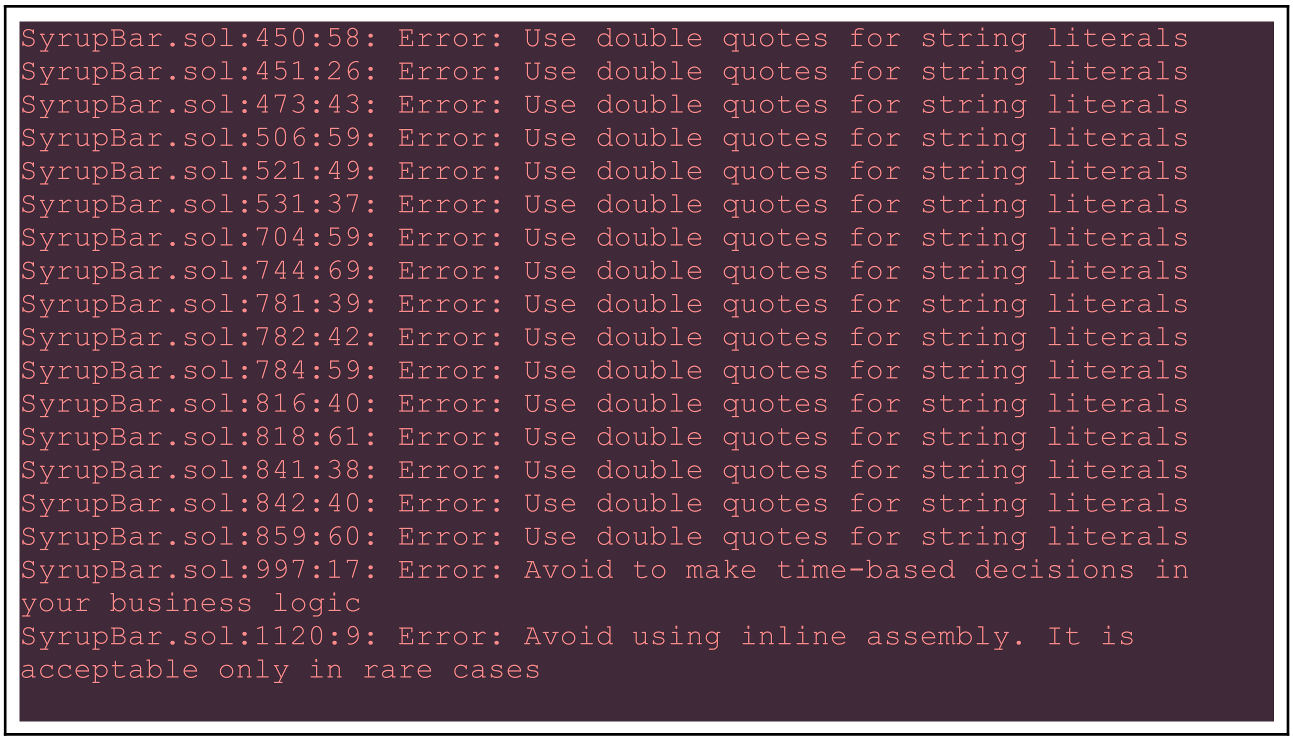


```
AgsVault.sol:3:1: Error: Compiler version 0.6.12 does not satisfy the
r semver requirement
AgsVault.sol:872:31: Error: Avoid to use tx.origin
AgsVault.sol:895:34: Error: Avoid to make time-based decisions in
your business logic
AgsVault.sol:900:35: Error: Avoid to make time-based decisions in
your business logic
AgsVault.sol:904:58: Error: Avoid to make time-based decisions in
your business logic
AgsVault.sol:930:29: Error: Avoid to make time-based decisions in
your business logic
AgsVault.sol:1083:13: Error: Avoid to make time-based decisions in
your business logic
AgsVault.sol:1095:35: Error: Avoid to make time-based decisions in
your business logic
AgsVault.sol:1135:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

SyrupBar.sol



```
SyrupBar.sol:3:1: Error: Compiler version 0.6.12 does not satisfy the
r semver requirement
SyrupBar.sol:124:28: Error: Code contains empty blocks
SyrupBar.sol:174:41: Error: Use double quotes for string literals
SyrupBar.sol:202:41: Error: Use double quotes for string literals
SyrupBar.sol:235:25: Error: Use double quotes for string literals
SyrupBar.sol:251:26: Error: Use double quotes for string literals
SyrupBar.sol:294:29: Error: Use double quotes for string literals
SyrupBar.sol:312:26: Error: Use double quotes for string literals
SyrupBar.sol:352:26: Error: Use double quotes for string literals
SyrupBar.sol:447:50: Error: Use double quotes for string literals
```



```
SyrupBar.sol:450:58: Error: Use double quotes for string literals
SyrupBar.sol:451:26: Error: Use double quotes for string literals
SyrupBar.sol:473:43: Error: Use double quotes for string literals
SyrupBar.sol:506:59: Error: Use double quotes for string literals
SyrupBar.sol:521:49: Error: Use double quotes for string literals
SyrupBar.sol:531:37: Error: Use double quotes for string literals
SyrupBar.sol:704:59: Error: Use double quotes for string literals
SyrupBar.sol:744:69: Error: Use double quotes for string literals
SyrupBar.sol:781:39: Error: Use double quotes for string literals
SyrupBar.sol:782:42: Error: Use double quotes for string literals
SyrupBar.sol:784:59: Error: Use double quotes for string literals
SyrupBar.sol:816:40: Error: Use double quotes for string literals
SyrupBar.sol:818:61: Error: Use double quotes for string literals
SyrupBar.sol:841:38: Error: Use double quotes for string literals
SyrupBar.sol:842:40: Error: Use double quotes for string literals
SyrupBar.sol:859:60: Error: Use double quotes for string literals
SyrupBar.sol:997:17: Error: Avoid to make time-based decisions in
your business logic
SyrupBar.sol:1120:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

Overall Software analysis result:

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

EtherAuthority