EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	AmpleSwap
Website:	ampleswap.com
Platform:	Binance Smart Chain
Language:	Solidity
Date:	September 27th, 2021

Table of contents

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Introduction

EtherAuthority was contracted by the AmpleSwap team to perform the Security audit of AmpleSwap 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 September 27th, 2021.

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

AmpleSwap is the decentralized exchange on Binance Smart Chain. AmpleSwap helps users make the most out of their crypto in three ways: Trade, Earn, and Win.

Audit scope

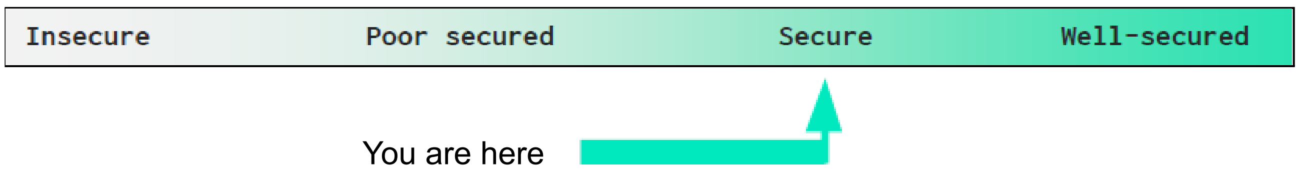
Name	**Code Review and Security Analysis Report for AmpleSwap Protocol Smart Contracts**
Platform	**BSC / Solidity**
File 1	AmpleFactory.sol
File 1 MD5 Hash	6D11EAB401D4F13603F930CEB8770D99
File 2	AmpleRouter.sol
File 2 MD5 Hash	43F779884FF61626BBDB31F411150032
File 3	AmpleToken.sol
File 3 MD5 Hash	B8C7DB649ED2C1842CF2D17148009F9E
File 4	MasterChef.sol
File 4 MD5 Hash	85595CEF85327154A5DB19C0F7BA1A79
File 5	SyrupBar.sol
File 5 MD5 Hash	E32F26B0601D91A3C1EDE08D5B4BFAE3
Audit Date	September 27th, 2021
Revision Date	September 29th, 2021

Claimed Smart Contracts Features

Claimed Feature Detail	Our Observation
File 1: AmpleFactory.sol ● anyone can create trading pairs of tokens	YES, This is valid.
File 2: AmpleRouter.sol ● Add/remove liquidity ● Token swapping	YES, This is valid.
File 3: AmpleToken.sol ● Name: AmpleSwap Token ● Symbol: AMPLE ● Decimals: 18 ● Maximum Supply: 1,000,000,000	YES, This is valid.
File 4: MasterChef.sol ● Bonus Multiplier: 1 ● Mints Ample and Syrup Bar tokens as needed ● MasterChef owner will be time locked	**MasterChef contract owner wallet will be time locked after farming launch.**
File 5: SyrupBar.sol ● Name: SyrupBar Token ● Symbol: SYRUP ● Decimals: 18	YES, This is valid.

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Technically Secured”**. These contracts 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, 1 high, 0 medium and 3 low and some very low level issues.

These issues are fixed / acknowledged in the revised version of the 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	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	Passed
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	"Out of Gas" Issue	Passed
	High consumption 'for/while' loop	Moderated
	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

These audit scope have 5 smart contracts files. Smart contracts also contain Libraries, Smart contracts, inherits and Interfaces. These are compact and well written contracts.

The libraries in AmpleSwap 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 AmpleSwap Protocol.

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

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

Documentation

We were given an AmpleSwap Protocol smart contracts code in the form BscScan web link. The hash of that code is mentioned above in the table.

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

Another source of information was its official website https://ampleswap.com/ which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.
Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

AmpleFactory.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	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

AmpleRouter.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	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	external	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

AmpleToken.sol

Sl.	**Functions**	**Type**	**Observation**	**Conclusion**
1	mintFor	write	access only Owner	No Issue
2	mint	write	access only Owner	No Issue
3	delegates	external	Passed	No Issue
4	delegate	external	Passed	No Issue
5	delegateBySig	external	Handle signatures securely	No Issue
6	getCurrentVotes	external	Passed	No Issue
7	getPriorVotes	external	Infinite loop possibility	Refer audit finding section
8	_delegate	internal	Passed	No Issue
9	_moveDelegates	internal	Passed	No Issue
10	_writeCheckpoint	internal	Passed	No Issue
11	safe32	internal	Passed	No Issue
12	getChainId	internal	Passed	No Issue
13	getOwner	external	Passed	No Issue
14	name	read	Passed	No Issue
15	decimals	read	Passed	No Issue
16	symbol	read	Passed	No Issue
17	totalSupply	read	Passed	No Issue
18	balanceOf	read	Passed	No Issue
19	transfer	write	Passed	No Issue
20	allowance	write	Passed	No Issue
21	approve	write	Passed	No Issue
22	transferFrom	write	Passed	No Issue
23	increaseAllowance	write	Passed	No Issue
24	decreaseAllowance	write	Passed	No Issue
25	mint	write	access only Owner	No Issue
26	_transfer	internal	Passed	No Issue
27	_mint	internal	Passed	No Issue
28	_burn	internal	Passed	No Issue
29	_approve	internal	Passed	No Issue

30	_burnFrom	internal	Passed	No Issue
31	owner	read	Passed	No Issue
32	onlyOwner	modifier	Passed	No Issue
33	renounceOwnership	write	access only Owner	No Issue
34	transferOwnership	write	access only Owner	No Issue

MasterChef.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	updateMultiplier	write	access only Owner	No Issue
3	poolLength	external	Passed	No Issue
4	add	write	Input validation missing	Refer audit finding section
5	set	write	access only Owner	No Issue
6	updateStakingPool	internal	Passed	No Issue
7	setMigrator / migrate	write	Rugpull possibility	Removed this code
8	getMultiplier	read	Passed	No Issue
9	pendingCake	external	Passed	No Issue
10	massUpdatePools	write	Infinite loop possibility	Refer audit finding section
11	updatePool	write	Passed	No Issue
12	deposit	write	Passed	No Issue
13	withdraw	write	Passed	No Issue
14	enterStaking	write	Passed	No Issue
15	leaveStaking	write	Passed	No Issue
16	emergencyWithdraw	write	Passed	No Issue
17	safeCakeTransfer	internal	Passed	No Issue
18	transferAmpleTokenOwnerShip	write	access only Owner	No Issue
19	transferSyrupOwnerShip	write	access only Owner	No Issue
20	mint	write	access only Owner	No Issue
21	burn	write	access only Owner	No Issue
22	safeCakeTransfer	write	access only Owner	No Issue

SyrupBar.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	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	getOwner	external	Passed	No Issue
7	name	read	Passed	No Issue
8	decimals	read	Passed	No Issue
9	symbol	read	Passed	No Issue
10	totalSupply	read	Passed	No Issue
11	balanceOf	read	Passed	No Issue
12	transfer	read	Passed	No Issue
13	allowance	read	Passed	No Issue
14	approve	write	Passed	No Issue
15	transferFrom	write	Passed	No Issue
16	increaseAllowance	write	Passed	No Issue
17	decreaseAllowance	write	Passed	No Issue
19	_transfer	internal	Passed	No Issue
20	_mint	internal	Passed	No Issue
21	_burn	internal	Passed	No Issue
22	_approve	internal	Passed	No Issue
23	_burnFrom	internal	Passed	No Issue
24	mintFor	write	access only Owner	No Issue
25	mint	write	access only Owner	No Issue
26	delegates	external	Passed	No Issue
27	delegate	external	Passed	No Issue
28	delegateBySig	external	Passed	No Issue
29	getCurrentVotes	external	Passed	No Issue
30	getPriorVotes	external	Passed	No Issue
31	_delegate	internal	Passed	No Issue
32	_moveDelegates	internal	Passed	No Issue
33	_writeCheckpoint	internal	Passed	No Issue
34	safe32	internal	Passed	No Issue
35	getChainId	internal	Passed	No Issue
36	mint	write	access only Owner	No Issue
37	burn	write	access only Owner	No Issue
38	safeCakeTransfer	write	Owner can transfer all syrup tokens	No Issue when owner is masterchef

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.

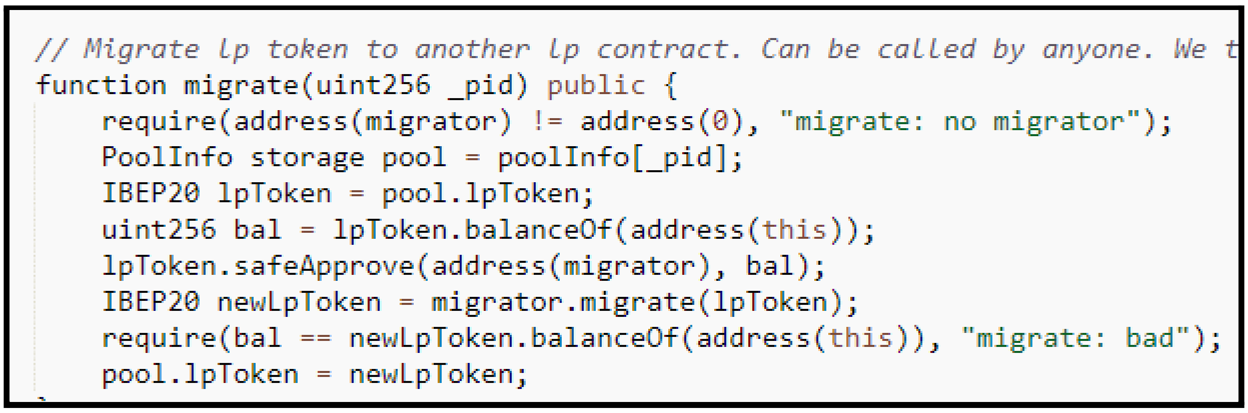
Audit Findings

Critical

No Critical severity vulnerabilities were found.

High

(1) Migrator code present - Masterchef.sol



```
// Migrate lp token to another lp contract. Can be called by anyone. We t
function migrate(uint256 _pid) public {
    require(address(migrator) != address(0), "migrate: no migrator");
    PoolInfo storage pool = poolInfo[_pid];
    IBEP20 lpToken = pool.lpToken;
    uint256 bal = lpToken.balanceOf(address(this));
    lpToken.safeApprove(address(migrator), bal);
    IBEP20 newLpToken = migrator.migrate(lpToken);
    require(bal == newLpToken.balanceOf(address(this)), "migrate: bad");
    pool.lpToken = newLpToken;
```

Migrator code in pancakeswap fork project is always criticized for rugpull. We suggest removing this if not really needed.

Reference: https://goosedefi.gitbook.io/goose-finance/security/rugpull-migrator-code

Update: This migrator code is removed in the revised contract code

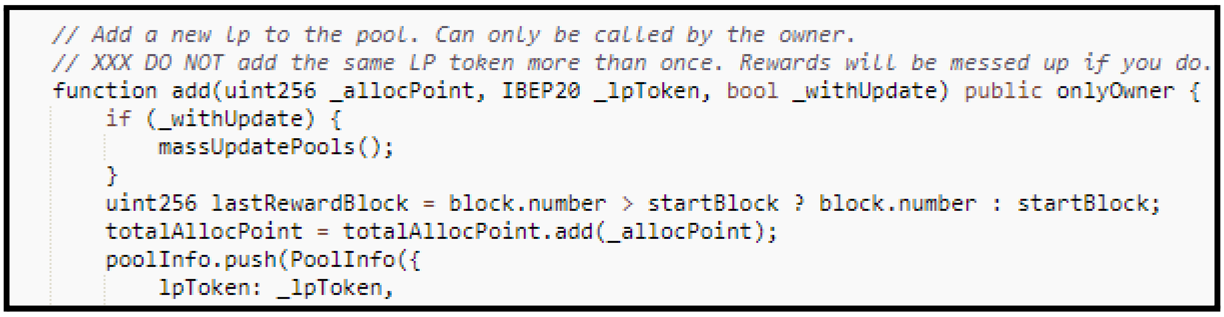
https://bscscan.com/address/0xeb642d600bf593cb21e1551e9a15426ff6d42f82#code

Medium

No Medium severity vulnerabilities were found.

Low

(1) Duplicate LP tokens may create discrepancy - MasterChef.sol

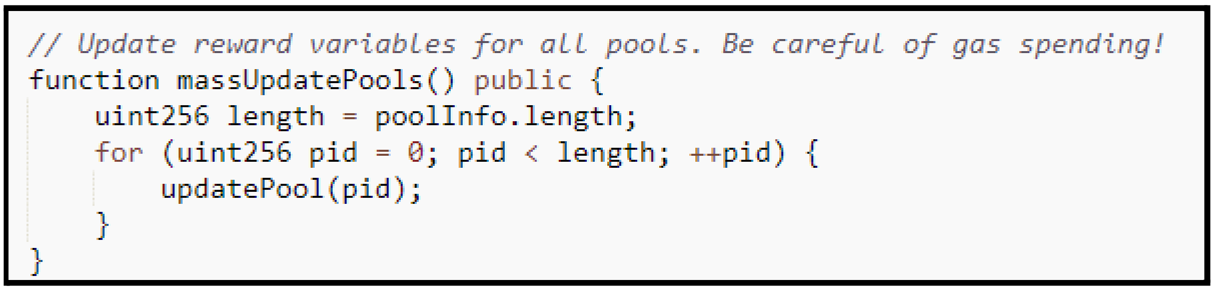


```
// Add a new lp to the pool. Can only be called by the owner.
// XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.
function add(uint256 _allocPoint, IBEP20 _lpToken, bool _withUpdate) public onlyOwner {
    if (_withUpdate) {
        massUpdatePools();
    }
    uint256 lastRewardBlock = block.number > startBlock ? block.number : startBlock;
    totalAllocPoint = totalAllocPoint.add(_allocPoint);
    poolInfo.push(PoolInfo({
        lpToken: _lpToken,
```

As per the comment, the add function must not allow the owner to add the same LP token more than once.

Resolution: We suggest using some conditions to not allow the user to add the same LP token more than once.

(2) Infinite loop possibility - MasterChef.sol



```
// Update reward variables for all pools. Be careful of gas spending!
function massUpdatePools() public {
    uint256 length = poolInfo.length;
    for (uint256 pid = 0; pid < length; ++pid) {
        updatePool(pid);
    }
}
```

If there are so many pools, then this logic will fail, as it might hit the block's gas limit. If there are very limited pools, then this will work, but will cost more gas.

Resolution: Just use a mapping that will map wallet to bool and make excluded wallets to be true. This logic will not have any gas or scalability issues.

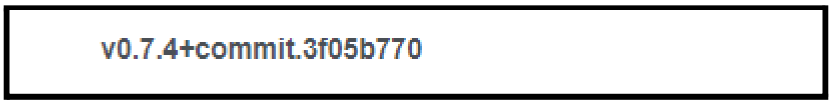
PS: This possibility is also there in AmpleToken and SyrupBar. We suggest adjusting the logic, or keeping array length limited to prevent this issue.

(3) Critical operation lacks event log. It is recommended to fire an event when an important state change operation is happening. Events are missing for below functions in MasterChef contract:

- updateStakingPool - MasterChef contract
- updatePool - MasterChef contract
- migrate - MasterChef contract
- setFeeTo - AmpleFactory contract
- setFeeToSetter - AmpleFactory contract

Very Low / Informational / Best practices:

(1) Solidity version:



```
v0.7.4+commit.3f05b770
```

Using the latest solidity will prevent any compiler level bugs.

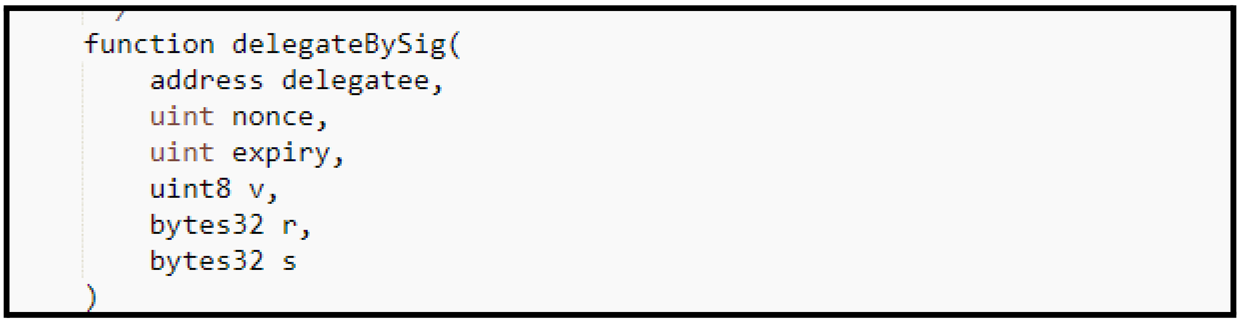
Resolution: Please use 0.8.7 which is the latest version at the time of this audit.

(2) Visibility external over public:

It is recommended to specify function visibility as external instead of public, if that function is not called from the contract internally. This is considered a gas saver.

https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

(3) Handle signature carefully - AmpleToken.sol



```
function delegateBySig(
    address delegatee,
    uint nonce,
    uint expiry,
    uint8 v,
    bytes32 r,
    bytes32 s
)
```

This feature is useful as it allows ease to users to delegate by making signatures. On the client side, these signatures must be handled securely because these signatures will allow anyone to interact with the contract on behalf of the user. Although the risk is very minimal, we suggest handling these signatures securely to prevent any phishing scams.

Centralization

These smart contracts have 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:

- mintFor: The MasterChef owner can create an `_amount` token to `_to`.
- mint: The AmpleToken owner can mint a token.
- updateMultiplier: The MasterChef owner can update multipliers.
- add: The MasterChef owner can add a new lp to the pool.
- set: The MasterChef owner can update the given pool's CAKE allocation point.
- setMigrator: The MasterChef owner can set the migrator contract.
- transferAmpleTokenOwnerShip: The MasterChef owner can transfer ample tokens to ownership.
- transferSyrupOwnerShip: The MasterChef owner can transfer syrup tokens to ownership.
- mint: The Syrup owner can create an `_amount` token to `_to`.
- burn: The Syrup owner can burn an amount from the account.
- safeCakeTransfer: The Syrup owner can call this function.

Conclusion

We were given contract codes. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and they are resolved/acknowledged. So, **it's good to go to production**.

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 **"Technically 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.

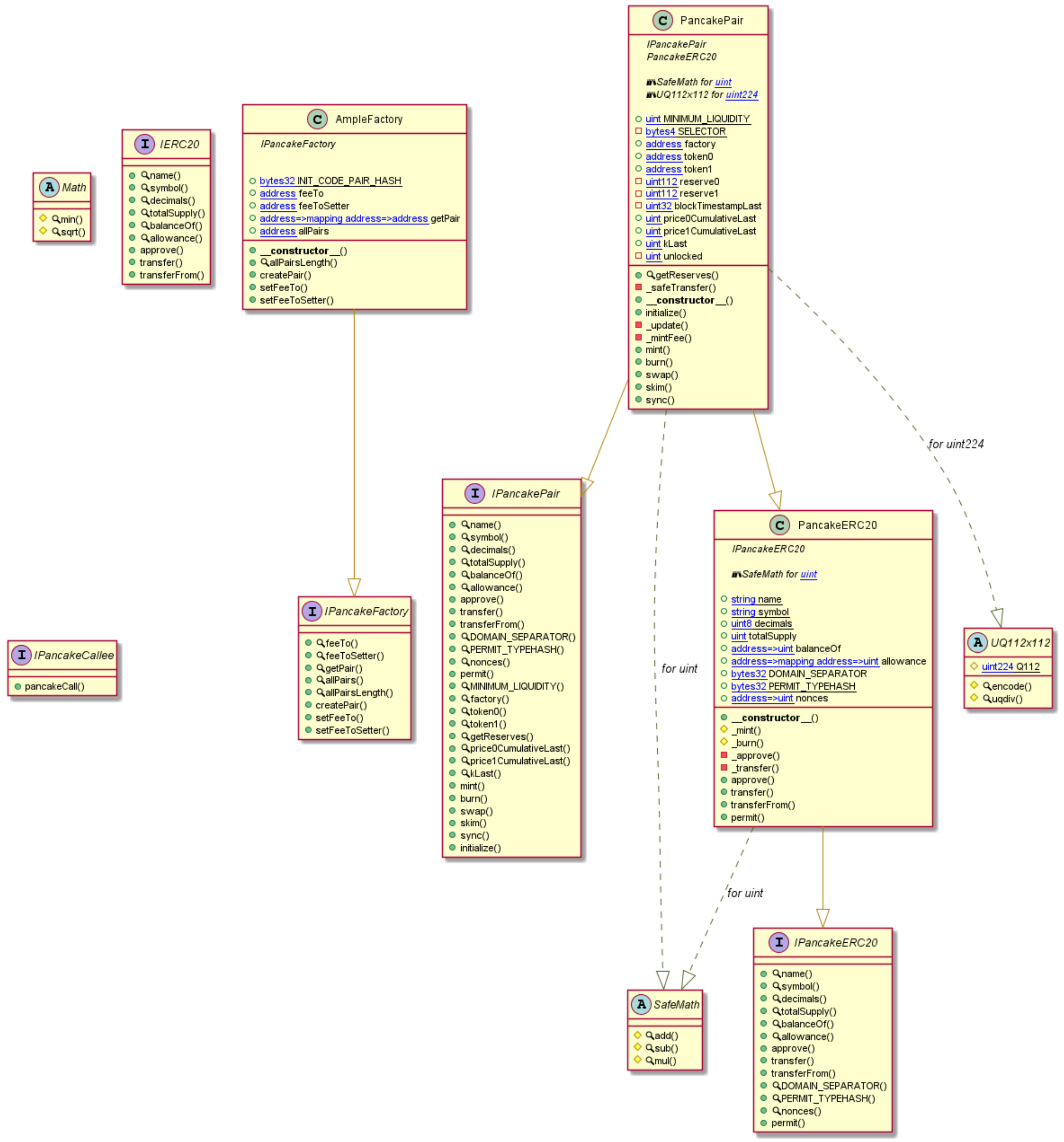
Technical Disclaimer

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.

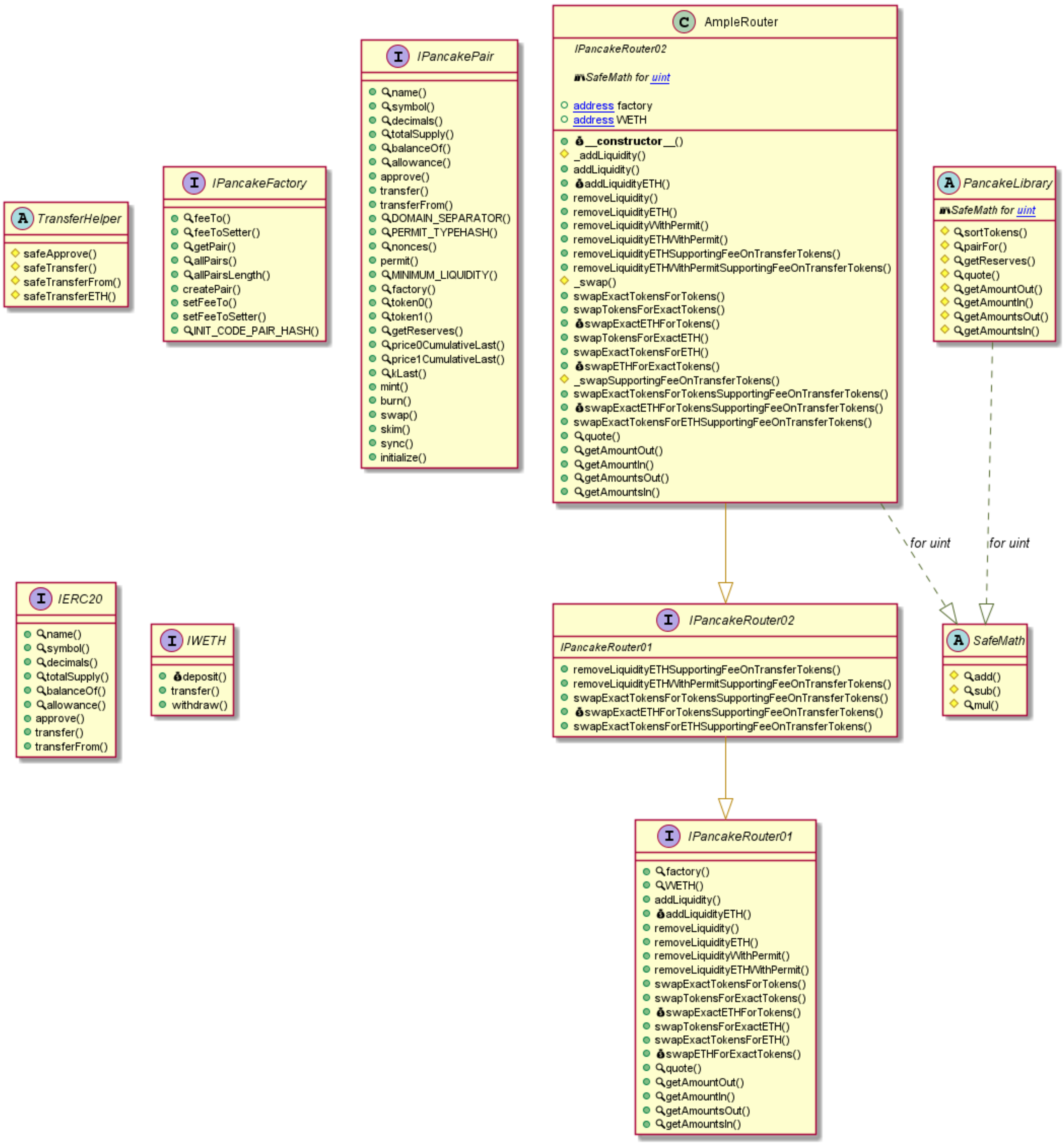
Appendix

Code Flow Diagram - AmpleSwap Protocol

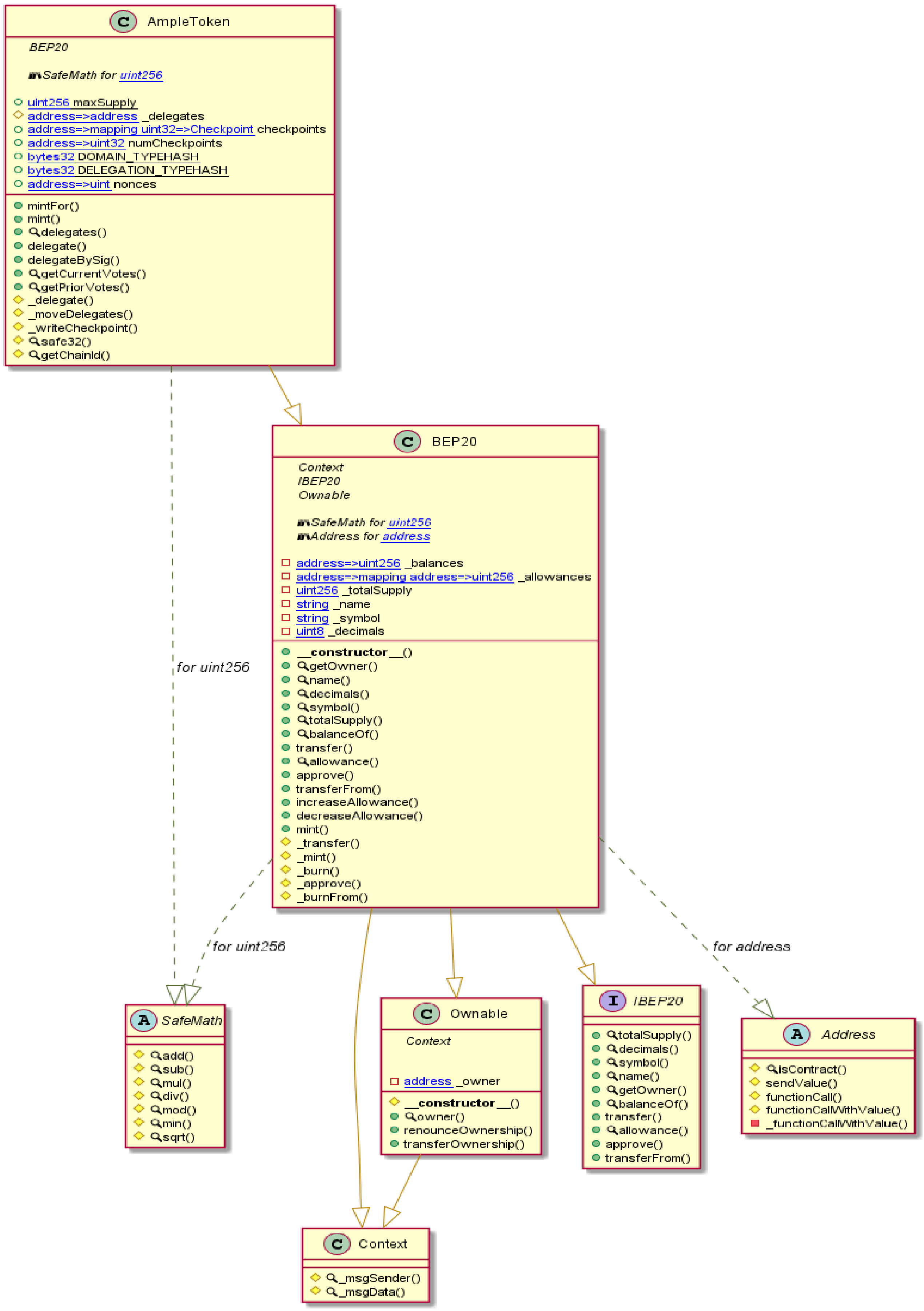
AmpleFactory Token



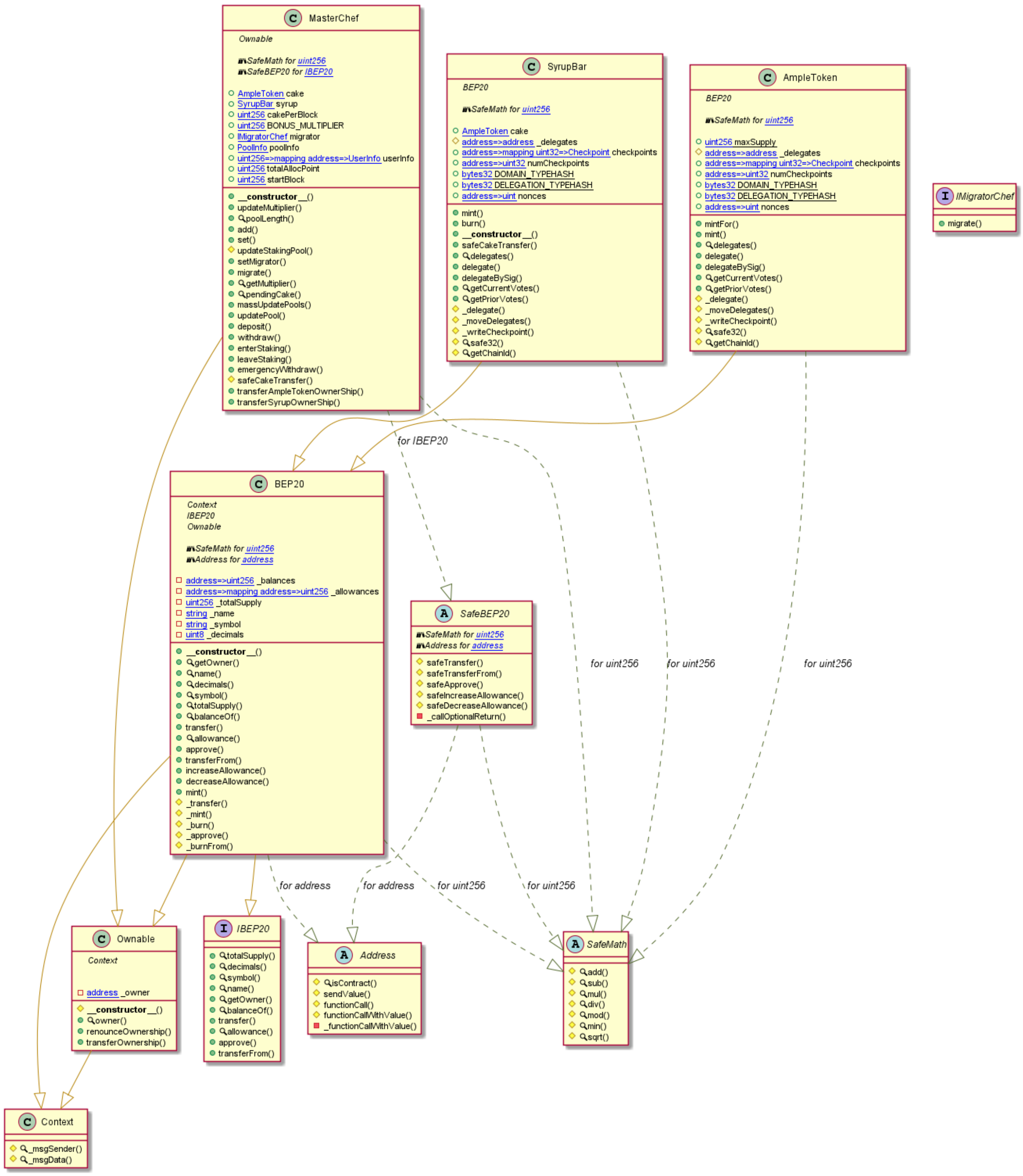
AmpleRouter Token



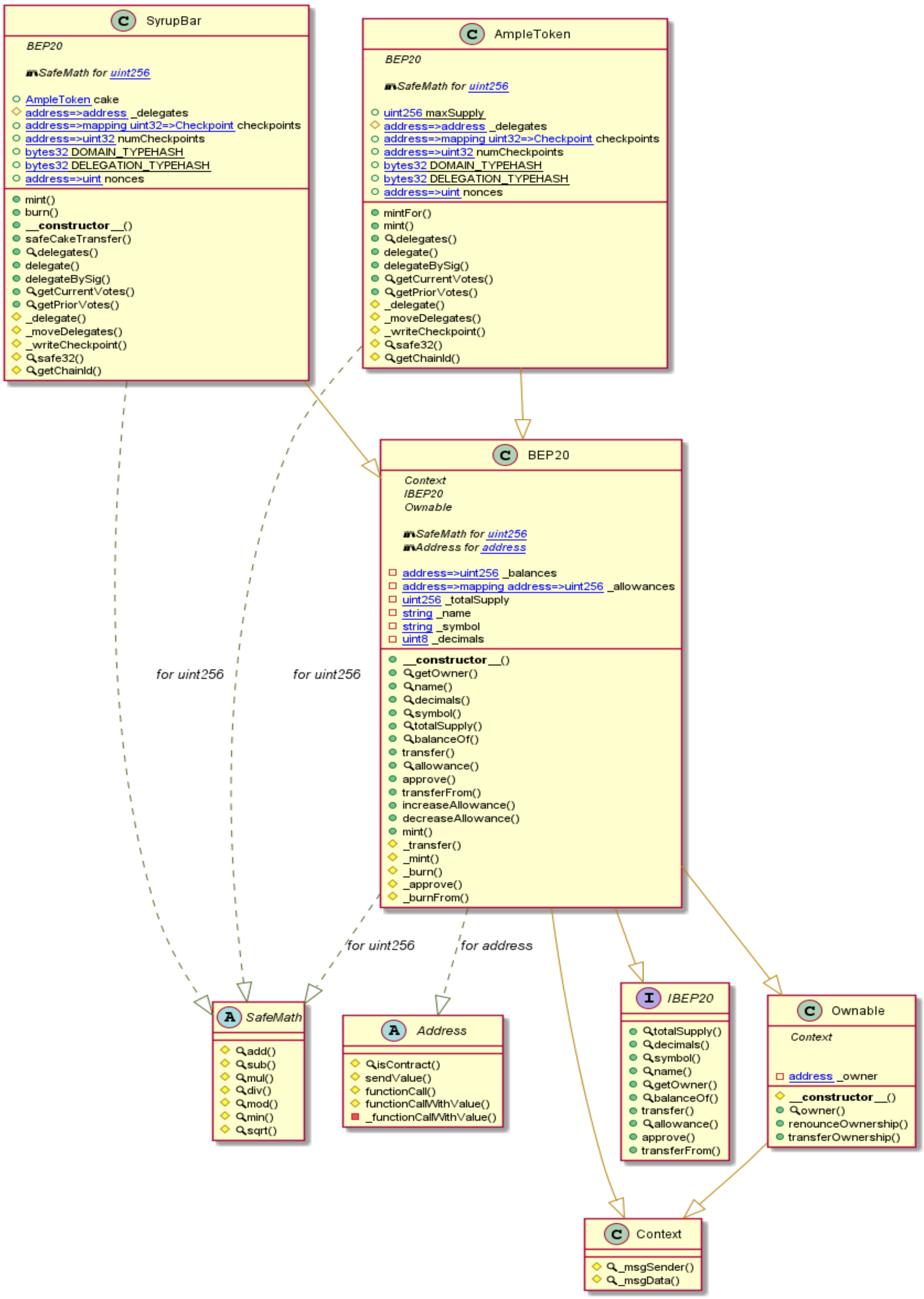
Ample Token


AmpleToken
BEP20
SafeMath for uint256
uint256 maxSupply
address=>address _delegates
address=>mapping uint32=>Checkpoint checkpoints
address=>uint32 numCheckpoints
bytes32 DOMAIN_TYPEHASH
bytes32 DELEGATION_TYPEHASH
address=>uint nonces
mintFor()
mint()
delegates()
delegate()
delegateBySig()
getCurrentVotes()
getPriorVotes()
_delegate()
_moveDelegates()
_writeCheckpoint()
safe32()
getChainId()
for uint256
BEP20
Context
IBEP20
Ownable
SafeMath for uint256
Address for address
address=>uint256 _balances
address=>mapping address=>uint256 _allowances
uint256 _totalSupply
string _name
string _symbol
uint8 _decimals
__constructor__()
getOwner()
name()
decimals()
symbol()
totalSupply()
balanceOf()
transfer()
allowance()
approve()
transferFrom()
increaseAllowance()
decreaseAllowance()
mint()
_transfer()
_mint()
_burn()
_approve()
_burnFrom()
for uint256
for address
SafeMath
add()
sub()
mul()
div()
mod()
min()
sqrt()
Ownable
Context
address _owner
__constructor__()
owner()
renounceOwnership()
transferOwnership()
IBEP20
totalSupply()
decimals()
symbol()
name()
getOwner()
balanceOf()
transfer()
allowance()
approve()
transferFrom()
Address
isContract()
sendValue()
functionCall()
functionCallWithValue()
_functionCallWithValue()
Context
_msgSender()
_msgData()

MasterChef Token

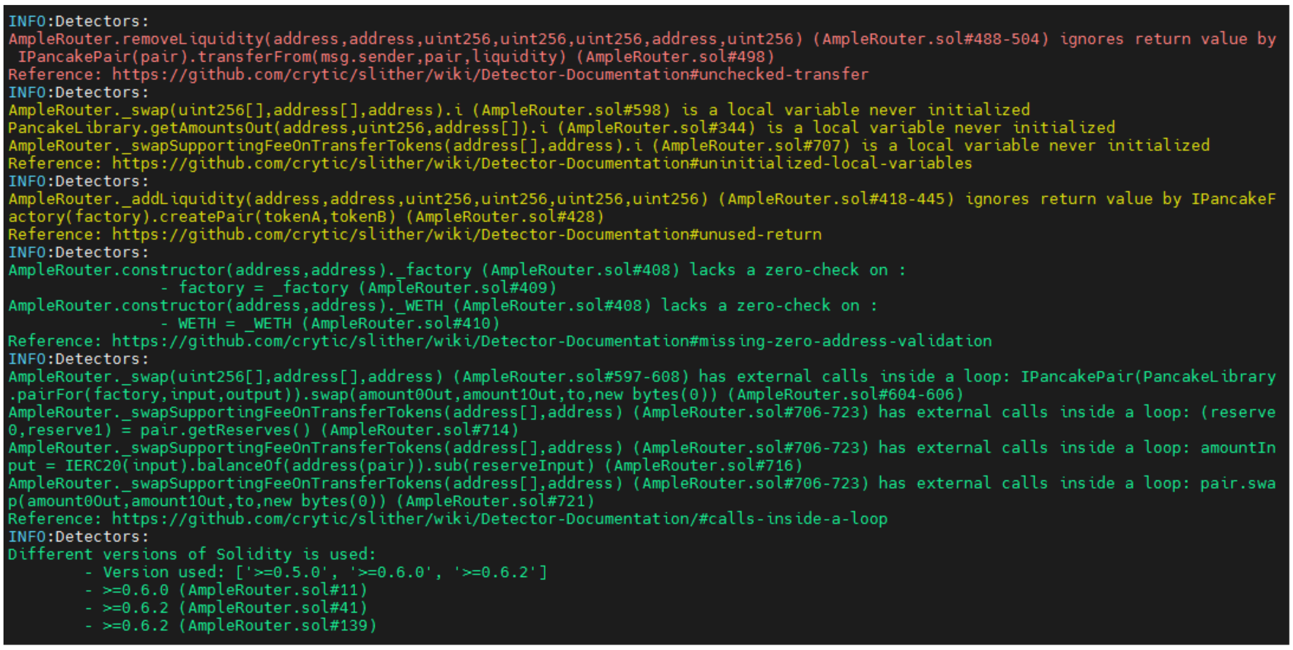


SyrupBarToken

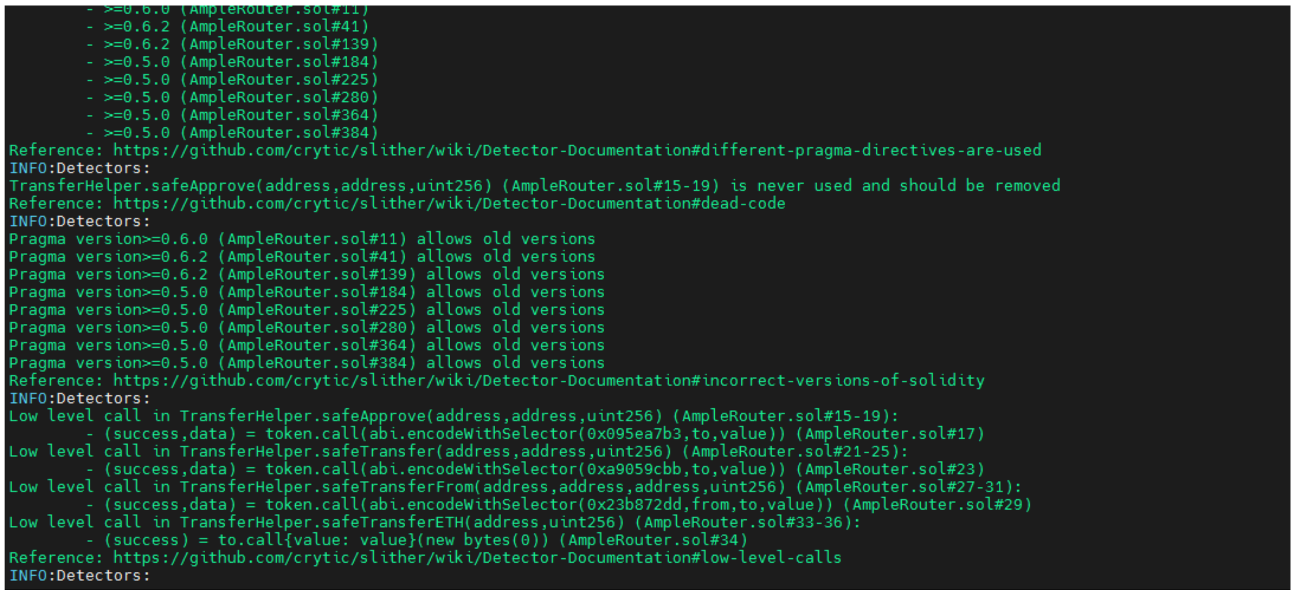


Slither Results Log

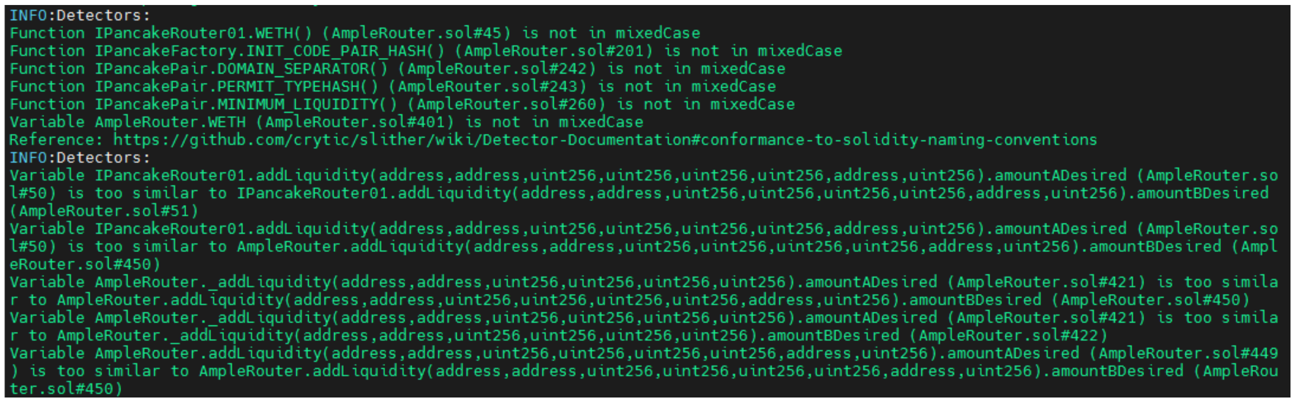
Slither log >> AmpleRouter.sol



```
INFO:Detectors:
AmpleRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) (AmpleRouter.sol#488-504) ignores return value by
 IPancakePair(pair).transferFrom(msg.sender,pair,liquidity) (AmpleRouter.sol#498)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
AmpleRouter._swap(uint256[],address[],address).i (AmpleRouter.sol#598) is a local variable never initialized
PancakeLibrary.getAmountsOut(address,uint256,address[]).i (AmpleRouter.sol#344) is a local variable never initialized
AmpleRouter._swapSupportingFeeOnTransferTokens(address[],address).i (AmpleRouter.sol#707) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
AmpleRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256) (AmpleRouter.sol#418-445) ignores return value by IPancakeF
actory(factory).createPair(tokenA,tokenB) (AmpleRouter.sol#428)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
AmpleRouter.constructor(address,address)._factory (AmpleRouter.sol#408) lacks a zero-check on :
                - factory = _factory (AmpleRouter.sol#409)
AmpleRouter.constructor(address,address)._WETH (AmpleRouter.sol#408) lacks a zero-check on :
                - WETH = _WETH (AmpleRouter.sol#410)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
AmpleRouter._swap(uint256[],address[],address) (AmpleRouter.sol#597-608) has external calls inside a loop: IPancakePair(PancakeLibrary
.pairFor(factory,input,output)).swap(amount0Out,amount1Out,to,new bytes(0)) (AmpleRouter.sol#604-606)
AmpleRouter._swapSupportingFeeOnTransferTokens(address[],address) (AmpleRouter.sol#706-723) has external calls inside a loop: (reserve
0,reserve1) = pair.getReserves() (AmpleRouter.sol#714)
AmpleRouter._swapSupportingFeeOnTransferTokens(address[],address) (AmpleRouter.sol#706-723) has external calls inside a loop: amountIn
put = IERC20(input).balanceOf(address(pair)).sub(reserveInput) (AmpleRouter.sol#716)
AmpleRouter._swapSupportingFeeOnTransferTokens(address[],address) (AmpleRouter.sol#706-723) has external calls inside a loop: pair.swa
p(amount0Out,amount1Out,to,new bytes(0)) (AmpleRouter.sol#721)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['>=0.5.0', '>=0.6.0', '>=0.6.2']
        - >=0.6.0 (AmpleRouter.sol#11)
        - >=0.6.2 (AmpleRouter.sol#41)
        - >=0.6.2 (AmpleRouter.sol#139)
```

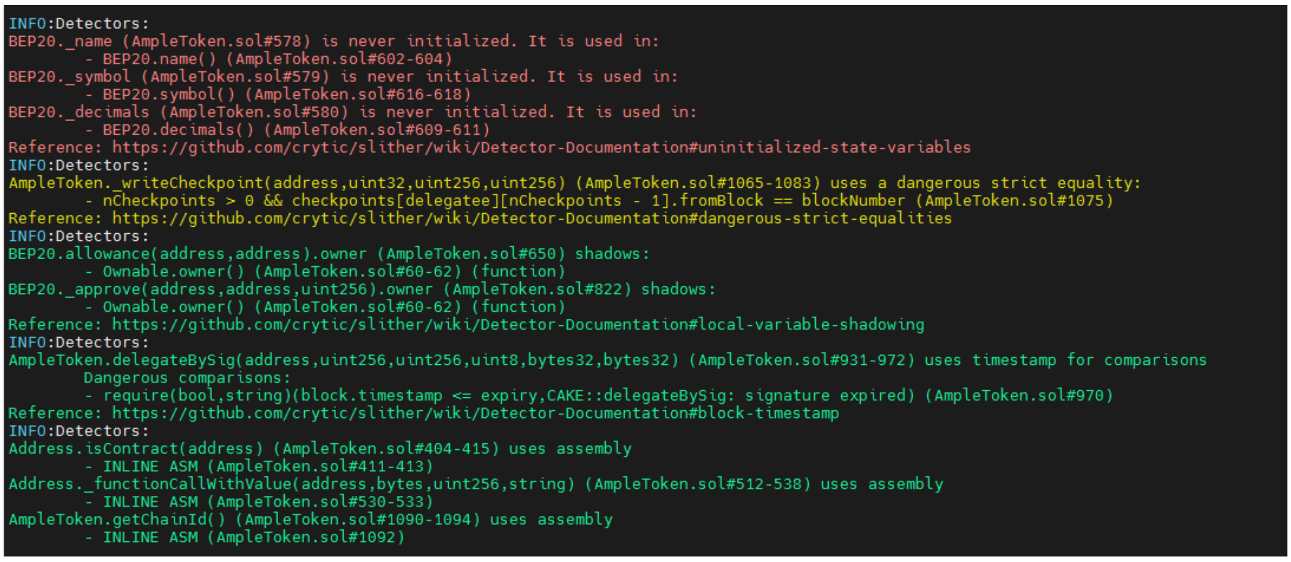


```
        - >=0.6.0 (AmpleRouter.sol#11)
        - >=0.6.2 (AmpleRouter.sol#41)
        - >=0.6.2 (AmpleRouter.sol#139)
        - >=0.5.0 (AmpleRouter.sol#184)
        - >=0.5.0 (AmpleRouter.sol#225)
        - >=0.5.0 (AmpleRouter.sol#280)
        - >=0.5.0 (AmpleRouter.sol#364)
        - >=0.5.0 (AmpleRouter.sol#384)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
TransferHelper.safeApprove(address,address,uint256) (AmpleRouter.sol#15-19) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0 (AmpleRouter.sol#11) allows old versions
Pragma version>=0.6.2 (AmpleRouter.sol#41) allows old versions
Pragma version>=0.6.2 (AmpleRouter.sol#139) allows old versions
Pragma version>=0.5.0 (AmpleRouter.sol#184) allows old versions
Pragma version>=0.5.0 (AmpleRouter.sol#225) allows old versions
Pragma version>=0.5.0 (AmpleRouter.sol#280) allows old versions
Pragma version>=0.5.0 (AmpleRouter.sol#364) allows old versions
Pragma version>=0.5.0 (AmpleRouter.sol#384) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in TransferHelper.safeApprove(address,address,uint256) (AmpleRouter.sol#15-19):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (AmpleRouter.sol#17)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (AmpleRouter.sol#21-25):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (AmpleRouter.sol#23)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (AmpleRouter.sol#27-31):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (AmpleRouter.sol#29)
Low level call in TransferHelper.safeTransferETH(address,uint256) (AmpleRouter.sol#33-36):
        - (success) = to.call{value: value}(new bytes(0)) (AmpleRouter.sol#34)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
```

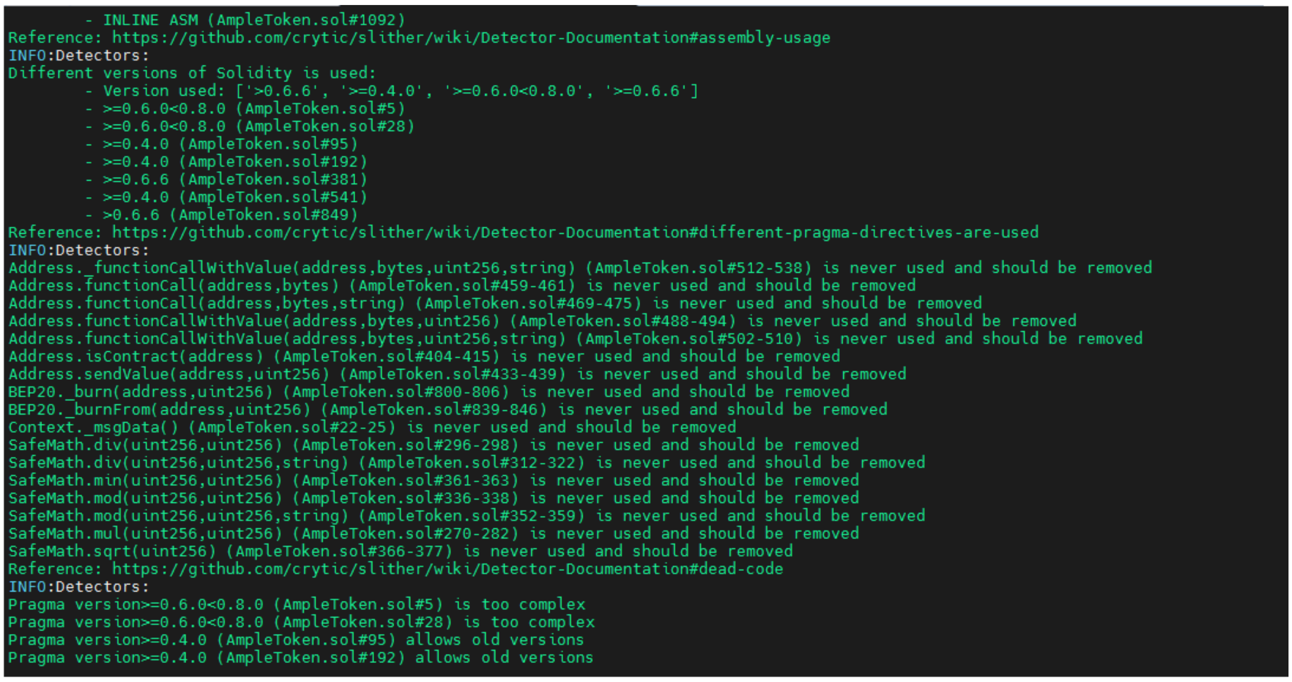


```
INFO:Detectors:
Function IPancakeRouter01.WETH() (AmpleRouter.sol#45) is not in mixedCase
Function IPancakeFactory.INIT_CODE_PAIR_HASH() (AmpleRouter.sol#201) is not in mixedCase
Function IPancakePair.DOMAIN_SEPARATOR() (AmpleRouter.sol#242) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (AmpleRouter.sol#243) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (AmpleRouter.sol#260) is not in mixedCase
Variable AmpleRouter.WETH (AmpleRouter.sol#401) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (AmpleRouter.so
l#50) is too similar to IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired
(AmpleRouter.sol#51)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (AmpleRouter.so
l#50) is too similar to AmpleRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Ampl
eRouter.sol#450)
Variable AmpleRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (AmpleRouter.sol#421) is too simila
r to AmpleRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (AmpleRouter.sol#450)
Variable AmpleRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (AmpleRouter.sol#421) is too simila
r to AmpleRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (AmpleRouter.sol#422)
Variable AmpleRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (AmpleRouter.sol#449
) is too similar to AmpleRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (AmpleRou
ter.sol#450)
```

Slither log >> AmpleToken.sol

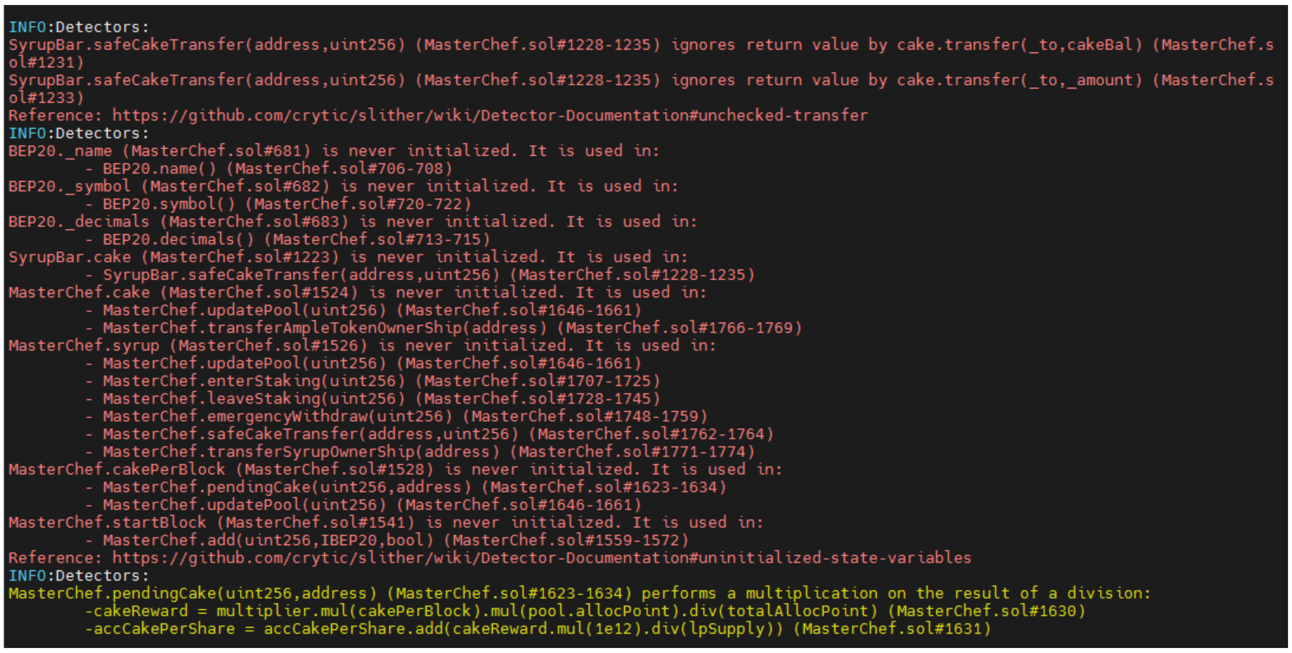


```
INFO:Detectors:
BEP20._name (AmpleToken.sol#578) is never initialized. It is used in:
        - BEP20.name() (AmpleToken.sol#602-604)
BEP20._symbol (AmpleToken.sol#579) is never initialized. It is used in:
        - BEP20.symbol() (AmpleToken.sol#616-618)
BEP20._decimals (AmpleToken.sol#580) is never initialized. It is used in:
        - BEP20.decimals() (AmpleToken.sol#609-611)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
AmpleToken._writeCheckpoint(address,uint32,uint256,uint256) (AmpleToken.sol#1065-1083) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (AmpleToken.sol#1075)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
BEP20.allowance(address,address).owner (AmpleToken.sol#650) shadows:
        - Ownable.owner() (AmpleToken.sol#60-62) (function)
BEP20._approve(address,address,uint256).owner (AmpleToken.sol#822) shadows:
        - Ownable.owner() (AmpleToken.sol#60-62) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
AmpleToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (AmpleToken.sol#931-972) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (AmpleToken.sol#970)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (AmpleToken.sol#404-415) uses assembly
        - INLINE ASM (AmpleToken.sol#411-413)
Address._functionCallWithValue(address,bytes,uint256,string) (AmpleToken.sol#512-538) uses assembly
        - INLINE ASM (AmpleToken.sol#530-533)
AmpleToken.getChainId() (AmpleToken.sol#1090-1094) uses assembly
        - INLINE ASM (AmpleToken.sol#1092)
```

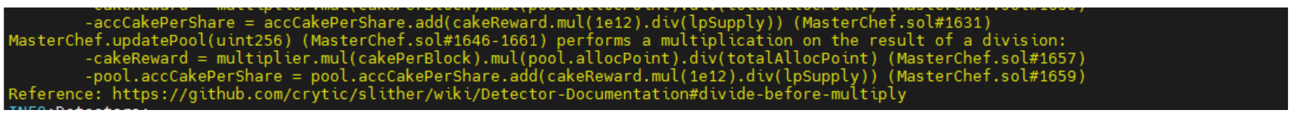


```
        - INLINE ASM (AmpleToken.sol#1092)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['>0.6.6', '>=0.4.0', '>=0.6.0<0.8.0', '>=0.6.6']
        - >=0.6.0<0.8.0 (AmpleToken.sol#5)
        - >=0.6.0<0.8.0 (AmpleToken.sol#28)
        - >=0.4.0 (AmpleToken.sol#95)
        - >=0.4.0 (AmpleToken.sol#192)
        - >=0.6.6 (AmpleToken.sol#381)
        - >=0.4.0 (AmpleToken.sol#541)
        - >0.6.6 (AmpleToken.sol#849)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (AmpleToken.sol#512-538) is never used and should be removed
Address.functionCall(address,bytes) (AmpleToken.sol#459-461) is never used and should be removed
Address.functionCall(address,bytes,string) (AmpleToken.sol#469-475) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (AmpleToken.sol#488-494) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (AmpleToken.sol#502-510) is never used and should be removed
Address.isContract(address) (AmpleToken.sol#404-415) is never used and should be removed
Address.sendValue(address,uint256) (AmpleToken.sol#433-439) is never used and should be removed
BEP20._burn(address,uint256) (AmpleToken.sol#800-806) is never used and should be removed
BEP20._burnFrom(address,uint256) (AmpleToken.sol#839-846) is never used and should be removed
Context._msgData() (AmpleToken.sol#22-25) is never used and should be removed
SafeMath.div(uint256,uint256) (AmpleToken.sol#296-298) is never used and should be removed
SafeMath.div(uint256,uint256,string) (AmpleToken.sol#312-322) is never used and should be removed
SafeMath.min(uint256,uint256) (AmpleToken.sol#361-363) is never used and should be removed
SafeMath.mod(uint256,uint256) (AmpleToken.sol#336-338) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (AmpleToken.sol#352-359) is never used and should be removed
SafeMath.mul(uint256,uint256) (AmpleToken.sol#270-282) is never used and should be removed
SafeMath.sqrt(uint256) (AmpleToken.sol#366-377) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (AmpleToken.sol#5) is too complex
Pragma version>=0.6.0<0.8.0 (AmpleToken.sol#28) is too complex
Pragma version>=0.4.0 (AmpleToken.sol#95) allows old versions
Pragma version>=0.4.0 (AmpleToken.sol#192) allows old versions
```

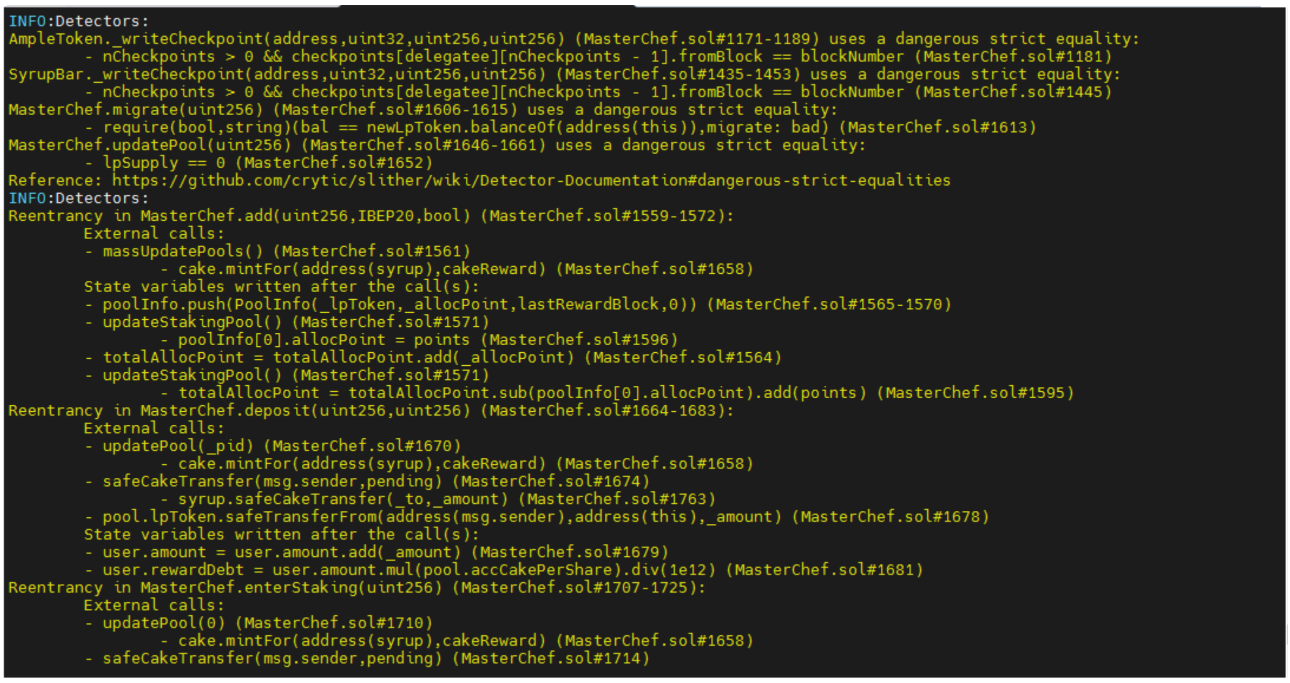
Slither log >> MasterChef.sol



```
INFO:Detectors:
SyrupBar.safeCakeTransfer(address,uint256) (MasterChef.sol#1228-1235) ignores return value by cake.transfer(_to,cakeBal) (MasterChef.s
ol#1231)
SyrupBar.safeCakeTransfer(address,uint256) (MasterChef.sol#1228-1235) ignores return value by cake.transfer(_to,_amount) (MasterChef.s
ol#1233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
BEP20._name (MasterChef.sol#681) is never initialized. It is used in:
        - BEP20.name() (MasterChef.sol#706-708)
BEP20._symbol (MasterChef.sol#682) is never initialized. It is used in:
        - BEP20.symbol() (MasterChef.sol#720-722)
BEP20._decimals (MasterChef.sol#683) is never initialized. It is used in:
        - BEP20.decimals() (MasterChef.sol#713-715)
SyrupBar.cake (MasterChef.sol#1223) is never initialized. It is used in:
        - SyrupBar.safeCakeTransfer(address,uint256) (MasterChef.sol#1228-1235)
MasterChef.cake (MasterChef.sol#1524) is never initialized. It is used in:
        - MasterChef.updatePool(uint256) (MasterChef.sol#1646-1661)
        - MasterChef.transferAmpleTokenOwnerShip(address) (MasterChef.sol#1766-1769)
MasterChef.syrup (MasterChef.sol#1526) is never initialized. It is used in:
        - MasterChef.updatePool(uint256) (MasterChef.sol#1646-1661)
        - MasterChef.enterStaking(uint256) (MasterChef.sol#1707-1725)
        - MasterChef.leaveStaking(uint256) (MasterChef.sol#1728-1745)
        - MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1748-1759)
        - MasterChef.safeCakeTransfer(address,uint256) (MasterChef.sol#1762-1764)
        - MasterChef.transferSyrupOwnerShip(address) (MasterChef.sol#1771-1774)
MasterChef.cakePerBlock (MasterChef.sol#1528) is never initialized. It is used in:
        - MasterChef.pendingCake(uint256,address) (MasterChef.sol#1623-1634)
        - MasterChef.updatePool(uint256) (MasterChef.sol#1646-1661)
MasterChef.startBlock (MasterChef.sol#1541) is never initialized. It is used in:
        - MasterChef.add(uint256,IBEP20,bool) (MasterChef.sol#1559-1572)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
MasterChef.pendingCake(uint256,address) (MasterChef.sol#1623-1634) performs a multiplication on the result of a division:
        -cakeReward = multiplier.mul(cakePerBlock).mul(pool.allocPoint).div(totalAllocPoint) (MasterChef.sol#1630)
        -accCakePerShare = accCakePerShare.add(cakeReward.mul(1e12).div(lpSupply)) (MasterChef.sol#1631)
```

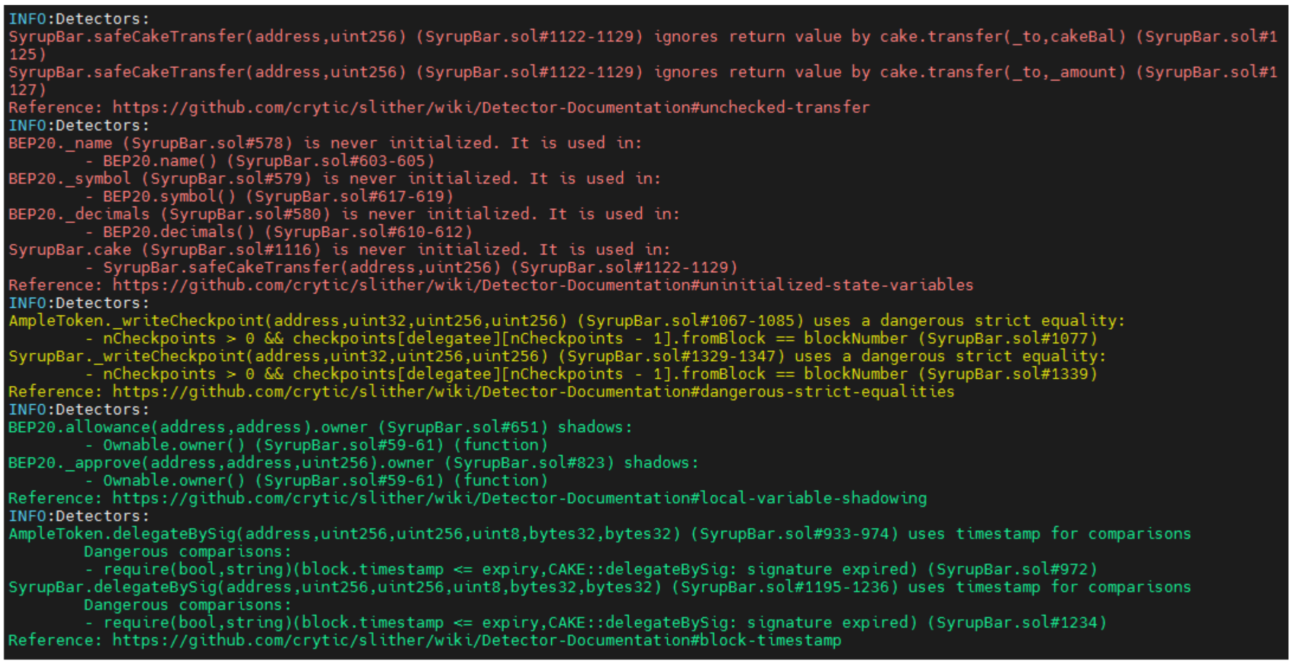


```
        -accCakePerShare = accCakePerShare.add(cakeReward.mul(1e12).div(lpSupply)) (MasterChef.sol#1631)
MasterChef.updatePool(uint256) (MasterChef.sol#1646-1661) performs a multiplication on the result of a division:
        -cakeReward = multiplier.mul(cakePerBlock).mul(pool.allocPoint).div(totalAllocPoint) (MasterChef.sol#1657)
        -pool.accCakePerShare = pool.accCakePerShare.add(cakeReward.mul(1e12).div(lpSupply)) (MasterChef.sol#1659)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

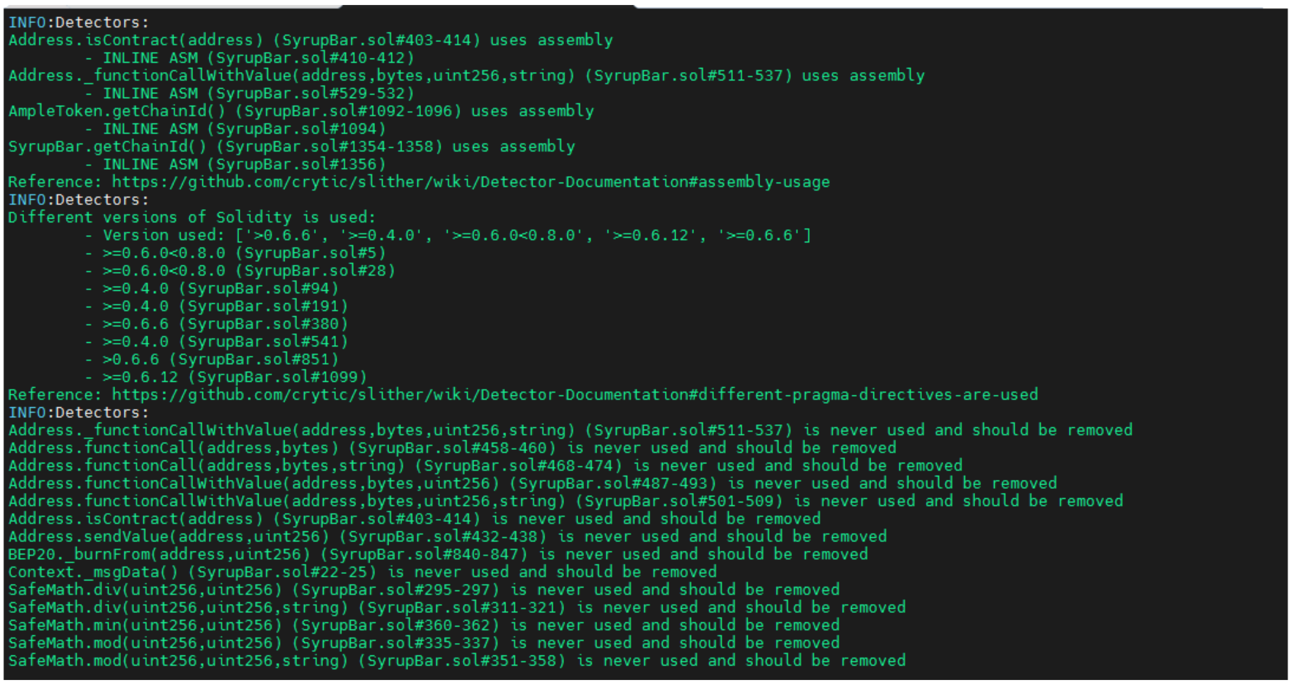


```
INFO:Detectors:
AmpleToken._writeCheckpoint(address,uint32,uint256,uint256) (MasterChef.sol#1171-1189) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (MasterChef.sol#1181)
SyrupBar._writeCheckpoint(address,uint32,uint256,uint256) (MasterChef.sol#1435-1453) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (MasterChef.sol#1445)
MasterChef.migrate(uint256) (MasterChef.sol#1606-1615) uses a dangerous strict equality:
        - require(bool,string)(bal == newLpToken.balanceOf(address(this)),migrate: bad) (MasterChef.sol#1613)
MasterChef.updatePool(uint256) (MasterChef.sol#1646-1661) uses a dangerous strict equality:
        - lpSupply == 0 (MasterChef.sol#1652)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in MasterChef.add(uint256,IBEP20,bool) (MasterChef.sol#1559-1572):
        External calls:
        - massUpdatePools() (MasterChef.sol#1561)
                - cake.mintFor(address(syrup),cakeReward) (MasterChef.sol#1658)
        State variables written after the call(s):
        - poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0)) (MasterChef.sol#1565-1570)
        - updateStakingPool() (MasterChef.sol#1571)
                - poolInfo[0].allocPoint = points (MasterChef.sol#1596)
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (MasterChef.sol#1564)
        - updateStakingPool() (MasterChef.sol#1571)
                - totalAllocPoint = totalAllocPoint.sub(poolInfo[0].allocPoint).add(points) (MasterChef.sol#1595)
Reentrancy in MasterChef.deposit(uint256,uint256) (MasterChef.sol#1664-1683):
        External calls:
        - updatePool(_pid) (MasterChef.sol#1670)
                - cake.mintFor(address(syrup),cakeReward) (MasterChef.sol#1658)
        - safeCakeTransfer(msg.sender,pending) (MasterChef.sol#1674)
                - syrup.safeCakeTransfer(_to,_amount) (MasterChef.sol#1763)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterChef.sol#1678)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (MasterChef.sol#1679)
        - user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12) (MasterChef.sol#1681)
Reentrancy in MasterChef.enterStaking(uint256) (MasterChef.sol#1707-1725):
        External calls:
        - updatePool(0) (MasterChef.sol#1710)
                - cake.mintFor(address(syrup),cakeReward) (MasterChef.sol#1658)
        - safeCakeTransfer(msg.sender,pending) (MasterChef.sol#1714)
```

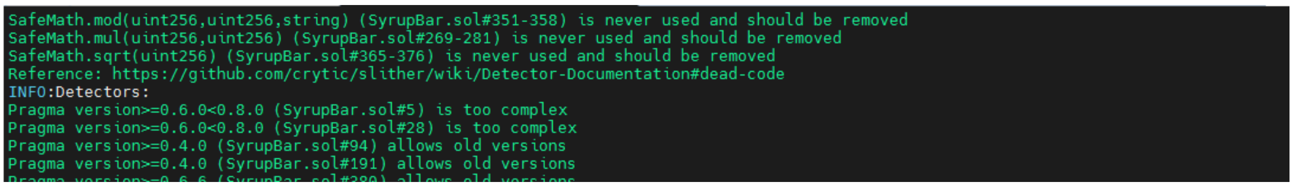
Slither log >> SyrupBar.sol



```
INFO:Detectors:
SyrupBar.safeCakeTransfer(address,uint256) (SyrupBar.sol#1122-1129) ignores return value by cake.transfer(_to,cakeBal) (SyrupBar.sol#1
125)
SyrupBar.safeCakeTransfer(address,uint256) (SyrupBar.sol#1122-1129) ignores return value by cake.transfer(_to,_amount) (SyrupBar.sol#1
127)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
BEP20._name (SyrupBar.sol#578) is never initialized. It is used in:
        - BEP20.name() (SyrupBar.sol#603-605)
BEP20._symbol (SyrupBar.sol#579) is never initialized. It is used in:
        - BEP20.symbol() (SyrupBar.sol#617-619)
BEP20._decimals (SyrupBar.sol#580) is never initialized. It is used in:
        - BEP20.decimals() (SyrupBar.sol#610-612)
SyrupBar.cake (SyrupBar.sol#1116) is never initialized. It is used in:
        - SyrupBar.safeCakeTransfer(address,uint256) (SyrupBar.sol#1122-1129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
AmpleToken._writeCheckpoint(address,uint32,uint256,uint256) (SyrupBar.sol#1067-1085) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (SyrupBar.sol#1077)
SyrupBar._writeCheckpoint(address,uint32,uint256,uint256) (SyrupBar.sol#1329-1347) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (SyrupBar.sol#1339)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
BEP20.allowance(address,address).owner (SyrupBar.sol#651) shadows:
        - Ownable.owner() (SyrupBar.sol#59-61) (function)
BEP20._approve(address,address,uint256).owner (SyrupBar.sol#823) shadows:
        - Ownable.owner() (SyrupBar.sol#59-61) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
AmpleToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#933-974) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (SyrupBar.sol#972)
SyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#1195-1236) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (SyrupBar.sol#1234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

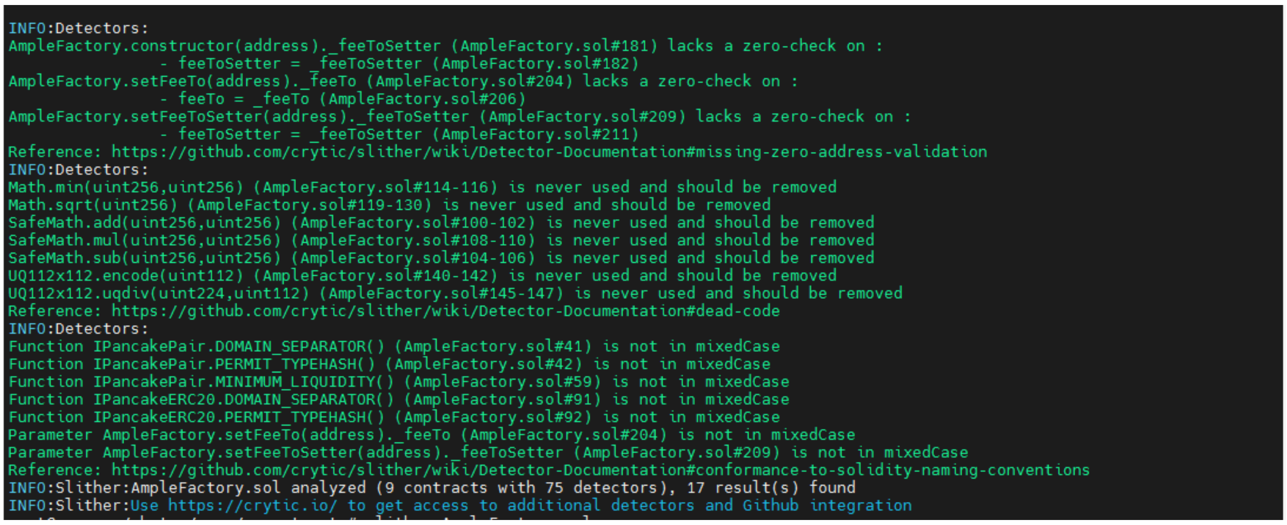


```
INFO:Detectors:
Address.isContract(address) (SyrupBar.sol#403-414) uses assembly
        - INLINE ASM (SyrupBar.sol#410-412)
Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#511-537) uses assembly
        - INLINE ASM (SyrupBar.sol#529-532)
AmpleToken.getChainId() (SyrupBar.sol#1092-1096) uses assembly
        - INLINE ASM (SyrupBar.sol#1094)
SyrupBar.getChainId() (SyrupBar.sol#1354-1358) uses assembly
        - INLINE ASM (SyrupBar.sol#1356)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['>0.6.6', '>=0.4.0', '>=0.6.0<0.8.0', '>=0.6.12', '>=0.6.6']
        - >=0.6.0<0.8.0 (SyrupBar.sol#5)
        - >=0.6.0<0.8.0 (SyrupBar.sol#28)
        - >=0.4.0 (SyrupBar.sol#94)
        - >=0.4.0 (SyrupBar.sol#191)
        - >=0.6.6 (SyrupBar.sol#380)
        - >=0.4.0 (SyrupBar.sol#541)
        - >0.6.6 (SyrupBar.sol#851)
        - >=0.6.12 (SyrupBar.sol#1099)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#511-537) is never used and should be removed
Address.functionCall(address,bytes) (SyrupBar.sol#458-460) is never used and should be removed
Address.functionCall(address,bytes,string) (SyrupBar.sol#468-474) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (SyrupBar.sol#487-493) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#501-509) is never used and should be removed
Address.isContract(address) (SyrupBar.sol#403-414) is never used and should be removed
Address.sendValue(address,uint256) (SyrupBar.sol#432-438) is never used and should be removed
BEP20._burnFrom(address,uint256) (SyrupBar.sol#840-847) is never used and should be removed
Context._msgData() (SyrupBar.sol#22-25) is never used and should be removed
SafeMath.div(uint256,uint256) (SyrupBar.sol#295-297) is never used and should be removed
SafeMath.div(uint256,uint256,string) (SyrupBar.sol#311-321) is never used and should be removed
SafeMath.min(uint256,uint256) (SyrupBar.sol#360-362) is never used and should be removed
SafeMath.mod(uint256,uint256) (SyrupBar.sol#335-337) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (SyrupBar.sol#351-358) is never used and should be removed
```



```
SafeMath.mod(uint256,uint256,string) (SyrupBar.sol#351-358) is never used and should be removed
SafeMath.mul(uint256,uint256) (SyrupBar.sol#269-281) is never used and should be removed
SafeMath.sqrt(uint256) (SyrupBar.sol#365-376) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (SyrupBar.sol#5) is too complex
Pragma version>=0.6.0<0.8.0 (SyrupBar.sol#28) is too complex
Pragma version>=0.4.0 (SyrupBar.sol#94) allows old versions
Pragma version>=0.4.0 (SyrupBar.sol#191) allows old versions
Pragma version>=0.6.6 (SyrupBar.sol#380) allows old versions
```

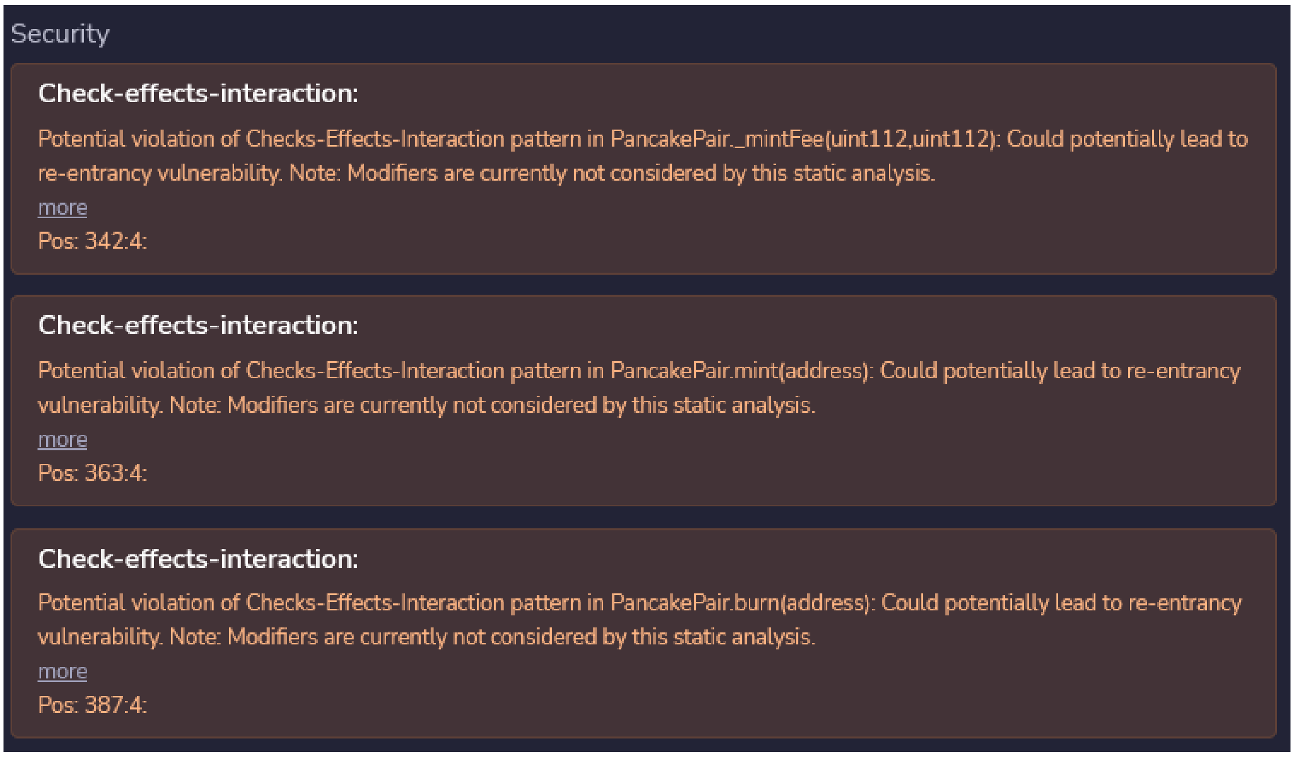
Slither log >> AmpleFactory.sol



```
INFO:Detectors:
AmpleFactory.constructor(address)._feeToSetter (AmpleFactory.sol#181) lacks a zero-check on :
                - feeToSetter = _feeToSetter (AmpleFactory.sol#182)
AmpleFactory.setFeeTo(address)._feeTo (AmpleFactory.sol#204) lacks a zero-check on :
                - feeTo = _feeTo (AmpleFactory.sol#206)
AmpleFactory.setFeeToSetter(address)._feeToSetter (AmpleFactory.sol#209) lacks a zero-check on :
                - feeToSetter = _feeToSetter (AmpleFactory.sol#211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Math.min(uint256,uint256) (AmpleFactory.sol#114-116) is never used and should be removed
Math.sqrt(uint256) (AmpleFactory.sol#119-130) is never used and should be removed
SafeMath.add(uint256,uint256) (AmpleFactory.sol#100-102) is never used and should be removed
SafeMath.mul(uint256,uint256) (AmpleFactory.sol#108-110) is never used and should be removed
SafeMath.sub(uint256,uint256) (AmpleFactory.sol#104-106) is never used and should be removed
UQ112x112.encode(uint112) (AmpleFactory.sol#140-142) is never used and should be removed
UQ112x112.uqdiv(uint224,uint112) (AmpleFactory.sol#145-147) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Function IPancakePair.DOMAIN_SEPARATOR() (AmpleFactory.sol#41) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (AmpleFactory.sol#42) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (AmpleFactory.sol#59) is not in mixedCase
Function IPancakeERC20.DOMAIN_SEPARATOR() (AmpleFactory.sol#91) is not in mixedCase
Function IPancakeERC20.PERMIT_TYPEHASH() (AmpleFactory.sol#92) is not in mixedCase
Parameter AmpleFactory.setFeeTo(address)._feeTo (AmpleFactory.sol#204) is not in mixedCase
Parameter AmpleFactory.setFeeToSetter(address)._feeToSetter (AmpleFactory.sol#209) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:AmpleFactory.sol analyzed (9 contracts with 75 detectors), 17 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

AmpleFactory.sol



Security

Check-effects-interaction:

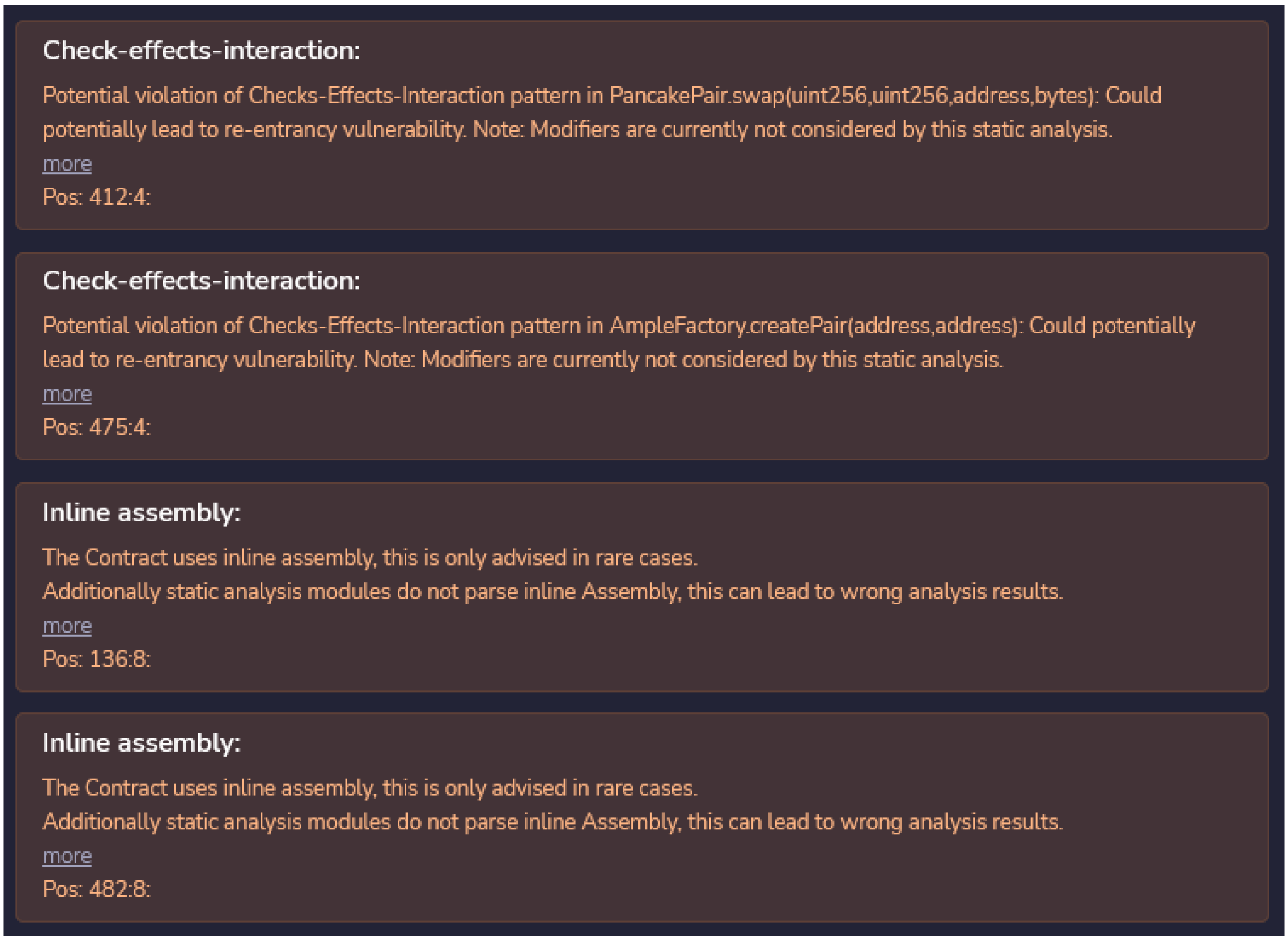
Potential violation of Checks-Effects-Interaction pattern in PancakePair._mintFee(uint112,uint112): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 342:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in PancakePair.mint(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 363:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in PancakePair.burn(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 387:4:



Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in PancakePair.swap(uint256,uint256,address,bytes): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 412:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in AmpleFactory.createPair(address,address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 475:4:

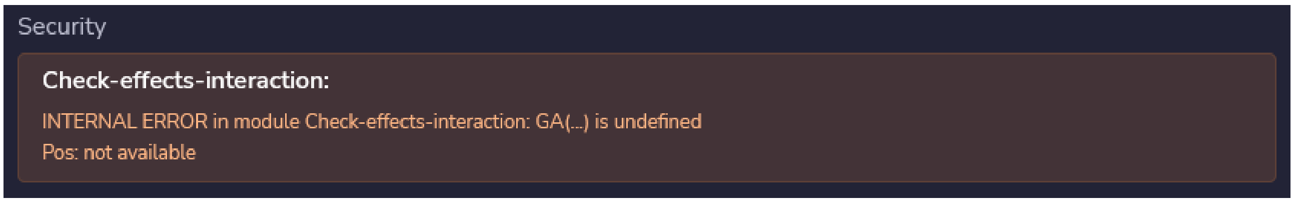
Inline assembly:

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: 136:8:

Inline assembly:

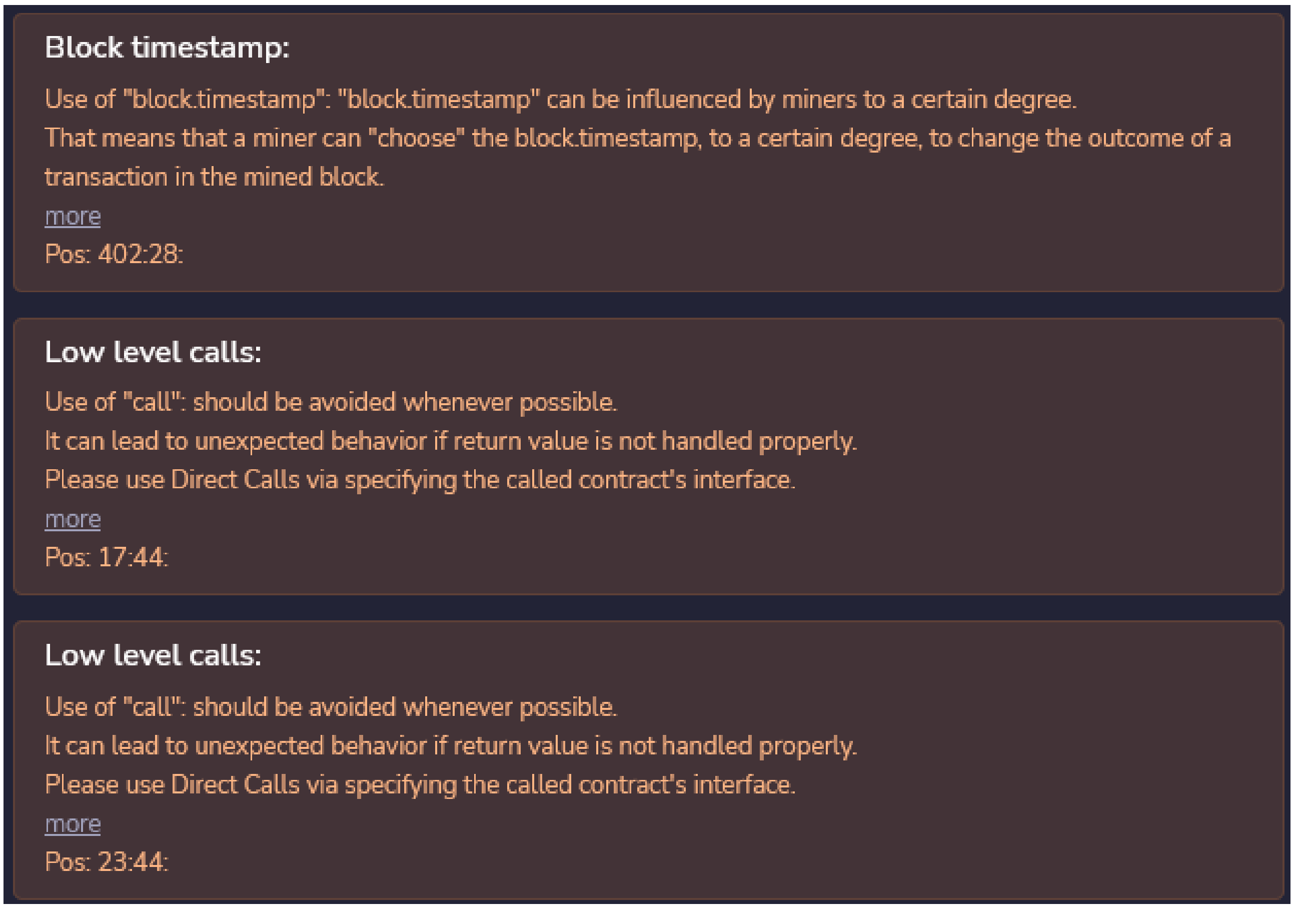
The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 482:8:

AmpleRouter.sol


Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: GA(...) is undefined
Pos: not available


Block timestamp:

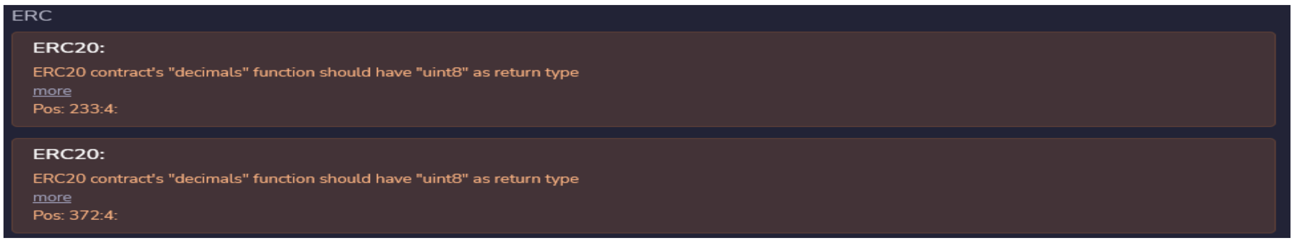
Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 402:28:

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: 17:44:

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: 23:44:

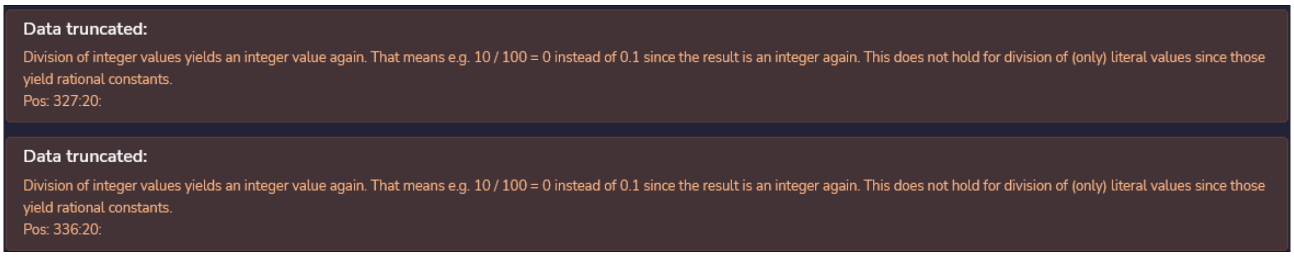

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 233:4:

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 372:4:

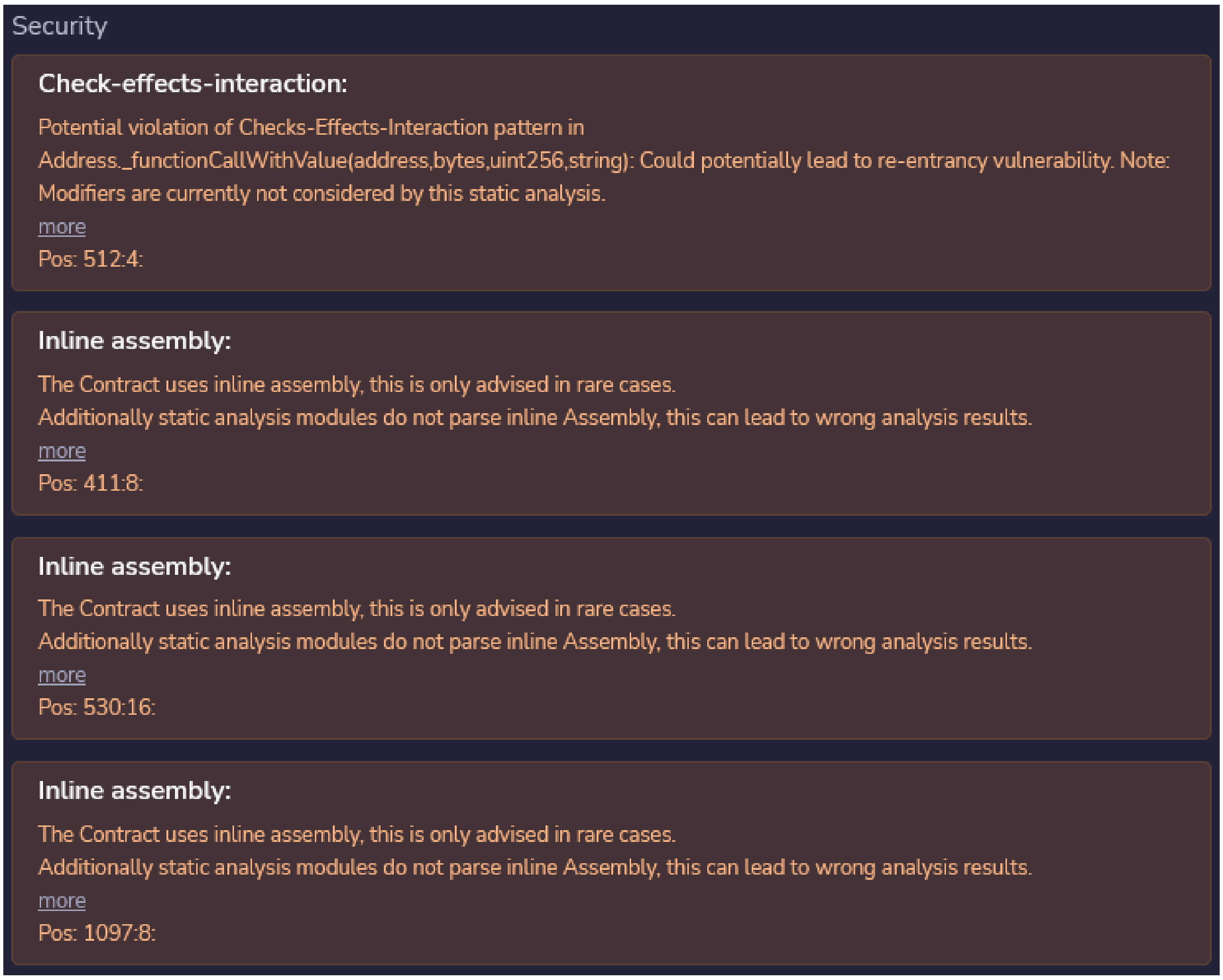

Data truncated:

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: 327:20:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 336:20:

AmpleToken.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: 512:4:

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: 411:8:

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

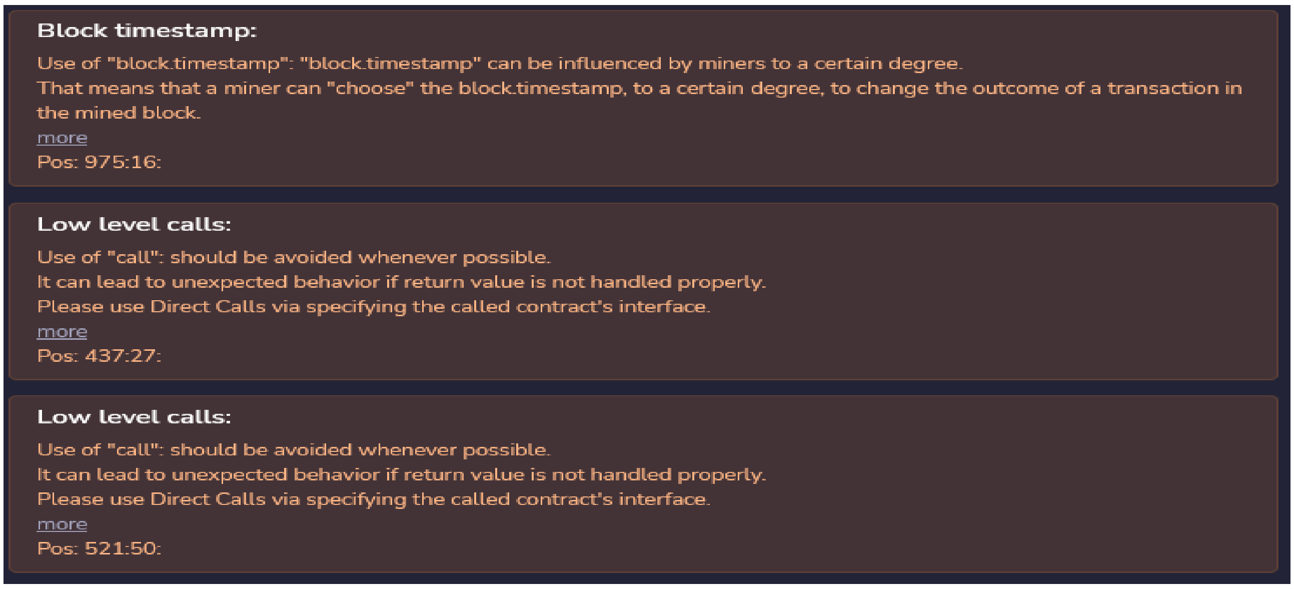
Pos: 530:16:

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: 1097:8:



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: 975:16:

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: 437:27:

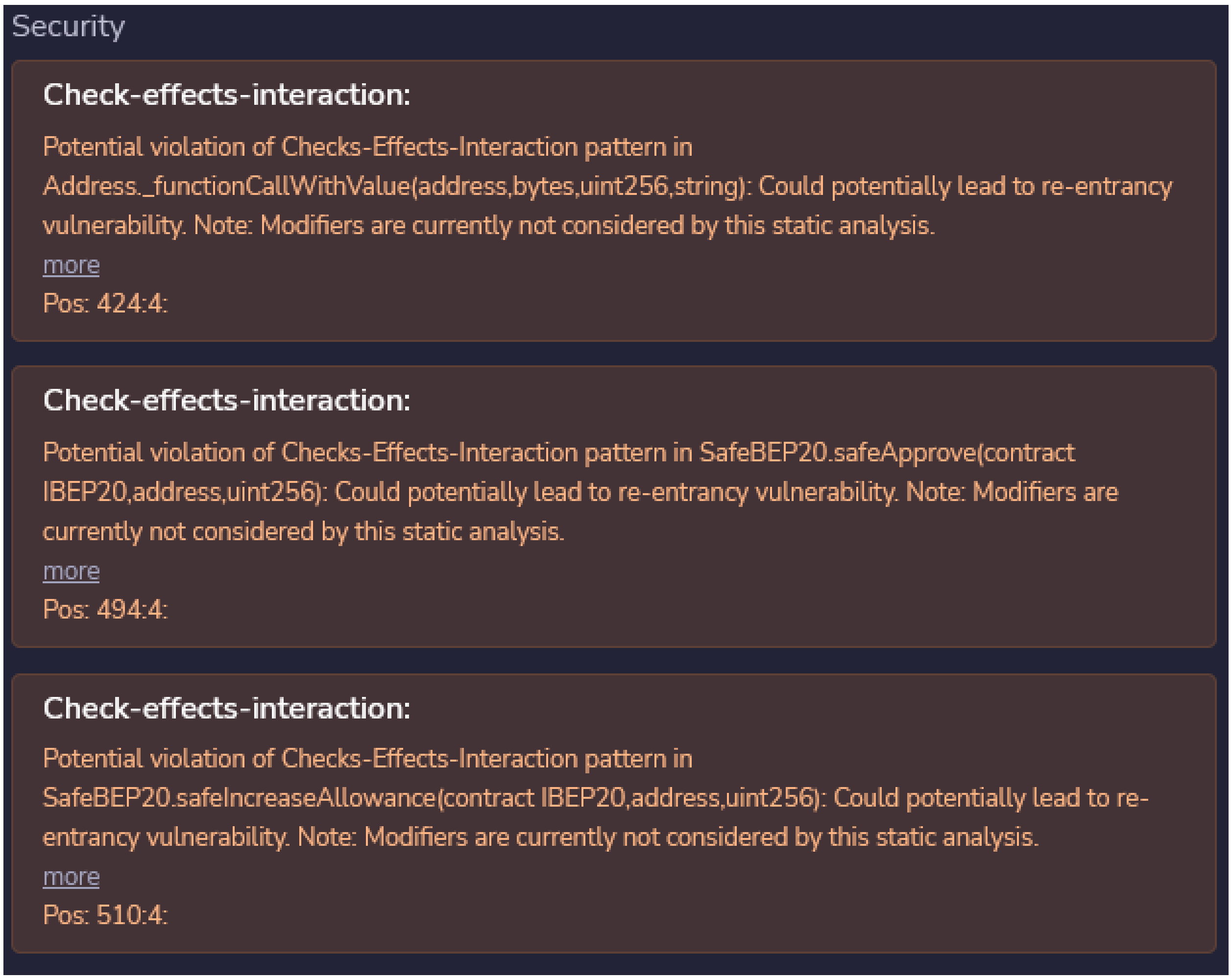
Low level calls:

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: 521:50:

MasterChef.sol


Security

Check-effects-interaction:

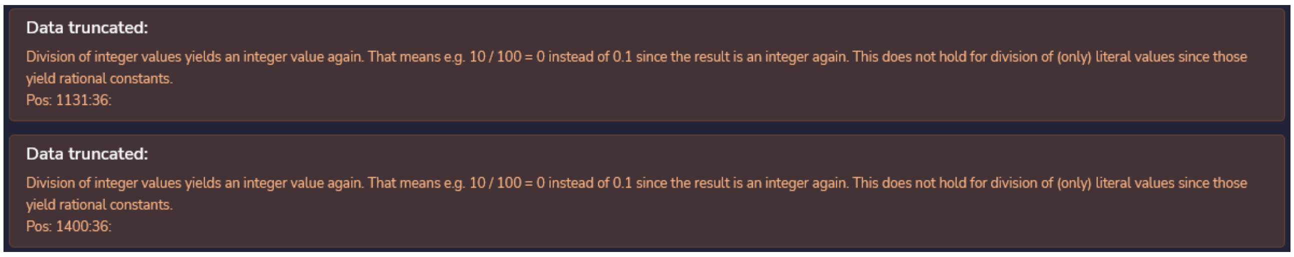
Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 424:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeBEP20.safeApprove(contract IBEP20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 494:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeBEP20.safeIncreaseAllowance(contract IBEP20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 510:4:

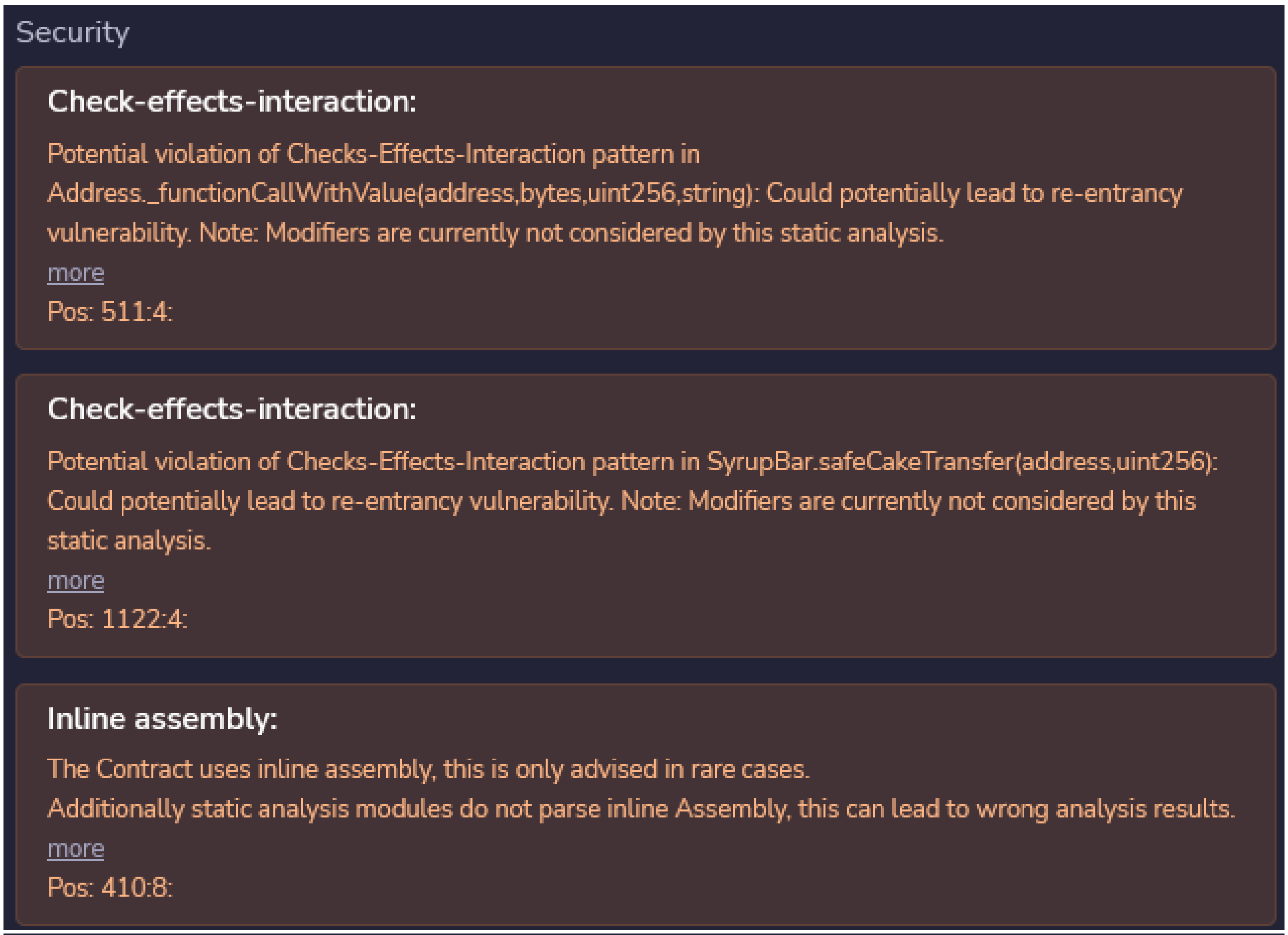

Data truncated:

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: 1131:36:

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: 1400:36:

SyrupBar.sol



Security

Check-effects-interaction:

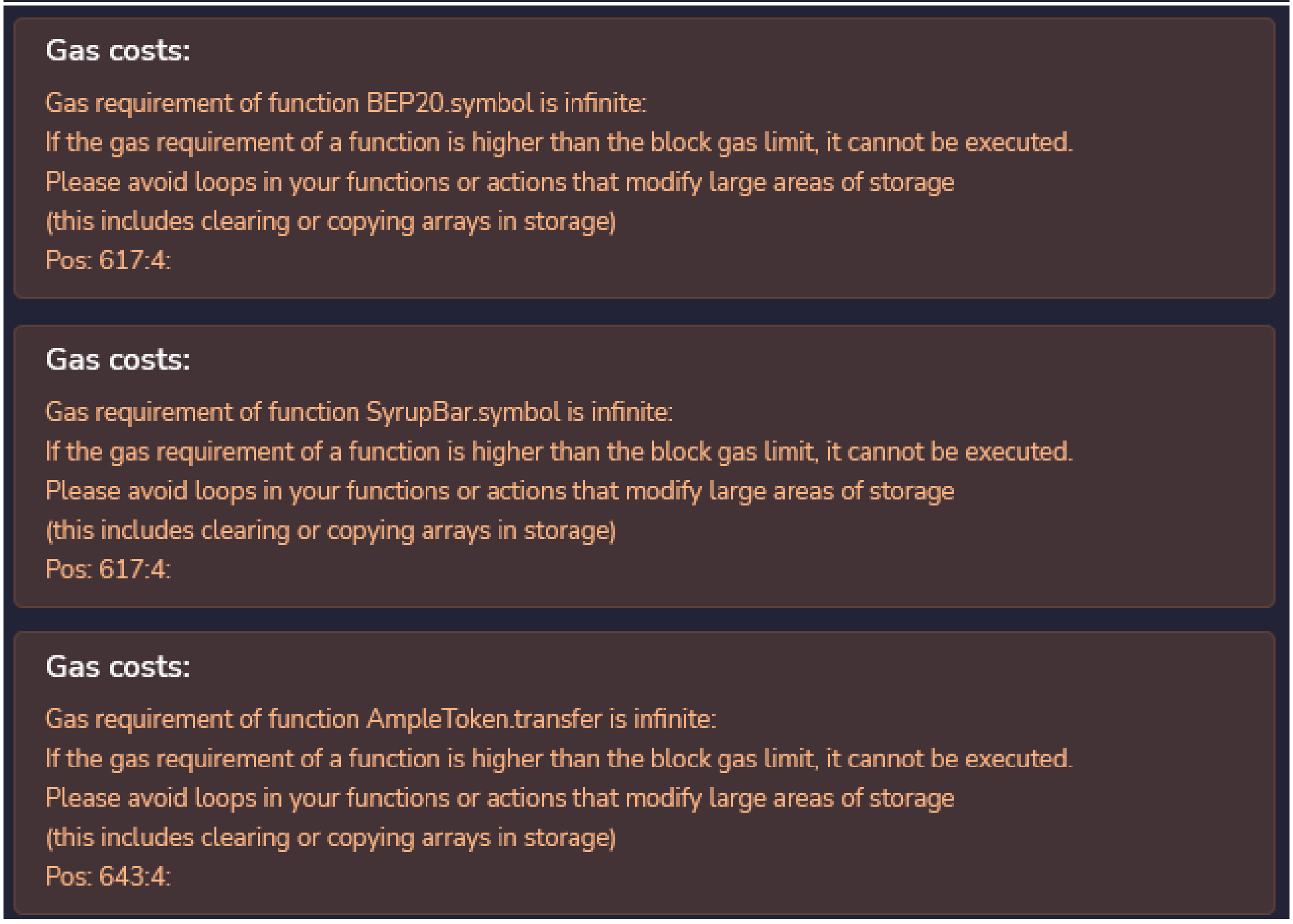
Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 511:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SyrupBar.safeCakeTransfer(address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1122:4:

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: 410:8:



Gas costs:

Gas requirement of function BEP20.symbol 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: 617:4:

Gas costs:

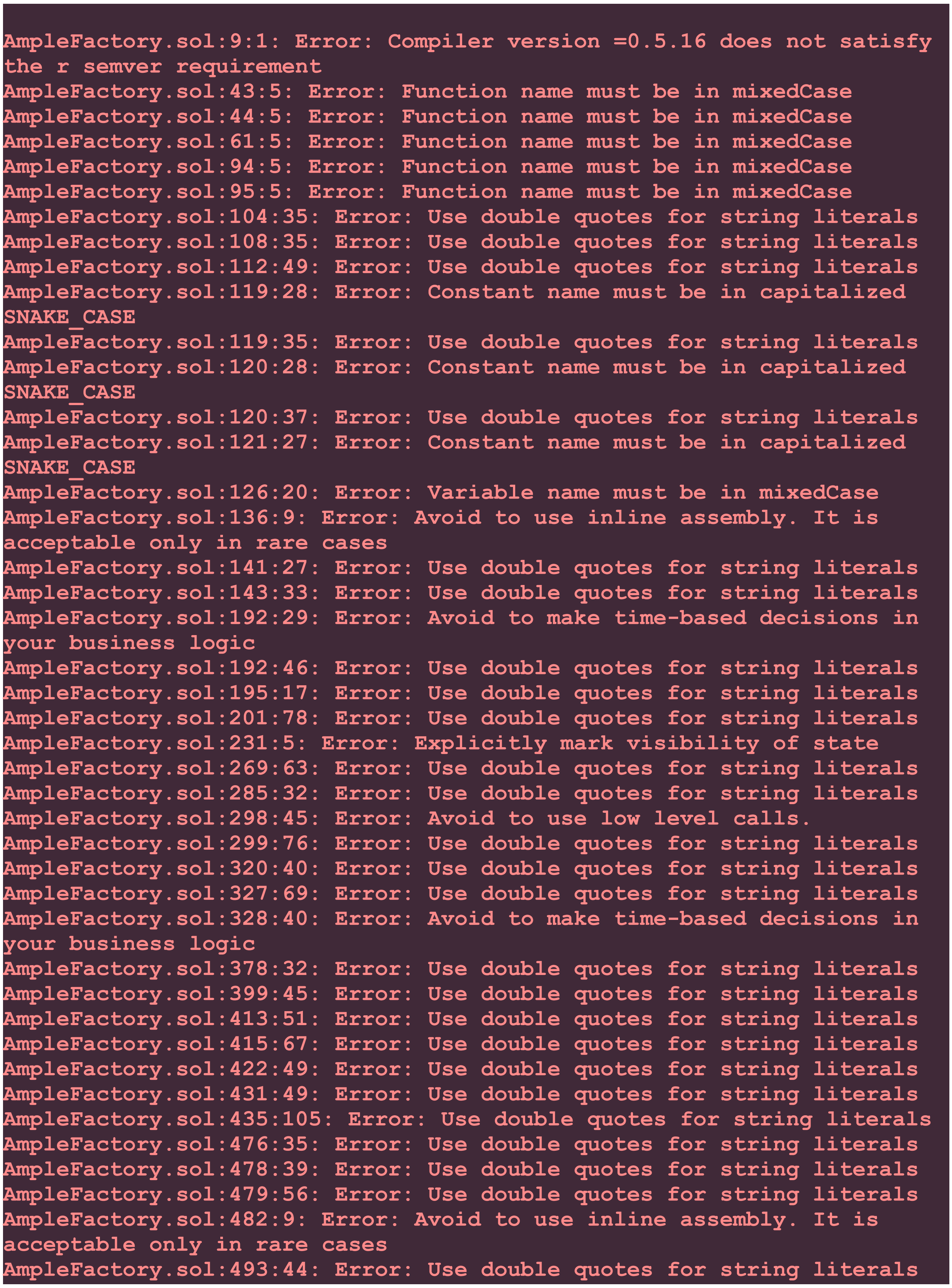
Gas requirement of function SyrupBar.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 617:4:

Gas costs:

Gas requirement of function AmpleToken.transfer 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: 643:4:

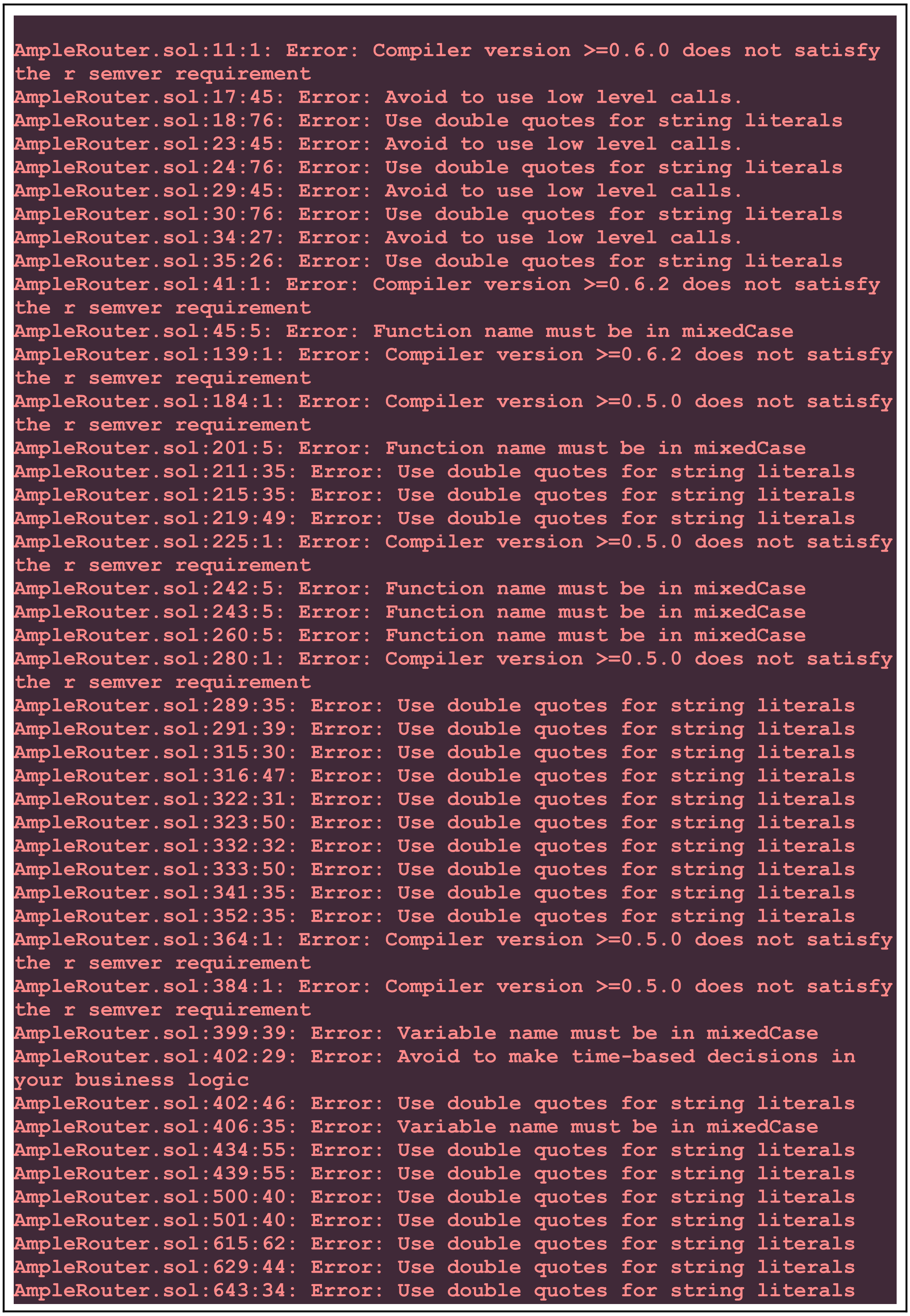
Solhint Linter

AmpleFactory.sol

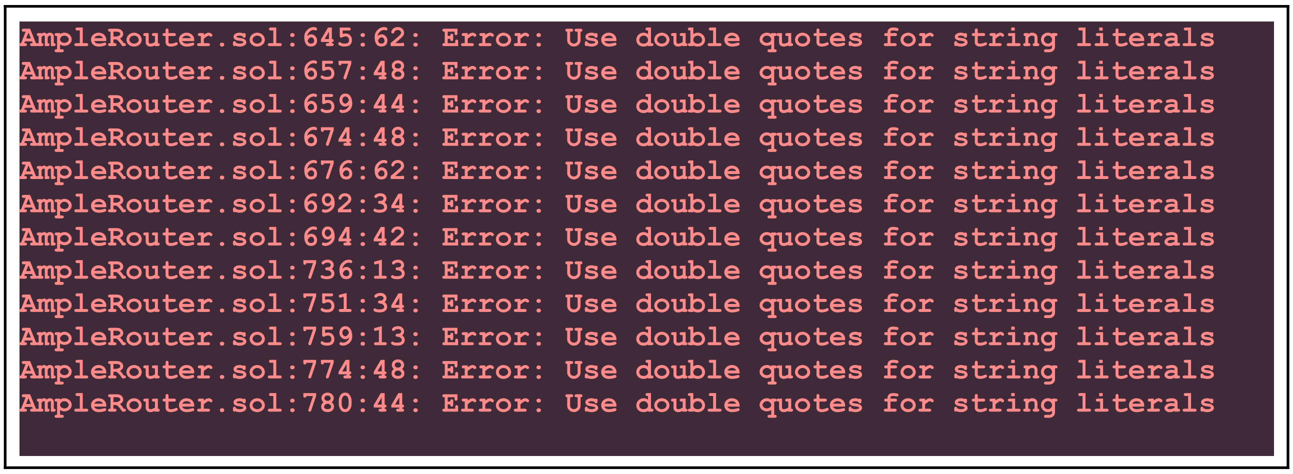


```
AmpleFactory.sol:9:1: Error: Compiler version =0.5.16 does not satisfy
the r semver requirement
AmpleFactory.sol:43:5: Error: Function name must be in mixedCase
AmpleFactory.sol:44:5: Error: Function name must be in mixedCase
AmpleFactory.sol:61:5: Error: Function name must be in mixedCase
AmpleFactory.sol:94:5: Error: Function name must be in mixedCase
AmpleFactory.sol:95:5: Error: Function name must be in mixedCase
AmpleFactory.sol:104:35: Error: Use double quotes for string literals
AmpleFactory.sol:108:35: Error: Use double quotes for string literals
AmpleFactory.sol:112:49: Error: Use double quotes for string literals
AmpleFactory.sol:119:28: Error: Constant name must be in capitalized
SNAKE_CASE
AmpleFactory.sol:119:35: Error: Use double quotes for string literals
AmpleFactory.sol:120:28: Error: Constant name must be in capitalized
SNAKE_CASE
AmpleFactory.sol:120:37: Error: Use double quotes for string literals
AmpleFactory.sol:121:27: Error: Constant name must be in capitalized
SNAKE_CASE
AmpleFactory.sol:126:20: Error: Variable name must be in mixedCase
AmpleFactory.sol:136:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
AmpleFactory.sol:141:27: Error: Use double quotes for string literals
AmpleFactory.sol:143:33: Error: Use double quotes for string literals
AmpleFactory.sol:192:29: Error: Avoid to make time-based decisions in
your business logic
AmpleFactory.sol:192:46: Error: Use double quotes for string literals
AmpleFactory.sol:195:17: Error: Use double quotes for string literals
AmpleFactory.sol:201:78: Error: Use double quotes for string literals
AmpleFactory.sol:231:5: Error: Explicitly mark visibility of state
AmpleFactory.sol:269:63: Error: Use double quotes for string literals
AmpleFactory.sol:285:32: Error: Use double quotes for string literals
AmpleFactory.sol:298:45: Error: Avoid to use low level calls.
AmpleFactory.sol:299:76: Error: Use double quotes for string literals
AmpleFactory.sol:320:40: Error: Use double quotes for string literals
AmpleFactory.sol:327:69: Error: Use double quotes for string literals
AmpleFactory.sol:328:40: Error: Avoid to make time-based decisions in
your business logic
AmpleFactory.sol:378:32: Error: Use double quotes for string literals
AmpleFactory.sol:399:45: Error: Use double quotes for string literals
AmpleFactory.sol:413:51: Error: Use double quotes for string literals
AmpleFactory.sol:415:67: Error: Use double quotes for string literals
AmpleFactory.sol:422:49: Error: Use double quotes for string literals
AmpleFactory.sol:431:49: Error: Use double quotes for string literals
AmpleFactory.sol:435:105: Error: Use double quotes for string literals
AmpleFactory.sol:476:35: Error: Use double quotes for string literals
AmpleFactory.sol:478:39: Error: Use double quotes for string literals
AmpleFactory.sol:479:56: Error: Use double quotes for string literals
AmpleFactory.sol:482:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
AmpleFactory.sol:493:44: Error: Use double quotes for string literals
```

AmpleRouter.sol

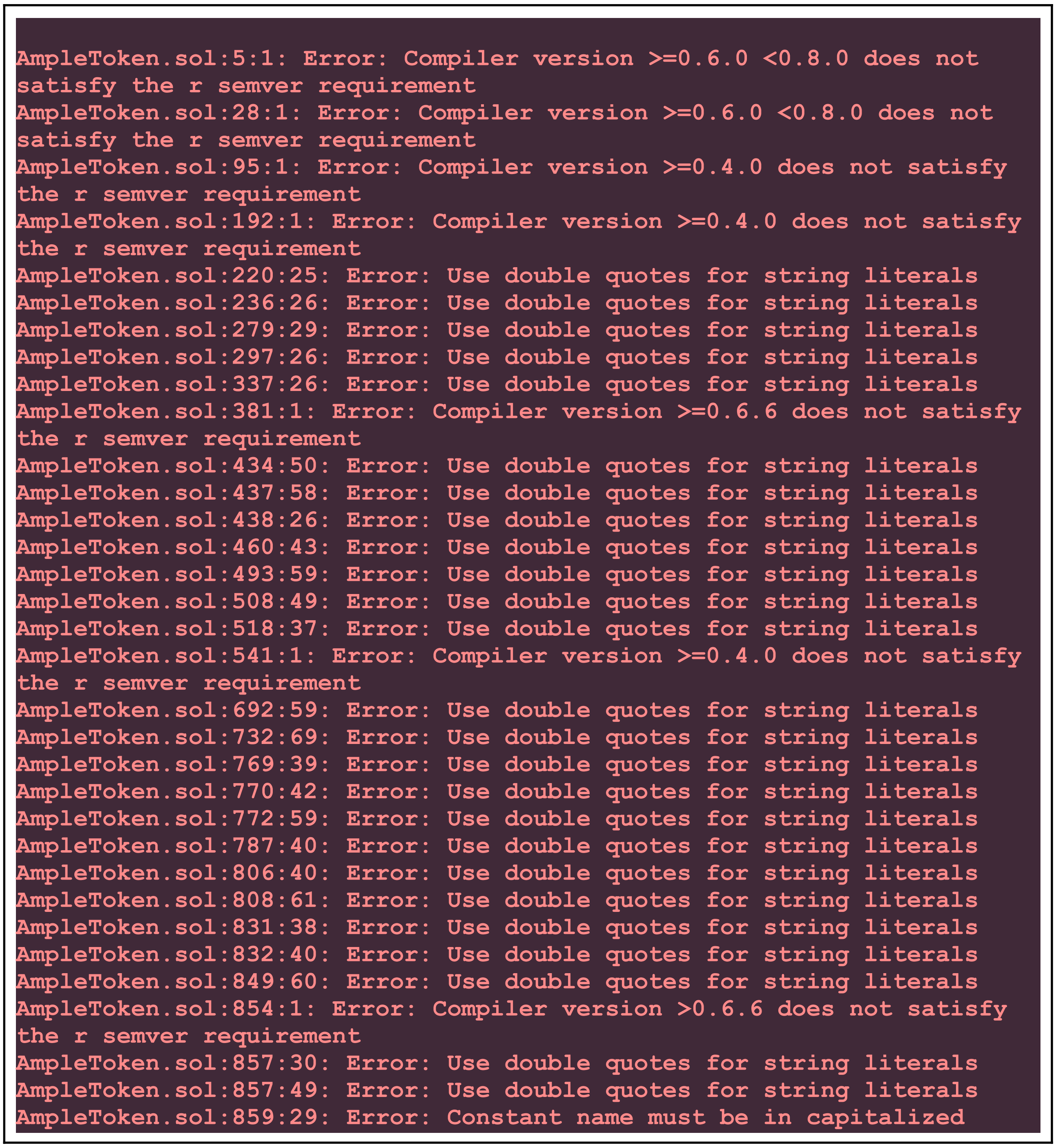


```
AmpleRouter.sol:11:1: Error: Compiler version >=0.6.0 does not satisfy
the r semver requirement
AmpleRouter.sol:17:45: Error: Avoid to use low level calls.
AmpleRouter.sol:18:76: Error: Use double quotes for string literals
AmpleRouter.sol:23:45: Error: Avoid to use low level calls.
AmpleRouter.sol:24:76: Error: Use double quotes for string literals
AmpleRouter.sol:29:45: Error: Avoid to use low level calls.
AmpleRouter.sol:30:76: Error: Use double quotes for string literals
AmpleRouter.sol:34:27: Error: Avoid to use low level calls.
AmpleRouter.sol:35:26: Error: Use double quotes for string literals
AmpleRouter.sol:41:1: Error: Compiler version >=0.6.2 does not satisfy
the r semver requirement
AmpleRouter.sol:45:5: Error: Function name must be in mixedCase
AmpleRouter.sol:139:1: Error: Compiler version >=0.6.2 does not satisfy
the r semver requirement
AmpleRouter.sol:184:1: Error: Compiler version >=0.5.0 does not satisfy
the r semver requirement
AmpleRouter.sol:201:5: Error: Function name must be in mixedCase
AmpleRouter.sol:211:35: Error: Use double quotes for string literals
AmpleRouter.sol:215:35: Error: Use double quotes for string literals
AmpleRouter.sol:219:49: Error: Use double quotes for string literals
AmpleRouter.sol:225:1: Error: Compiler version >=0.5.0 does not satisfy
the r semver requirement
AmpleRouter.sol:242:5: Error: Function name must be in mixedCase
AmpleRouter.sol:243:5: Error: Function name must be in mixedCase
AmpleRouter.sol:260:5: Error: Function name must be in mixedCase
AmpleRouter.sol:280:1: Error: Compiler version >=0.5.0 does not satisfy
the r semver requirement
AmpleRouter.sol:289:35: Error: Use double quotes for string literals
AmpleRouter.sol:291:39: Error: Use double quotes for string literals
AmpleRouter.sol:315:30: Error: Use double quotes for string literals
AmpleRouter.sol:316:47: Error: Use double quotes for string literals
AmpleRouter.sol:322:31: Error: Use double quotes for string literals
AmpleRouter.sol:323:50: Error: Use double quotes for string literals
AmpleRouter.sol:332:32: Error: Use double quotes for string literals
AmpleRouter.sol:333:50: Error: Use double quotes for string literals
AmpleRouter.sol:341:35: Error: Use double quotes for string literals
AmpleRouter.sol:352:35: Error: Use double quotes for string literals
AmpleRouter.sol:364:1: Error: Compiler version >=0.5.0 does not satisfy
the r semver requirement
AmpleRouter.sol:384:1: Error: Compiler version >=0.5.0 does not satisfy
the r semver requirement
AmpleRouter.sol:399:39: Error: Variable name must be in mixedCase
AmpleRouter.sol:402:29: Error: Avoid to make time-based decisions in
your business logic
AmpleRouter.sol:402:46: Error: Use double quotes for string literals
AmpleRouter.sol:406:35: Error: Variable name must be in mixedCase
AmpleRouter.sol:434:55: Error: Use double quotes for string literals
AmpleRouter.sol:439:55: Error: Use double quotes for string literals
AmpleRouter.sol:500:40: Error: Use double quotes for string literals
AmpleRouter.sol:501:40: Error: Use double quotes for string literals
AmpleRouter.sol:615:62: Error: Use double quotes for string literals
AmpleRouter.sol:629:44: Error: Use double quotes for string literals
AmpleRouter.sol:643:34: Error: Use double quotes for string literals
```

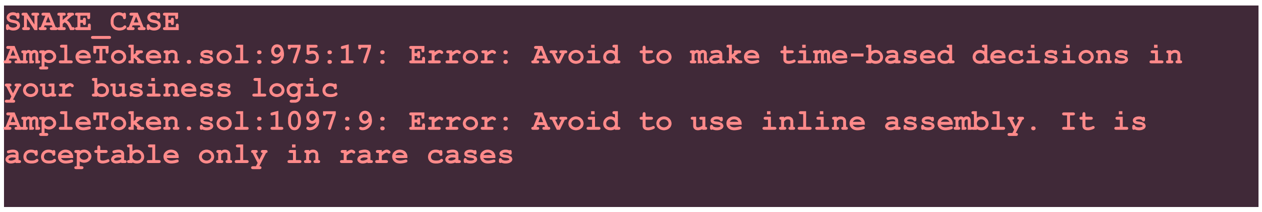


```
AmpleRouter.sol:645:62: Error: Use double quotes for string literals
AmpleRouter.sol:657:48: Error: Use double quotes for string literals
AmpleRouter.sol:659:44: Error: Use double quotes for string literals
AmpleRouter.sol:674:48: Error: Use double quotes for string literals
AmpleRouter.sol:676:62: Error: Use double quotes for string literals
AmpleRouter.sol:692:34: Error: Use double quotes for string literals
AmpleRouter.sol:694:42: Error: Use double quotes for string literals
AmpleRouter.sol:736:13: Error: Use double quotes for string literals
AmpleRouter.sol:751:34: Error: Use double quotes for string literals
AmpleRouter.sol:759:13: Error: Use double quotes for string literals
AmpleRouter.sol:774:48: Error: Use double quotes for string literals
AmpleRouter.sol:780:44: Error: Use double quotes for string literals
```

AmpleToken.sol

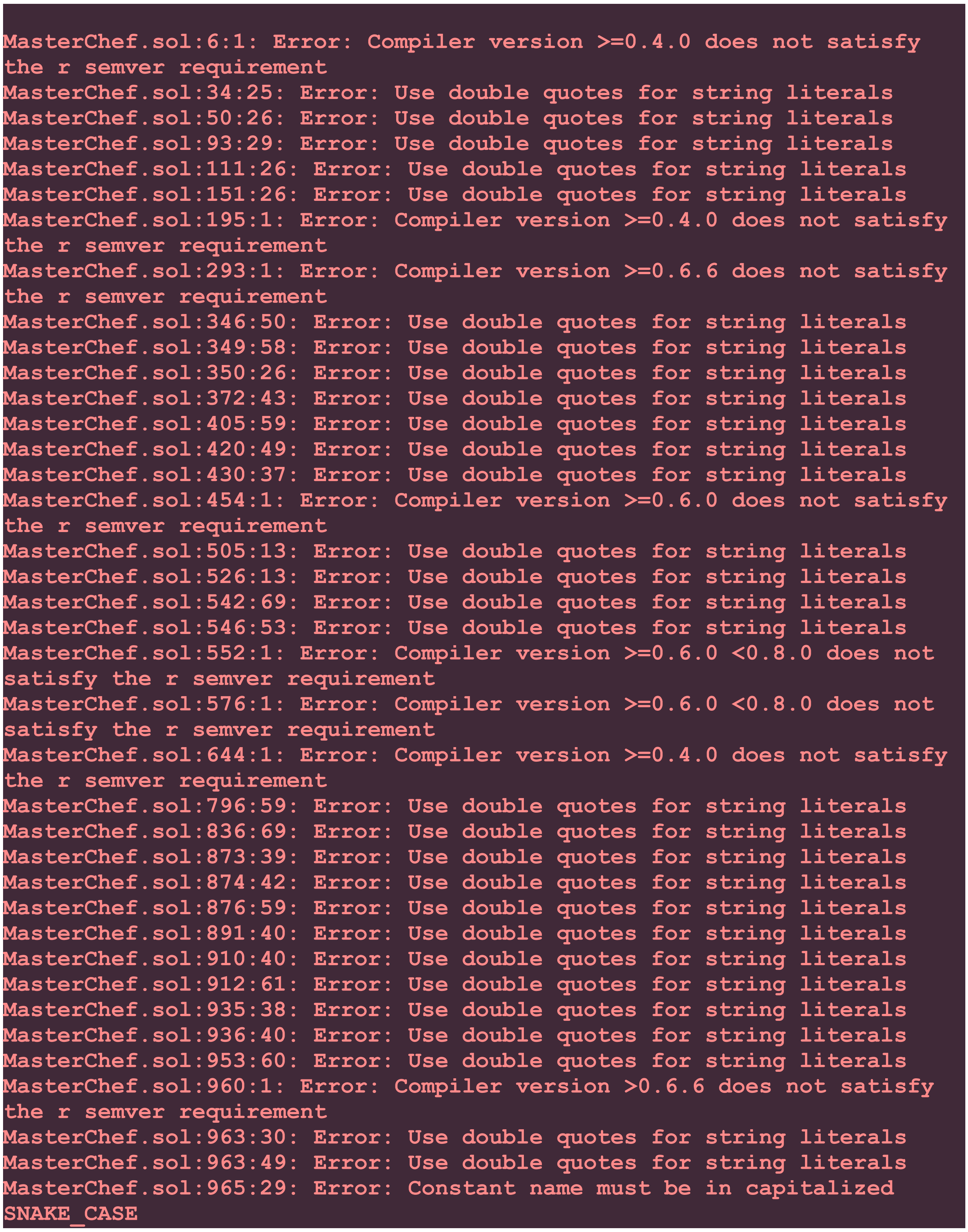


```
AmpleToken.sol:5:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
AmpleToken.sol:28:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
AmpleToken.sol:95:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
AmpleToken.sol:192:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
AmpleToken.sol:220:25: Error: Use double quotes for string literals
AmpleToken.sol:236:26: Error: Use double quotes for string literals
AmpleToken.sol:279:29: Error: Use double quotes for string literals
AmpleToken.sol:297:26: Error: Use double quotes for string literals
AmpleToken.sol:337:26: Error: Use double quotes for string literals
AmpleToken.sol:381:1: Error: Compiler version >=0.6.6 does not satisfy
the r semver requirement
AmpleToken.sol:434:50: Error: Use double quotes for string literals
AmpleToken.sol:437:58: Error: Use double quotes for string literals
AmpleToken.sol:438:26: Error: Use double quotes for string literals
AmpleToken.sol:460:43: Error: Use double quotes for string literals
AmpleToken.sol:493:59: Error: Use double quotes for string literals
AmpleToken.sol:508:49: Error: Use double quotes for string literals
AmpleToken.sol:518:37: Error: Use double quotes for string literals
AmpleToken.sol:541:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
AmpleToken.sol:692:59: Error: Use double quotes for string literals
AmpleToken.sol:732:69: Error: Use double quotes for string literals
AmpleToken.sol:769:39: Error: Use double quotes for string literals
AmpleToken.sol:770:42: Error: Use double quotes for string literals
AmpleToken.sol:772:59: Error: Use double quotes for string literals
AmpleToken.sol:787:40: Error: Use double quotes for string literals
AmpleToken.sol:806:40: Error: Use double quotes for string literals
AmpleToken.sol:808:61: Error: Use double quotes for string literals
AmpleToken.sol:831:38: Error: Use double quotes for string literals
AmpleToken.sol:832:40: Error: Use double quotes for string literals
AmpleToken.sol:849:60: Error: Use double quotes for string literals
AmpleToken.sol:854:1: Error: Compiler version >0.6.6 does not satisfy
the r semver requirement
AmpleToken.sol:857:30: Error: Use double quotes for string literals
AmpleToken.sol:857:49: Error: Use double quotes for string literals
AmpleToken.sol:859:29: Error: Constant name must be in capitalized
```

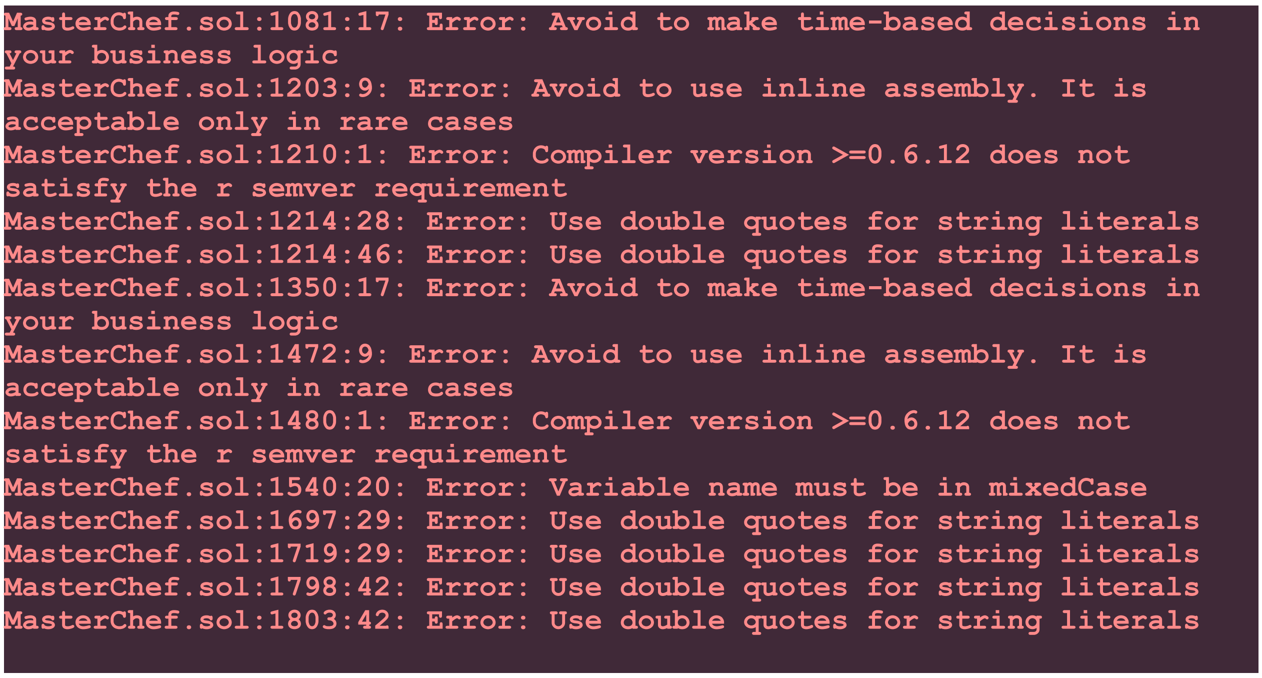


```
SNAKE_CASE
AmpleToken.sol:975:17: Error: Avoid to make time-based decisions in
your business logic
AmpleToken.sol:1097:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
```

MasterChef.sol

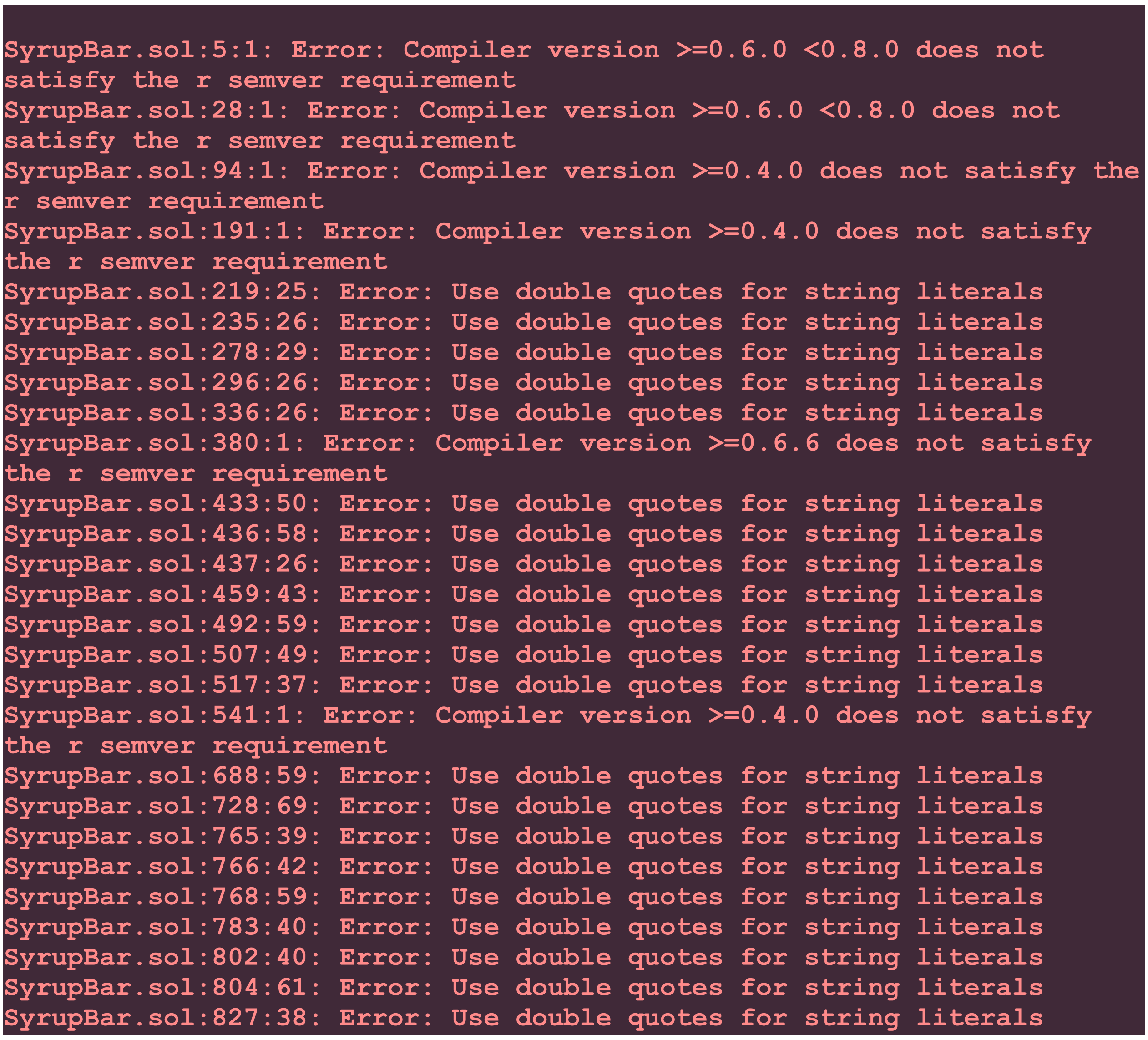


```
MasterChef.sol:6:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
MasterChef.sol:34:25: Error: Use double quotes for string literals
MasterChef.sol:50:26: Error: Use double quotes for string literals
MasterChef.sol:93:29: Error: Use double quotes for string literals
MasterChef.sol:111:26: Error: Use double quotes for string literals
MasterChef.sol:151:26: Error: Use double quotes for string literals
MasterChef.sol:195:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
MasterChef.sol:293:1: Error: Compiler version >=0.6.6 does not satisfy
the r semver requirement
MasterChef.sol:346:50: Error: Use double quotes for string literals
MasterChef.sol:349:58: Error: Use double quotes for string literals
MasterChef.sol:350:26: Error: Use double quotes for string literals
MasterChef.sol:372:43: Error: Use double quotes for string literals
MasterChef.sol:405:59: Error: Use double quotes for string literals
MasterChef.sol:420:49: Error: Use double quotes for string literals
MasterChef.sol:430:37: Error: Use double quotes for string literals
MasterChef.sol:454:1: Error: Compiler version >=0.6.0 does not satisfy
the r semver requirement
MasterChef.sol:505:13: Error: Use double quotes for string literals
MasterChef.sol:526:13: Error: Use double quotes for string literals
MasterChef.sol:542:69: Error: Use double quotes for string literals
MasterChef.sol:546:53: Error: Use double quotes for string literals
MasterChef.sol:552:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
MasterChef.sol:576:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
MasterChef.sol:644:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
MasterChef.sol:796:59: Error: Use double quotes for string literals
MasterChef.sol:836:69: Error: Use double quotes for string literals
MasterChef.sol:873:39: Error: Use double quotes for string literals
MasterChef.sol:874:42: Error: Use double quotes for string literals
MasterChef.sol:876:59: Error: Use double quotes for string literals
MasterChef.sol:891:40: Error: Use double quotes for string literals
MasterChef.sol:910:40: Error: Use double quotes for string literals
MasterChef.sol:912:61: Error: Use double quotes for string literals
MasterChef.sol:935:38: Error: Use double quotes for string literals
MasterChef.sol:936:40: Error: Use double quotes for string literals
MasterChef.sol:953:60: Error: Use double quotes for string literals
MasterChef.sol:960:1: Error: Compiler version >0.6.6 does not satisfy
the r semver requirement
MasterChef.sol:963:30: Error: Use double quotes for string literals
MasterChef.sol:963:49: Error: Use double quotes for string literals
MasterChef.sol:965:29: Error: Constant name must be in capitalized
SNAKE_CASE
```

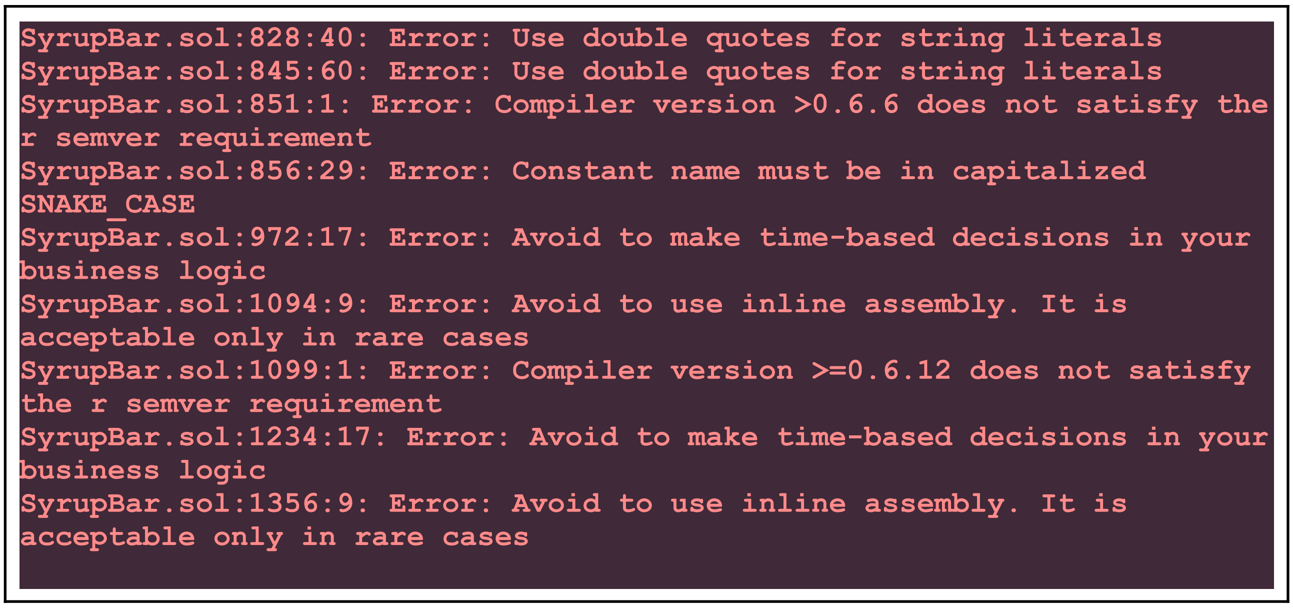


```
MasterChef.sol:1081:17: Error: Avoid to make time-based decisions in
your business logic
MasterChef.sol:1203:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
MasterChef.sol:1210:1: Error: Compiler version >=0.6.12 does not
satisfy the r semver requirement
MasterChef.sol:1214:28: Error: Use double quotes for string literals
MasterChef.sol:1214:46: Error: Use double quotes for string literals
MasterChef.sol:1350:17: Error: Avoid to make time-based decisions in
your business logic
MasterChef.sol:1472:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
MasterChef.sol:1480:1: Error: Compiler version >=0.6.12 does not
satisfy the r semver requirement
MasterChef.sol:1540:20: Error: Variable name must be in mixedCase
MasterChef.sol:1697:29: Error: Use double quotes for string literals
MasterChef.sol:1719:29: Error: Use double quotes for string literals
MasterChef.sol:1798:42: Error: Use double quotes for string literals
MasterChef.sol:1803:42: Error: Use double quotes for string literals
```

SyrupBar.sol



```
SyrupBar.sol:5:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
SyrupBar.sol:28:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
SyrupBar.sol:94:1: Error: Compiler version >=0.4.0 does not satisfy the
r semver requirement
SyrupBar.sol:191:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
SyrupBar.sol:219:25: Error: Use double quotes for string literals
SyrupBar.sol:235:26: Error: Use double quotes for string literals
SyrupBar.sol:278:29: Error: Use double quotes for string literals
SyrupBar.sol:296:26: Error: Use double quotes for string literals
SyrupBar.sol:336:26: Error: Use double quotes for string literals
SyrupBar.sol:380:1: Error: Compiler version >=0.6.6 does not satisfy
the r semver requirement
SyrupBar.sol:433:50: Error: Use double quotes for string literals
SyrupBar.sol:436:58: Error: Use double quotes for string literals
SyrupBar.sol:437:26: Error: Use double quotes for string literals
SyrupBar.sol:459:43: Error: Use double quotes for string literals
SyrupBar.sol:492:59: Error: Use double quotes for string literals
SyrupBar.sol:507:49: Error: Use double quotes for string literals
SyrupBar.sol:517:37: Error: Use double quotes for string literals
SyrupBar.sol:541:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
SyrupBar.sol:688:59: Error: Use double quotes for string literals
SyrupBar.sol:728:69: Error: Use double quotes for string literals
SyrupBar.sol:765:39: Error: Use double quotes for string literals
SyrupBar.sol:766:42: Error: Use double quotes for string literals
SyrupBar.sol:768:59: Error: Use double quotes for string literals
SyrupBar.sol:783:40: Error: Use double quotes for string literals
SyrupBar.sol:802:40: Error: Use double quotes for string literals
SyrupBar.sol:804:61: Error: Use double quotes for string literals
SyrupBar.sol:827:38: Error: Use double quotes for string literals
```



```
SyrupBar.sol:828:40: Error: Use double quotes for string literals
SyrupBar.sol:845:60: Error: Use double quotes for string literals
SyrupBar.sol:851:1: Error: Compiler version >0.6.6 does not satisfy the
r semver requirement
SyrupBar.sol:856:29: Error: Constant name must be in capitalized
SNAKE_CASE
SyrupBar.sol:972:17: Error: Avoid to make time-based decisions in your
business logic
SyrupBar.sol:1094:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
SyrupBar.sol:1099:1: Error: Compiler version >=0.6.12 does not satisfy
the r semver requirement
SyrupBar.sol:1234:17: Error: Avoid to make time-based decisions in your
business logic
SyrupBar.sol:1356:9: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
```

Software analysis result:

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

EtherAuthority