Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Versa Protocol
Website:	https://versa.finance
Platform:	Astar Network
Language:	Solidity
Date:	April 25th, 2022

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 Versa team to perform the Security audit of the Versa Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on April 25th, 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

Versa Finance is a UX-oriented family of dApps on the Astar Network. This audit project consists of automatic market maker (AMM) decentralized exchange smart contracts.

Audit scope

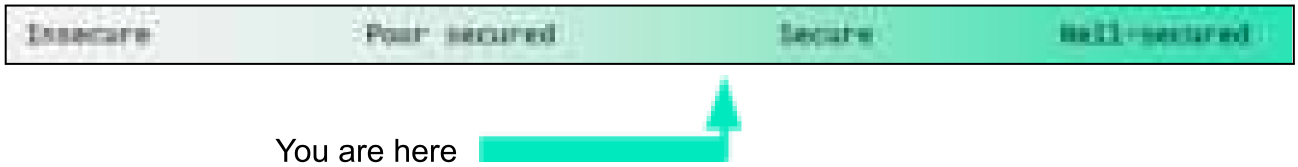
Name	**Code Review and Security Analysis Report for Versa Finance Protocol Smart Contracts**
Platform	**Astar / Solidity**
File 1	MasterChef.sol
File 1 Github Commit	41f5710dbc4a05a5fb6eaf7ea4a723fca2682377
File 2	SyrupBar.sol
File 2 Github Commit	ccb074dca3aa40b0a0379115bbde37aa0df6886d
File 3	TokenTimelock.sol
File 3 Github Commit	9ed0ef04f55f8f7544344b0c9de50ef823e9bc98
File 4	Versa.sol
File 4 Github Commit	582782ed9740c8e3a42c8c87b6514d46760d439b
File 5	VersaRouter.sol
File 5 Github Commit	51ebcbe2ea96eb9abd8086d4b8551e2f25731eb3
File 6	VersaFactory.sol
File 6 Github Commit	28f2831ee6e53384838c9a5d126bb0f402fc36cb

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 MasterChef.sol ● MasterChef is the master of Versa. ● Bonus Multiplier: 1 ● Total Alloc Point: 1000 ● Dev commission: 10%	**YES, This is valid.**
File 2 SyrupBar.sol ● Name: SyrupBar Token ● Symbol: SYRUP ● Decimals: 18 ● Minting by masterChef contract	**YES, This is valid.**
File 3 TokenTimelock.sol ● Lock time can be set at the time of contract deployment	**YES, This is valid.**
File 4 Versa.sol ● Name: Versa ● Symbol: VERSA ● Decimals: 18 ● Dev Fund Pool Allocation: 500000 Tokens ● Vesting Duration: 300 Days ● Minting should be done by MasterChef contract	**YES, This is valid.**
File 5 VersaRouter.sol ● Performs trading/swapping of tokens ● Performs add/remove liquidity	**YES, This is valid.**
File 6 VersaFactory.sol ● Creates token pairs	**YES, This is valid.**

Audit Summary

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



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.

All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 1 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	Passed
	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	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	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 6 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 Versa 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 Versa Finance Protocol.

The Versa 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 **not** well commented on smart contracts.

Documentation

We were given a Versa Finance Protocol smart contract code in the form of github links. The commits of that code are 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://versa.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

MasterChef.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	updateMultiplier	write	access only Owner	No Issue
8	poolLength	external	Passed	No Issue
9	add	write	Input validation missing	LP Token must not be added twice
10	set	write	access only Owner	No Issue
11	updateStakingPool	internal	Infinite loop possibility	Array length must be limited
12	setMigrator	write	access only Owner	No Issue
13	migrate	write	This should be removed, as it can be potential rugpull	Acknowledged by the dev team that It will not be used by the owner
14	getMultiplier	read	Passed	No Issue
15	pendingVersa	external	Passed	No Issue
16	massUpdatePools	write	Infinite loop possibility	Array length must be limited
17	updatePool	write	Passed	No Issue
18	deposit	write	Passed	No Issue
19	withdraw	write	Passed	No Issue
20	enterStaking	write	Passed	No Issue
21	leaveStaking	write	Passed	No Issue
22	emergencyWithdraw	write	Passed	No Issue
23	safeVersaTransfer	internal	Passed	No Issue
24	dev	write	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	read	Passed	No Issue
4	decimals	read	Passed	No Issue

5	symbol	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	transfer	write	Passed	No Issue
9	allowance	read	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	write	access only Owner	No Issue
21	burn	write	access only Owner	No Issue
22	safeVersaTransfer	write	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	Passed	No Issue
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

TokenTimelock.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	token	read	Passed	No Issue
3	beneficiary	read	Passed	No Issue
4	releaseTime	read	Passed	No Issue
5	release	write	Passed	No Issue

Versa.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getOwner	external	Passed	No Issue
3	name	read	Passed	No Issue

4	decimals	read	Passed	No Issue
5	symbol	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	transfer	write	Passed	No Issue
9	allowance	read	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	addDevAddr	write	access only Owner	No Issue
21	delegate	external	Passed	No Issue
22	delegates	external	Passed	No Issue
23	delegateBySig	external	Passed	No Issue
24	getCurrentVotes	external	Passed	No Issue
25	getPriorVotes	external	Passed	No Issue
26	_delegate	internal	Passed	No Issue
27	_moveDelegates	internal	Passed	No Issue
28	_writeCheckpoint	internal	Passed	No Issue
29	safe32	internal	Passed	No Issue
30	unclaimedDevFund	read	Passed	No Issue
31	claimRewards	external	Passed	No Issue
32	getChainId	internal	Passed	No Issue

VersaRouter.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	write	Passed	No Issue
12	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	write	Passed	No Issue
13	_swap	internal	Infinite loop possibility	Keep path limited
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	Infinite loop possibility	Keep path limited
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

VersaFactory.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

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.

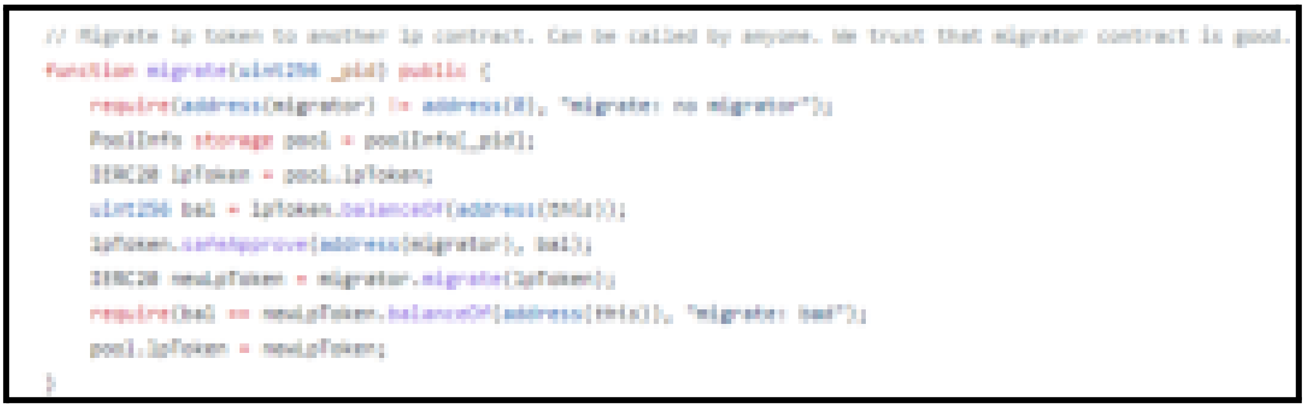
Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

(1) The migrator code is present - MasterChef.sol



```
// Migrate lp token to another lp contract. Can be called by anyone. We trust that migrator contract is good.
function migrate(uint256 _pid) public {
    require(address(migrator) != address(0), "migrate: no migrator");
    PoolInfo storage pool = poolInfo[_pid];
    IERC20 lpToken = pool.lpToken;
    uint256 bal = lpToken.balanceOf(address(this));
    lpToken.safeApprove(address(migrator), bal);
    IERC20 newLpToken = migrator.migrate(lpToken);
    require(bal == newLpToken.balanceOf(address(this)), "migrate: bad");
    pool.lpToken = newLpToken;
}
```

This code is used to migrate the LP tokens to any other contract. This creates the scenario of potential rugpull.

Resolution: we advise removing this if there is no need for migrating the LP tokens.

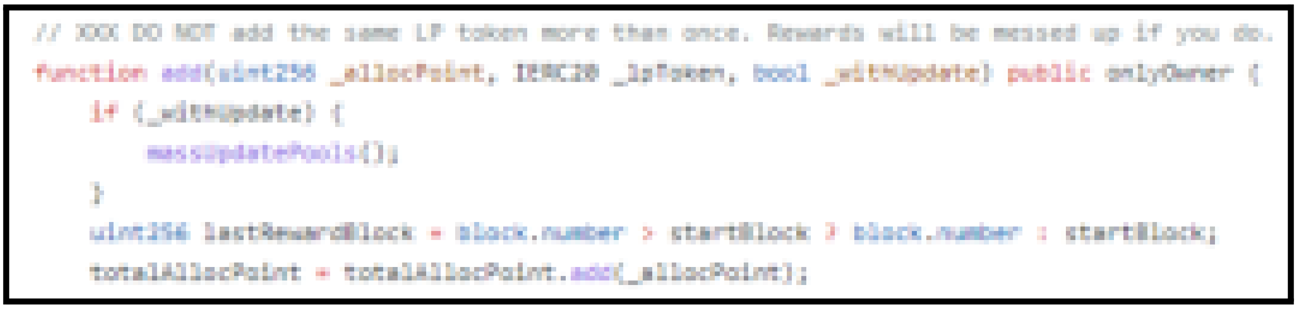
Status: We got confirmation from the Versa team that this functionality will never be used.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Input validation missing - MasterChef.sol



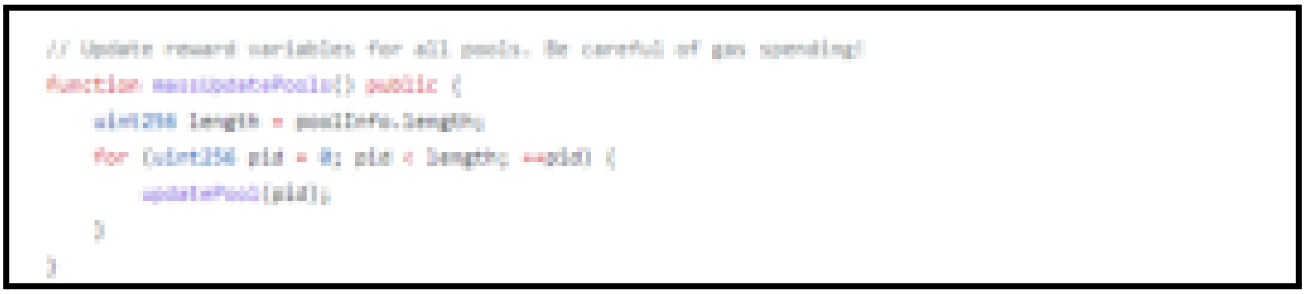
```
// XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.
function add(uint256 _allocPoint, IERC20 _lpToken, bool _withUpdate) public onlyOwner {
    if (_withUpdate) {
        massUpdatePools();
    }
    uint256 lastRewardBlock = block.number > startBlock ? block.number : startBlock;
    totalAllocPoint = totalAllocPoint.add(_allocPoint);
```

As mentioned in the comment, the token must never be added twice. So, there must be a condition to prevent that from happening by mistake.

Resolution: One condition to prevent any duplicate input will fix this.

Status: we got confirmation from the Versa team as this will be taken extra care as this is the owner function.

(2) Infinite loops possibility at multiple places:



```
// 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);
    }
}
```

As seen in the AS-IS section, there are several places in the smart contracts, where array.length is used directly in the loops. It is recommended to put some kind of limits, so it does not go wild and create any scenario where it can hit the block gas limit.

Resolution: Limiting the array length is recommended.

Status: We got confirmation from the Versa team that the array will be provided as limited length. And this will be taken care of from the client side

(3) Missing event logs in VersaFactory.sol

It is best practice to fire an event when a significant state change is happening. It helps clients interact with the blockchain. We suggest to add events in following functions:

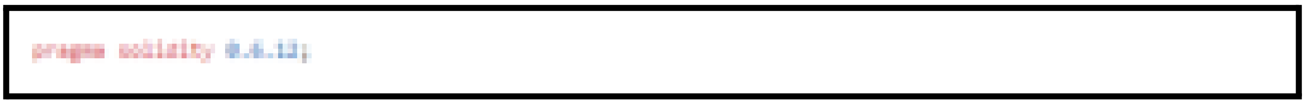
- setFeeTo
- setFeeToSetter

Resolution: Add appropriate events in above functions.

Status: Acknowledged

Very Low / Informational / Best practices:

(1) Use latest solidity version



```
pragma solidity 0.6.12;
```

Consider using the latest solidity version while contract deployment to prevent any compiler version level bugs. There are many features introduced and many security bugs are fixed so it is a good practice to use the latest solidity version.

Resolution: Please use the latest solidity version.

Status: Acknowledged

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:

- add: MasterChef owner can add a new lp to the pool.
- updateMultiplier: MasterChef owner can update multiplier number.
- set: MasterChef owner can update the given pool's VERSA allocation point.
- setMigrator: MasterChef owner can set the migrator contract.
- mint: SyrupBar owner can create `_amount` token to `_to` by MasterChef owner.
- burn: SyrupBar owners can burn an amount from the address.
- safeVersaTransfer: SyrupBar owner can safe versa transfer function, just in case if rounding error causes pool to not have enough VERSAs.
- mint: Versa owner can create `_amount` token to `_to` by MasterChef owner.
- addDevAddr: Versa owner can set dev address.

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 had observed some issues in the smart contracts, and we suggested resolving them using any alternative solutions. **So, smart contracts can be workable and secure**.

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 **"Nearly Secure".**

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

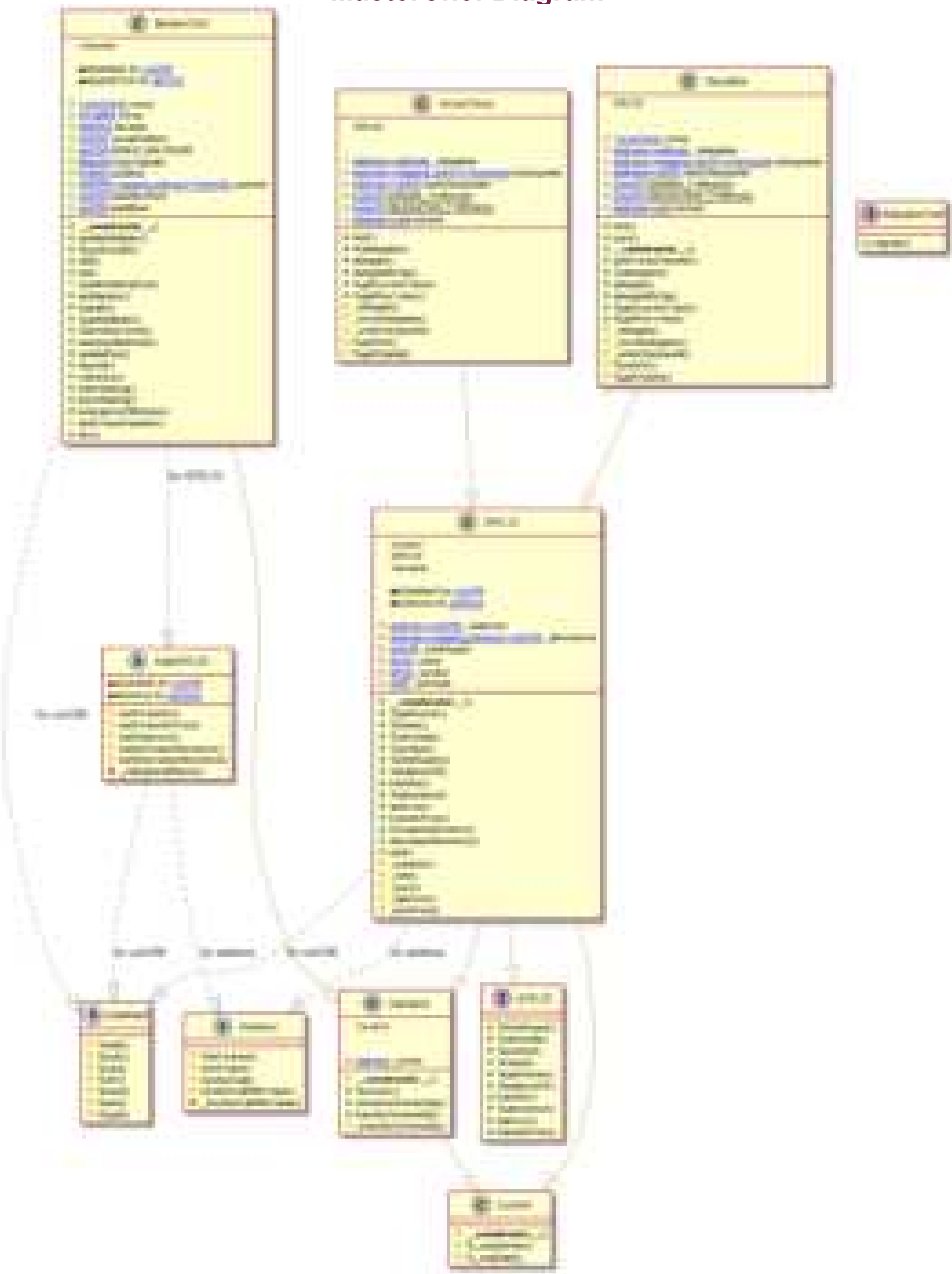
Technical Disclaimer

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

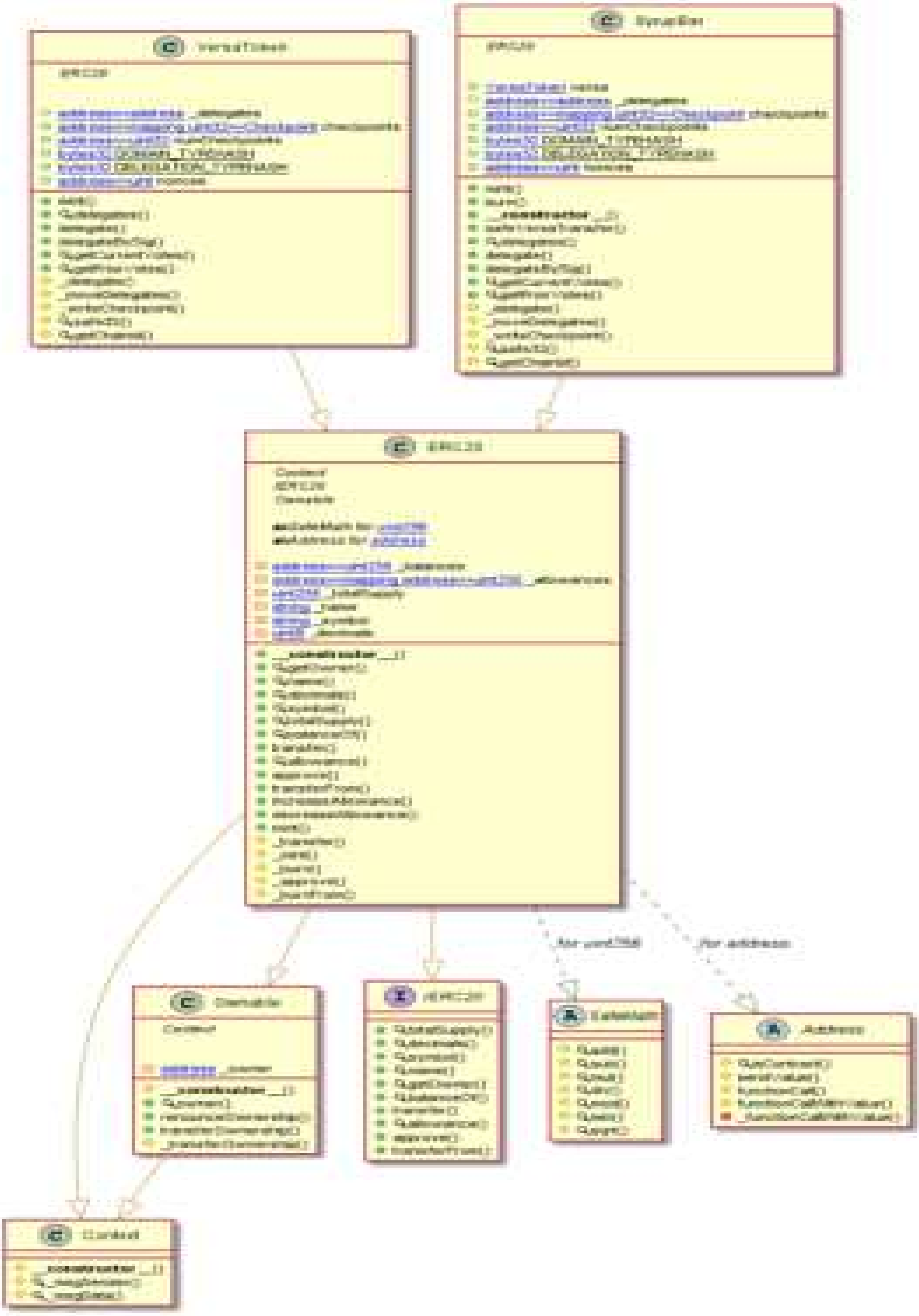
Appendix

Code Flow Diagram - Versa Finance Protocol

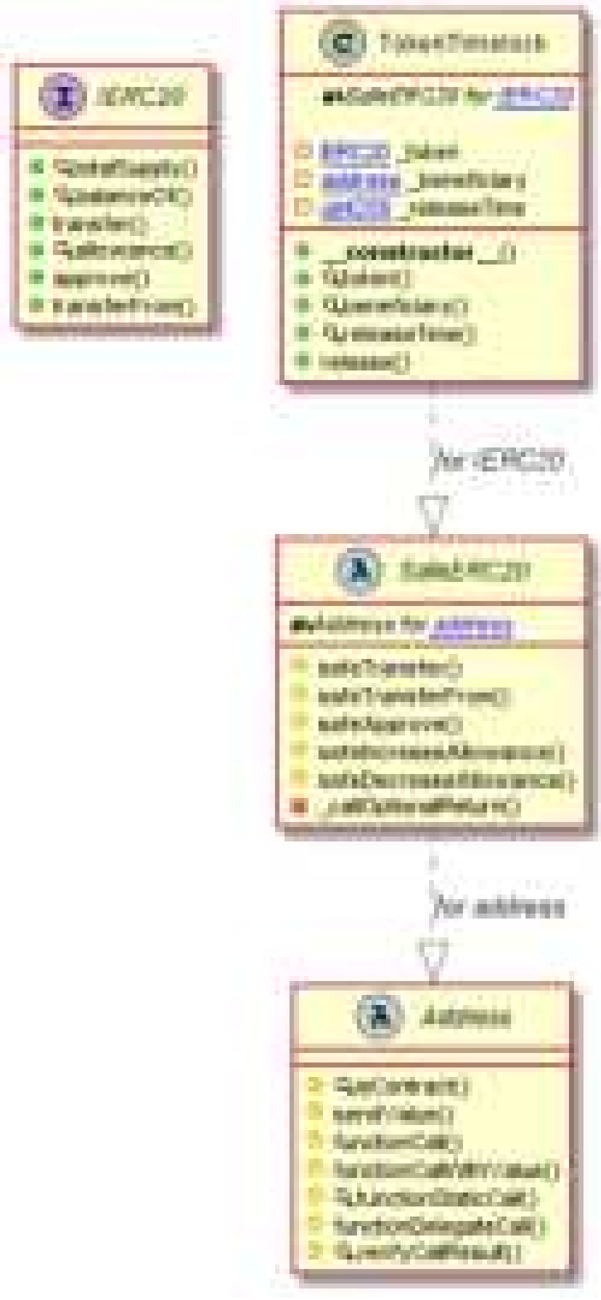
MasterChef Diagram



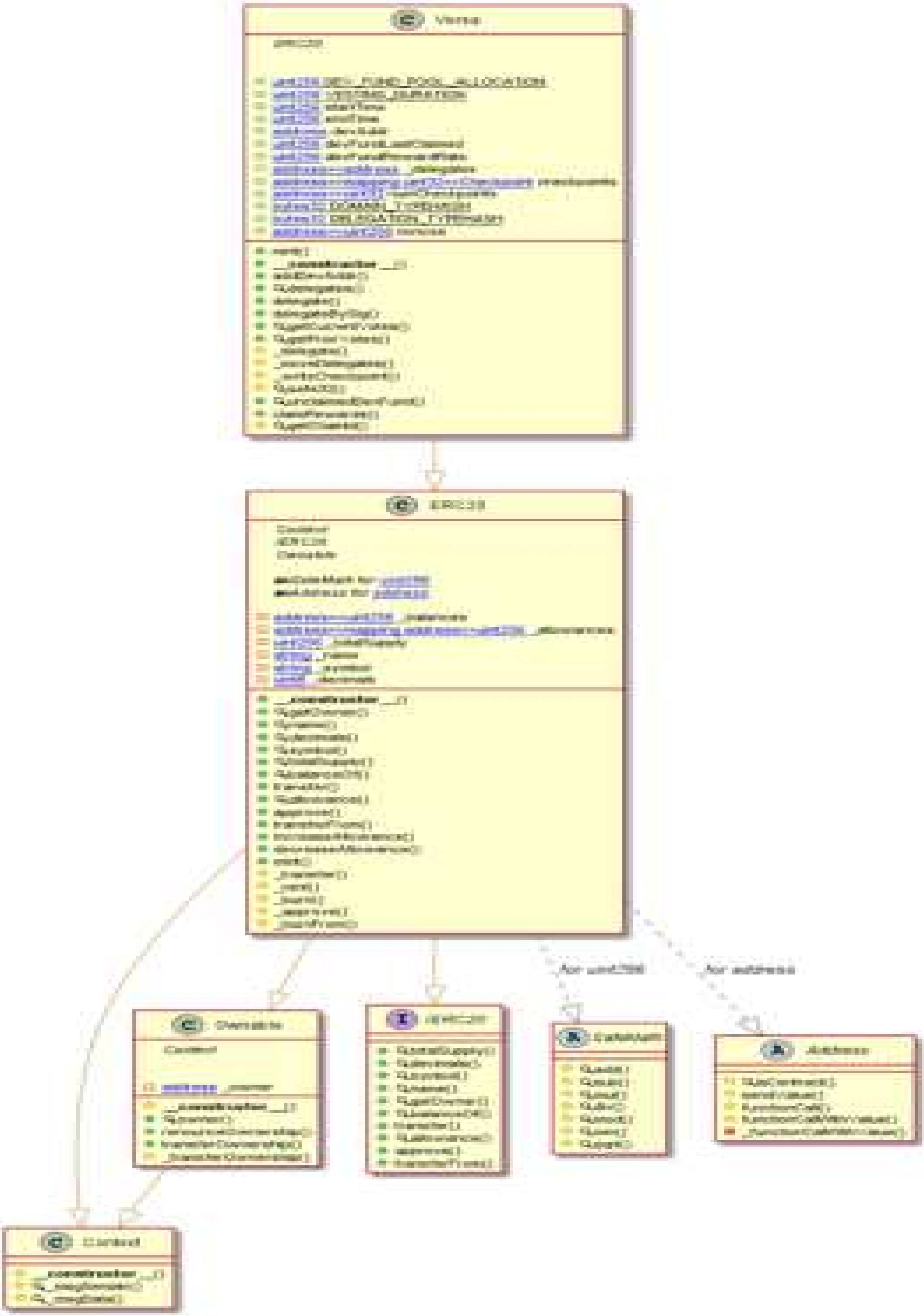
SyrupBar Diagram



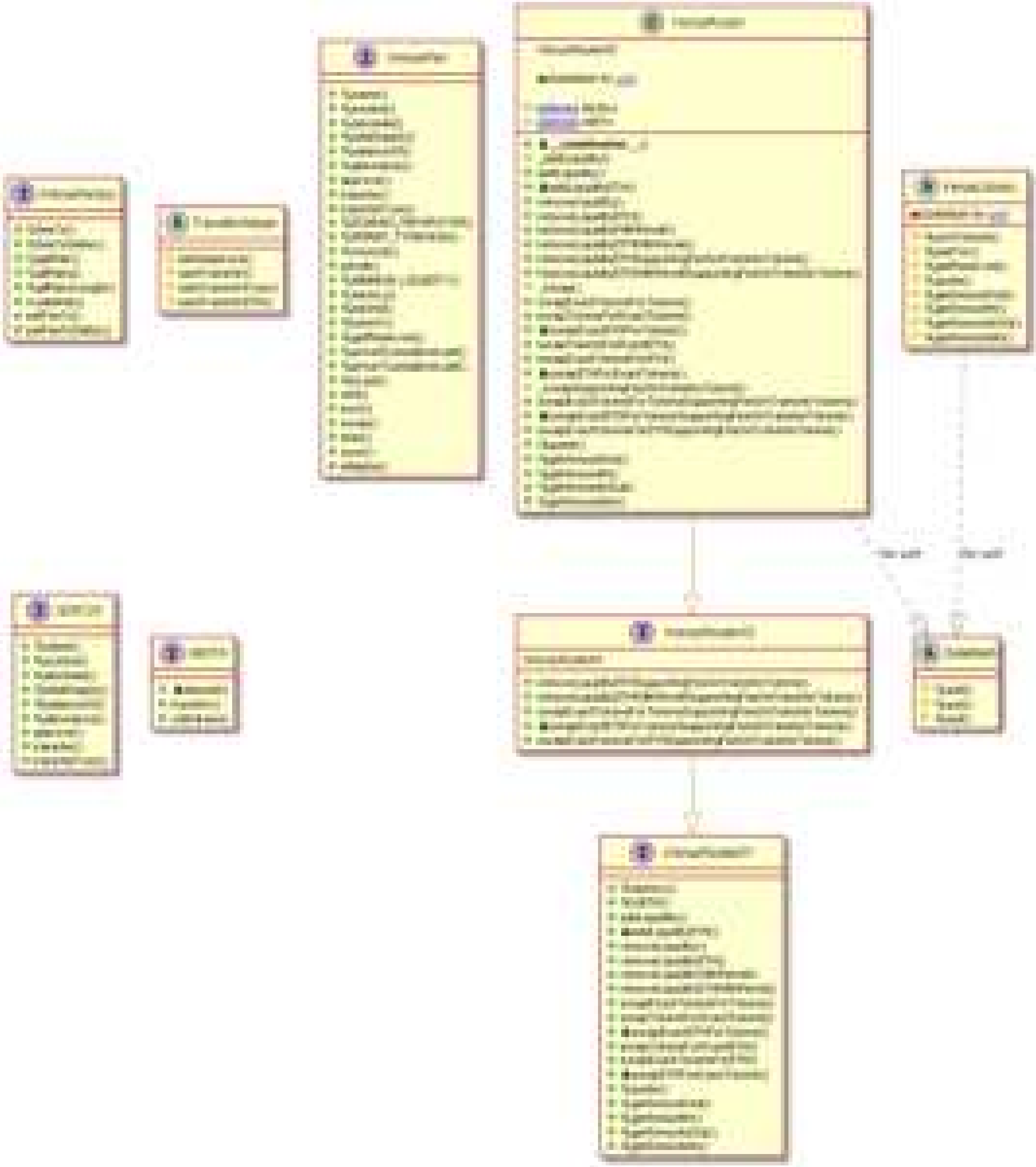
TokenTimelock Diagram



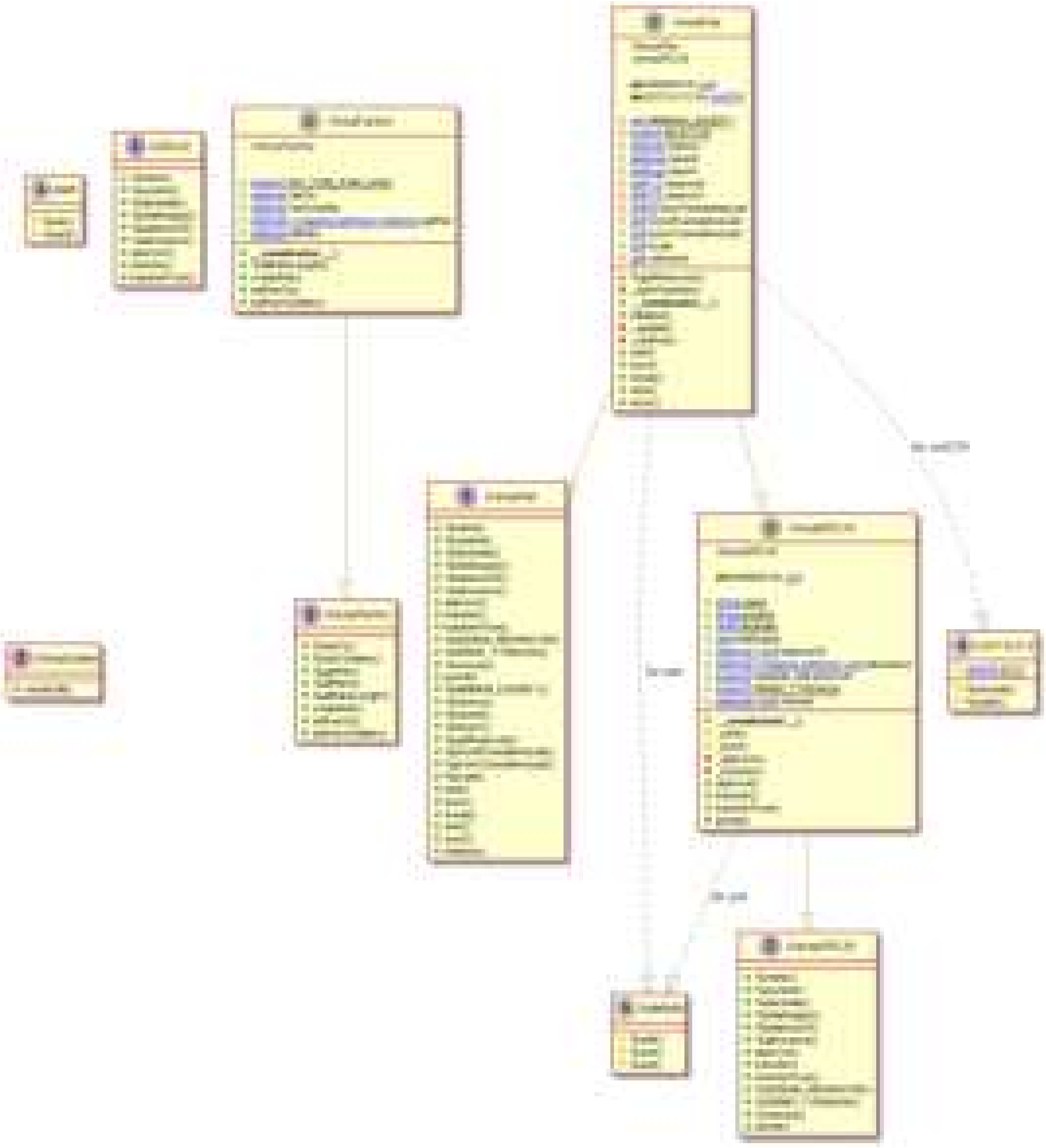
Versa Diagram



VersaRouter Diagram

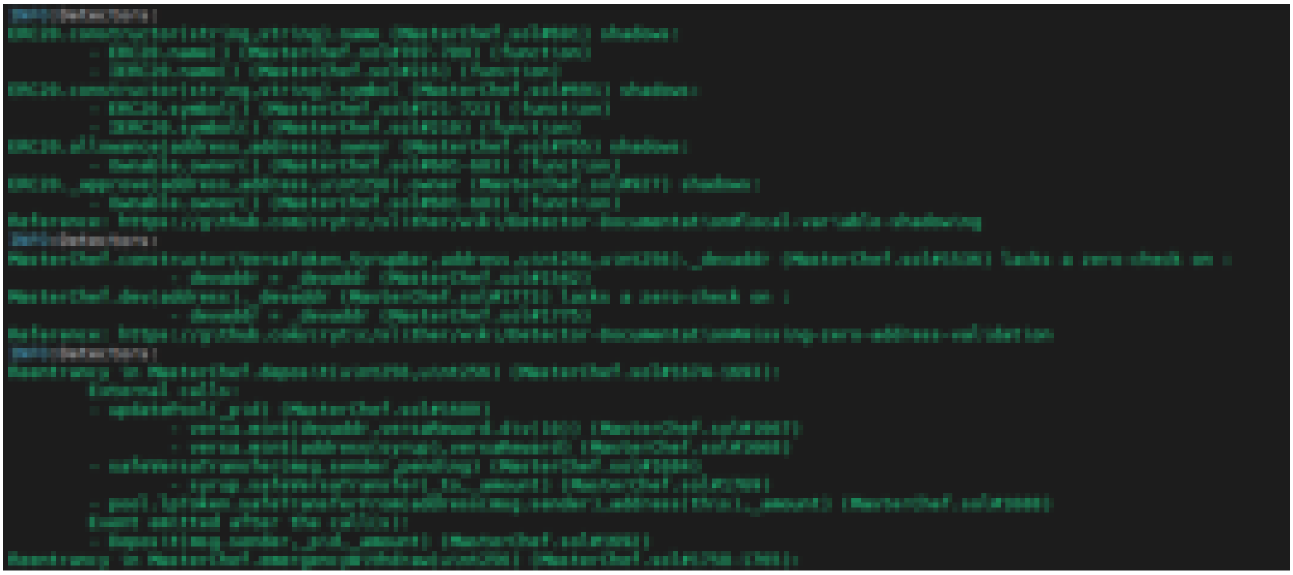


VersaFactory Diagram

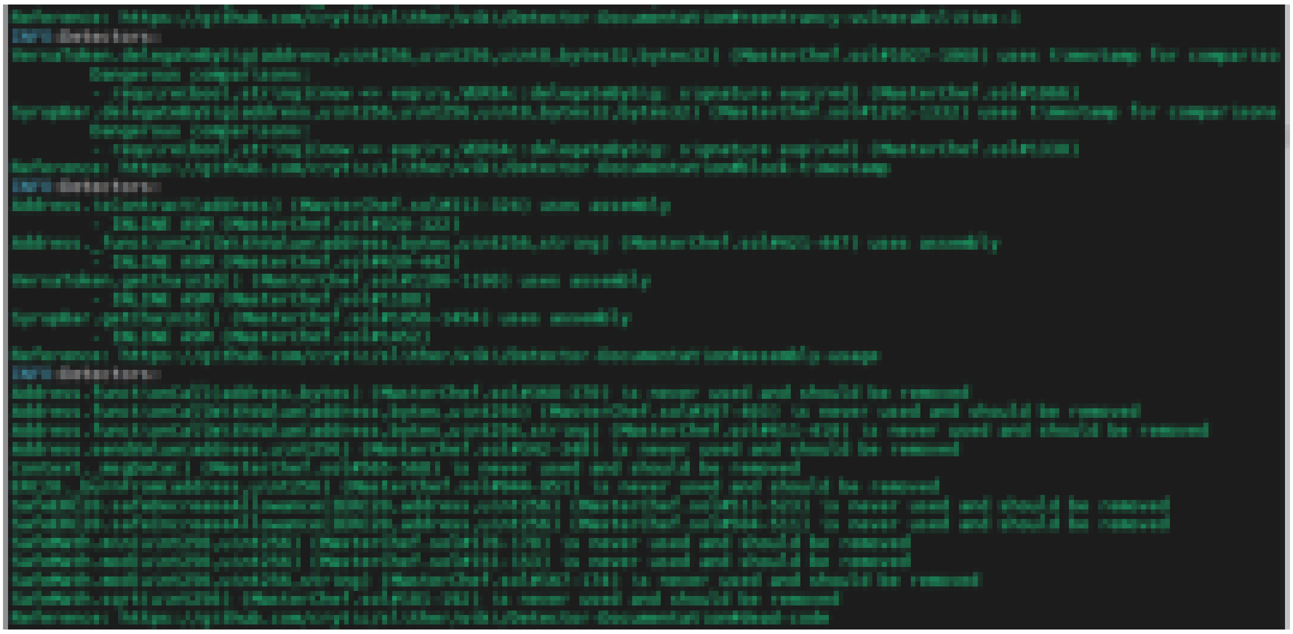


Slither Results Log

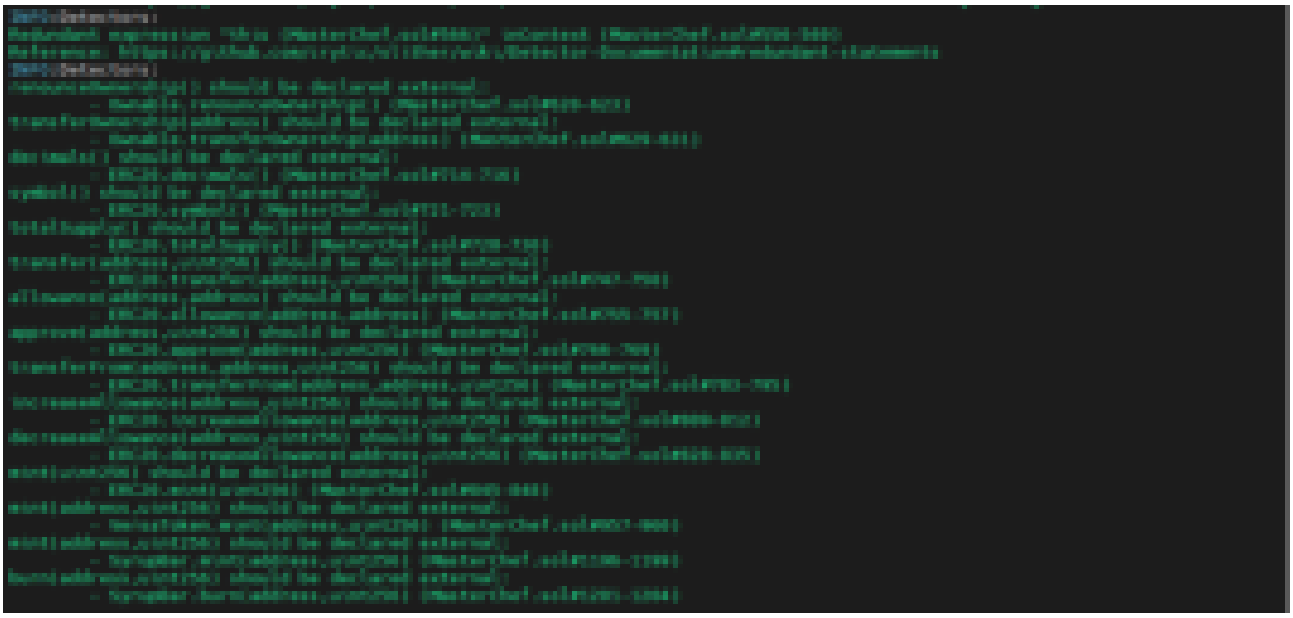
Slither log >> MasterChef.sol



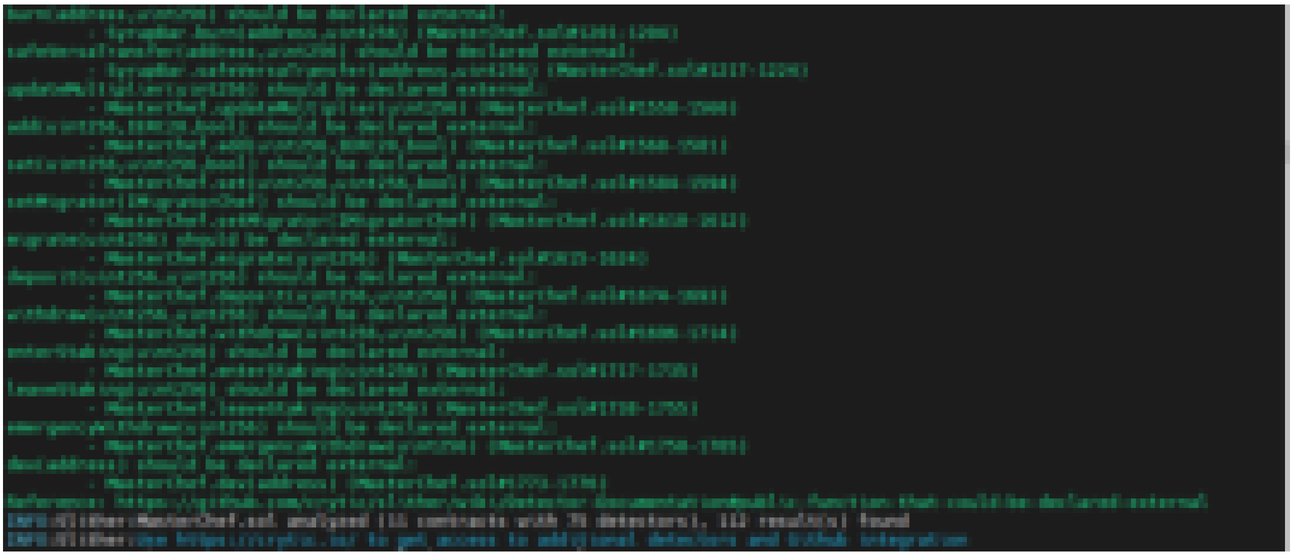
```
INFO:Detectors:
ERC20.constructor(string,string).name (MasterChef.sol#681) shadows:
        - ERC20.name() (MasterChef.sol#697-705) (function)
        - IERC20.name() (MasterChef.sol#315) (function)
ERC20.constructor(string,string).symbol (MasterChef.sol#681) shadows:
        - ERC20.symbol() (MasterChef.sol#711-723) (function)
        - IERC20.symbol() (MasterChef.sol#320) (function)
ERC20.allowance(address,address).owner (MasterChef.sol#750) shadows:
        - Ownable.owner() (MasterChef.sol#580-582) (function)
ERC20._approve(address,address,uint256).owner (MasterChef.sol#937) shadows:
        - Ownable.owner() (MasterChef.sol#580-582) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
MasterChef.constructor(VersaToken,SyrupBar,address,uint256,uint256)._devaddr (MasterChef.sol#1530) lacks a zero-check on :
                - devaddr = _devaddr (MasterChef.sol#1542)
MasterChef.dev(address)._devaddr (MasterChef.sol#1773) lacks a zero-check on :
                - devaddr = _devaddr (MasterChef.sol#1775)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in MasterChef.deposit(uint256,uint256) (MasterChef.sol#1676-1693):
        External calls:
        - updatePool(_pid) (MasterChef.sol#1680)
                - versa.mint(devaddr,versaReward.div(10)) (MasterChef.sol#1667)
                - versa.mint(address(syrup),versaReward) (MasterChef.sol#1668)
        - safeVersaTransfer(msg.sender,pending) (MasterChef.sol#1684)
                - syrup.safeVersaTransfer(_to,_amount) (MasterChef.sol#1769)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterChef.sol#1688)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (MasterChef.sol#1692)
Reentrancy in MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1758-1765):
```



```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
VersaToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (MasterChef.sol#1027-1068) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,VERSA::delegateBySig: signature expired) (MasterChef.sol#1066)
SyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (MasterChef.sol#1291-1332) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,VERSA::delegateBySig: signature expired) (MasterChef.sol#1330)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (MasterChef.sol#311-320) uses assembly
        - INLINE ASM (MasterChef.sol#318-322)
Address._functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#421-447) uses assembly
        - INLINE ASM (MasterChef.sol#439-442)
VersaToken.getChainId() (MasterChef.sol#1186-1190) uses assembly
        - INLINE ASM (MasterChef.sol#1188)
SyrupBar.getChainId() (MasterChef.sol#1450-1454) uses assembly
        - INLINE ASM (MasterChef.sol#1452)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (MasterChef.sol#368-370) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (MasterChef.sol#397-403) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#411-419) is never used and should be removed
Address.sendValue(address,uint256) (MasterChef.sol#342-348) is never used and should be removed
Context._msgData() (MasterChef.sol#565-568) is never used and should be removed
ERC20._burn(address,uint256) (MasterChef.sol#886-891) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (MasterChef.sol#511-521) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (MasterChef.sol#501-510) is never used and should be removed
SafeMath.min(uint256,uint256) (MasterChef.sol#176-178) is never used and should be removed
SafeMath.mod(uint256,uint256) (MasterChef.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (MasterChef.sol#167-174) is never used and should be removed
SafeMath.sqrt(uint256) (MasterChef.sol#181-192) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

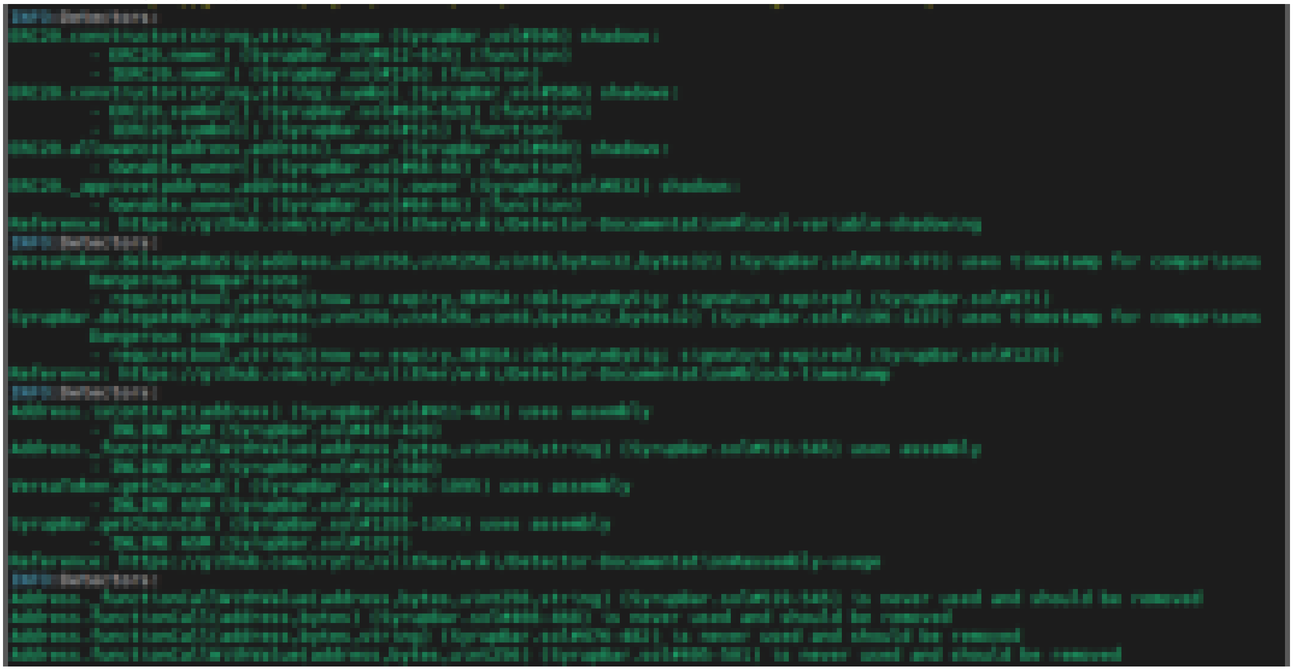


```
INFO:Detectors:
Redundant expression "this (MasterChef.sol#566)" inContext (MasterChef.sol#556-569)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MasterChef.sol#619-622)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MasterChef.sol#628-631)
decimals() should be declared external:
        - ERC20.decimals() (MasterChef.sol#718-720)
symbol() should be declared external:
        - ERC20.symbol() (MasterChef.sol#711-713)
totalSupply() should be declared external:
        - ERC20.totalSupply() (MasterChef.sol#725-727)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (MasterChef.sol#747-750)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (MasterChef.sol#755-757)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (MasterChef.sol#766-769)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (MasterChef.sol#783-791)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (MasterChef.sol#805-808)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (MasterChef.sol#824-827)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (MasterChef.sol#835-838)
mint(address,uint256) should be declared external:
        - VersaToken.mint(address,uint256) (MasterChef.sol#957-960)
mint(address,uint256) should be declared external:
        - SyrupBar.mint(address,uint256) (MasterChef.sol#1196-1199)
burn(address,uint256) should be declared external:
        - SyrupBar.burn(address,uint256) (MasterChef.sol#1201-1204)
```

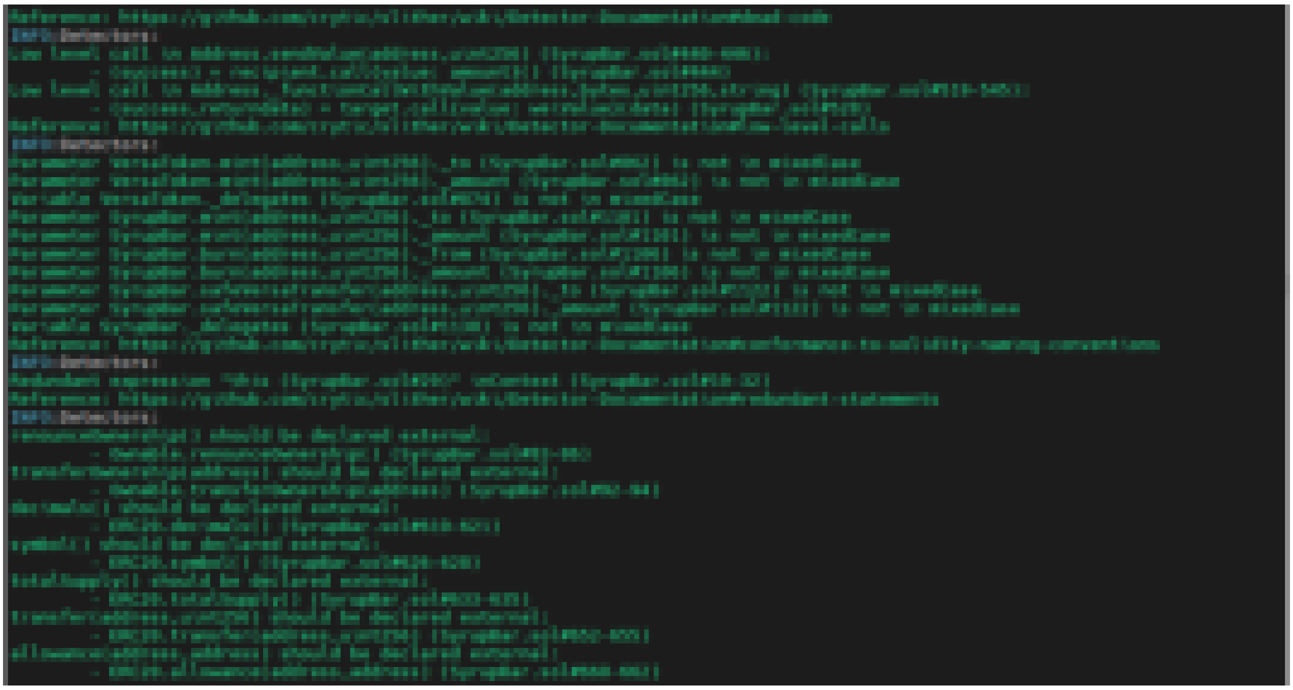


```
burn(address,uint256) should be declared external:
	- SyrupBar.burn(address,uint256) (MasterChef.sol#1201-1206)
safeVersaTransfer(address,uint256) should be declared external:
	- SyrupBar.safeVersaTransfer(address,uint256) (MasterChef.sol#1217-1226)
updateMultiplier(uint256) should be declared external:
	- MasterChef.updateMultiplier(uint256) (MasterChef.sol#1558-1560)
add(uint256,IBEP20,bool) should be declared external:
	- MasterChef.add(uint256,IBEP20,bool) (MasterChef.sol#1568-1581)
set(uint256,uint256,bool) should be declared external:
	- MasterChef.set(uint256,uint256,bool) (MasterChef.sol#1588-1594)
setMigrator(IMigratorChef) should be declared external:
	- MasterChef.setMigrator(IMigratorChef) (MasterChef.sol#1610-1612)
migrate(uint256) should be declared external:
	- MasterChef.migrate(uint256) (MasterChef.sol#1615-1624)
deposit(uint256,uint256) should be declared external:
	- MasterChef.deposit(uint256,uint256) (MasterChef.sol#1676-1693)
withdraw(uint256,uint256) should be declared external:
	- MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1696-1714)
enterStaking(uint256) should be declared external:
	- MasterChef.enterStaking(uint256) (MasterChef.sol#1717-1735)
leaveStaking(uint256) should be declared external:
	- MasterChef.leaveStaking(uint256) (MasterChef.sol#1738-1755)
emergencyWithdraw(uint256) should be declared external:
	- MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1758-1765)
dev(address) should be declared external:
	- MasterChef.dev(address) (MasterChef.sol#1770-1774)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MasterChef.sol analyzed (11 contracts with 75 detectors), 112 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

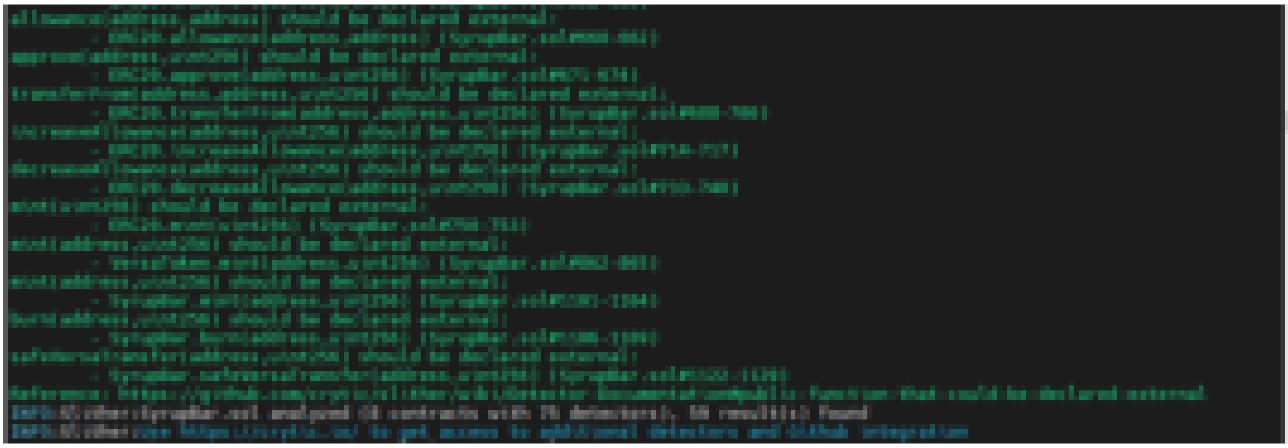
Slither log >> SyrupBar.sol



```
INFO:Detectors:
ERC20.constructor(string,string).name (SyrupBar.sol#598) shadows:
	- ERC20.name() (SyrupBar.sol#611-613) (function)
	- IBEP20.name() (SyrupBar.sol#128) (function)
ERC20.constructor(string,string).symbol (SyrupBar.sol#598) shadows:
	- ERC20.symbol() (SyrupBar.sol#626-628) (function)
	- IBEP20.symbol() (SyrupBar.sol#121) (function)
ERC20.allowance(address,address).owner (SyrupBar.sol#686) shadows:
	- Ownable.owner() (SyrupBar.sol#56-58) (function)
ERC20._approve(address,address,uint256).owner (SyrupBar.sol#832) shadows:
	- Ownable.owner() (SyrupBar.sol#56-58) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
VersaToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#932-973) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(now <= expiry,VERSA::delegateBySig: signature expired) (SyrupBar.sol#971)
SyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#1190-1231) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(now <= expiry,VERSA::delegateBySig: signature expired) (SyrupBar.sol#1229)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (SyrupBar.sol#415-422) uses assembly
	- INLINE ASM (SyrupBar.sol#418-420)
Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#519-545) uses assembly
	- INLINE ASM (SyrupBar.sol#537-540)
VersaToken.getChainId() (SyrupBar.sol#1081-1085) uses assembly
	- INLINE ASM (SyrupBar.sol#1083)
SyrupBar.getChainId() (SyrupBar.sol#1333-1339) uses assembly
	- INLINE ASM (SyrupBar.sol#1337)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#519-545) is never used and should be removed
Address.functionCall(address,bytes) (SyrupBar.sol#466-468) is never used and should be removed
Address.functionCall(address,bytes,string) (SyrupBar.sol#476-482) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (SyrupBar.sol#495-501) is never used and should be removed
```

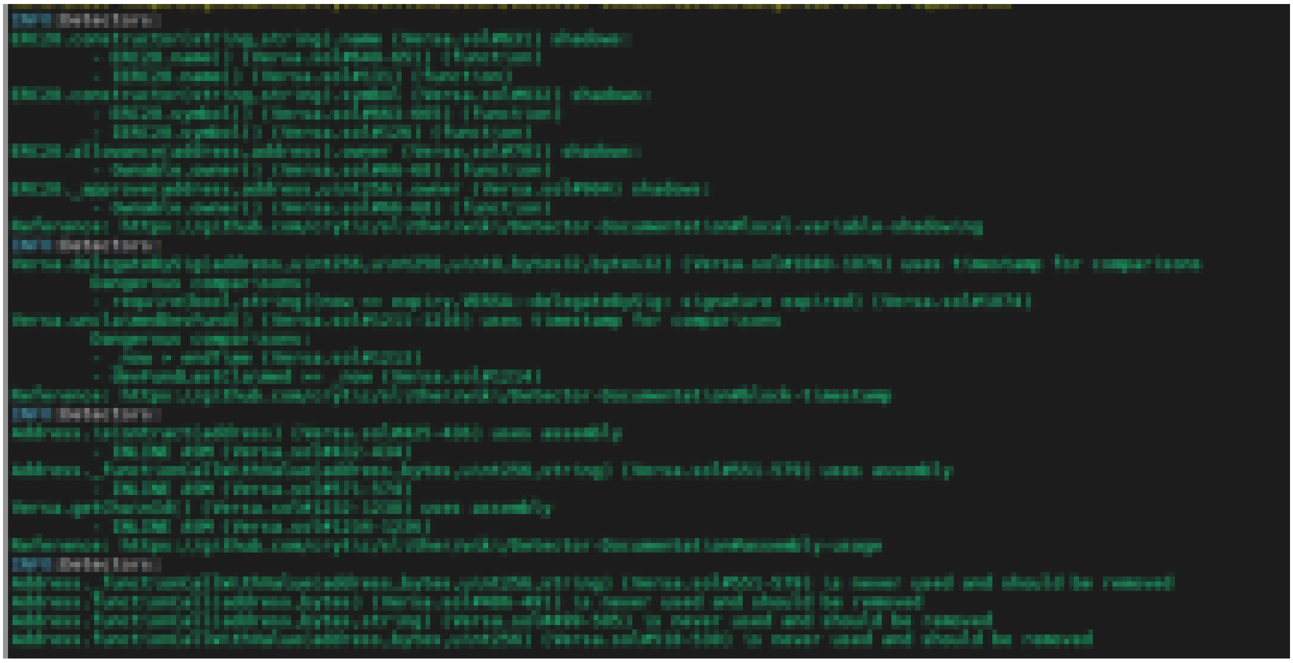


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (SyrupBar.sol#440-446):
	- (success) = recipient.call{value: amount}() (SyrupBar.sol#444)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#519-545):
	- (success,returndata) = target.call{value: weiValue}(data) (SyrupBar.sol#528)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter VersaToken.mint(address,uint256)._to (SyrupBar.sol#882) is not in mixedCase
Parameter VersaToken.mint(address,uint256)._amount (SyrupBar.sol#882) is not in mixedCase
Variable VersaToken._delegates (SyrupBar.sol#874) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._to (SyrupBar.sol#1120) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._amount (SyrupBar.sol#1120) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._from (SyrupBar.sol#1125) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._amount (SyrupBar.sol#1125) is not in mixedCase
Parameter SyrupBar.safeVersaTransfer(address,uint256)._to (SyrupBar.sol#1132) is not in mixedCase
Parameter SyrupBar.safeVersaTransfer(address,uint256)._amount (SyrupBar.sol#1132) is not in mixedCase
Variable SyrupBar._delegates (SyrupBar.sol#1138) 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#29)" inContext (SyrupBar.sol#18-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
	- Ownable.renounceOwnership() (SyrupBar.sol#82-85)
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (SyrupBar.sol#92-94)
decimals() should be declared external:
	- ERC20.decimals() (SyrupBar.sol#618-620)
symbol() should be declared external:
	- ERC20.symbol() (SyrupBar.sol#626-628)
totalSupply() should be declared external:
	- ERC20.totalSupply() (SyrupBar.sol#633-635)
transfer(address,uint256) should be declared external:
	- ERC20.transfer(address,uint256) (SyrupBar.sol#652-655)
allowance(address,address) should be declared external:
	- ERC20.allowance(address,address) (SyrupBar.sol#660-662)
```

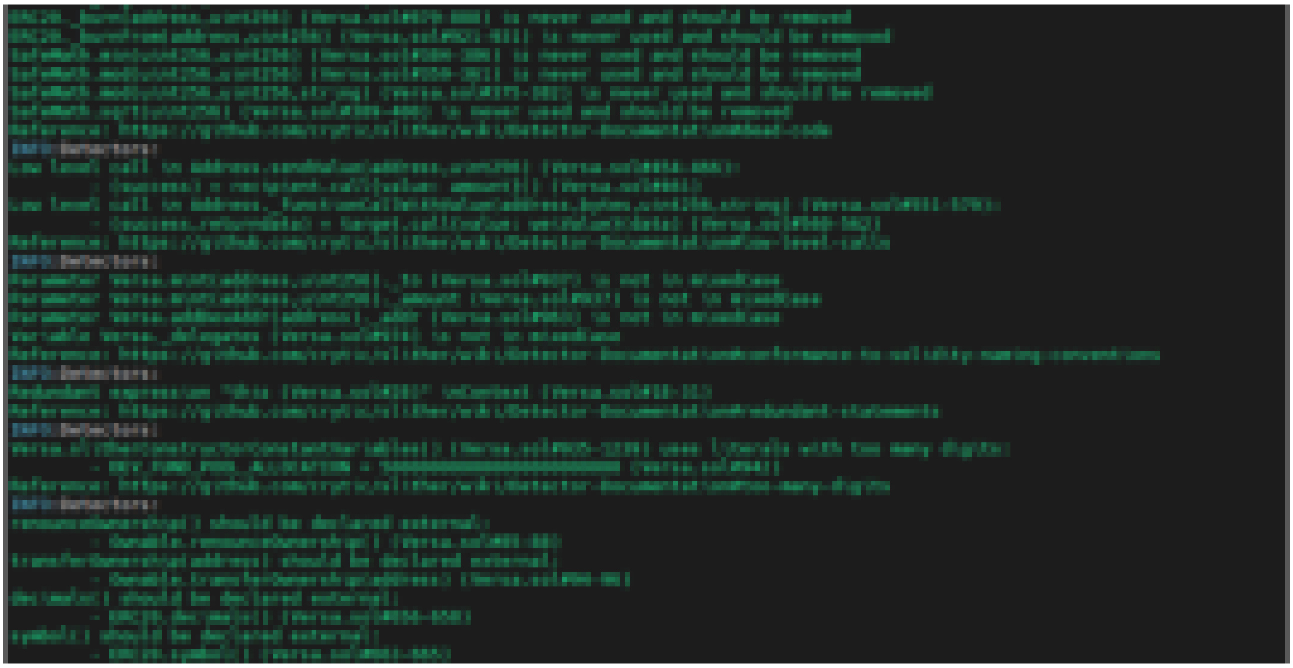


```
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (SyrupBar.sol#660-662)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (SyrupBar.sol#671-674)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (SyrupBar.sol#688-700)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (SyrupBar.sol#714-717)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (SyrupBar.sol#733-740)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (SyrupBar.sol#750-753)
mint(address,uint256) should be declared external:
        - VersaToken.mint(address,uint256) (SyrupBar.sol#882-885)
mint(address,uint256) should be declared external:
        - SyrupBar.mint(address,uint256) (SyrupBar.sol#1101-1104)
burn(address,uint256) should be declared external:
        - SyrupBar.burn(address,uint256) (SyrupBar.sol#1106-1109)
safeVersaTransfer(address,uint256) should be declared external:
        - SyrupBar.safeVersaTransfer(address,uint256) (SyrupBar.sol#1122-1129)
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), 58 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

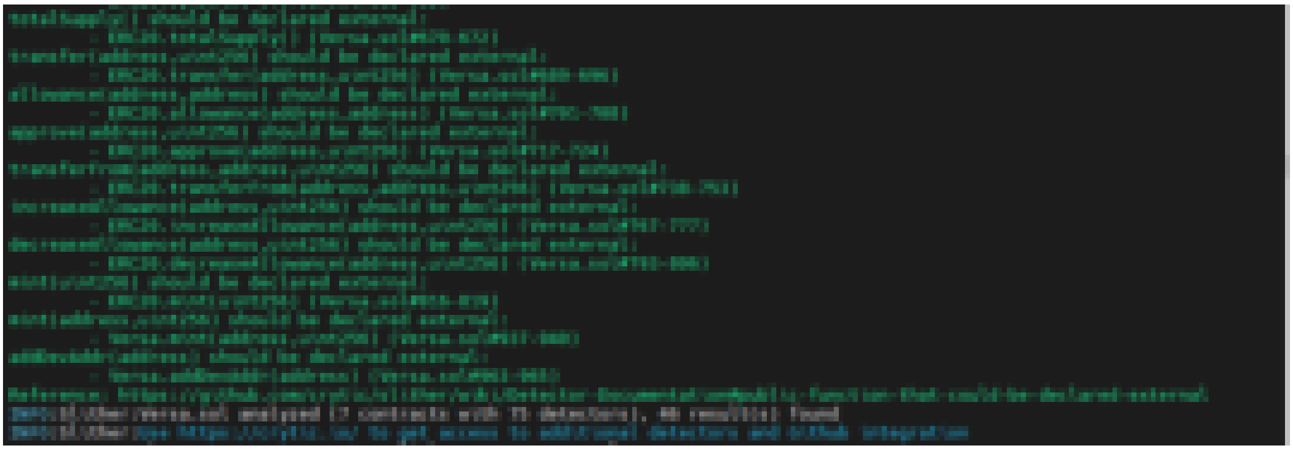
Slither log >> Versa.sol



```
INFO:Detectors:
ERC20.constructor(string,string).name (Versa.sol#631) shadows:
        - ERC20.name() (Versa.sol#640-642) (function)
        - IBEP20.name() (Versa.sol#535) (function)
ERC20.constructor(string,string).symbol (Versa.sol#632) shadows:
        - ERC20.symbol() (Versa.sol#663-665) (function)
        - IBEP20.symbol() (Versa.sol#530) (function)
ERC20.allowance(address,address).owner (Versa.sol#701) shadows:
        - Ownable.owner() (Versa.sol#480-482) (function)
ERC20._approve(address,address,uint256).owner (Versa.sol#904) shadows:
        - Ownable.owner() (Versa.sol#480-482) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Versa.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (Versa.sol#1040-1079) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,VERSA::delegateBySig: signature expired) (Versa.sol#1071)
Versa.unclaimedRefund() (Versa.sol#1211-1218) uses timestamp for comparisons
        Dangerous comparisons:
        - _now > endTime (Versa.sol#1213)
        - RefundLastClaimed >= _now (Versa.sol#1214)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (Versa.sol#425-436) uses assembly
        - INLINE ASM (Versa.sol#432-434)
Address._functionCallWithValue(address,bytes,uint256,string) (Versa.sol#551-578) uses assembly
        - INLINE ASM (Versa.sol#570-573)
Versa.getChainId() (Versa.sol#1222-1230) uses assembly
        - INLINE ASM (Versa.sol#1226-1228)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (Versa.sol#551-578) is never used and should be removed
Address.functionCall(address,bytes) (Versa.sol#488-493) is never used and should be removed
Address.functionCall(address,bytes,string) (Versa.sol#498-505) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Versa.sol#518-530) is never used and should be removed
```

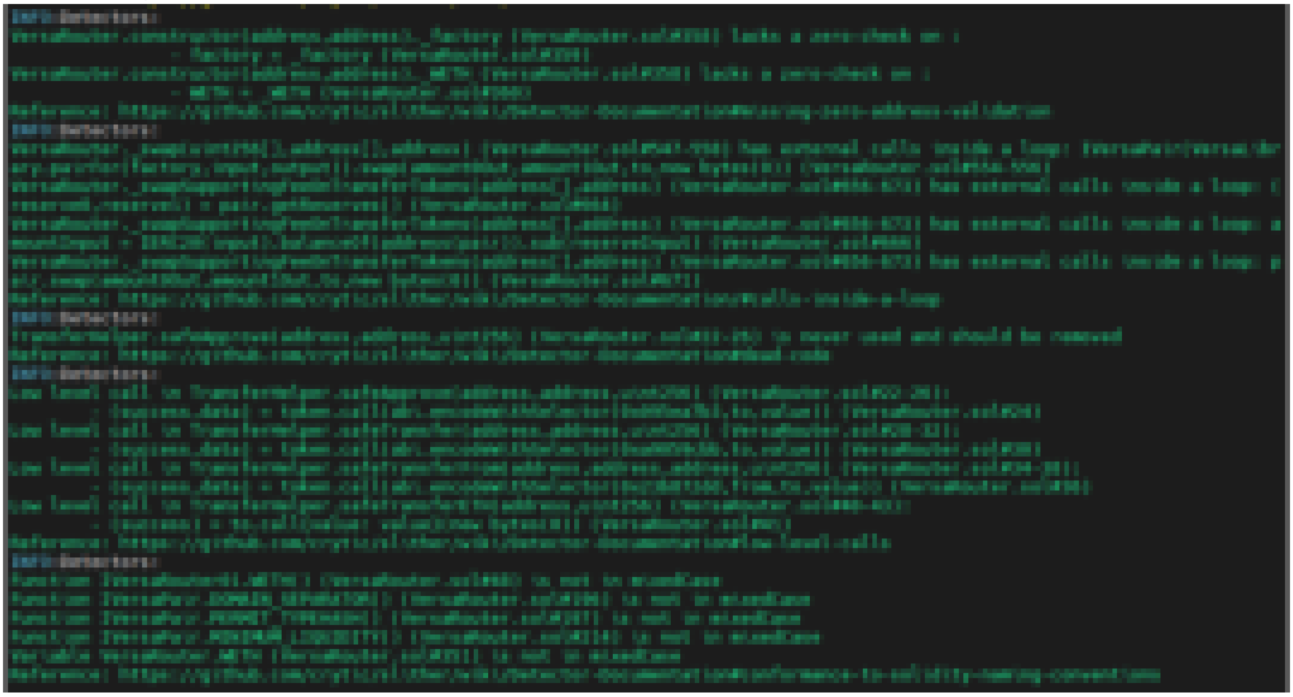


```
ERC20._burn(address,uint256) (Versa.sol#829-838) is never used and should be removed
ERC20._burnFrom(address,uint256) (Versa.sol#921-931) is never used and should be removed
SafeMath.min(uint256,uint256) (Versa.sol#386-388) is never used and should be removed
SafeMath.mod(uint256,uint256) (Versa.sol#350-352) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Versa.sol#375-382) is never used and should be removed
SafeMath.sqrt(uint256) (Versa.sol#389-400) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Versa.sol#454-460):
        - (success) = recipient.call{value: amount}() (Versa.sol#458)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (Versa.sol#551-578):
        - (success,returndata) = target.call{value: weiValue}(data) (Versa.sol#560-562)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Versa.mint(address,uint256)._to (Versa.sol#937) is not in mixedCase
Parameter Versa.mint(address,uint256)._amount (Versa.sol#937) is not in mixedCase
Parameter Versa.addBookAddr(address)._addr (Versa.sol#963) is not in mixedCase
Variable Versa._delegates (Versa.sol#951) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Versa.sol#101)" inContext (Versa.sol#95-103)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Versa.slitherConstructorConstantVariables() (Versa.sol#925-1230) uses literals with too many digits:
        - DEV_FUND_POOL_ALLOCATION = 5000000000000000000000000 (Versa.sol#942)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Versa.sol#499-502)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Versa.sol#508-510)
decimals() should be declared external:
        - ERC20.decimals() (Versa.sol#656-658)
symbol() should be declared external:
        - ERC20.symbol() (Versa.sol#663-665)
```

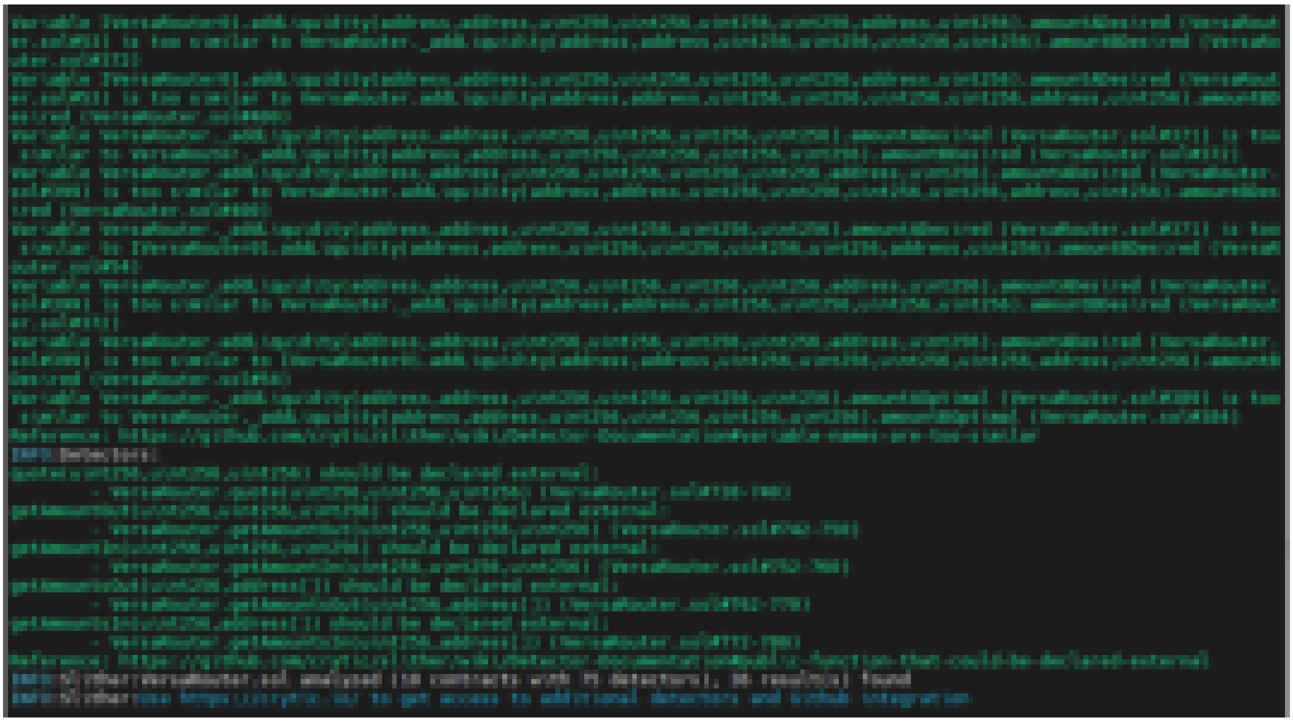


```
totalSupply() should be declared external:
        - ERC20.totalSupply() (Versa.sol#[illegible])
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (Versa.sol#[illegible])
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (Versa.sol#[illegible])
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (Versa.sol#[illegible])
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (Versa.sol#[illegible])
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (Versa.sol#[illegible])
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (Versa.sol#[illegible])
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (Versa.sol#[illegible])
mint(address,uint256) should be declared external:
        - Versa.mint(address,uint256) (Versa.sol#[illegible])
addDevAddr(address) should be declared external:
        - Versa.addDevAddr(address) (Versa.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Versa.sol analyzed (7 contracts with 75 detectors), 48 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> VersaRouter.sol

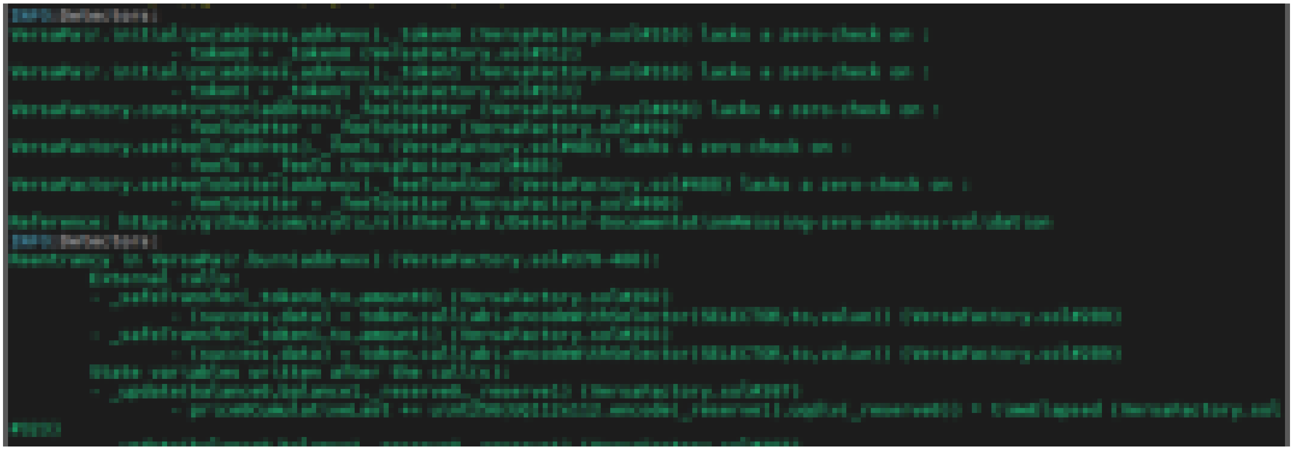


```
INFO:Detectors:
VersaRouter.constructor(address,address)._factory (VersaRouter.sol#[illegible]) lacks a zero-check on :
                - factory = _factory (VersaRouter.sol#[illegible])
VersaRouter.constructor(address,address)._WETH (VersaRouter.sol#[illegible]) lacks a zero-check on :
                - WETH = _WETH (VersaRouter.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
VersaRouter._swap(uint256[],address[],address) (VersaRouter.sol#[illegible]) has external calls inside a loop: IVersaPair(VersaLibr
ary.pairFor(factory,input,output)).swap(amount0Out,amount1Out,to,new bytes(0)) (VersaRouter.sol#[illegible])
VersaRouter._swapSupportingFeeOnTransferTokens(address[],address) (VersaRouter.sol#[illegible]) has external calls inside a loop: (
reserve0,reserve1) = pair.getReserves() (VersaRouter.sol#[illegible])
VersaRouter._swapSupportingFeeOnTransferTokens(address[],address) (VersaRouter.sol#[illegible]) has external calls inside a loop: a
mountInput = IERC20(input).balanceOf(address(pair)).sub(reserveInput) (VersaRouter.sol#[illegible])
VersaRouter._swapSupportingFeeOnTransferTokens(address[],address) (VersaRouter.sol#[illegible]) has external calls inside a loop: p
air.swap(amount0Out,amount1Out,to,new bytes(0)) (VersaRouter.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
TransferHelper.safeApprove(address,address,uint256) (VersaRouter.sol#[illegible]) 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) (VersaRouter.sol#[illegible]):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (VersaRouter.sol#[illegible])
Low level call in TransferHelper.safeTransfer(address,address,uint256) (VersaRouter.sol#[illegible]):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (VersaRouter.sol#[illegible])
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (VersaRouter.sol#[illegible]):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (VersaRouter.sol#[illegible])
Low level call in TransferHelper.safeTransferETH(address,uint256) (VersaRouter.sol#[illegible]):
        - (success) = to.call{value: value}(new bytes(0)) (VersaRouter.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IVersaRouter01.WETH() (VersaRouter.sol#[illegible]) is not in mixedCase
Function IVersaPair.DOMAIN_SEPARATOR() (VersaRouter.sol#[illegible]) is not in mixedCase
Function IVersaPair.PERMIT_TYPEHASH() (VersaRouter.sol#[illegible]) is not in mixedCase
Function IVersaPair.MINIMUM_LIQUIDITY() (VersaRouter.sol#[illegible]) is not in mixedCase
Variable VersaRouter.WETH (VersaRouter.sol#[illegible]) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

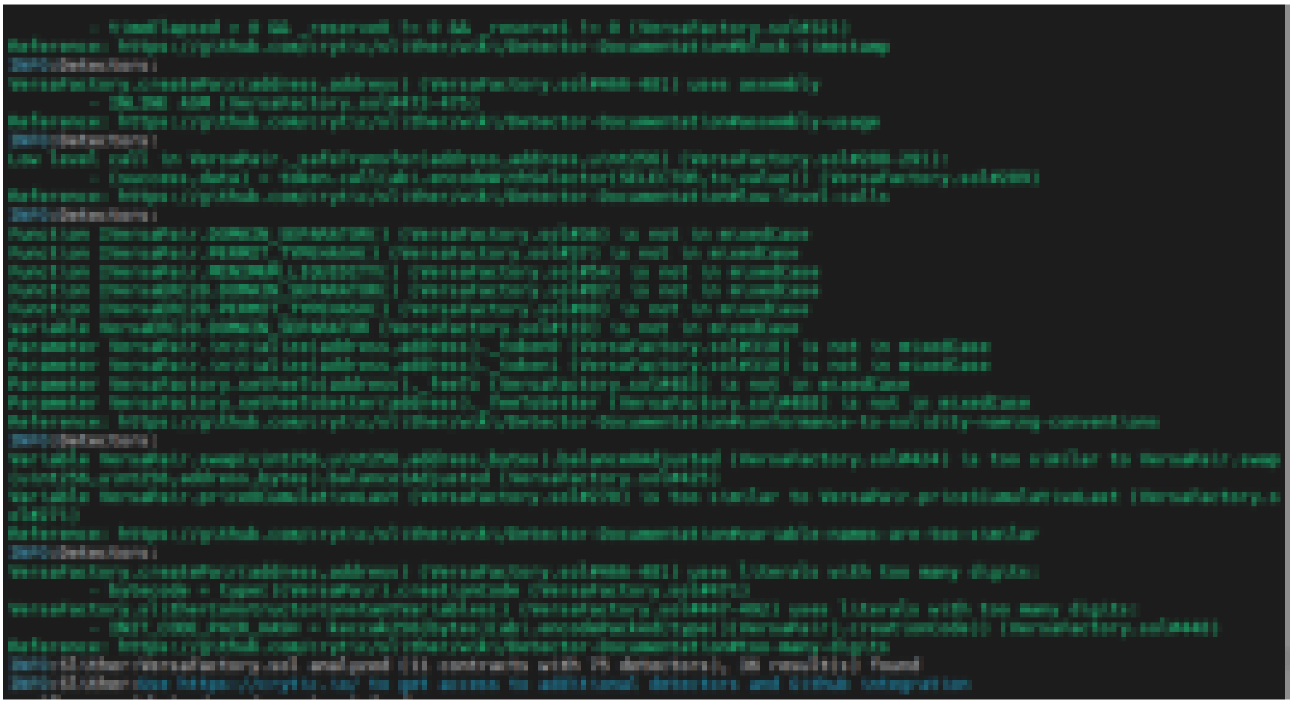


```
Variable IVersaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (VersaRout
er.sol#[illegible]) is too similar to VersaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (VersaRo
uter.sol#[illegible])
Variable IVersaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (VersaRout
er.sol#[illegible]) is too similar to VersaRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBD
esired (VersaRouter.sol#[illegible])
Variable VersaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (VersaRouter.sol#[illegible]) is too
 similar to VersaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (VersaRouter.sol#[illegible])
Variable VersaRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (VersaRouter.
sol#[illegible]) is too similar to VersaRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDes
ired (VersaRouter.sol#[illegible])
Variable VersaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (VersaRouter.sol#[illegible]) is too
 similar to IVersaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (VersaR
outer.sol#[illegible])
Variable VersaRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (VersaRouter.
sol#[illegible]) is too similar to VersaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (VersaRout
er.sol#[illegible])
Variable VersaRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (VersaRouter.
sol#[illegible]) is too similar to IVersaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountB
Desired (VersaRouter.sol#[illegible])
Variable VersaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (VersaRouter.sol#[illegible]) is too
 similar to VersaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (VersaRouter.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
quote(uint256,uint256,uint256) should be declared external:
        - VersaRouter.quote(uint256,uint256,uint256) (VersaRouter.sol#[illegible])
getAmountOut(uint256,uint256,uint256) should be declared external:
        - VersaRouter.getAmountOut(uint256,uint256,uint256) (VersaRouter.sol#[illegible])
getAmountIn(uint256,uint256,uint256) should be declared external:
        - VersaRouter.getAmountIn(uint256,uint256,uint256) (VersaRouter.sol#[illegible])
getAmountsOut(uint256,address[]) should be declared external:
        - VersaRouter.getAmountsOut(uint256,address[]) (VersaRouter.sol#[illegible])
getAmountsIn(uint256,address[]) should be declared external:
        - VersaRouter.getAmountsIn(uint256,address[]) (VersaRouter.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:VersaRouter.sol analyzed (10 contracts with 75 detectors), 36 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> VersaFactory.sol

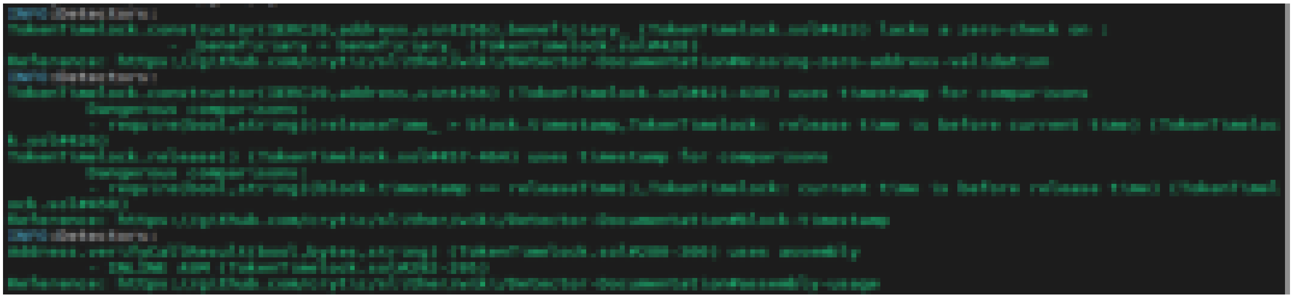


```
INFO:Detectors:
VersaPair.initialize(address,address)._token0 (VersaFactory.sol#310) lacks a zero-check on :
                - token0 = _token0 (VersaFactory.sol#312)
VersaPair.initialize(address,address)._token1 (VersaFactory.sol#310) lacks a zero-check on :
                - token1 = _token1 (VersaFactory.sol#313)
VersaFactory.constructor(address)._feeToSetter (VersaFactory.sol#460) lacks a zero-check on :
                - feeToSetter = _feeToSetter (VersaFactory.sol#461)
VersaFactory.setFeeTo(address)._feeTo (VersaFactory.sol#483) lacks a zero-check on :
                - feeTo = _feeTo (VersaFactory.sol#485)
VersaFactory.setFeeToSetter(address)._feeToSetter (VersaFactory.sol#488) lacks a zero-check on :
                - feeToSetter = _feeToSetter (VersaFactory.sol#490)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in VersaPair.burn(address) (VersaFactory.sol#378-400):
        External calls:
        - _safeTransfer(_token0,to,amount0) (VersaFactory.sol#392)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (VersaFactory.sol#280)
        - _safeTransfer(_token1,to,amount1) (VersaFactory.sol#393)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (VersaFactory.sol#280)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (VersaFactory.sol#397)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (VersaFactory.sol
#322)
        - _update(balance0,balance1,_reserve0,_reserve1) (VersaFactory.sol#397)
```

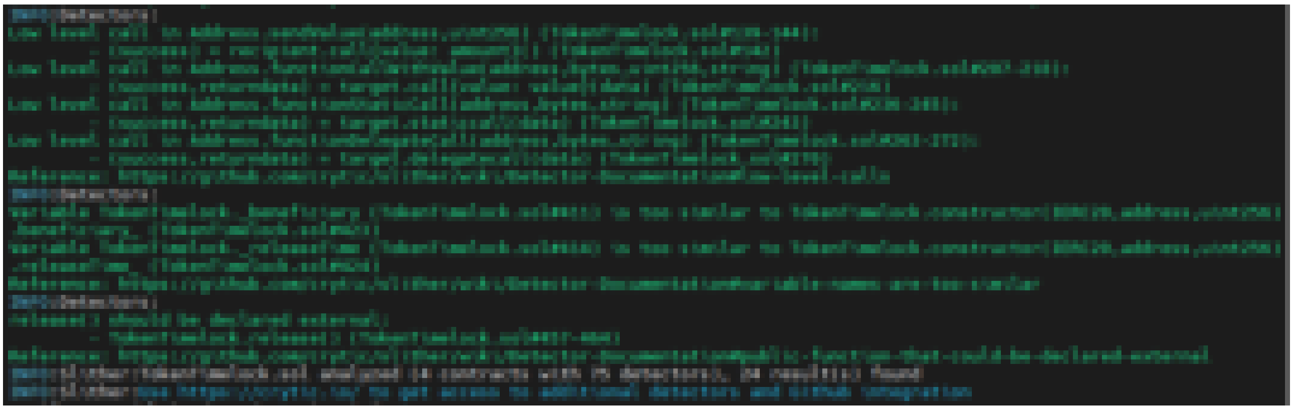


```
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (VersaFactory.sol#321)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
VersaFactory.createPair(address,address) (VersaFactory.sol#466-481) uses assembly
        - INLINE ASM (VersaFactory.sol#472-475)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Low level call in VersaPair._safeTransfer(address,address,uint256) (VersaFactory.sol#279-282):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (VersaFactory.sol#280)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IVersaPair.DOMAIN_SEPARATOR() (VersaFactory.sol#36) is not in mixedCase
Function IVersaPair.PERMIT_TYPEHASH() (VersaFactory.sol#37) is not in mixedCase
Function IVersaPair.MINIMUM_LIQUIDITY() (VersaFactory.sol#54) is not in mixedCase
Function IVersaERC20.DOMAIN_SEPARATOR() (VersaFactory.sol#97) is not in mixedCase
Function IVersaERC20.PERMIT_TYPEHASH() (VersaFactory.sol#98) is not in mixedCase
Variable VersaERC20.DOMAIN_SEPARATOR (VersaFactory.sol#178) is not in mixedCase
Parameter VersaPair.initialize(address,address)._token0 (VersaFactory.sol#310) is not in mixedCase
Parameter VersaPair.initialize(address,address)._token1 (VersaFactory.sol#310) is not in mixedCase
Parameter VersaFactory.setFeeTo(address)._feeTo (VersaFactory.sol#483) is not in mixedCase
Parameter VersaFactory.setFeeToSetter(address)._feeToSetter (VersaFactory.sol#488) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable VersaPair.swap(uint256,uint256,address,bytes).balance0Adjusted (VersaFactory.sol#424) is too similar to VersaPair.swap
(uint256,uint256,address,bytes).balance1Adjusted (VersaFactory.sol#425)
Variable VersaPair.price0CumulativeLast (VersaFactory.sol#270) is too similar to VersaPair.price1CumulativeLast (VersaFactory.s
ol#271)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
VersaFactory.createPair(address,address) (VersaFactory.sol#466-481) uses literals with too many digits:
        - bytecode = type()(VersaPair).creationCode (VersaFactory.sol#471)
VersaFactory.slitherConstructorConstantVariables() (VersaFactory.sol#447-492) uses literals with too many digits:
        - INIT_CODE_PAIR_HASH = keccak256(bytes)(abi.encodePacked(type()(VersaPair).creationCode)) (VersaFactory.sol#448)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Slither:VersaFactory.sol analyzed (11 contracts with 75 detectors), 36 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> TokenTimelock.sol



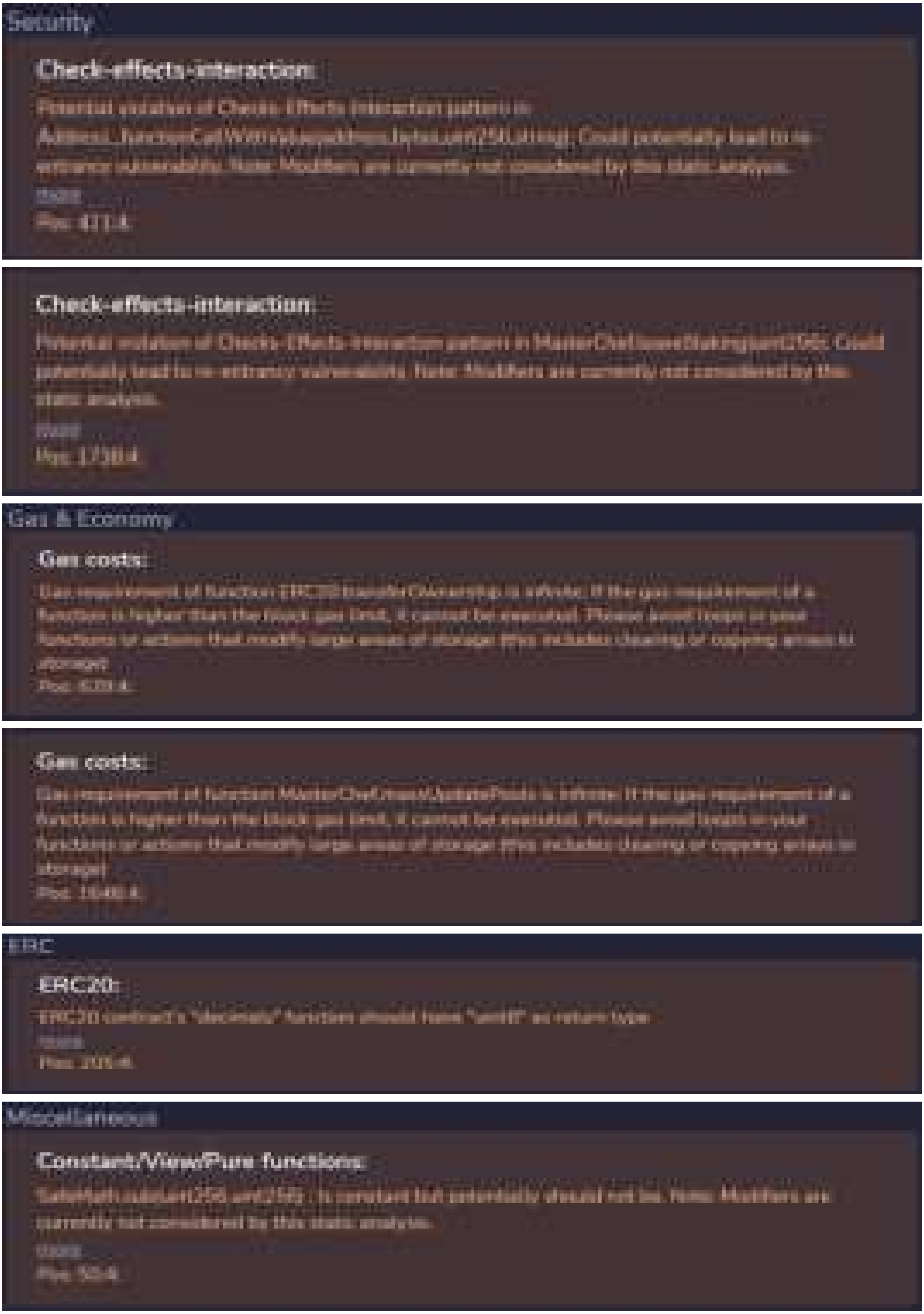
```
INFO:Detectors:
TokenTimelock.constructor(IERC20,address,uint256).beneficiary_ (TokenTimelock.sol#422) lacks a zero-check on :
                - _beneficiary = beneficiary_ (TokenTimelock.sol#428)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
TokenTimelock.constructor(IERC20,address,uint256) (TokenTimelock.sol#421-430) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(releaseTime_ > block.timestamp,TokenTimelock: release time is before current time) (TokenTimeloc
k.sol#426)
TokenTimelock.release() (TokenTimelock.sol#457-464) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= releaseTime(),TokenTimelock: current time is before release time) (TokenTimel
ock.sol#458)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (TokenTimelock.sol#280-300) uses assembly
        - INLINE ASM (TokenTimelock.sol#292-295)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```



```
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (TokenTimelock.sol#136-141):
        - (success) = recipient.call{value: amount}() (TokenTimelock.sol#139)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (TokenTimelock.sol#207-218):
        - (success,returndata) = target.call{value: value}(data) (TokenTimelock.sol#216)
Low level call in Address.functionStaticCall(address,bytes,string) (TokenTimelock.sol#236-245):
        - (success,returndata) = target.staticcall(data) (TokenTimelock.sol#243)
Low level call in Address.functionDelegateCall(address,bytes,string) (TokenTimelock.sol#263-272):
        - (success,returndata) = target.delegatecall(data) (TokenTimelock.sol#270)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable TokenTimelock._beneficiary (TokenTimelock.sol#411) is too similar to TokenTimelock.constructor(IERC20,address,uint256)
.beneficiary_ (TokenTimelock.sol#422)
Variable TokenTimelock._releaseTime (TokenTimelock.sol#414) is too similar to TokenTimelock.constructor(IERC20,address,uint256)
.releaseTime_ (TokenTimelock.sol#424)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
release() should be declared external:
        - TokenTimelock.release() (TokenTimelock.sol#457-464)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:TokenTimelock.sol analyzed (4 contracts with 75 detectors), 24 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

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: 471:4:

Check-effects-interaction:

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

more

Pos: 1738:4:

Gas & Economy

Gas costs:

Gas requirement of function ERC20.transferOwnership 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: 620:4:

Gas costs:

Gas requirement of function MasterChef.massUpdatePools 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: 1640:4:

ERC

ERC20:

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

more

Pos: 205:4:

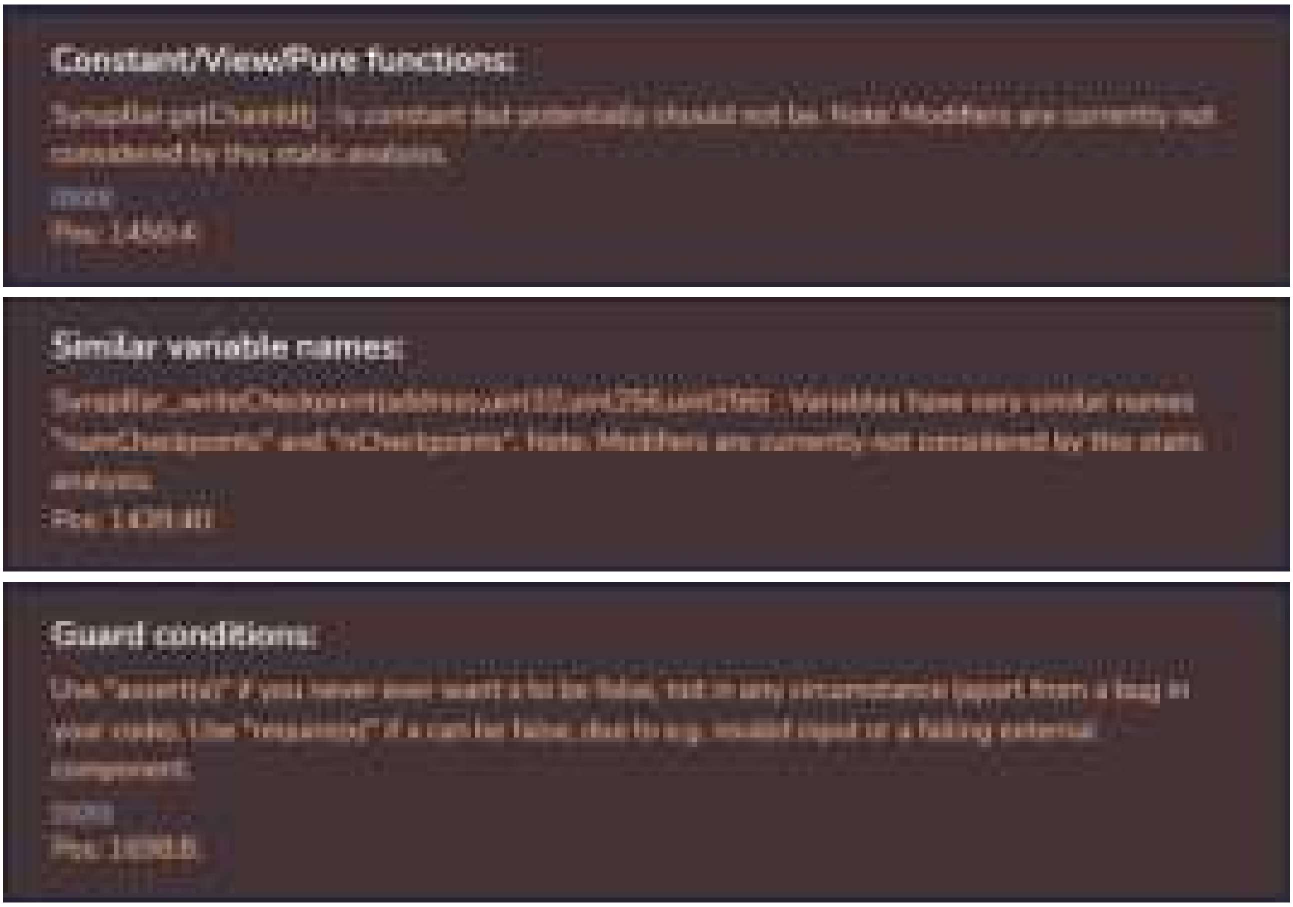
Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256,string) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 50:4:



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: 1450:4

Similar variable names:

SyrupBar._writeCheckpoint(address,uint32,uint256,uint256) : Variables have very similar names "numCheckpoints" and "nCheckpoints". Note: Modifiers are currently not considered by this static analysis.

Pos: 1409:40

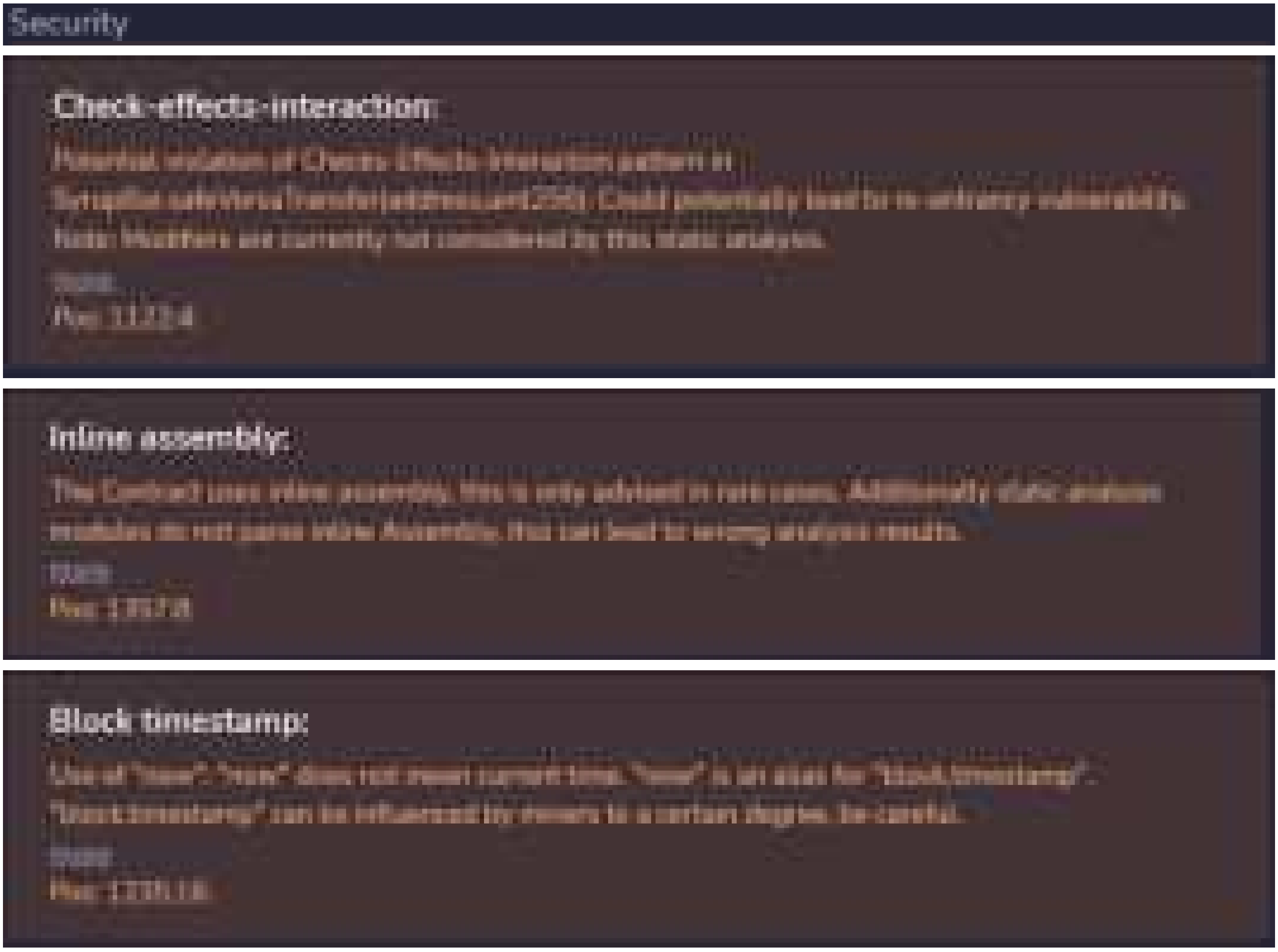
Guard conditions:

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: 1698:6

SyrupBar.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SyrupBar.safeVenusTransfer(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

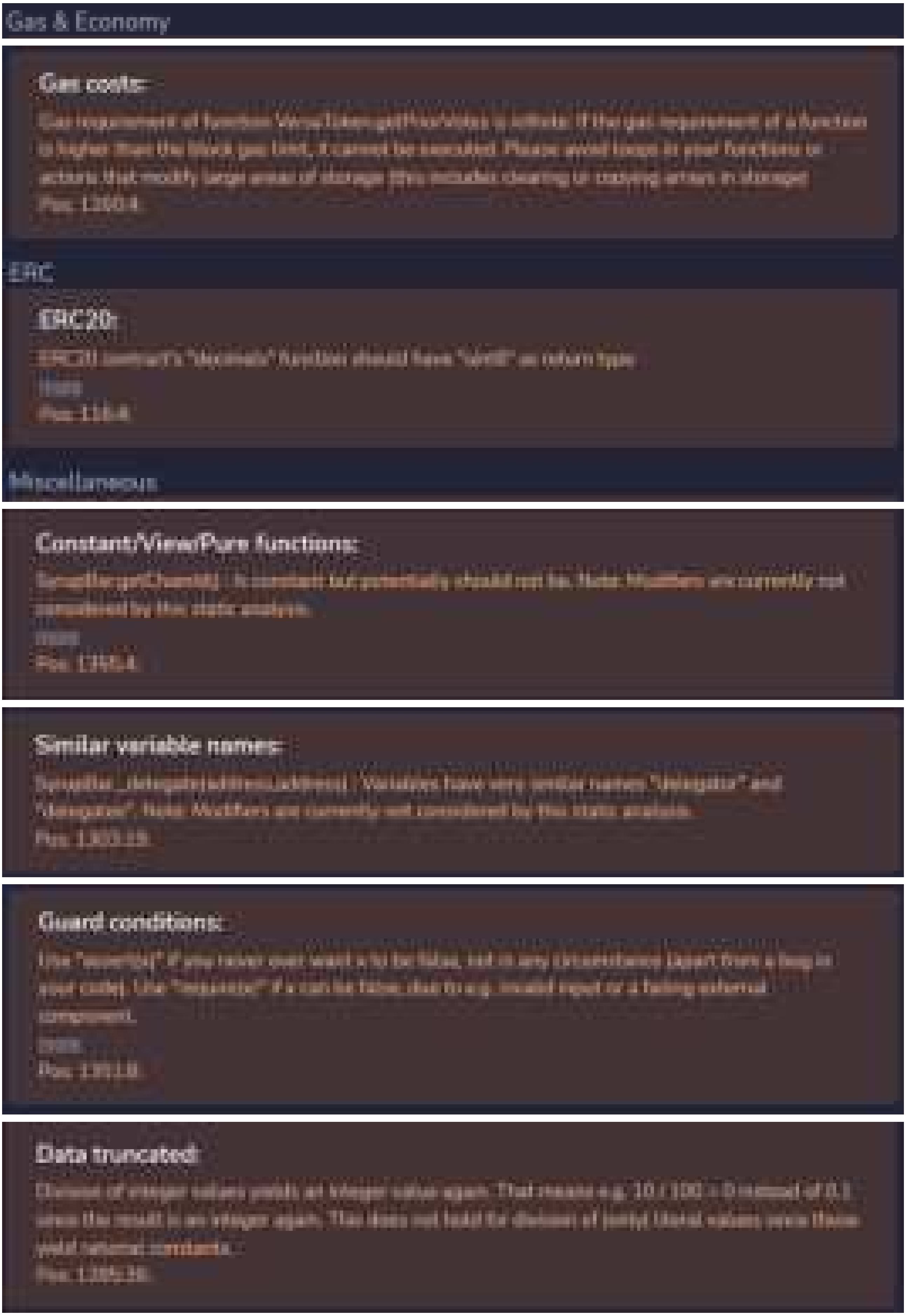
Pos: 1357:8

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: 1335:16



Gas & Economy

Gas costs:

Gas requirement of function VersaToken.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: 1260:4:

ERC

ERC20:

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

more

Pos: 116:4:

Miscellaneous

Constant/View/Pure functions:

SynapseToken.getChainId() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1345:4:

Similar variable names:

SynapseToken._delegate(address,address) : Variables have very similar names "delegator" and "delegatee". Note: Modifiers are currently not considered by this static analysis.

Pos: 1303:19:

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

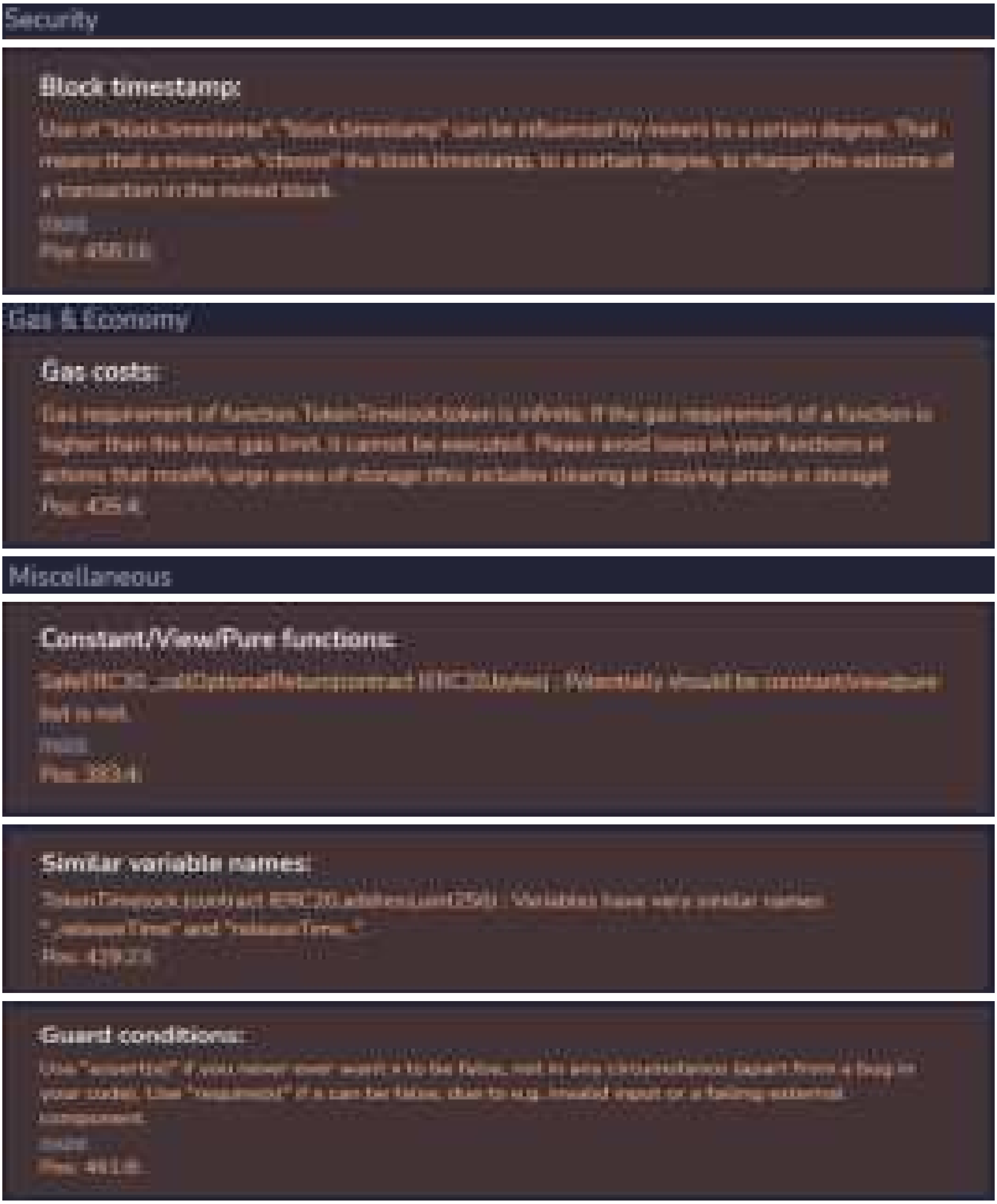
Pos: 1311:8:

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: 1285:36:

TokenTimelock.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: 458:16:

Gas & Economy

Gas costs:

Gas requirement of function TokenTimelock.token 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: 428:4:

Miscellaneous

Constant/View/Pure functions:

SafeERC20._callOptionalReturn(contract IERC20,bytes) : Potentially should be constant/view/pure but is not.

more

Pos: 393:4:

Similar variable names:

TokenTimelock.(contract IERC20,address,uint256) : Variables have very similar names "_releaseTime" and "releaseTime_".

Pos: 429:23:

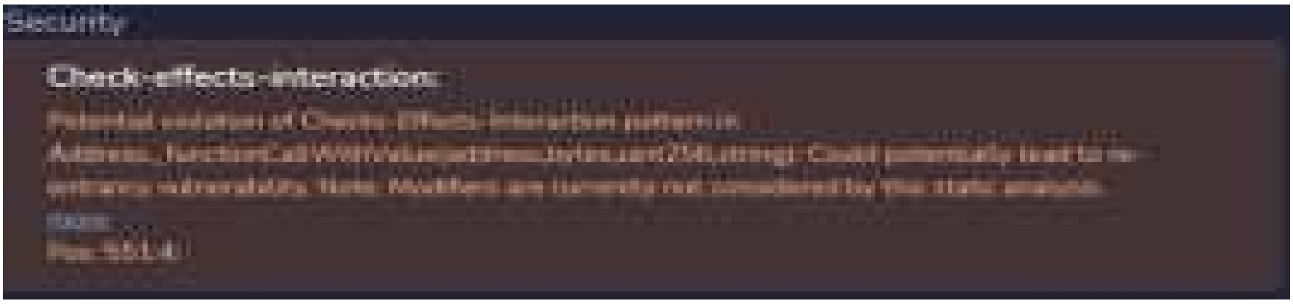
Guard conditions:

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: 461:8:

Versa.sol



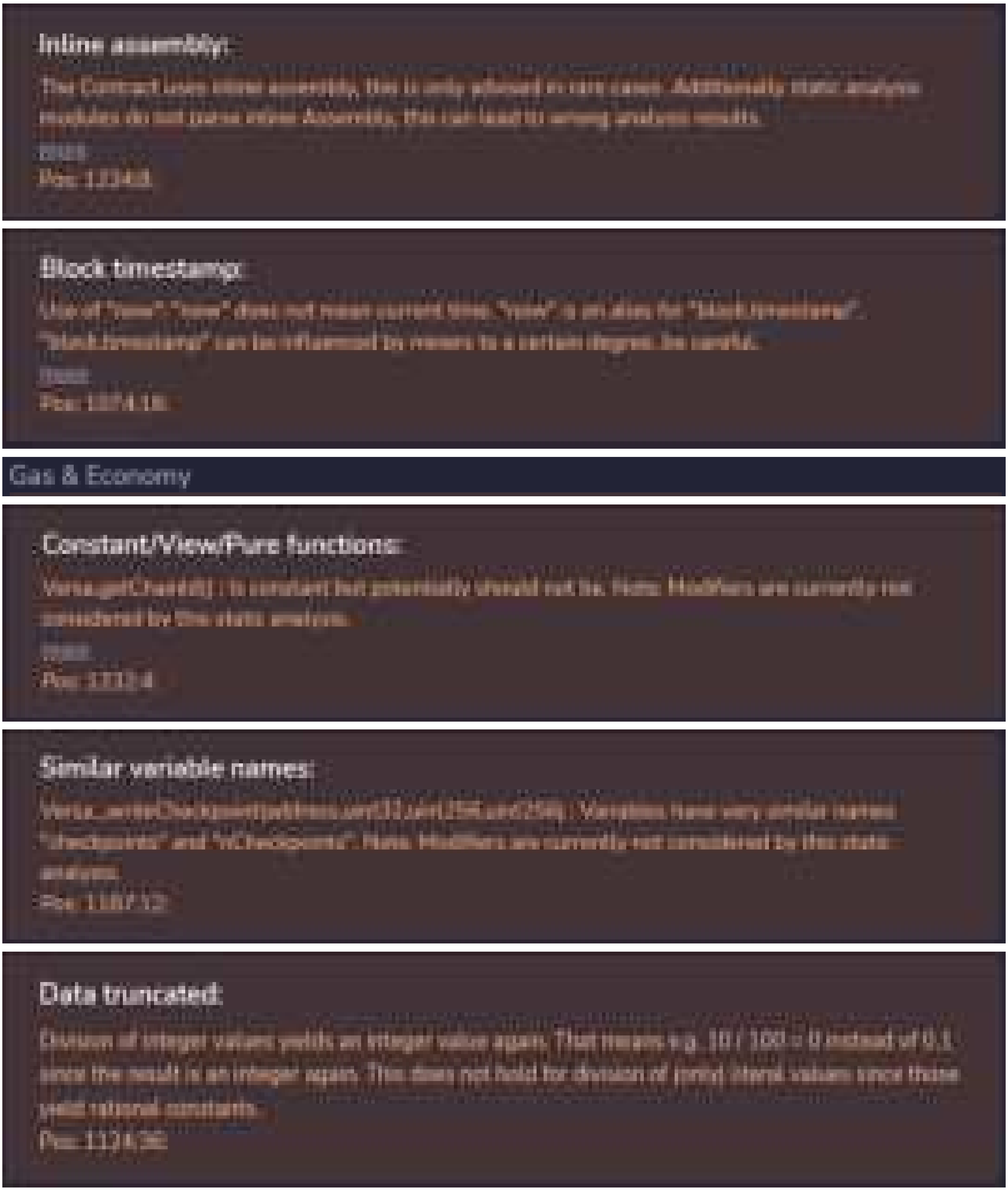
Security

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: 551:4:



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: 1234:8

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: 1874:18

Gas & Economy

Constant/View/Pure functions:

Versa.getChainId() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1733:4

Similar variable names:

Versa._writeCheckpoint(address,uint32,uint256,uint256) : Variables have very similar names "checkpoints" and "nCheckpoints". Note: Modifiers are currently not considered by this static analysis.

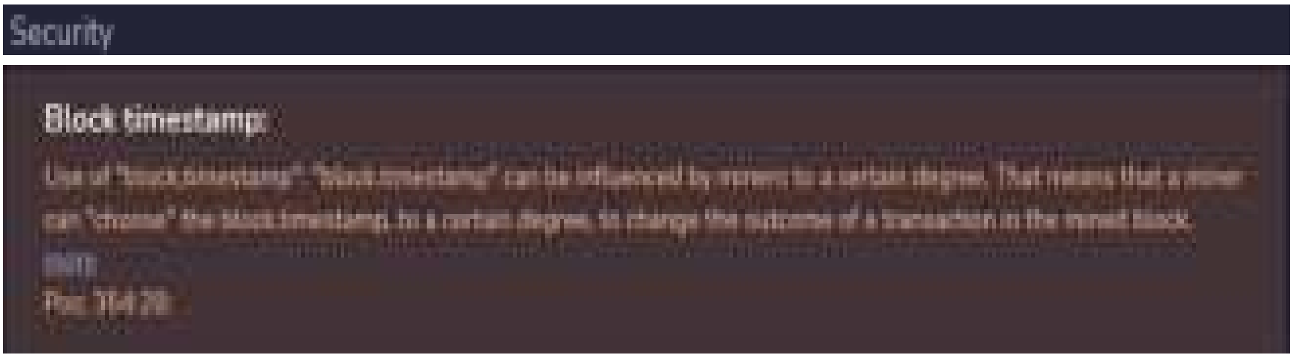
Pos: 1187:12

Data truncated:

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

Pos: 1124:36

VersaRouter.sol



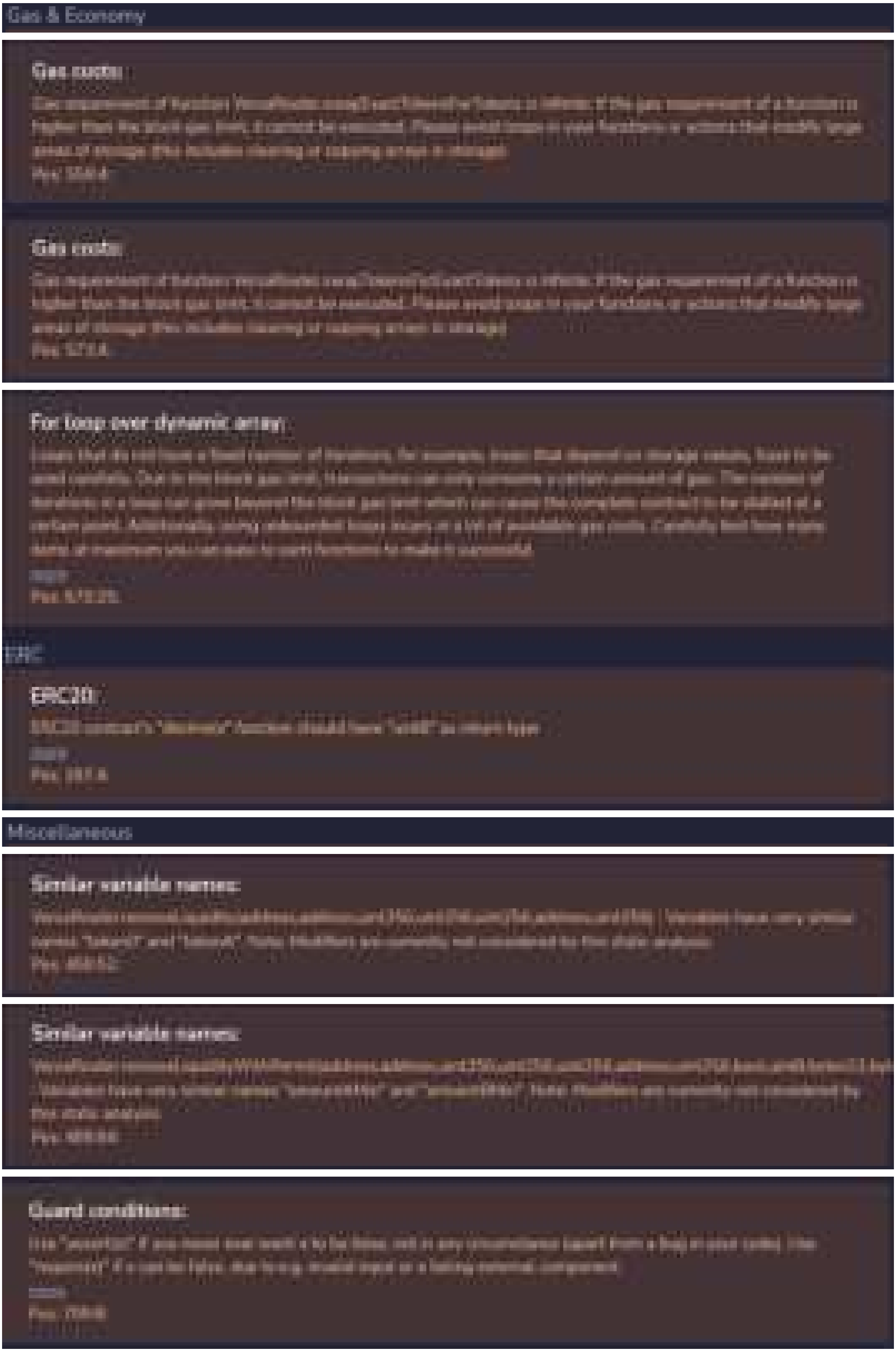
Security

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: 314:28



Gas & Economy

Gas costs:

Gas requirement of function [illegible] 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: [illegible]

Gas costs:

Gas requirement of function [illegible] 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: [illegible]

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: [illegible]

ERC

ERC20:

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

more

Pos: [illegible]

Miscellaneous

Similar variable names:

[illegible] : Variables have very similar names "tokenA" and "tokenB". Note: Modifiers are currently not considered by this static analysis.

Pos: [illegible]

Similar variable names:

[illegible] : Variables have very similar names "amountAMin" and "amountBMin". Note: Modifiers are currently not considered by this static analysis.

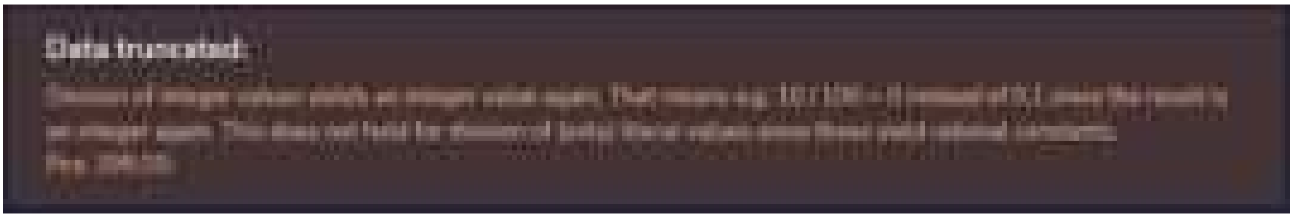
Pos: [illegible]

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: [illegible]

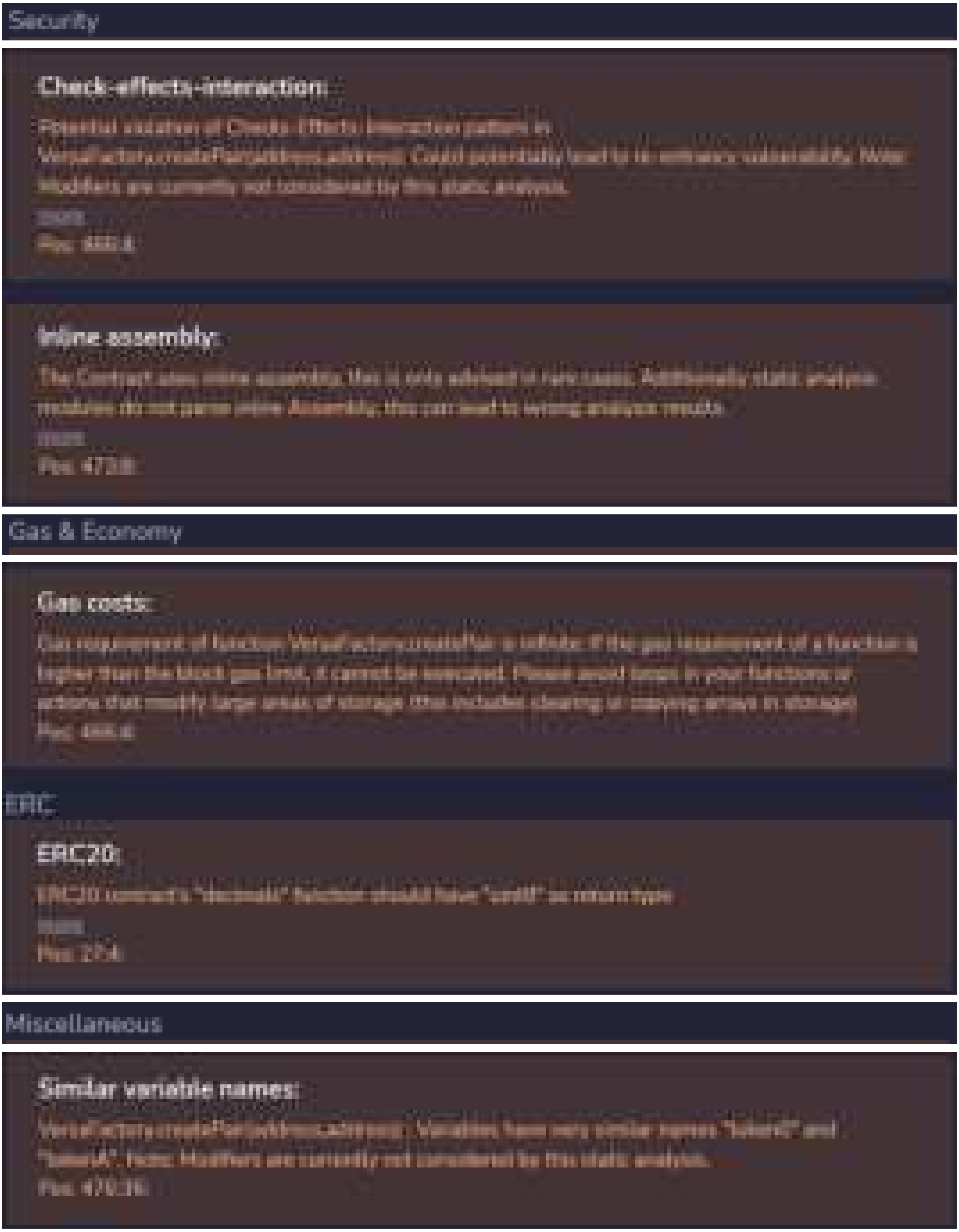


Data truncated:

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: 396:55

VersaFactory.sol



Security

Check-effects-interaction:

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

more

Pos: 448:4

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: 473:8

Gas & Economy

Gas costs:

Gas requirement of function VersaFactory.createPair 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: 466:4

ERC

ERC20:

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

more

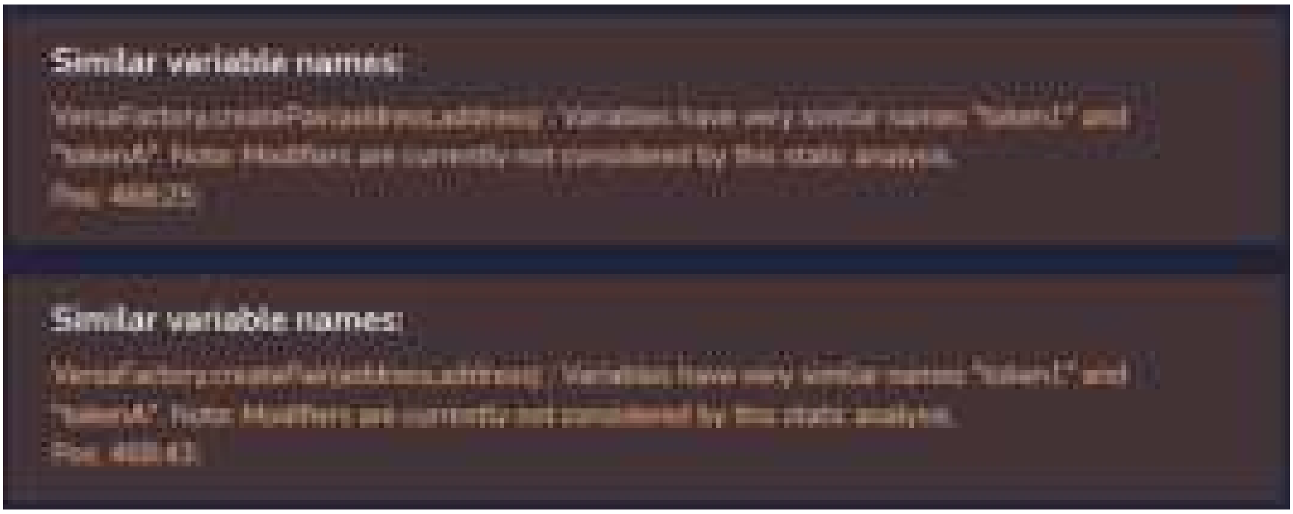
Pos: 27:4

Miscellaneous

Similar variable names:

VersaFactory.createPair(address,address): Variables have very similar names "token0" and "tokenA". Note: Modifiers are currently not considered by this static analysis.

Pos: 470:36

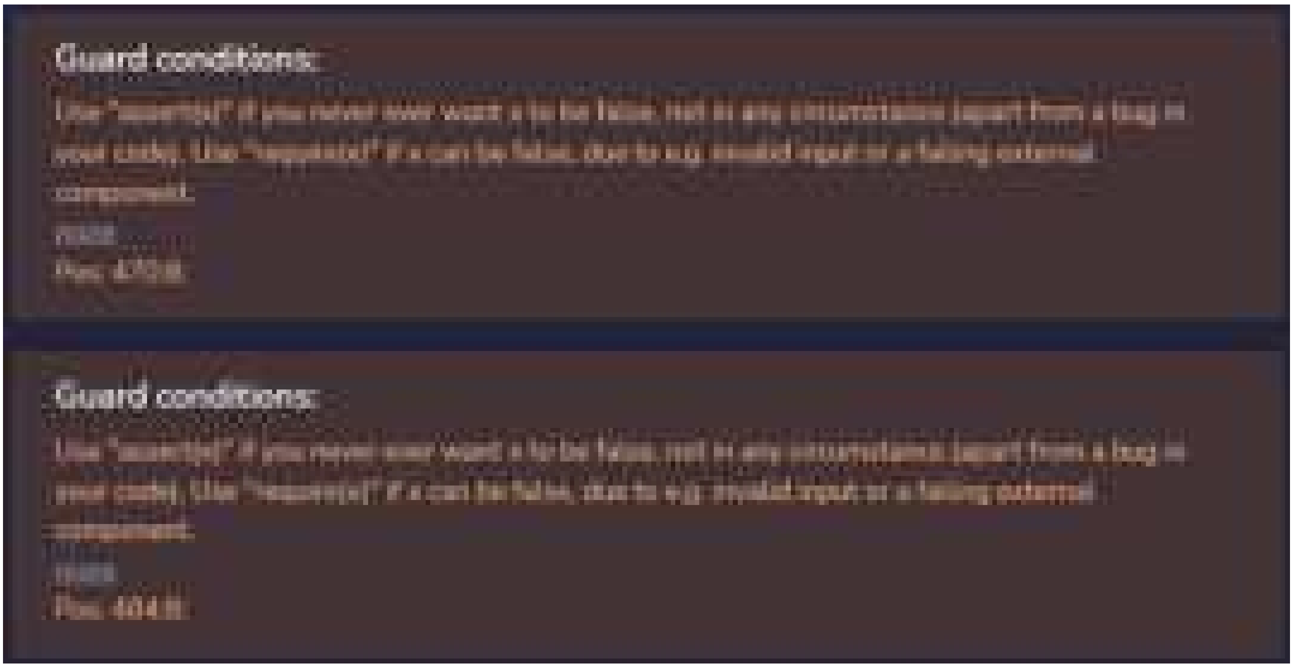


Similar variable names:

VersaFactory.createPair(address,address) : Variables have very similar names "token1" and "tokenA". Note: Modifiers are currently not considered by this static analysis.
Pos: 468:25

Similar variable names:

VersaFactory.createPair(address,address) : Variables have very similar names "token1" and "tokenA". Note: Modifiers are currently not considered by this static analysis.
Pos: 468:43

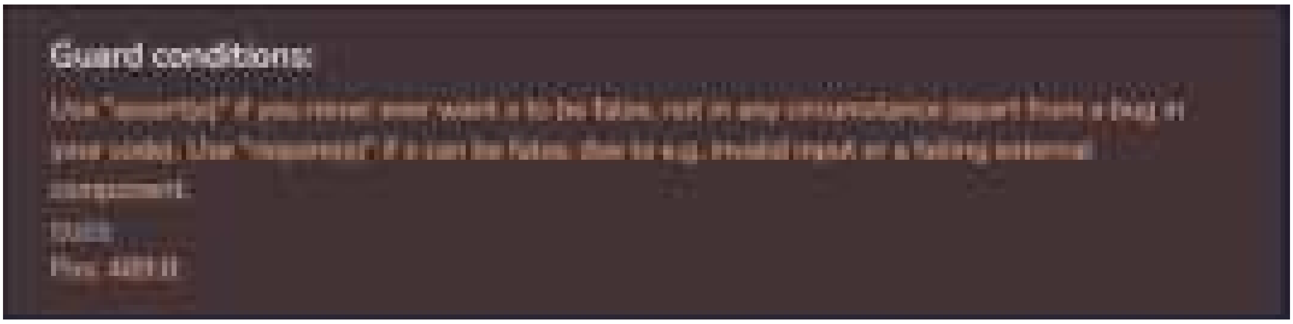


Guard conditions:

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: 470:8

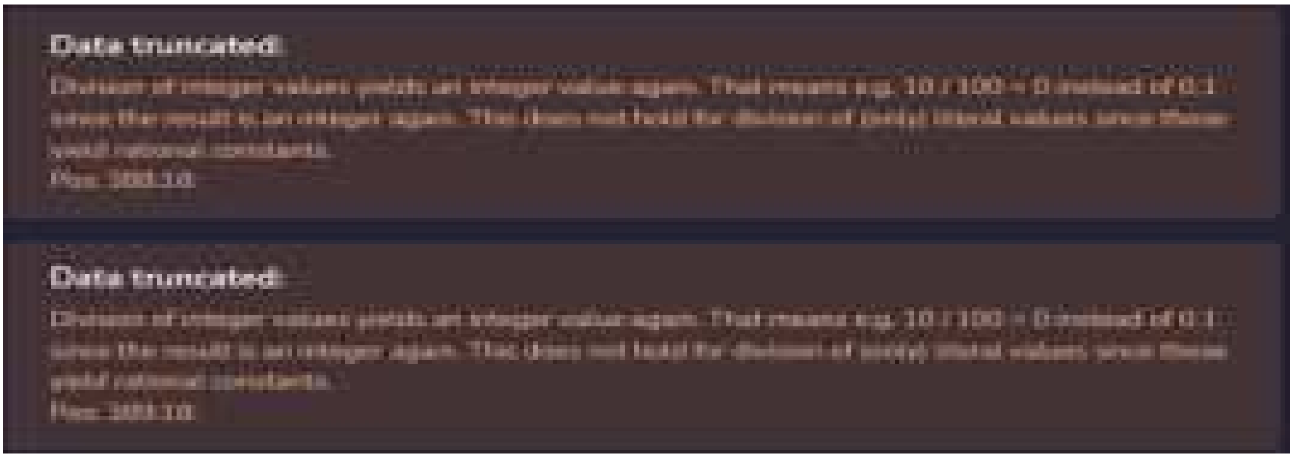
Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 404:8



Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 449:8



Data truncated:

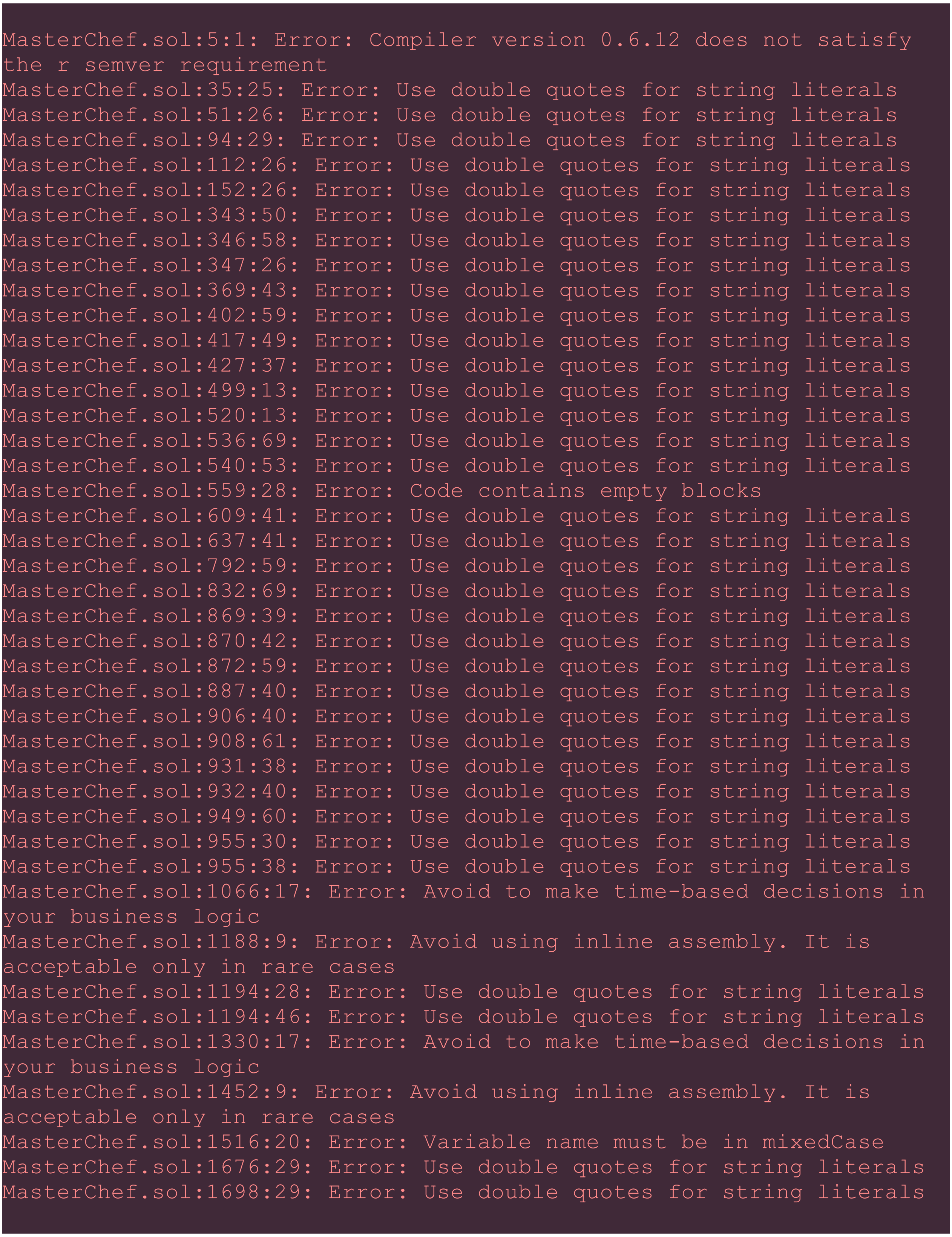
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 388:10

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: 388:10

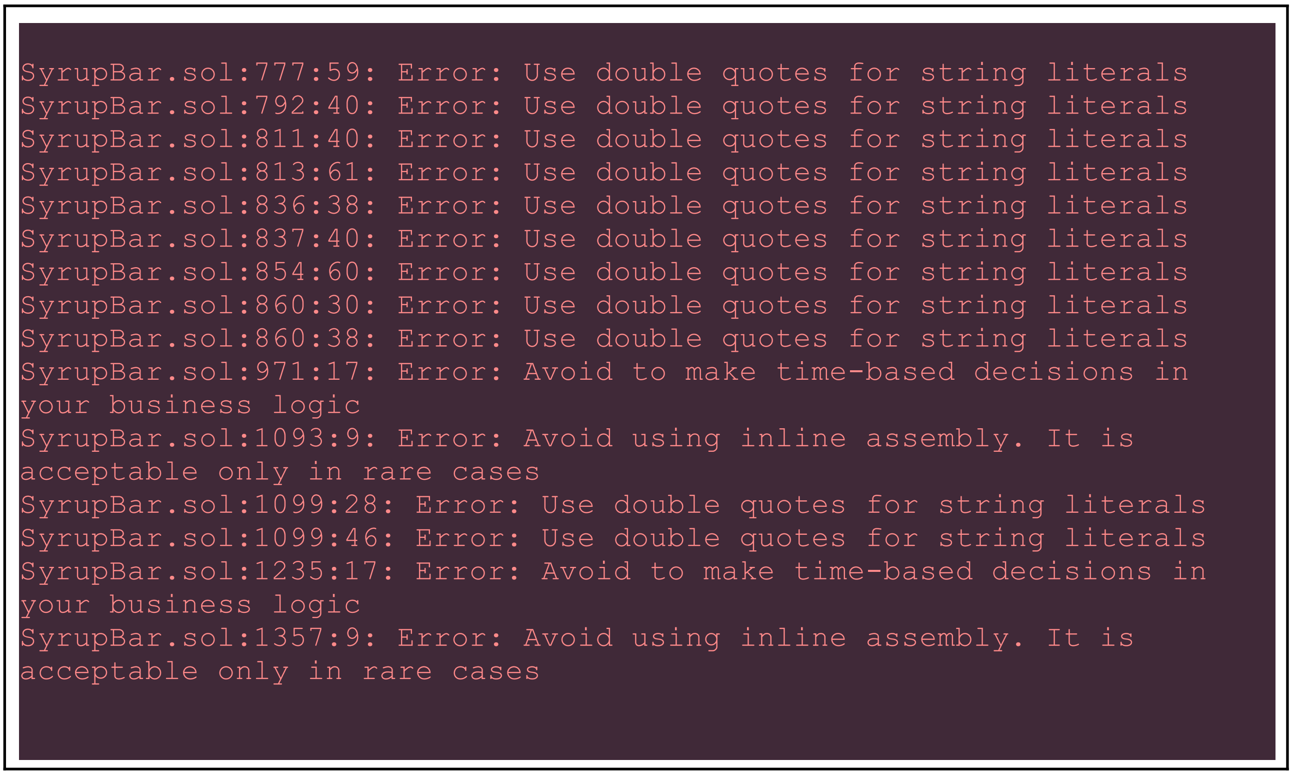
Solhint Linter

MasterChef.sol



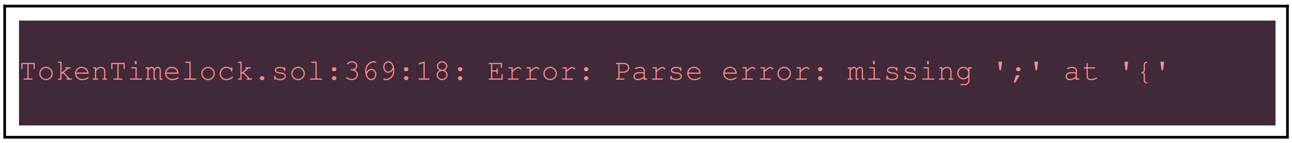
```
MasterChef.sol:5:1: Error: Compiler version 0.6.12 does not satisfy
the r semver requirement
MasterChef.sol:35:25: Error: Use double quotes for string literals
MasterChef.sol:51:26: Error: Use double quotes for string literals
MasterChef.sol:94:29: Error: Use double quotes for string literals
MasterChef.sol:112:26: Error: Use double quotes for string literals
MasterChef.sol:152:26: Error: Use double quotes for string literals
MasterChef.sol:343:50: Error: Use double quotes for string literals
MasterChef.sol:346:58: Error: Use double quotes for string literals
MasterChef.sol:347:26: Error: Use double quotes for string literals
MasterChef.sol:369:43: Error: Use double quotes for string literals
MasterChef.sol:402:59: Error: Use double quotes for string literals
MasterChef.sol:417:49: Error: Use double quotes for string literals
MasterChef.sol:427:37: Error: Use double quotes for string literals
MasterChef.sol:499:13: Error: Use double quotes for string literals
MasterChef.sol:520:13: Error: Use double quotes for string literals
MasterChef.sol:536:69: Error: Use double quotes for string literals
MasterChef.sol:540:53: Error: Use double quotes for string literals
MasterChef.sol:559:28: Error: Code contains empty blocks
MasterChef.sol:609:41: Error: Use double quotes for string literals
MasterChef.sol:637:41: Error: Use double quotes for string literals
MasterChef.sol:792:59: Error: Use double quotes for string literals
MasterChef.sol:832:69: Error: Use double quotes for string literals
MasterChef.sol:869:39: Error: Use double quotes for string literals
MasterChef.sol:870:42: Error: Use double quotes for string literals
MasterChef.sol:872:59: Error: Use double quotes for string literals
MasterChef.sol:887:40: Error: Use double quotes for string literals
MasterChef.sol:906:40: Error: Use double quotes for string literals
MasterChef.sol:908:61: Error: Use double quotes for string literals
MasterChef.sol:931:38: Error: Use double quotes for string literals
MasterChef.sol:932:40: Error: Use double quotes for string literals
MasterChef.sol:949:60: Error: Use double quotes for string literals
MasterChef.sol:955:30: Error: Use double quotes for string literals
MasterChef.sol:955:38: Error: Use double quotes for string literals
MasterChef.sol:1066:17: Error: Avoid to make time-based decisions in
your business logic
MasterChef.sol:1188:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
MasterChef.sol:1194:28: Error: Use double quotes for string literals
MasterChef.sol:1194:46: Error: Use double quotes for string literals
MasterChef.sol:1330:17: Error: Avoid to make time-based decisions in
your business logic
MasterChef.sol:1452:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
MasterChef.sol:1516:20: Error: Variable name must be in mixedCase
MasterChef.sol:1676:29: Error: Use double quotes for string literals
MasterChef.sol:1698:29: Error: Use double quotes for string literals
```

SyrupBar.sol



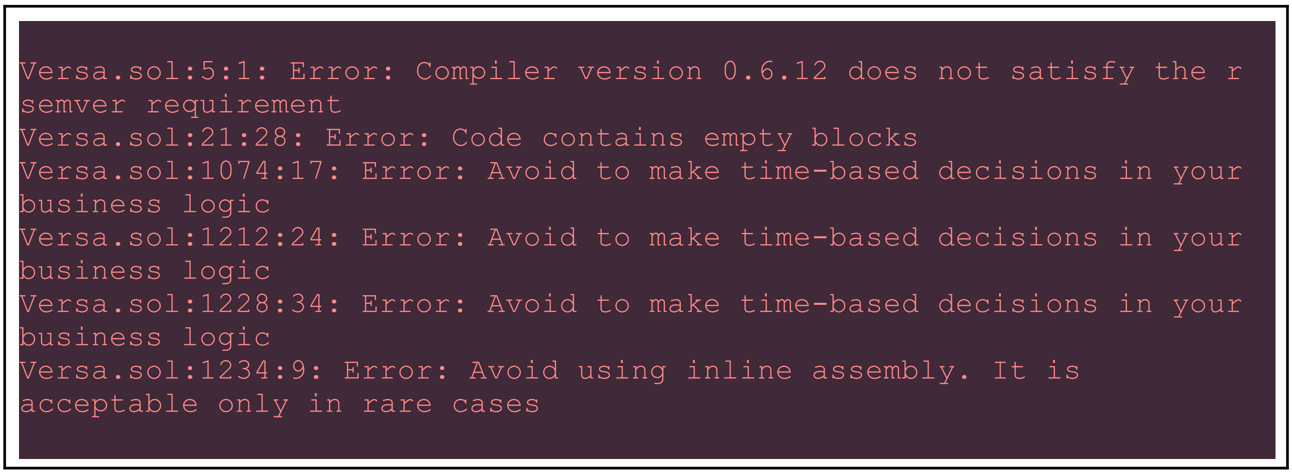
```
SyrupBar.sol:777:59: Error: Use double quotes for string literals
SyrupBar.sol:792:40: Error: Use double quotes for string literals
SyrupBar.sol:811:40: Error: Use double quotes for string literals
SyrupBar.sol:813:61: Error: Use double quotes for string literals
SyrupBar.sol:836:38: Error: Use double quotes for string literals
SyrupBar.sol:837:40: Error: Use double quotes for string literals
SyrupBar.sol:854:60: Error: Use double quotes for string literals
SyrupBar.sol:860:30: Error: Use double quotes for string literals
SyrupBar.sol:860:38: Error: Use double quotes for string literals
SyrupBar.sol:971:17: Error: Avoid to make time-based decisions in
your business logic
SyrupBar.sol:1093:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
SyrupBar.sol:1099:28: Error: Use double quotes for string literals
SyrupBar.sol:1099:46: Error: Use double quotes for string literals
SyrupBar.sol:1235:17: Error: Avoid to make time-based decisions in
your business logic
SyrupBar.sol:1357:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

TokenTimelock.sol



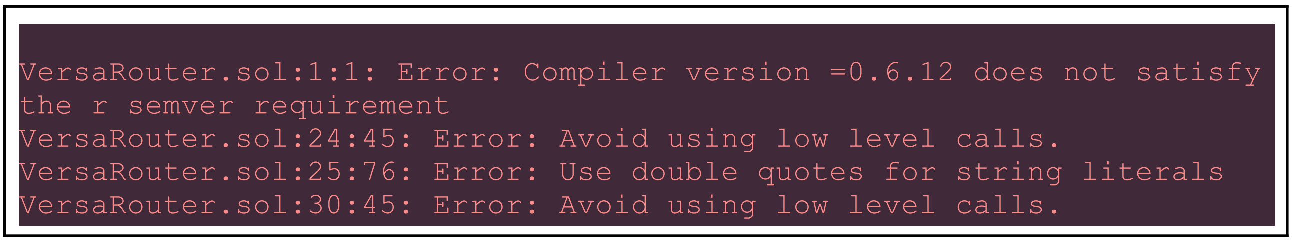
```
TokenTimelock.sol:369:18: Error: Parse error: missing ';' at '{'
```

Versa.sol

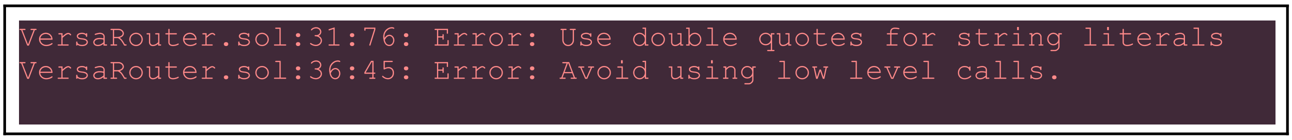


```
Versa.sol:5:1: Error: Compiler version 0.6.12 does not satisfy the r
semver requirement
Versa.sol:21:28: Error: Code contains empty blocks
Versa.sol:1074:17: Error: Avoid to make time-based decisions in your
business logic
Versa.sol:1212:24: Error: Avoid to make time-based decisions in your
business logic
Versa.sol:1228:34: Error: Avoid to make time-based decisions in your
business logic
Versa.sol:1234:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

VersaRouter.sol

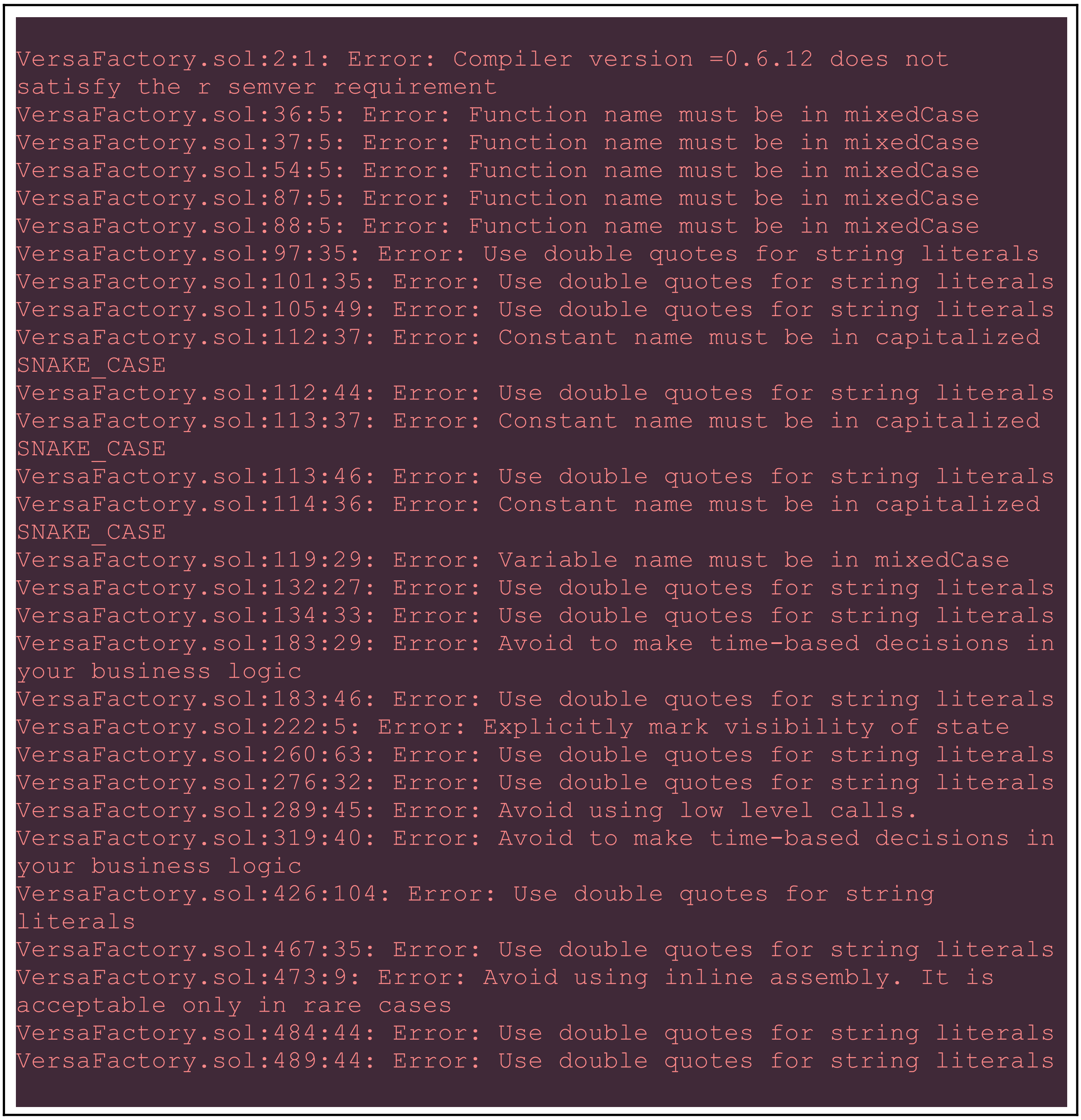


```
VersaRouter.sol:1:1: Error: Compiler version =0.6.12 does not satisfy
the r semver requirement
VersaRouter.sol:24:45: Error: Avoid using low level calls.
VersaRouter.sol:25:76: Error: Use double quotes for string literals
VersaRouter.sol:30:45: Error: Avoid using low level calls.
```



```
VersaRouter.sol:31:76: Error: Use double quotes for string literals
VersaRouter.sol:36:45: Error: Avoid using low level calls.
```

VersaFactory.sol



```
VersaFactory.sol:2:1: Error: Compiler version =0.6.12 does not
satisfy the r semver requirement
VersaFactory.sol:36:5: Error: Function name must be in mixedCase
VersaFactory.sol:37:5: Error: Function name must be in mixedCase
VersaFactory.sol:54:5: Error: Function name must be in mixedCase
VersaFactory.sol:87:5: Error: Function name must be in mixedCase
VersaFactory.sol:88:5: Error: Function name must be in mixedCase
VersaFactory.sol:97:35: Error: Use double quotes for string literals
VersaFactory.sol:101:35: Error: Use double quotes for string literals
VersaFactory.sol:105:49: Error: Use double quotes for string literals
VersaFactory.sol:112:37: Error: Constant name must be in capitalized
SNAKE_CASE
VersaFactory.sol:112:44: Error: Use double quotes for string literals
VersaFactory.sol:113:37: Error: Constant name must be in capitalized
SNAKE_CASE
VersaFactory.sol:113:46: Error: Use double quotes for string literals
VersaFactory.sol:114:36: Error: Constant name must be in capitalized
SNAKE_CASE
VersaFactory.sol:119:29: Error: Variable name must be in mixedCase
VersaFactory.sol:132:27: Error: Use double quotes for string literals
VersaFactory.sol:134:33: Error: Use double quotes for string literals
VersaFactory.sol:183:29: Error: Avoid to make time-based decisions in
your business logic
VersaFactory.sol:183:46: Error: Use double quotes for string literals
VersaFactory.sol:222:5: Error: Explicitly mark visibility of state
VersaFactory.sol:260:63: Error: Use double quotes for string literals
VersaFactory.sol:276:32: Error: Use double quotes for string literals
VersaFactory.sol:289:45: Error: Avoid using low level calls.
VersaFactory.sol:319:40: Error: Avoid to make time-based decisions in
your business logic
VersaFactory.sol:426:104: Error: Use double quotes for string
literals
VersaFactory.sol:467:35: Error: Use double quotes for string literals
VersaFactory.sol:473:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
VersaFactory.sol:484:44: Error: Use double quotes for string literals
VersaFactory.sol:489:44: Error: Use double quotes for string literals
```

Software analysis result:

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

Ether Authority

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io