EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Yumi-Swap Protocol
Website:	https://yumiswap.com
Platform:	Cronos Network
Language:	Solidity
Date:	April 12th, 2022

Table of contents

Appendix

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 Yumi-Swap team to perform the Security audit of the Yumi-Swap 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 12th, 2022.

The purpose of this audit was to address the following:

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

Project Background

The Yumi-Swap Contracts have functions like add and set pool, withdraw, deposit, addPair, setPair, reward, mint, burn, swap, enter, leave, getPriorVotes, getChainId, etc.

Audit scope

Name	**Code Review and Security Analysis Report for Yumi-Swap Protocol Smart Contracts**
Platform	**Cronos Network / Solidity**
File 1	MasterChef.sol
File 1 MD5 Hash	3C7EF2712DB28DA3BB3D9A6D84AC62B7
Updated File 1 MD5 Hash	05EDAEE91DE06EDE5D013B625161463D
File 2	SwapMining.sol
File 2 MD5 Hash	D6AC8FFDE07EB05014645D39A6EDEAD9
Updated File 2 MD5 Hash	2B2D1195B4AD5A48BDE770C38244AF53
File 3	SyrupBar.sol
File 3 MD5 Hash	08028B372959A0E82AA650B23EFF14D4
Updated File 3 MD5 Hash	2F5CF6D4112838680BAD677E859240AD
File 4	MockToken.sol

File 4 MD5 Hash	9D1DB94665C7D4C111645D20B8A0CCD6
File 5	Factory.sol
File 5 MD5 Hash	A417A902F34E26F8F66B65C85A4C1CF6
Updated File 5 MD5 Hash	F85C21A8D2EA2DC3B6C0DE138768507B
File 6	Pair.sol
File 6 MD5 Hash	7B1C70F7F9FADE20D2732C47AB2F18E1
File 7	xYUMI.sol
File 7 MD5 Hash	01E7908C3D8965C736E88F0D2ED65EC4
Updated File 7 MD5 Hash	05FF07C65E901C4B159BC547C88ECE4E
File 8	YumiToken.sol
File 8 MD5 Hash	B301957E808A5C6BCDC3279116736685
Updated File 8 MD5 Hash	F01C1B3A89799FC208FEBD0D73E32776
File 9	LakeOfYumi.sol
File 9 MD5 Hash	CF0146DD5B80F075FD8D9973E5916DB4
File 10	Multicall.sol
File 10 MD5 Hash	B31A5401C236F10109672BC3D903C9DA
Updated File 10 MD5 Hash	CD78A297F742B45105931F70C0458053
File 11	FeeSharingPool.sol
File 11 MD5 Hash	B5CDD3C64337EFA9B0A4638D1B98F9CC
File 12	Oracle.sol
File 12 MD5 Hash	3F75D4A26F5FA909AFB50C4FD1B5D080
File 13	Router.sol
File 13 MD5 Hash	8476A37A0A5B9F0CA875E7D2259C305F
Audit Date	April 12th,2022
Revise Audit Date	July 28th,2022

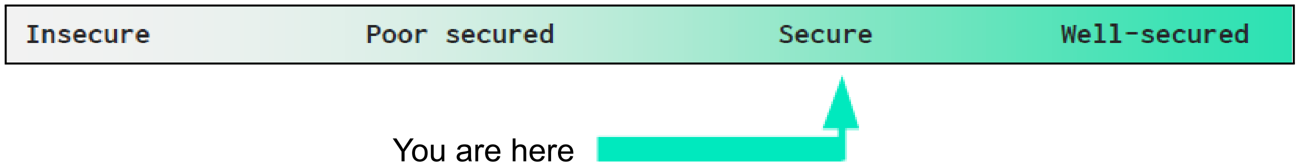
Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 MasterChef.sol ● Maximum Cake per Sec: 10 Quintillion ● Yumi Maximum Supply: 100 Septillion	YES, This is valid.
File 2 SwapMining.sol ● Swap Mining contract has functions like: setPair, setYumiswapPerSecond, addWhitelist, etc.	YES, This is valid.
File 3 SyrupBar.sol ● Name: YumiSwapBar Token ● Symbol: SYRUP ● SyrupBar used for YUMI staking.	YES, This is valid.
File 4 MockToken.sol ● Decimals: 18	YES, This is valid.
File 5 Factory.sol ● YumiswapFactory contract has functions like: allPairsLength, expectPairFor, createPair, etc.	YES, This is valid.
File 6 Pair.sol ● Name: Yumiswap LPs ● Symbol: YUMI-LP ● Decimals: 18 ● Minimum Liquidity: 1000	YES, This is valid.
File 7 xYUMI.sol ● Name: Yumi Staking Token ● Symbol: xYUMI ● Decimals: 18	YES, This is valid.

File 8 YumiToken.sol ● Name: YumiSwap Token ● Symbol: YUMI ● Decimals: 18	**YES, This is valid.**
File 9 LakeOfYumi.sol ● LakeOfYumi contract has functions like: convertMultiple, setDevAddr, bridgeFor, etc.	**YES, This is valid.**
File 10 Multicall.sol ● Multicall contract has multiple read-only function calls.	**YES, This is valid.**

Audit Summary

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



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

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

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

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	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 13 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 Yumi-Swap 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 Yumi-Swap Protocol.

The Yumi-Swap Protocol 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 Yumi-Swap Protocol smart contract code in the form files and cronos blockscout 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://yumiswap.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 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	updateMultiplier	write	access only Owner	No Issue
7	poolLength	external	Passed	No Issue
8	add	write	Critical operation lacks event log	Refer Audit Findings
9	set	write	Critical operation lacks event log	Refer Audit Findings
10	getMultiplier	read	Passed	No Issue
11	pendingCake	external	Passed	No Issue
12	massUpdatePools	write	Passed	No Issue
13	updatePool	write	Critical operation lacks event log	Refer Audit Findings
14	deposit	write	Passed	No Issue
15	withdraw	write	Passed	No Issue
16	emergencyWithdraw	write	Passed	No Issue
17	safeCakeTransfer	internal	Passed	No Issue
18	setCakePerSecond	external	access only Owner	No Issue
19	setEcoaddr	write	Passed	No Issue
20	setReserveaddr	write	Passed	No Issue

SwapMining.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	poolLength	read	Passed	No Issue
7	addPair	write	Critical operation lacks event log	Refer Audit Findings
8	setPair	write	Critical operation lacks event log	Refer Audit Findings
9	setYumiswapPerSecond	write	access only Owner	No Issue
10	addWhitelist	write	access only Owner	No Issue

11	delWhitelist	write	access only Owner	No Issue
12	getWhitelistLength	read	Passed	No Issue
13	isWhitelist	read	Passed	No Issue
14	getWhitelist	read	Passed	No Issue
15	setHalvingPeriod	write	access only Owner	No Issue
16	setRouter	write	access only Owner	No Issue
17	setOracle	write	access only Owner	No Issue
18	phase	read	Passed	No Issue
19	phase	read	Passed	No Issue
20	reward	read	Passed	No Issue
21	reward	read	Passed	No Issue
22	getYumiReward	read	Passed	No Issue
23	massMintPools	write	Passed	No Issue
24	mint	write	Critical operation lacks event log	Refer Audit Findings
25	onlyRouter	modifier	Passed	No Issue
26	swap	write	access only Router	No Issue
27	getQuantity	read	Passed	No Issue
28	takerWithdraw	write	Critical operation lacks event log	Refer Audit Findings
29	getUserReward	read	Passed	No Issue
30	getTotalUserReward	read	Passed	No Issue
31	getPoolInfo	read	Passed	No Issue
32	ownerWithdraw	write	Critical operation lacks event log	Refer Audit Findings
33	addBlacklist	external	access only Owner	No Issue
34	removeBlacklist	external	access only Owner	No Issue
35	safeYumiTransfer	internal	Passed	No Issue

SyrupBar.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getOwner	external	Passed	No Issue
3	name	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	write	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	mint	write	access only Owner	No Issue

15	_transfer	internal	Passed	No Issue
16	_mint	internal	Passed	No Issue
17	_burn	internal	Passed	No Issue
18	_approve	internal	Passed	No Issue
19	_burnFrom	internal	Passed	No Issue
20	mint	write	access only Owner	No Issue
21	burn	write	access only Owner	No Issue
22	safeCakeTransfer	write	access only Owner	No Issue
23	delegates	external	Passed	No Issue
24	delegate	external	Passed	No Issue
25	getCurrentVotes	external	Passed	No Issue
26	delegateBySig	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

MockToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	mint	write	Passed	No Issue
3	owner	read	Passed	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	getOwner	external	Passed	No Issue
8	name	read	Passed	No Issue
9	decimals	read	Passed	No Issue
10	symbol	read	Passed	No Issue
11	totalSupply	read	Passed	No Issue
12	balanceOf	read	Passed	No Issue
13	transfer	write	Passed	No Issue
14	allowance	write	Passed	No Issue
15	approve	write	Passed	No Issue
16	transferFrom	write	Passed	No Issue
17	increaseAllowance	write	Passed	No Issue
18	decreaseAllowance	write	Passed	No Issue
19	mint	write	access only Owner	No Issue
20	_transfer	internal	Passed	No Issue
21	_mint	internal	Passed	No Issue
22	_burn	internal	Passed	No Issue
23	_approve	internal	Passed	No Issue
24	_burnFrom	internal	Passed	No Issue

Factory.sol

Functions

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

Pair.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_mint	internal	Passed	No Issue
3	_burn	internal	Passed	No Issue
4	_approve	write	Passed	No Issue
5	_transfer	write	Passed	No Issue
6	approve	external	Passed	No Issue
7	transfer	external	Passed	No Issue
8	transferFrom	external	Passed	No Issue
9	permit	external	Passed	No Issue
10	getReserves	read	Passed	No Issue
11	_safeTransfer	write	Passed	No Issue
12	initialize	external	Passed	No Issue
13	_update	write	Passed	No Issue
14	_mintFee	write	Passed	No Issue
15	mint	external	Passed	No Issue
16	burn	external	Passed	No Issue
17	swap	external	Passed	No Issue
18	skim	external	Passed	No Issue
19	sync	external	Passed	No Issue

xYUMI.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	write	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	mint	write	access only Owner	No Issue
15	_transfer	internal	Passed	No Issue
16	_mint	internal	Passed	No Issue
17	_burn	internal	Passed	No Issue
18	_approve	internal	Passed	No Issue
19	_burnFrom	internal	Passed	No Issue
20	stakedTime	read	Passed	No Issue
21	canWithdraw	read	Passed	No Issue
22	setDelayToWithdraw	external	Passed	No Issue
23	enter	write	Critical operation lacks event log	Refer Audit Findings
24	leave	write	Critical operation lacks event log	Refer Audit Findings
25	YUMIBalance	external	Passed	No Issue
26	xYUMIForYUMI	external	Passed	No Issue
27	YUMIForxYUMI	external	Passed	No Issue
28	burn	write	Passed	No Issue
29	mint	write	Passed	No Issue
30	transferFrom	write	Passed	No Issue
31	transfer	write	Passed	No Issue
32	_initDelegates	internal	Passed	No Issue
33	delegates	external	Passed	No Issue
34	delegate	external	Passed	No Issue
35	delegateBySig	external	Passed	No Issue
36	getCurrentVotes	external	Passed	No Issue
37	getPriorVotes	external	Passed	No Issue
38	_delegate	internal	Passed	No Issue
39	_moveDelegates	internal	Passed	No Issue
40	_writeCheckpoint	internal	Passed	No Issue
41	safe32	internal	Passed	No Issue
42	getChainId	internal	Passed	No Issue
43	setAdmin	write	Passed	No Issue

YumiToken.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	write	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	mint	write	access only Owner	No Issue
15	_transfer	internal	Passed	No Issue
16	_mint	internal	Passed	No Issue
17	_burn	internal	Passed	No Issue
18	_approve	internal	Passed	No Issue
19	_burnFrom	internal	Passed	No Issue
20	mintFor	write	access only Owner	No Issue
21	mint	write	access only Owner	No Issue
22	delegates	external	Passed	No Issue
23	delegate	external	Passed	No Issue
24	delegateBySig	external	Passed	No Issue
25	getCurrentVotes	external	Passed	No Issue
26	getPriorVotes	external	Passed	No Issue
27	_delegate	internal	Passed	No Issue
28	_moveDelegates	internal	Passed	No Issue
29	_writeCheckpoint	internal	Passed	No Issue
30	safe32	internal	Passed	No Issue
31	getChainId	internal	Passed	No Issue

LakeOfYumi.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	onlyAuth	modifier	Passed	No Issue
7	addAuth	external	access only Owner	No Issue
8	revokeAuth	external	access only Owner	No Issue
9	setAnyAuth	external	access only Owner	No Issue
10	setBridge	external	access only Owner	No Issue

11	setDevCut	external	access only Owner	No Issue
12	setDevAddr	external	access only Owner	No Issue
13	bridgeFor	read	Passed	No Issue
14	onlyEOA	modifier	Passed	No Issue
15	convert	external	access only Auth	No Issue
16	convertMultiple	external	access only Auth	No Issue
17	_convert	internal	Passed	No Issue
18	_convertStep	internal	Passed	No Issue
19	_swap	internal	Passed	No Issue
20	_toYUMI	internal	Passed	No Issue
21	getAmountOut	internal	Passed	No Issue

Multicall.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	aggregate	write	Passed	No Issue
3	getEthBalance	read	Passed	No Issue
4	getBlockHash	read	Passed	No Issue
5	getLastBlockHash	read	Passed	No Issue
6	getCurrentBlockTimestamp	read	Passed	No Issue
7	getCurrentBlockDifficulty	read	Passed	No Issue
8	getCurrentBlockGasLimit	read	Passed	No Issue
9	getCurrentBlockCoinbase	read	Passed	No Issue

Severity Definitions

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

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Critical operation lacks event log:

Missing event log for:

MasterChef.sol

- add
- set
- updatePool

xYUMI.sol

- enter.
- leave

SwapMining.sol

- addPair
- setPair
- mint
- ownerWithdraw
- takerWithdraw

Resolution: Write an event log for listed events.

Very Low / Informational / Best practices:

(1) Unused variable: **MasterChef.sol**.

prevAllocPoint has been defined but not used anywhere.

Resolution: We suggest removing unused variables.

(2) Use the latest solidity version: **- YumiToken.sol, MockToken.sol, Syrupbar.sol, xYUMI.sol**

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

.

Resolution: We suggest using the latest solidity version.

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:

- updateMultiplier: Masterchef owner can update multiplier number value.
- add: Masterchef owner can add a new lp to the pool.
- set: Masterchef owner can update the given pool's YUMI allocation point.
- setCakePerSecond: Masterchef owner can update cake token reward per second, with a cap of max cake per second.
- mint: SyrupBar owner can create `_amount` token to `_to` by MasterChef owner.
- burn: SyrupBar owners can burn an amount from the address.
- safeCakeTransfer: SyrupBar owners can save cake transfer function, just in case if rounding error causes pool to not have enough YUMIs.
- addPair: SwapMining owner can add new pair.
- setPair: SwapMining owner can update the allocPoint of the pool.
- setYumiswapPerSecond: SwapMining owner can set the number of yumi produced by each second.

- addWhitelist: SwapMining owner can add new wallet address in whitelist.
- delWhitelist: SwapMining owner can remove wallet address from the whitelist.
- setHalvingPeriod: SwapMining owner can set halving period value.
- setRouter: SwapMining owner can set new router address.
- setOracle: SwapMining owner can set new oracle address.
- ownerWithdraw: SwapMining owner can withdraw amount from wallet address.
- addBlacklist: SwapMining owner can add wallet address in blacklist.
- removeBlacklist: SwapMining owner can remove wallet address from the blacklist.
- mintFor: YumiToken owner can create `_amount` token to `_to` by masterchef owner.
- mint: YumiToken owner can mint value from owner wallet.
- addAuth: LakeOfYumi owner can add a new auth wallet address.
- revokeAuth: LakeOfYumi owner can remove auth wallet address.
- setAnyAuth: LakeOfYumi owner can set anyAuth to true and allows anyone to call functions protected by onlyAuth.
- setBridge: LakeOfYumi owner can set bridge address.
- setDevCut: LakeOfYumi owner can set dev cut amount.
- setDevAddr: LakeOfYumi owner can set dev address.
- convert: LakeOfYumi auth can convert token value.
- convertMultiple: LakeOfYumi auth can convert multiple token values.

Conclusion

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

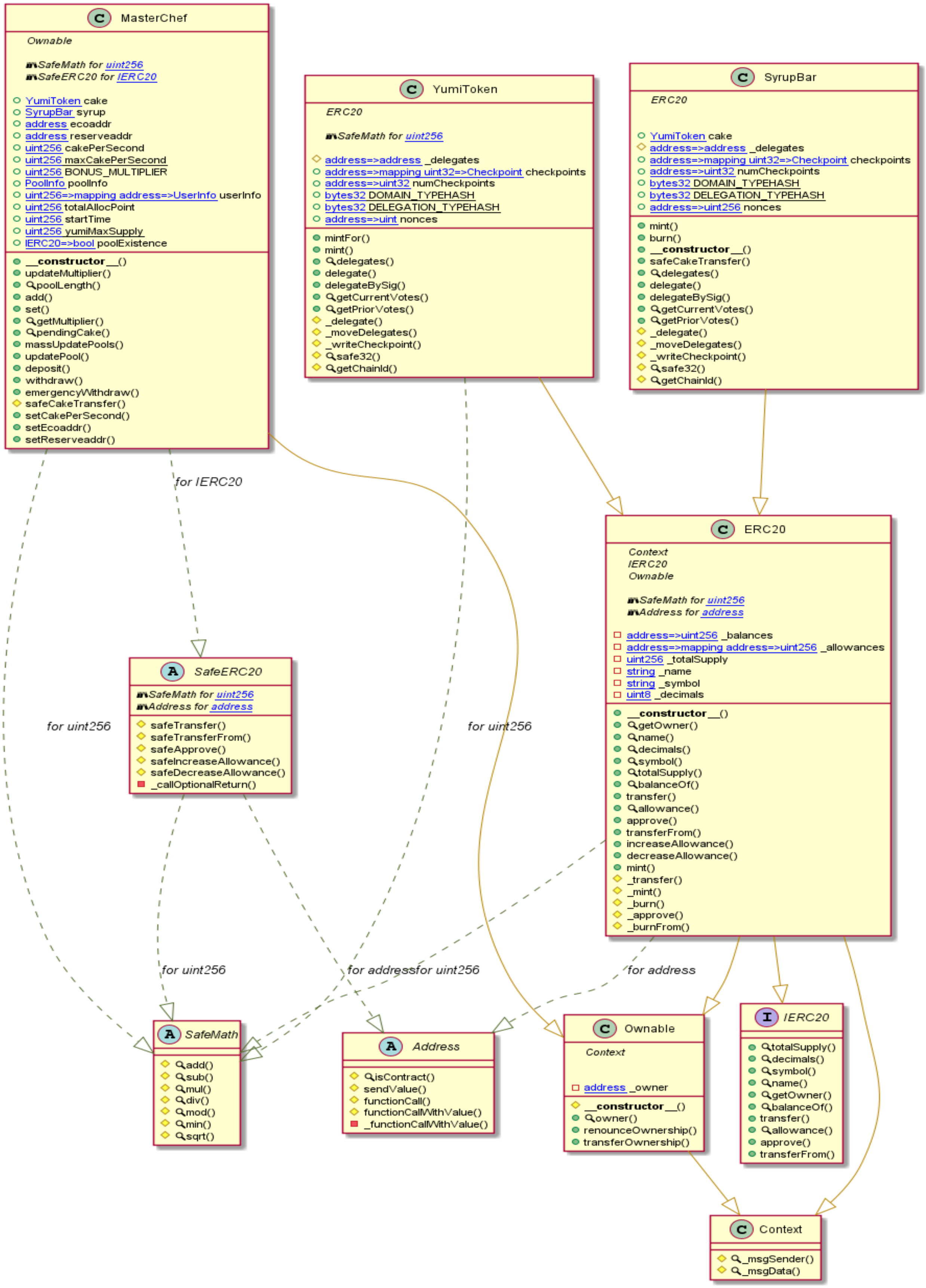
Technical Disclaimer

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.

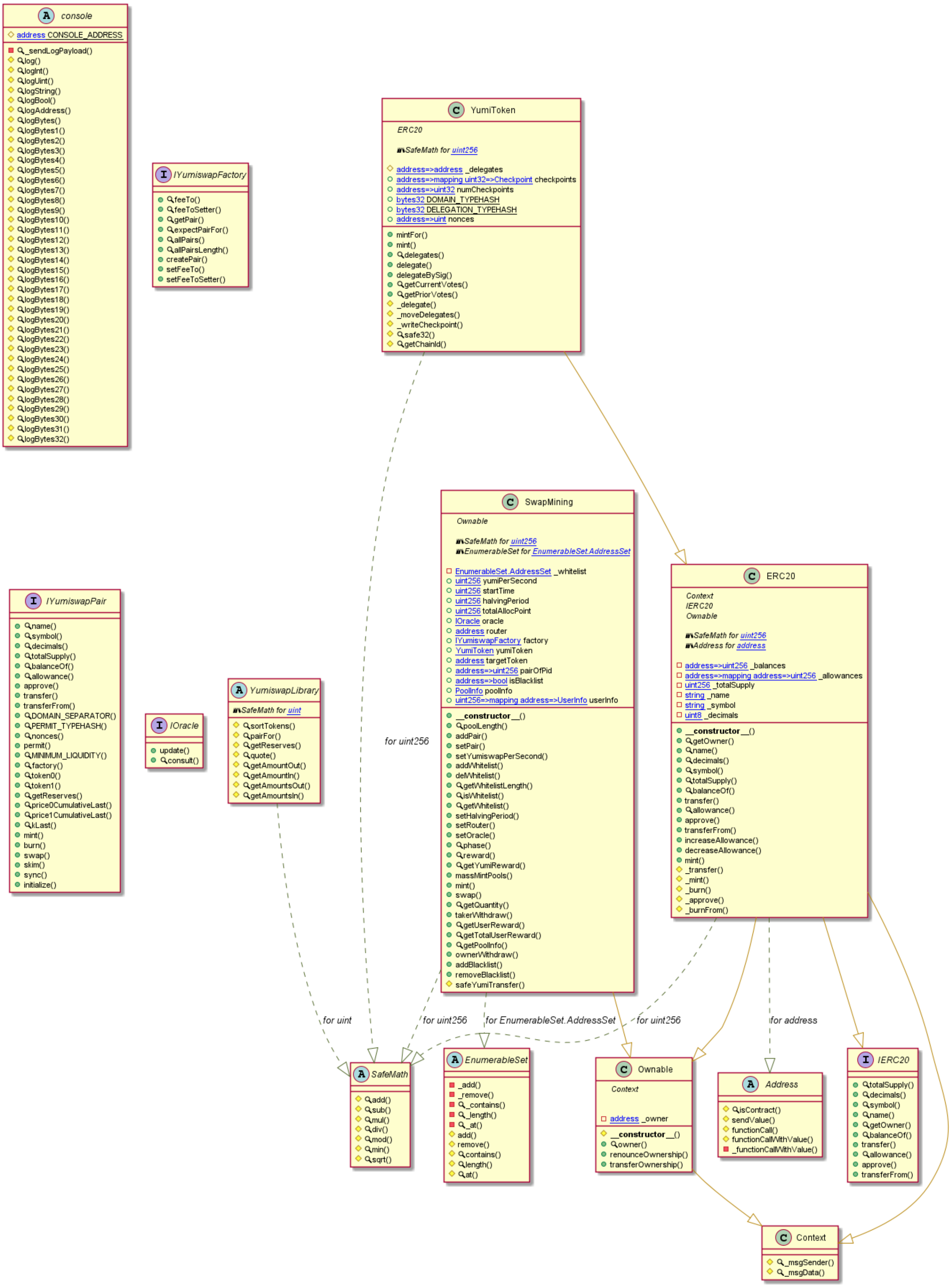
Appendix

Code Flow Diagram - Yumi-Swap Protocol

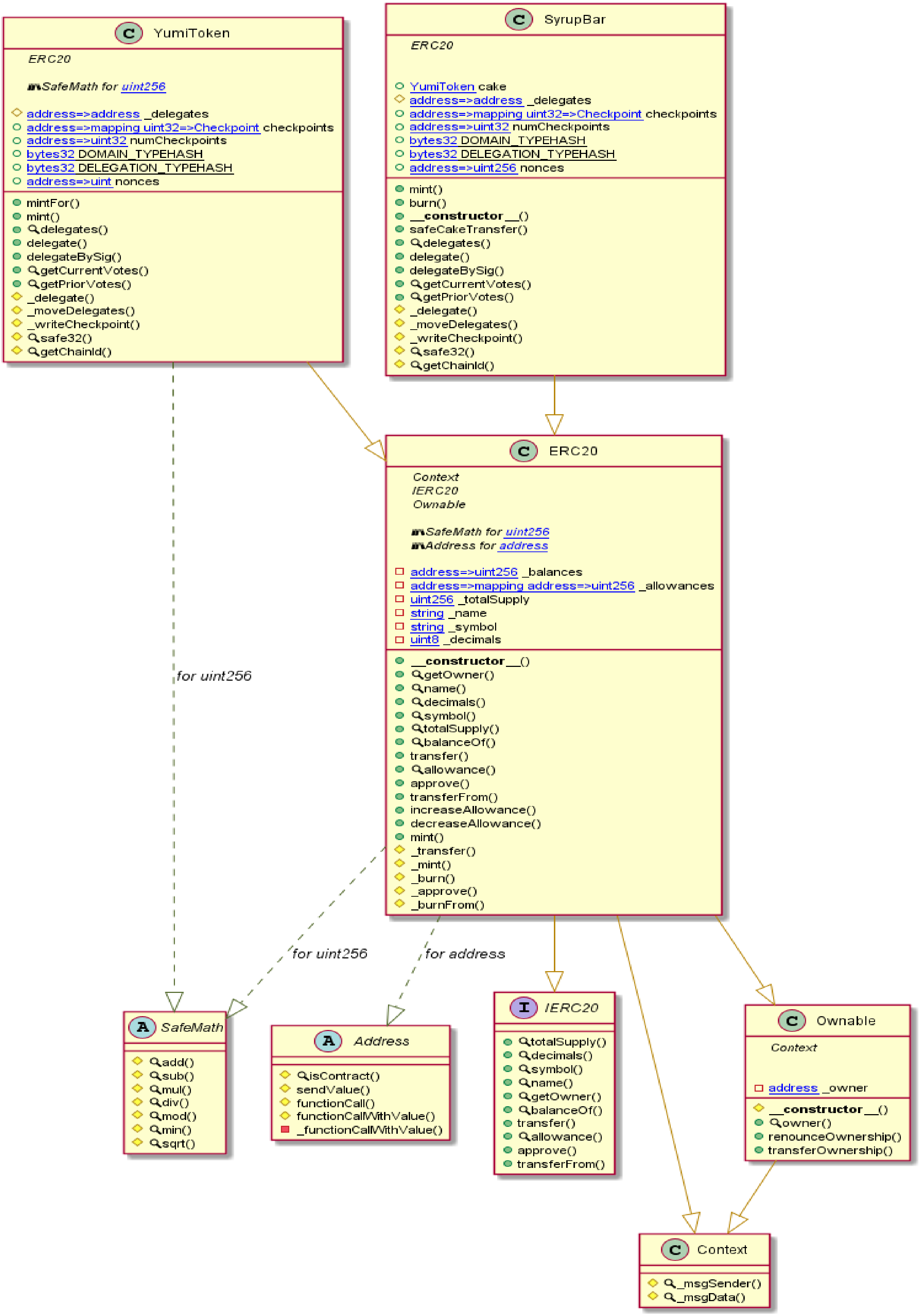
MasterChef Diagram



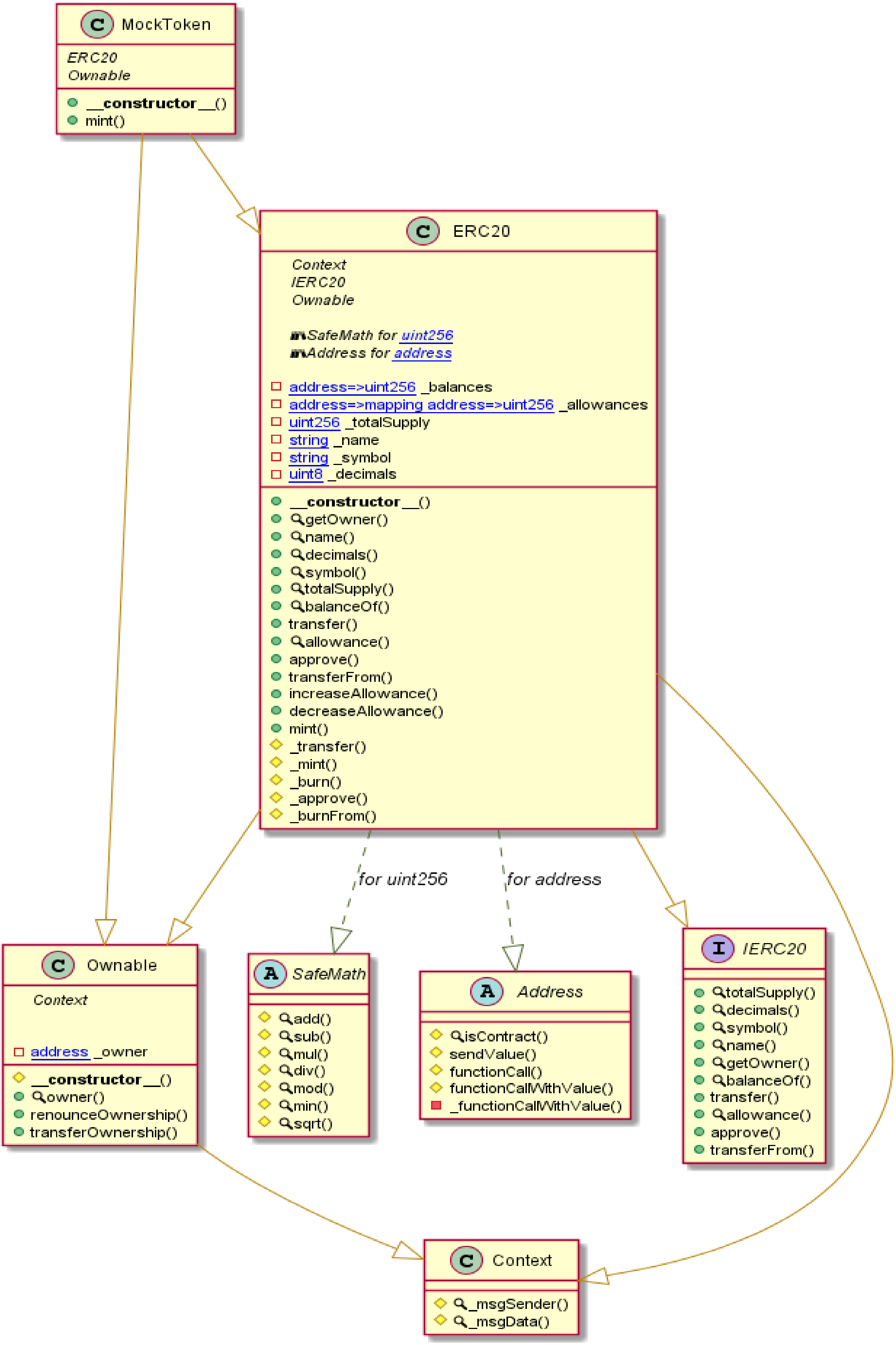
SwapMining Diagram



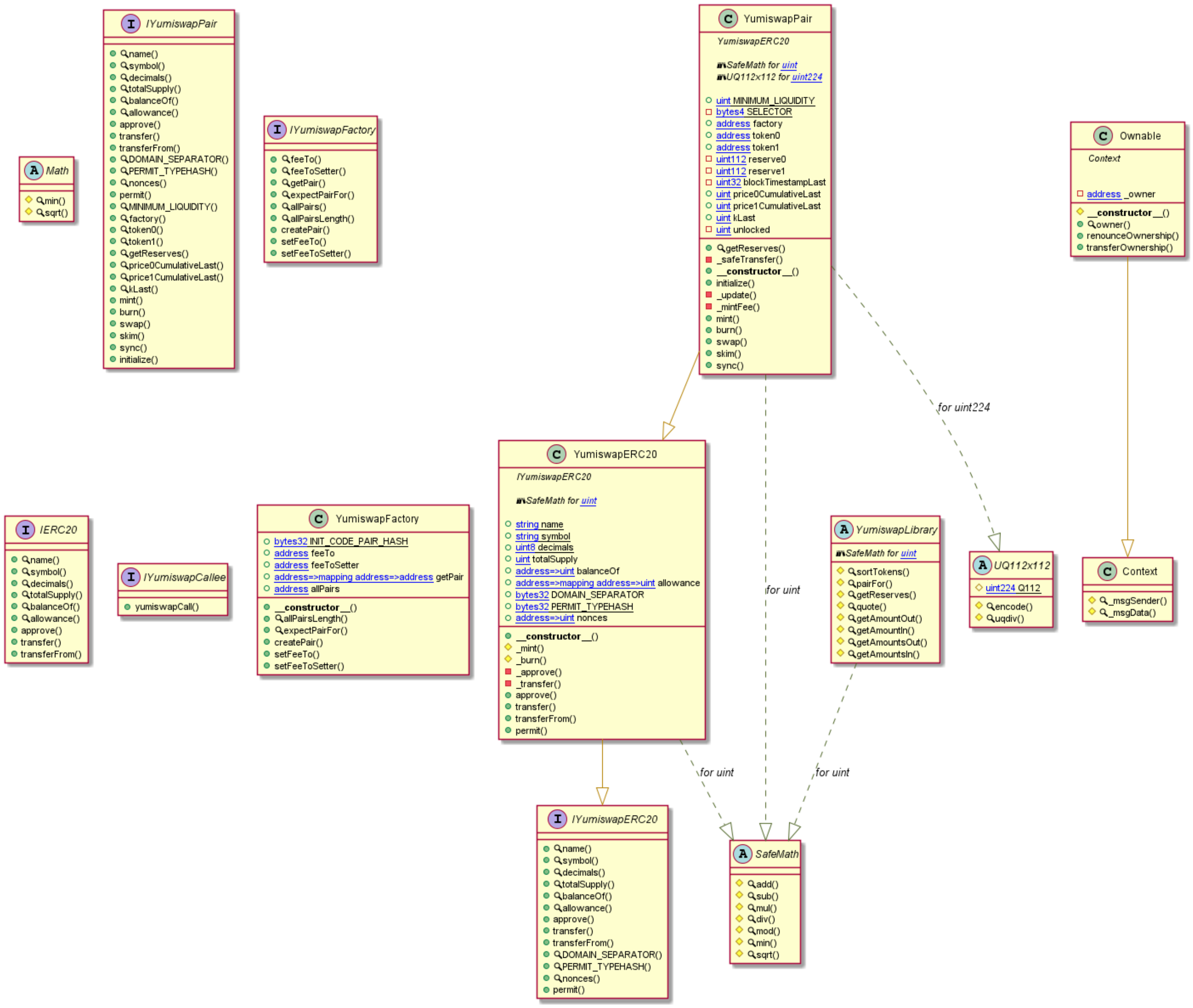
SyrupBar Diagram



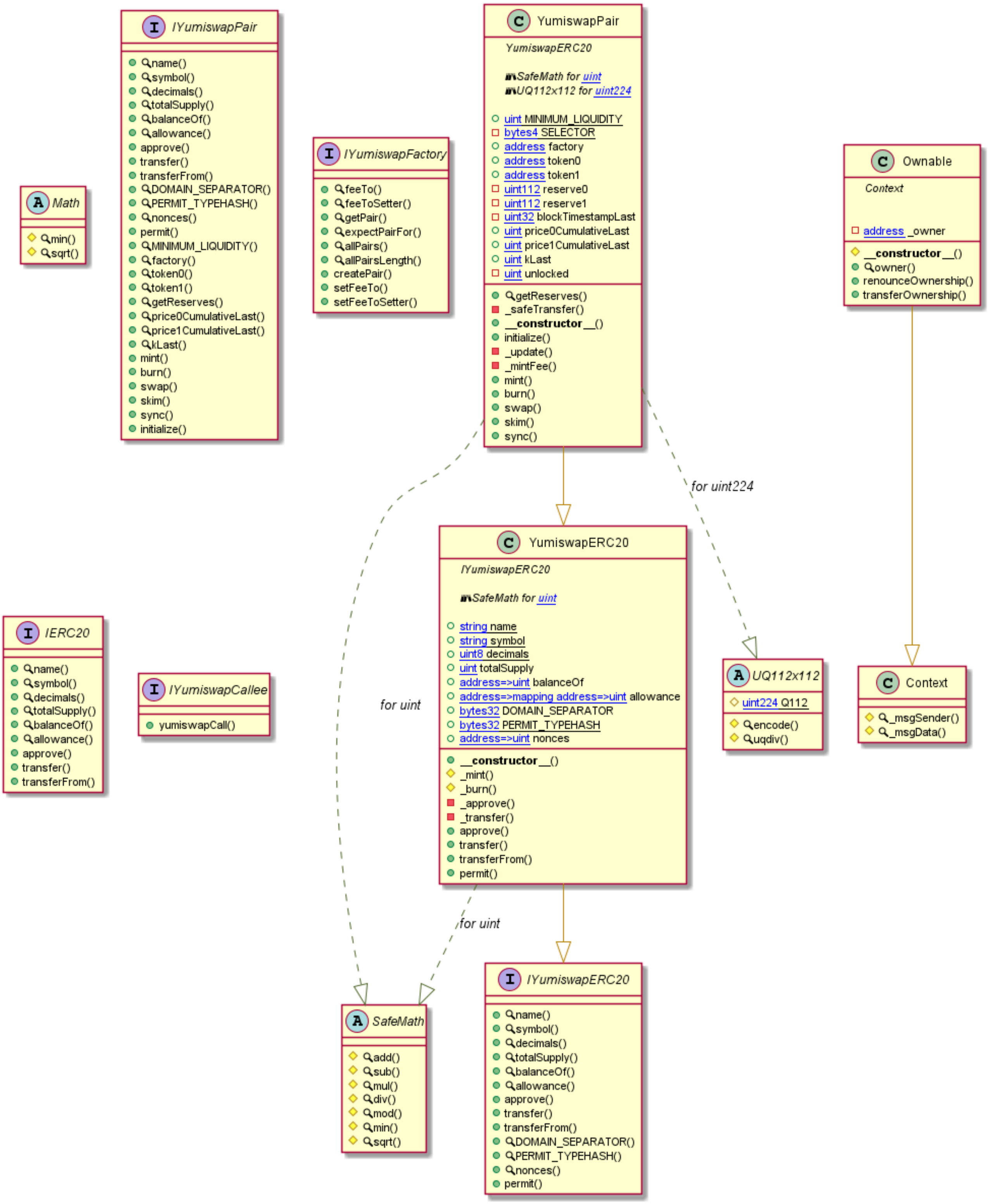
MockToken Diagram



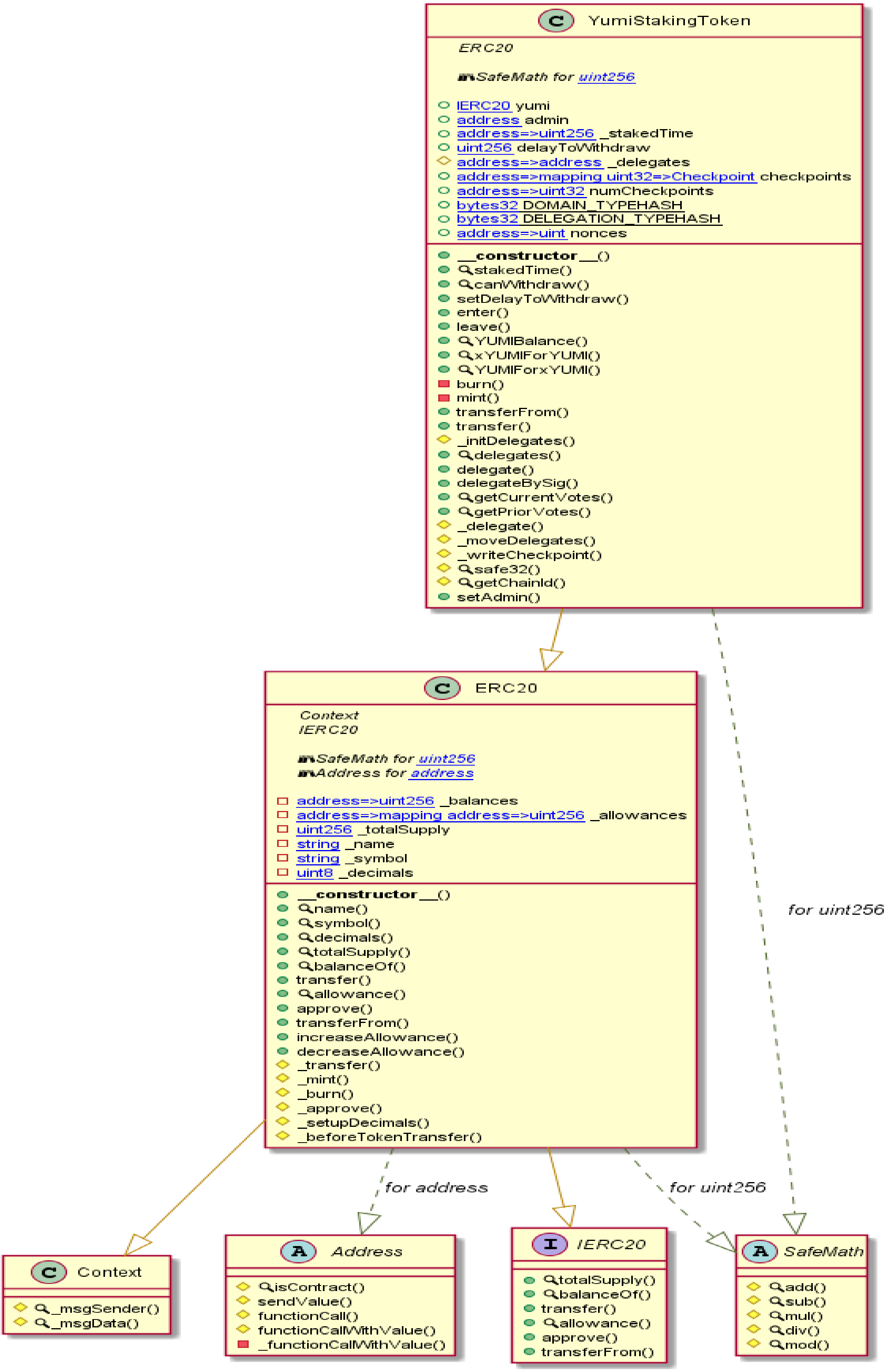
Factory Diagram



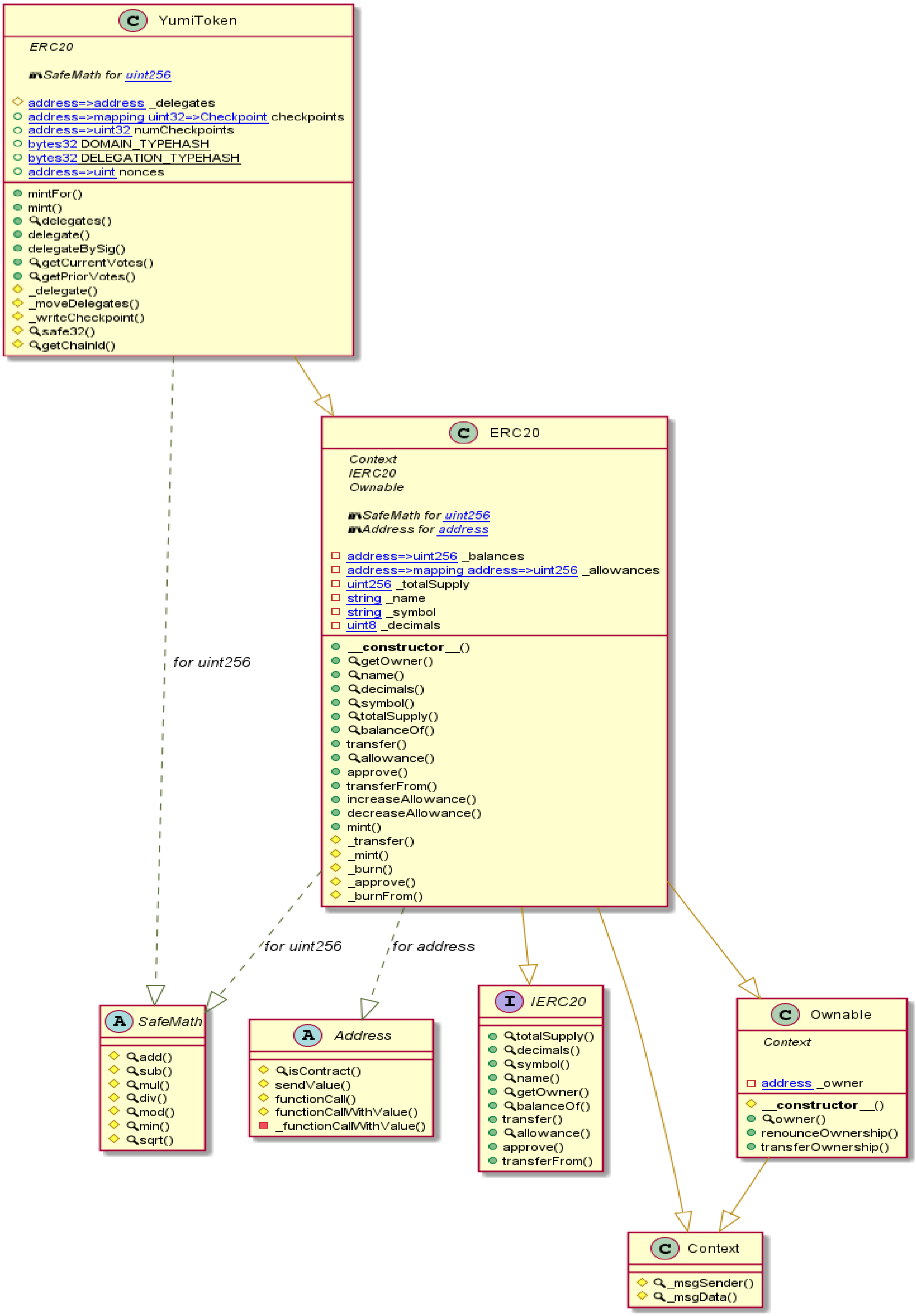
Pair Diagram



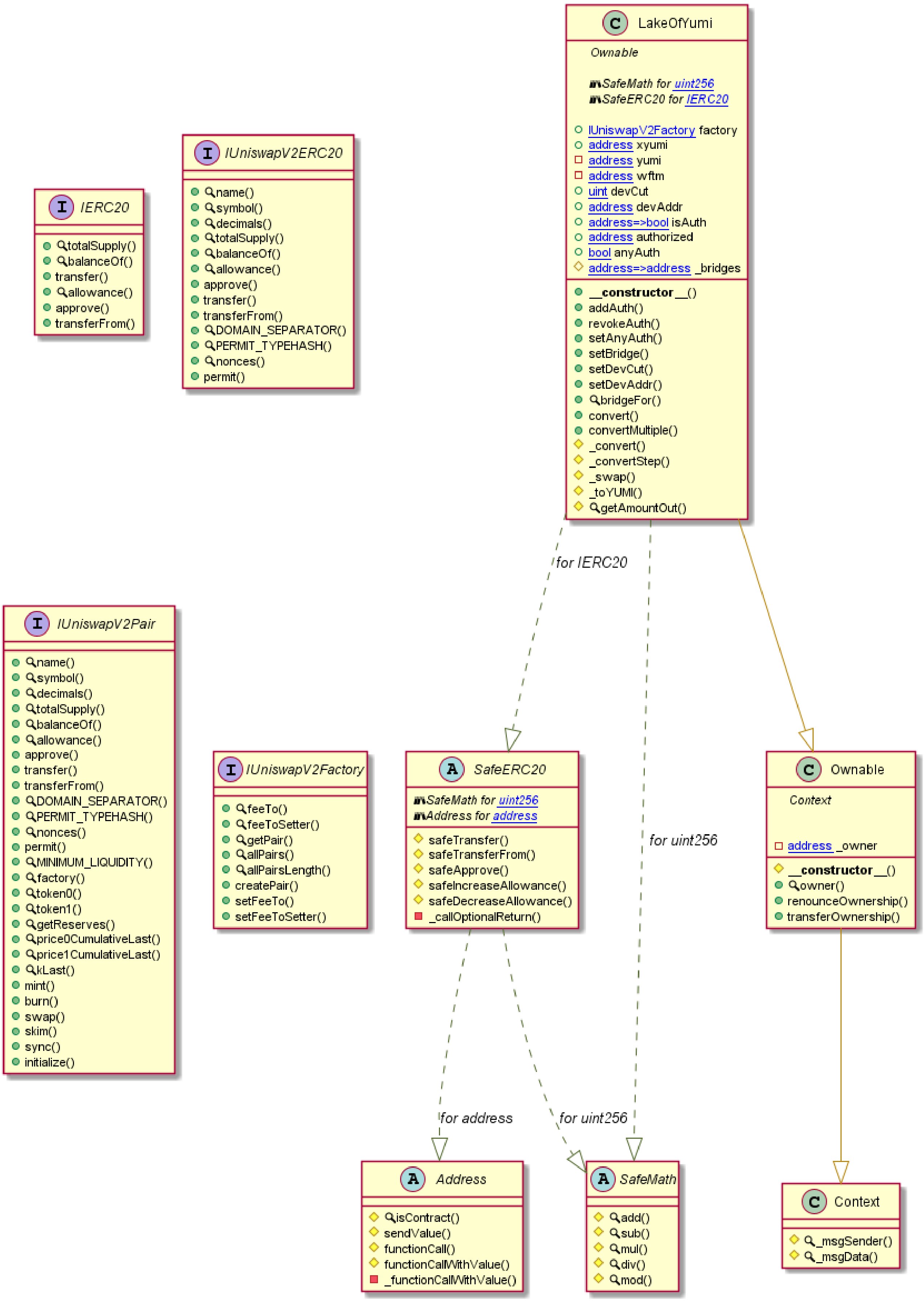
xYUMI Diagram



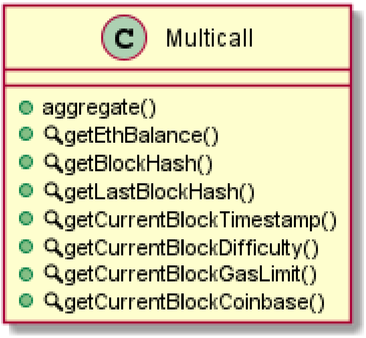
YumiToken Diagram



LakeOfYumi Diagram

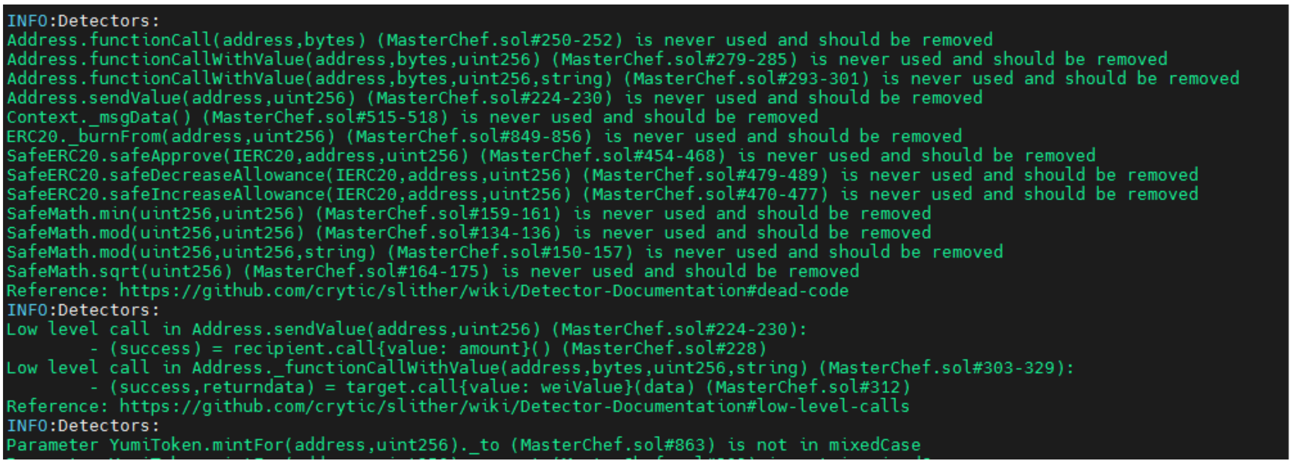


Multicall Diagram

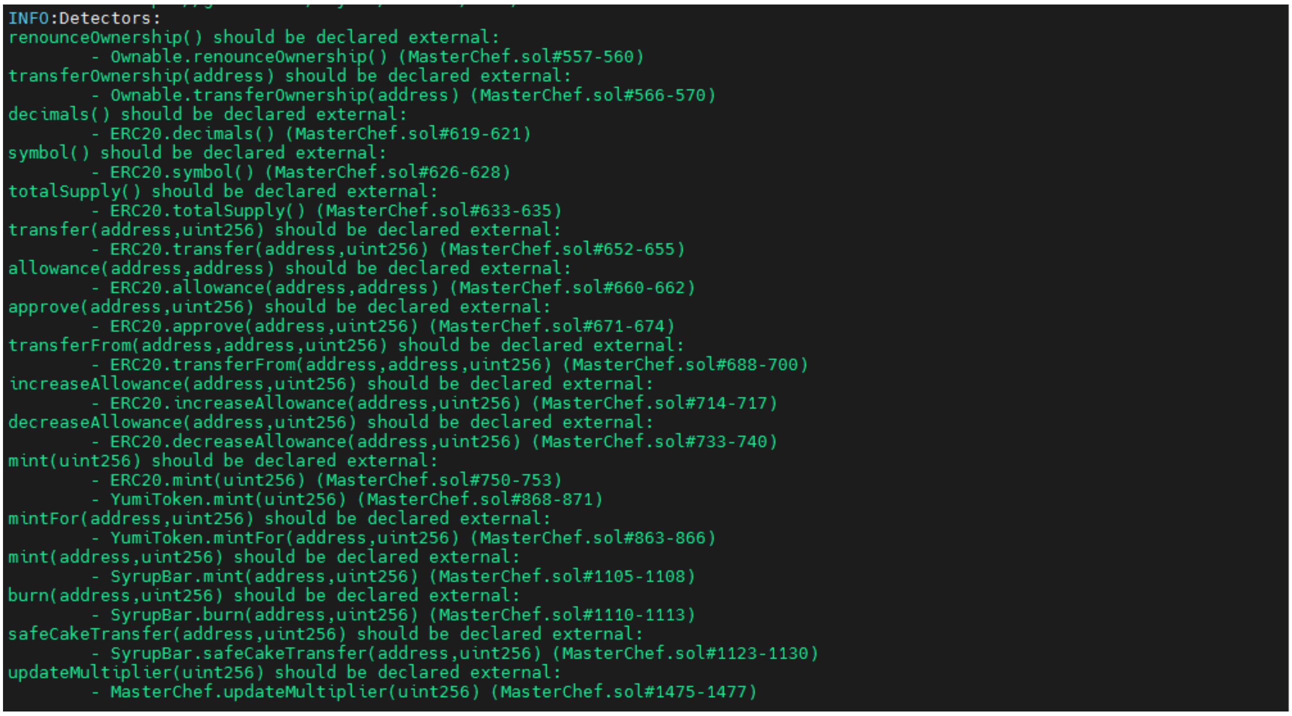

Multicall
aggregate()
getEthBalance()
getBlockHash()
getLastBlockHash()
getCurrentBlockTimestamp()
getCurrentBlockDifficulty()
getCurrentBlockGasLimit()
getCurrentBlockCoinbase()

Slither Results Log

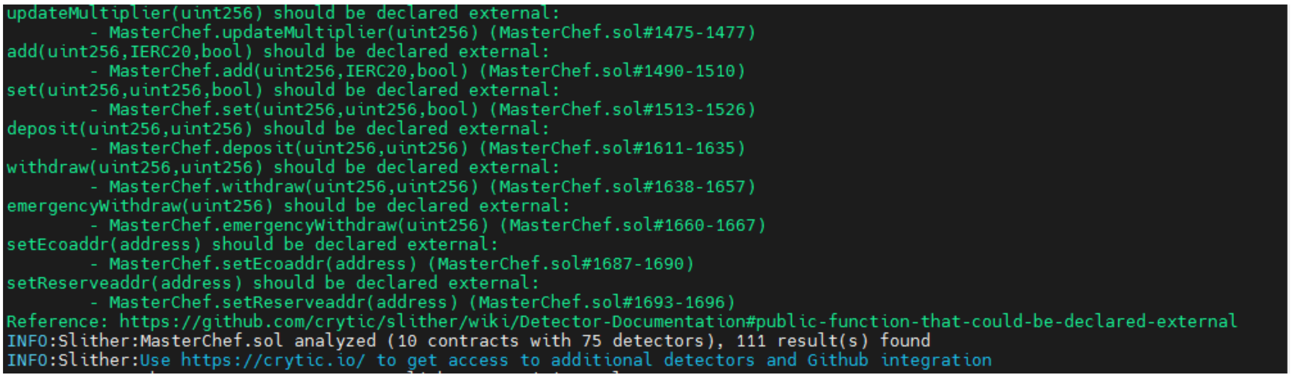
Slither log >> MasterChef.sol



```
INFO:Detectors:
Address.functionCall(address,bytes) (MasterChef.sol#250-252) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (MasterChef.sol#279-285) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#293-301) is never used and should be removed
Address.sendValue(address,uint256) (MasterChef.sol#224-230) is never used and should be removed
Context._msgData() (MasterChef.sol#515-518) is never used and should be removed
ERC20._burnFrom(address,uint256) (MasterChef.sol#849-856) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (MasterChef.sol#454-468) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (MasterChef.sol#479-489) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (MasterChef.sol#470-477) is never used and should be removed
SafeMath.min(uint256,uint256) (MasterChef.sol#159-161) is never used and should be removed
SafeMath.mod(uint256,uint256) (MasterChef.sol#134-136) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (MasterChef.sol#150-157) is never used and should be removed
SafeMath.sqrt(uint256) (MasterChef.sol#164-175) 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) (MasterChef.sol#224-230):
        - (success) = recipient.call{value: amount}() (MasterChef.sol#228)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#303-329):
        - (success,returndata) = target.call{value: weiValue}(data) (MasterChef.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter YumiToken.mintFor(address,uint256)._to (MasterChef.sol#863) is not in mixedCase
```

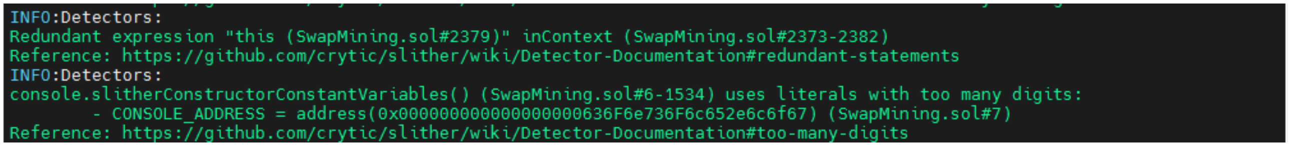


```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MasterChef.sol#557-560)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MasterChef.sol#566-570)
decimals() should be declared external:
        - ERC20.decimals() (MasterChef.sol#619-621)
symbol() should be declared external:
        - ERC20.symbol() (MasterChef.sol#626-628)
totalSupply() should be declared external:
        - ERC20.totalSupply() (MasterChef.sol#633-635)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (MasterChef.sol#652-655)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (MasterChef.sol#660-662)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (MasterChef.sol#671-674)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (MasterChef.sol#688-700)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (MasterChef.sol#714-717)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (MasterChef.sol#733-740)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (MasterChef.sol#750-753)
        - YumiToken.mint(uint256) (MasterChef.sol#868-871)
mintFor(address,uint256) should be declared external:
        - YumiToken.mintFor(address,uint256) (MasterChef.sol#863-866)
mint(address,uint256) should be declared external:
        - SyrupBar.mint(address,uint256) (MasterChef.sol#1105-1108)
burn(address,uint256) should be declared external:
        - SyrupBar.burn(address,uint256) (MasterChef.sol#1110-1113)
safeCakeTransfer(address,uint256) should be declared external:
        - SyrupBar.safeCakeTransfer(address,uint256) (MasterChef.sol#1123-1130)
updateMultiplier(uint256) should be declared external:
        - MasterChef.updateMultiplier(uint256) (MasterChef.sol#1475-1477)
```

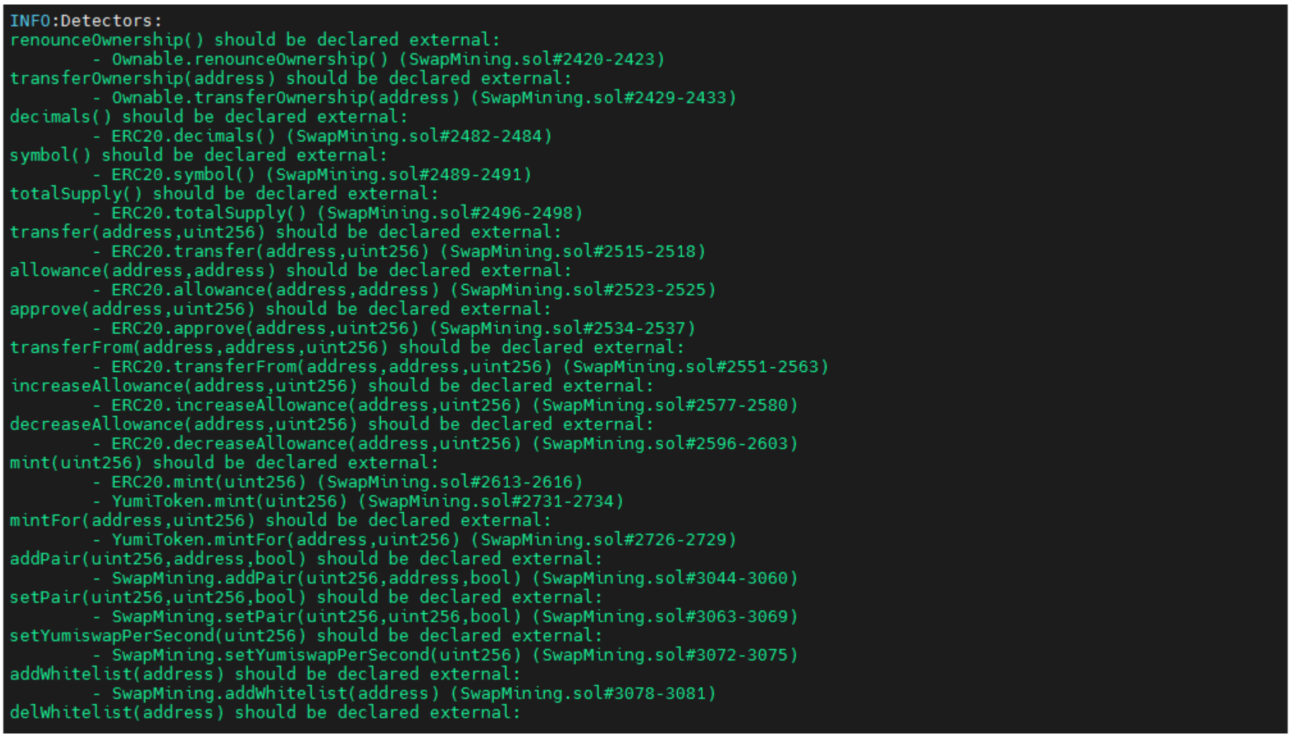


```
updateMultiplier(uint256) should be declared external:
        - MasterChef.updateMultiplier(uint256) (MasterChef.sol#1475-1477)
add(uint256,IERC20,bool) should be declared external:
        - MasterChef.add(uint256,IERC20,bool) (MasterChef.sol#1490-1510)
set(uint256,uint256,bool) should be declared external:
        - MasterChef.set(uint256,uint256,bool) (MasterChef.sol#1513-1526)
deposit(uint256,uint256) should be declared external:
        - MasterChef.deposit(uint256,uint256) (MasterChef.sol#1611-1635)
withdraw(uint256,uint256) should be declared external:
        - MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1638-1657)
emergencyWithdraw(uint256) should be declared external:
        - MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1660-1667)
setEcoaddr(address) should be declared external:
        - MasterChef.setEcoaddr(address) (MasterChef.sol#1687-1690)
setReserveaddr(address) should be declared external:
        - MasterChef.setReserveaddr(address) (MasterChef.sol#1693-1696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MasterChef.sol analyzed (10 contracts with 75 detectors), 111 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

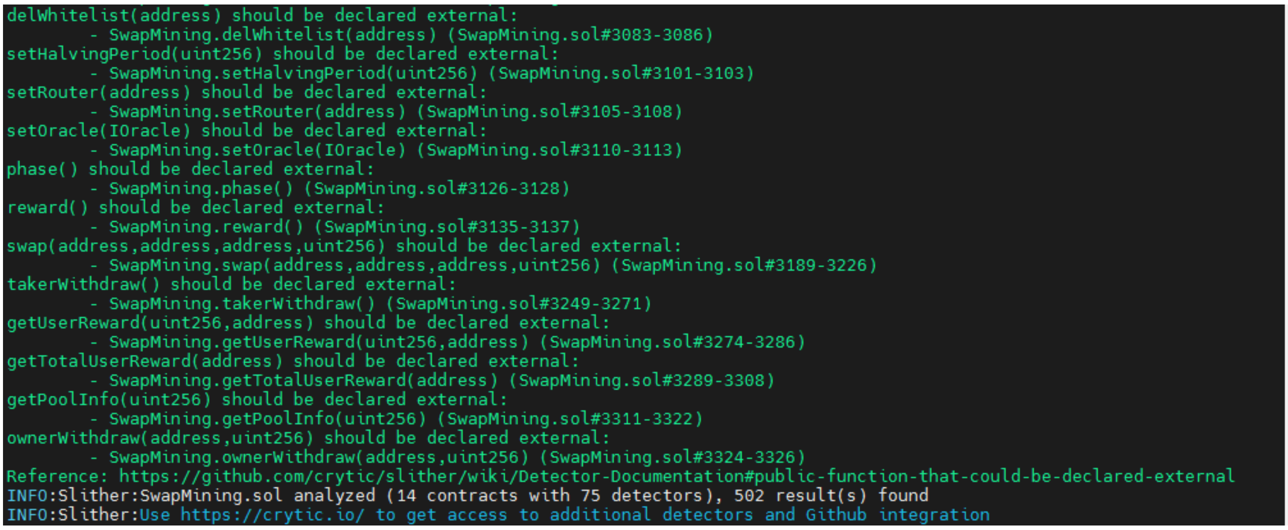
Slither log >> SwapMining.sol



```
INFO:Detectors:
Redundant expression "this (SwapMining.sol#2379)" inContext (SwapMining.sol#2373-2382)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
console.slitherConstructorConstantVariables() (SwapMining.sol#6-1534) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (SwapMining.sol#7)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

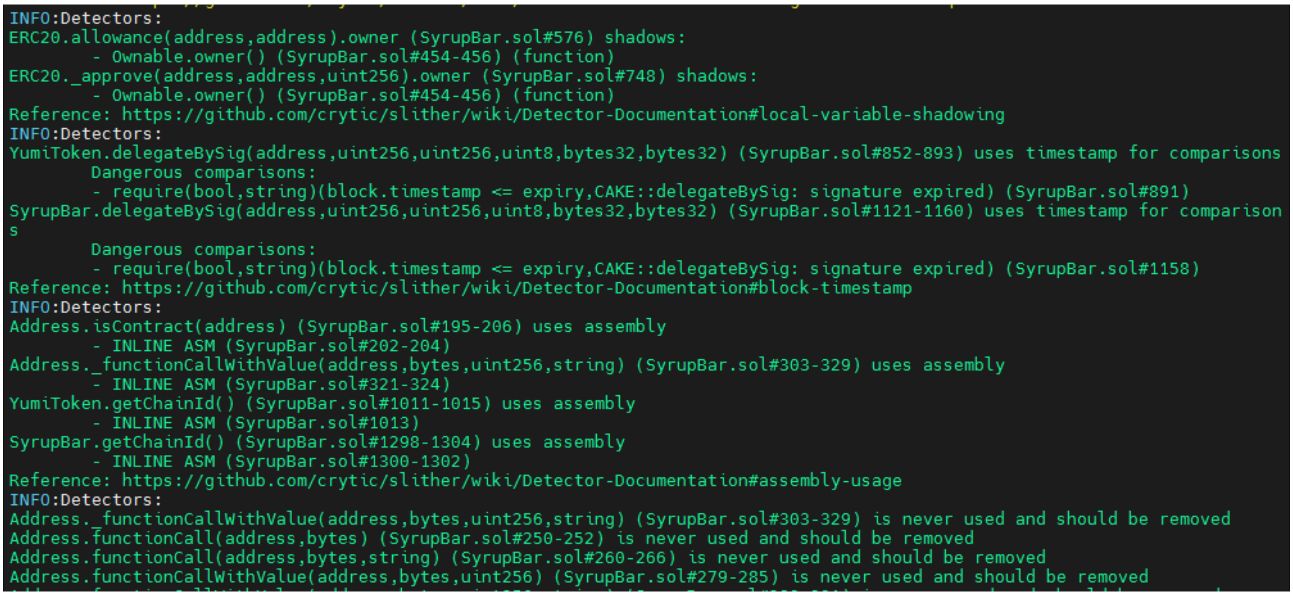


```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SwapMining.sol#2420-2423)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SwapMining.sol#2429-2433)
decimals() should be declared external:
        - ERC20.decimals() (SwapMining.sol#2482-2484)
symbol() should be declared external:
        - ERC20.symbol() (SwapMining.sol#2489-2491)
totalSupply() should be declared external:
        - ERC20.totalSupply() (SwapMining.sol#2496-2498)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (SwapMining.sol#2515-2518)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (SwapMining.sol#2523-2525)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (SwapMining.sol#2534-2537)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (SwapMining.sol#2551-2563)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (SwapMining.sol#2577-2580)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (SwapMining.sol#2596-2603)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (SwapMining.sol#2613-2616)
        - YumiToken.mint(uint256) (SwapMining.sol#2731-2734)
mintFor(address,uint256) should be declared external:
        - YumiToken.mintFor(address,uint256) (SwapMining.sol#2726-2729)
addPair(uint256,address,bool) should be declared external:
        - SwapMining.addPair(uint256,address,bool) (SwapMining.sol#3044-3060)
setPair(uint256,uint256,bool) should be declared external:
        - SwapMining.setPair(uint256,uint256,bool) (SwapMining.sol#3063-3069)
setYumiswapPerSecond(uint256) should be declared external:
        - SwapMining.setYumiswapPerSecond(uint256) (SwapMining.sol#3072-3075)
addWhitelist(address) should be declared external:
        - SwapMining.addWhitelist(address) (SwapMining.sol#3078-3081)
delWhitelist(address) should be declared external:
```

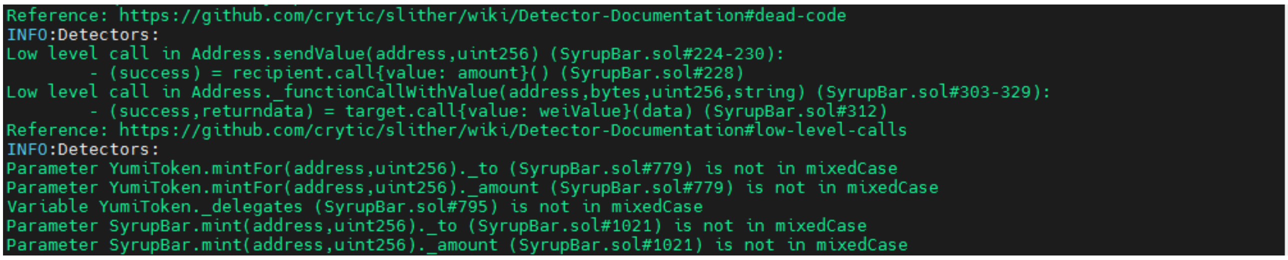


```
delWhitelist(address) should be declared external:
        - SwapMining.delWhitelist(address) (SwapMining.sol#3083-3086)
setHalvingPeriod(uint256) should be declared external:
        - SwapMining.setHalvingPeriod(uint256) (SwapMining.sol#3101-3103)
setRouter(address) should be declared external:
        - SwapMining.setRouter(address) (SwapMining.sol#3105-3108)
setOracle(IOracle) should be declared external:
        - SwapMining.setOracle(IOracle) (SwapMining.sol#3110-3113)
phase() should be declared external:
        - SwapMining.phase() (SwapMining.sol#3126-3128)
reward() should be declared external:
        - SwapMining.reward() (SwapMining.sol#3135-3137)
swap(address,address,address,uint256) should be declared external:
        - SwapMining.swap(address,address,address,uint256) (SwapMining.sol#3189-3226)
takerWithdraw() should be declared external:
        - SwapMining.takerWithdraw() (SwapMining.sol#3249-3271)
getUserReward(uint256,address) should be declared external:
        - SwapMining.getUserReward(uint256,address) (SwapMining.sol#3274-3286)
getTotalUserReward(address) should be declared external:
        - SwapMining.getTotalUserReward(address) (SwapMining.sol#3289-3308)
getPoolInfo(uint256) should be declared external:
        - SwapMining.getPoolInfo(uint256) (SwapMining.sol#3311-3322)
ownerWithdraw(address,uint256) should be declared external:
        - SwapMining.ownerWithdraw(address,uint256) (SwapMining.sol#3324-3326)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SwapMining.sol analyzed (14 contracts with 75 detectors), 502 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

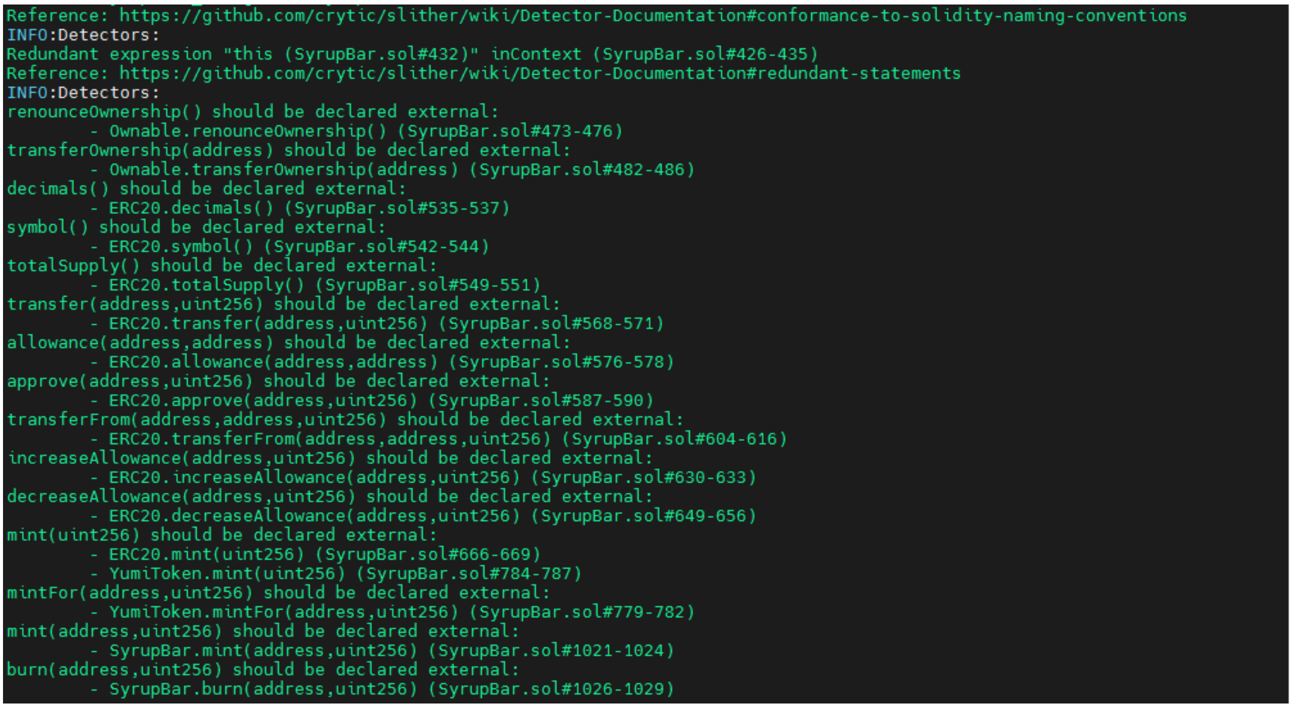
Slither log >> SyrupBar.sol



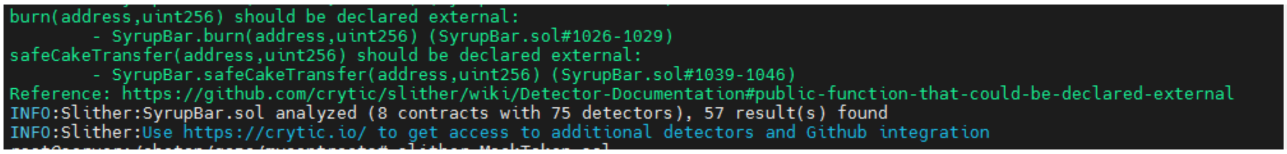
```
INFO:Detectors:
ERC20.allowance(address,address).owner (SyrupBar.sol#576) shadows:
        - Ownable.owner() (SyrupBar.sol#454-456) (function)
ERC20._approve(address,address,uint256).owner (SyrupBar.sol#748) shadows:
        - Ownable.owner() (SyrupBar.sol#454-456) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
YumiToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#852-893) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (SyrupBar.sol#891)
SyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#1121-1160) uses timestamp for comparison
s
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (SyrupBar.sol#1158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (SyrupBar.sol#195-206) uses assembly
        - INLINE ASM (SyrupBar.sol#202-204)
Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#303-329) uses assembly
        - INLINE ASM (SyrupBar.sol#321-324)
YumiToken.getChainId() (SyrupBar.sol#1011-1015) uses assembly
        - INLINE ASM (SyrupBar.sol#1013)
SyrupBar.getChainId() (SyrupBar.sol#1298-1304) uses assembly
        - INLINE ASM (SyrupBar.sol#1300-1302)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#303-329) is never used and should be removed
Address.functionCall(address,bytes) (SyrupBar.sol#250-252) is never used and should be removed
Address.functionCall(address,bytes,string) (SyrupBar.sol#260-266) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (SyrupBar.sol#279-285) 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#224-230):
        - (success) = recipient.call{value: amount}() (SyrupBar.sol#228)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#303-329):
        - (success,returndata) = target.call{value: weiValue}(data) (SyrupBar.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter YumiToken.mintFor(address,uint256)._to (SyrupBar.sol#779) is not in mixedCase
Parameter YumiToken.mintFor(address,uint256)._amount (SyrupBar.sol#779) is not in mixedCase
Variable YumiToken._delegates (SyrupBar.sol#795) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._to (SyrupBar.sol#1021) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._amount (SyrupBar.sol#1021) is not in mixedCase
```

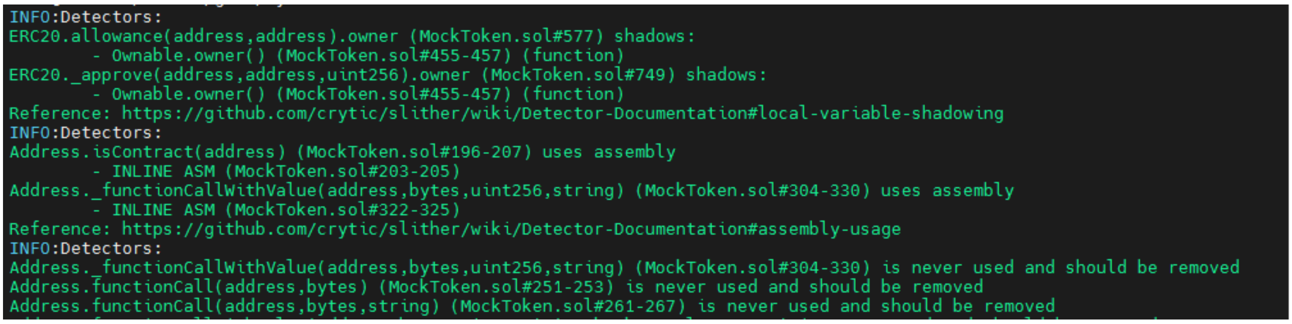


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (SyrupBar.sol#432)" inContext (SyrupBar.sol#426-435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SyrupBar.sol#473-476)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SyrupBar.sol#482-486)
decimals() should be declared external:
        - ERC20.decimals() (SyrupBar.sol#535-537)
symbol() should be declared external:
        - ERC20.symbol() (SyrupBar.sol#542-544)
totalSupply() should be declared external:
        - ERC20.totalSupply() (SyrupBar.sol#549-551)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (SyrupBar.sol#568-571)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (SyrupBar.sol#576-578)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (SyrupBar.sol#587-590)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (SyrupBar.sol#604-616)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (SyrupBar.sol#630-633)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (SyrupBar.sol#649-656)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (SyrupBar.sol#666-669)
        - YumiToken.mint(uint256) (SyrupBar.sol#784-787)
mintFor(address,uint256) should be declared external:
        - YumiToken.mintFor(address,uint256) (SyrupBar.sol#779-782)
mint(address,uint256) should be declared external:
        - SyrupBar.mint(address,uint256) (SyrupBar.sol#1021-1024)
burn(address,uint256) should be declared external:
        - SyrupBar.burn(address,uint256) (SyrupBar.sol#1026-1029)
```

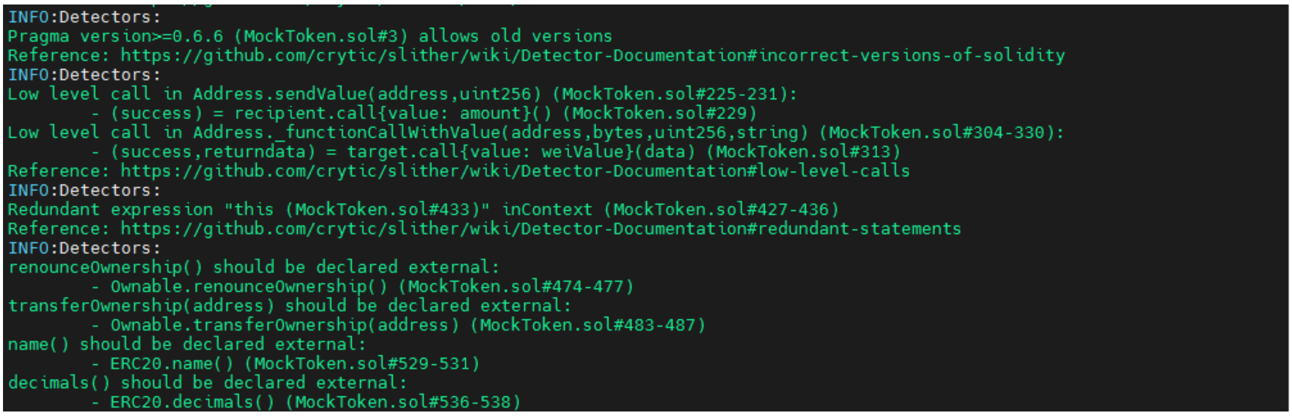


```
burn(address,uint256) should be declared external:
        - SyrupBar.burn(address,uint256) (SyrupBar.sol#1026-1029)
safeCakeTransfer(address,uint256) should be declared external:
        - SyrupBar.safeCakeTransfer(address,uint256) (SyrupBar.sol#1039-1046)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SyrupBar.sol analyzed (8 contracts with 75 detectors), 57 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

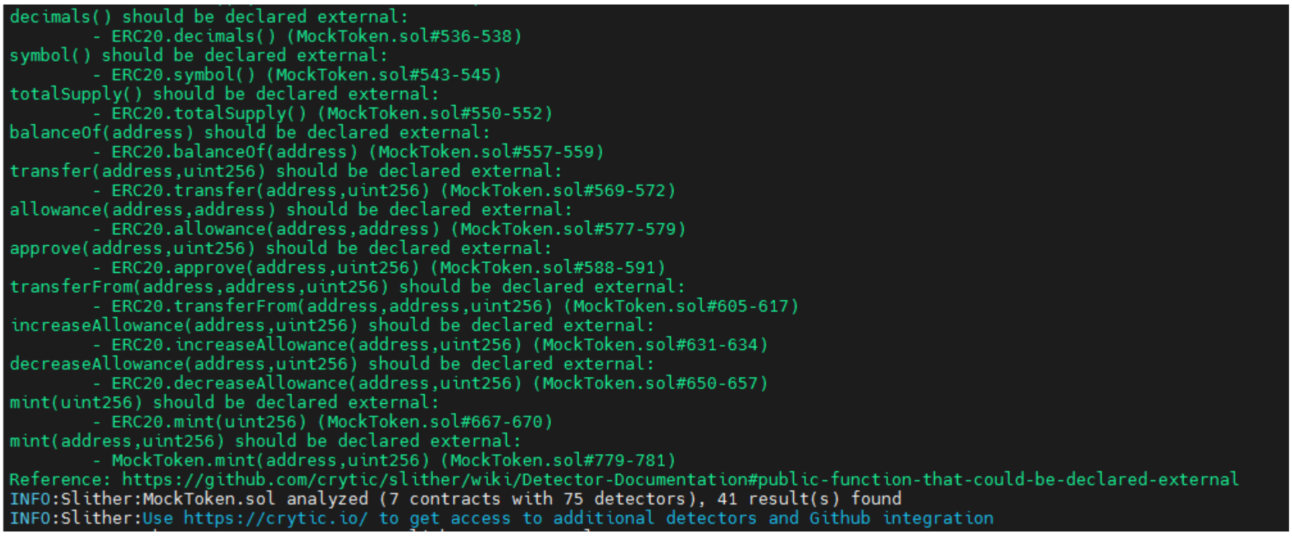
Slither log >> MockToken.sol



```
INFO:Detectors:
ERC20.allowance(address,address).owner (MockToken.sol#577) shadows:
        - Ownable.owner() (MockToken.sol#455-457) (function)
ERC20._approve(address,address,uint256).owner (MockToken.sol#749) shadows:
        - Ownable.owner() (MockToken.sol#455-457) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Address.isContract(address) (MockToken.sol#196-207) uses assembly
        - INLINE ASM (MockToken.sol#203-205)
Address._functionCallWithValue(address,bytes,uint256,string) (MockToken.sol#304-330) uses assembly
        - INLINE ASM (MockToken.sol#322-325)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (MockToken.sol#304-330) is never used and should be removed
Address.functionCall(address,bytes) (MockToken.sol#251-253) is never used and should be removed
Address.functionCall(address,bytes,string) (MockToken.sol#261-267) is never used and should be removed
```

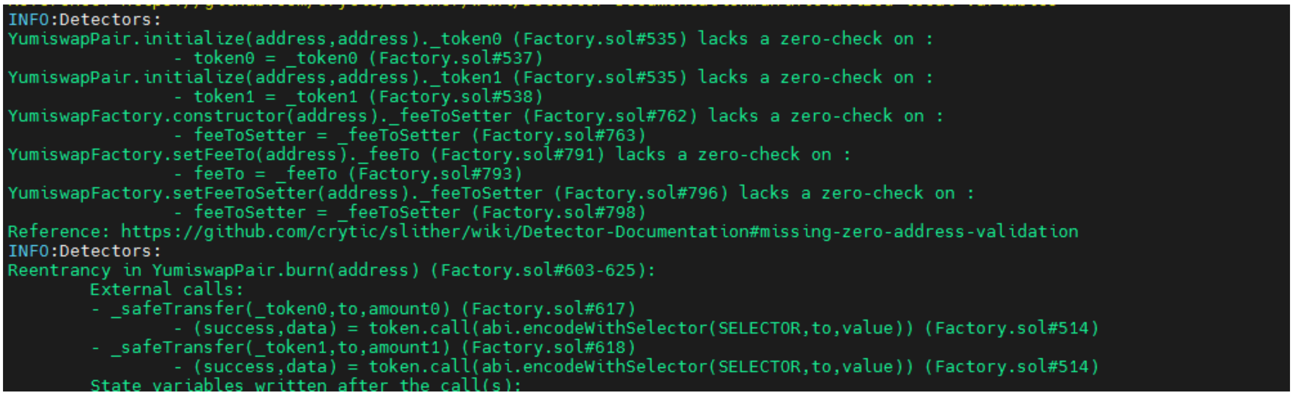


```
INFO:Detectors:
Pragma version>=0.6.6 (MockToken.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MockToken.sol#225-231):
        - (success) = recipient.call{value: amount}() (MockToken.sol#229)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (MockToken.sol#304-330):
        - (success,returndata) = target.call{value: weiValue}(data) (MockToken.sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Redundant expression "this (MockToken.sol#433)" inContext (MockToken.sol#427-436)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MockToken.sol#474-477)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MockToken.sol#483-487)
name() should be declared external:
        - ERC20.name() (MockToken.sol#529-531)
decimals() should be declared external:
        - ERC20.decimals() (MockToken.sol#536-538)
```

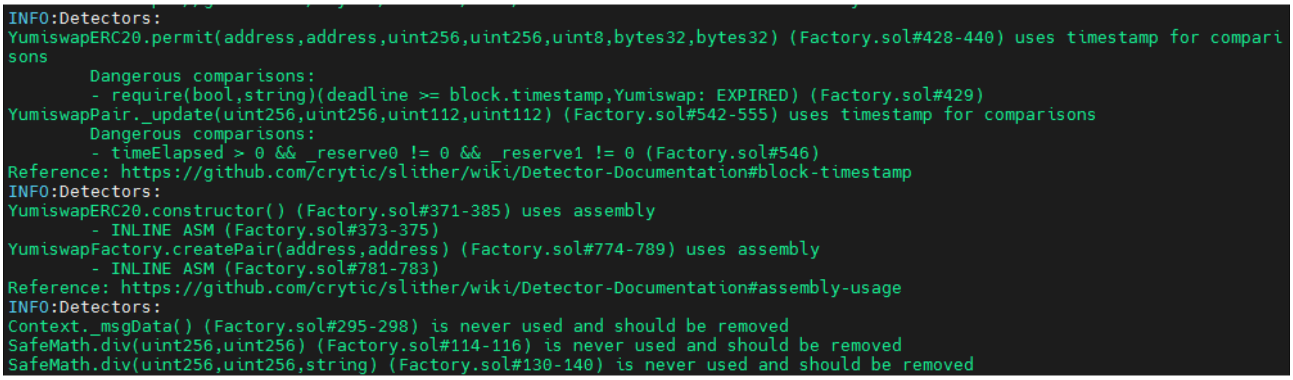


```
decimals() should be declared external:
        - ERC20.decimals() (MockToken.sol#536-538)
symbol() should be declared external:
        - ERC20.symbol() (MockToken.sol#543-545)
totalSupply() should be declared external:
        - ERC20.totalSupply() (MockToken.sol#550-552)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (MockToken.sol#557-559)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (MockToken.sol#569-572)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (MockToken.sol#577-579)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (MockToken.sol#588-591)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (MockToken.sol#605-617)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (MockToken.sol#631-634)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (MockToken.sol#650-657)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (MockToken.sol#667-670)
mint(address,uint256) should be declared external:
        - MockToken.mint(address,uint256) (MockToken.sol#779-781)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MockToken.sol analyzed (7 contracts with 75 detectors), 41 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

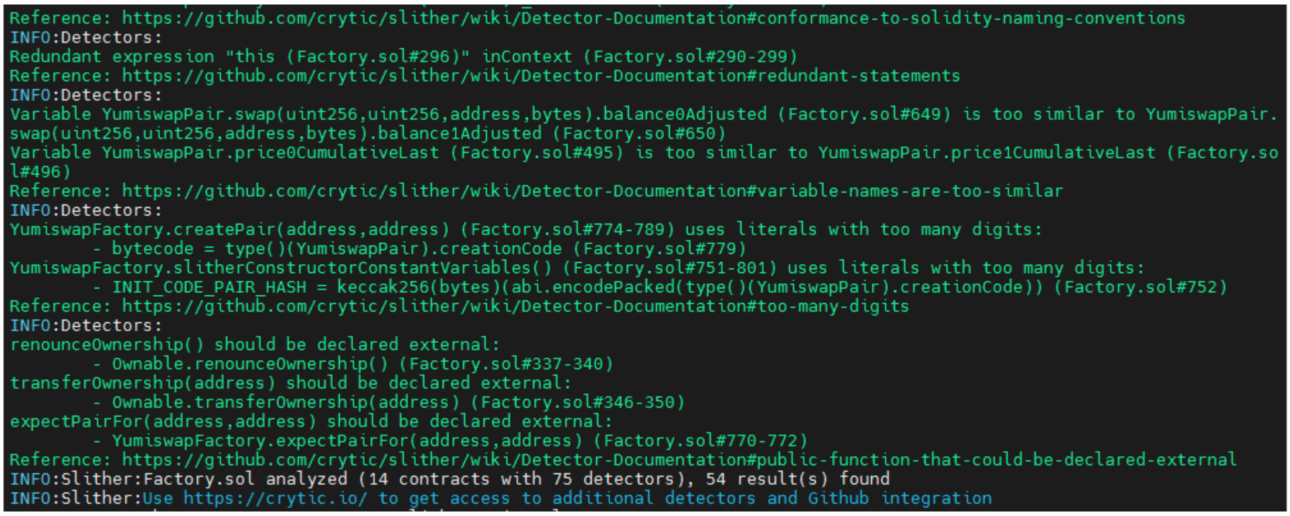
Slither log >> Factory.sol



```
INFO:Detectors:
YumiswapPair.initialize(address,address)._token0 (Factory.sol#535) lacks a zero-check on :
                - token0 = _token0 (Factory.sol#537)
YumiswapPair.initialize(address,address)._token1 (Factory.sol#535) lacks a zero-check on :
                - token1 = _token1 (Factory.sol#538)
YumiswapFactory.constructor(address)._feeToSetter (Factory.sol#762) lacks a zero-check on :
                - feeToSetter = _feeToSetter (Factory.sol#763)
YumiswapFactory.setFeeTo(address)._feeTo (Factory.sol#791) lacks a zero-check on :
                - feeTo = _feeTo (Factory.sol#793)
YumiswapFactory.setFeeToSetter(address)._feeToSetter (Factory.sol#796) lacks a zero-check on :
                - feeToSetter = _feeToSetter (Factory.sol#798)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in YumiswapPair.burn(address) (Factory.sol#603-625):
        External calls:
        - _safeTransfer(_token0,to,amount0) (Factory.sol#617)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Factory.sol#514)
        - _safeTransfer(_token1,to,amount1) (Factory.sol#618)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Factory.sol#514)
        State variables written after the call(s):
```

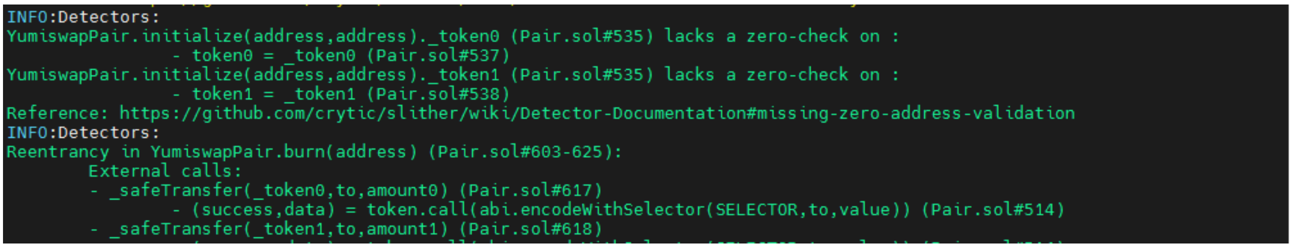


```
INFO:Detectors:
YumiswapERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (Factory.sol#428-440) uses timestamp for compari
sons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Yumiswap: EXPIRED) (Factory.sol#429)
YumiswapPair._update(uint256,uint256,uint112,uint112) (Factory.sol#542-555) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (Factory.sol#546)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
YumiswapERC20.constructor() (Factory.sol#371-385) uses assembly
        - INLINE ASM (Factory.sol#373-375)
YumiswapFactory.createPair(address,address) (Factory.sol#774-789) uses assembly
        - INLINE ASM (Factory.sol#781-783)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Context._msgData() (Factory.sol#295-298) is never used and should be removed
SafeMath.div(uint256,uint256) (Factory.sol#114-116) is never used and should be removed
SafeMath.div(uint256,uint256,string) (Factory.sol#130-140) is never used and should be removed
```

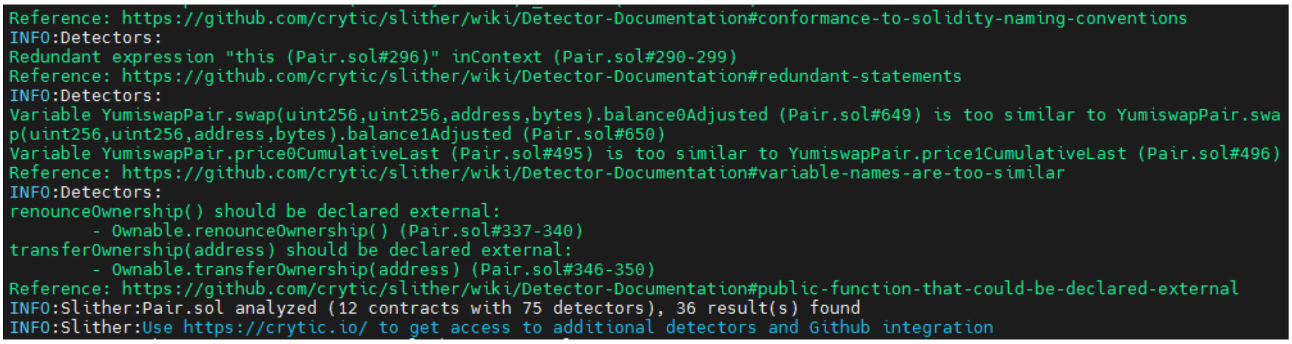


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Factory.sol#296)" inContext (Factory.sol#290-299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable YumiswapPair.swap(uint256,uint256,address,bytes).balance0Adjusted (Factory.sol#649) is too similar to YumiswapPair.
swap(uint256,uint256,address,bytes).balance1Adjusted (Factory.sol#650)
Variable YumiswapPair.price0CumulativeLast (Factory.sol#495) is too similar to YumiswapPair.price1CumulativeLast (Factory.so
l#496)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
YumiswapFactory.createPair(address,address) (Factory.sol#774-789) uses literals with too many digits:
        - bytecode = type()(YumiswapPair).creationCode (Factory.sol#779)
YumiswapFactory.slitherConstructorConstantVariables() (Factory.sol#751-801) uses literals with too many digits:
        - INIT_CODE_PAIR_HASH = keccak256(bytes)(abi.encodePacked(type()(YumiswapPair).creationCode)) (Factory.sol#752)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Factory.sol#337-340)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Factory.sol#346-350)
expectPairFor(address,address) should be declared external:
        - YumiswapFactory.expectPairFor(address,address) (Factory.sol#770-772)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Factory.sol analyzed (14 contracts with 75 detectors), 54 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Pair.sol

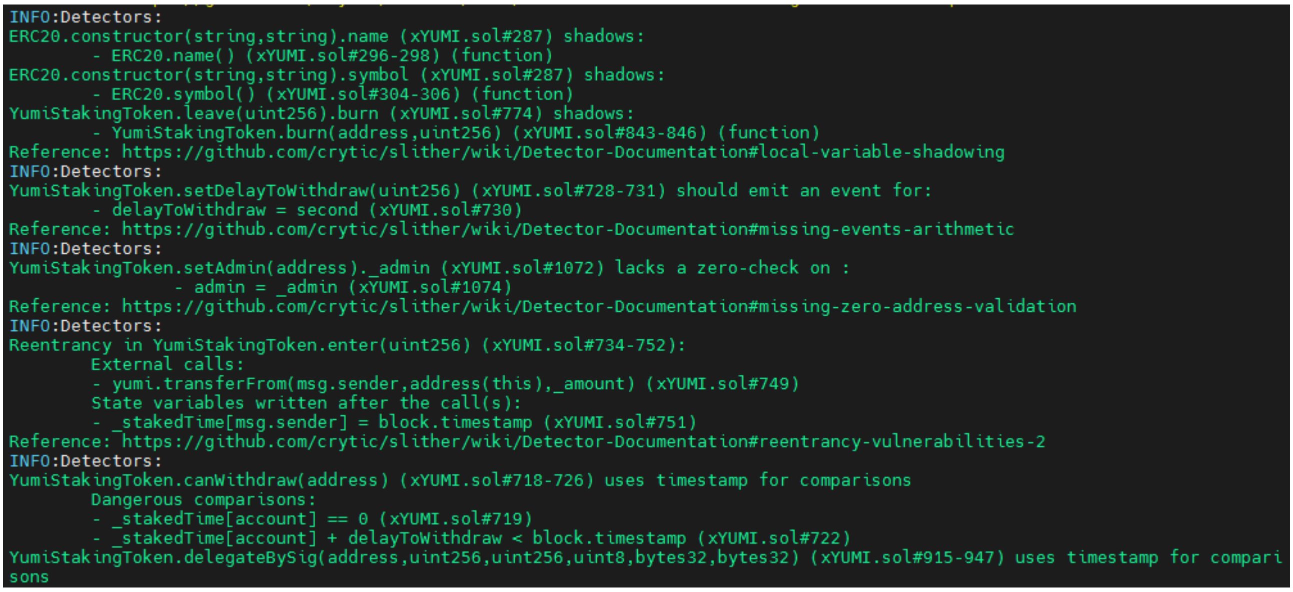


```
INFO:Detectors:
YumiswapPair.initialize(address,address)._token0 (Pair.sol#535) lacks a zero-check on :
                - token0 = _token0 (Pair.sol#537)
YumiswapPair.initialize(address,address)._token1 (Pair.sol#535) lacks a zero-check on :
                - token1 = _token1 (Pair.sol#538)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in YumiswapPair.burn(address) (Pair.sol#603-625):
        External calls:
        - _safeTransfer(_token0,to,amount0) (Pair.sol#617)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#514)
        - _safeTransfer(_token1,to,amount1) (Pair.sol#618)
```

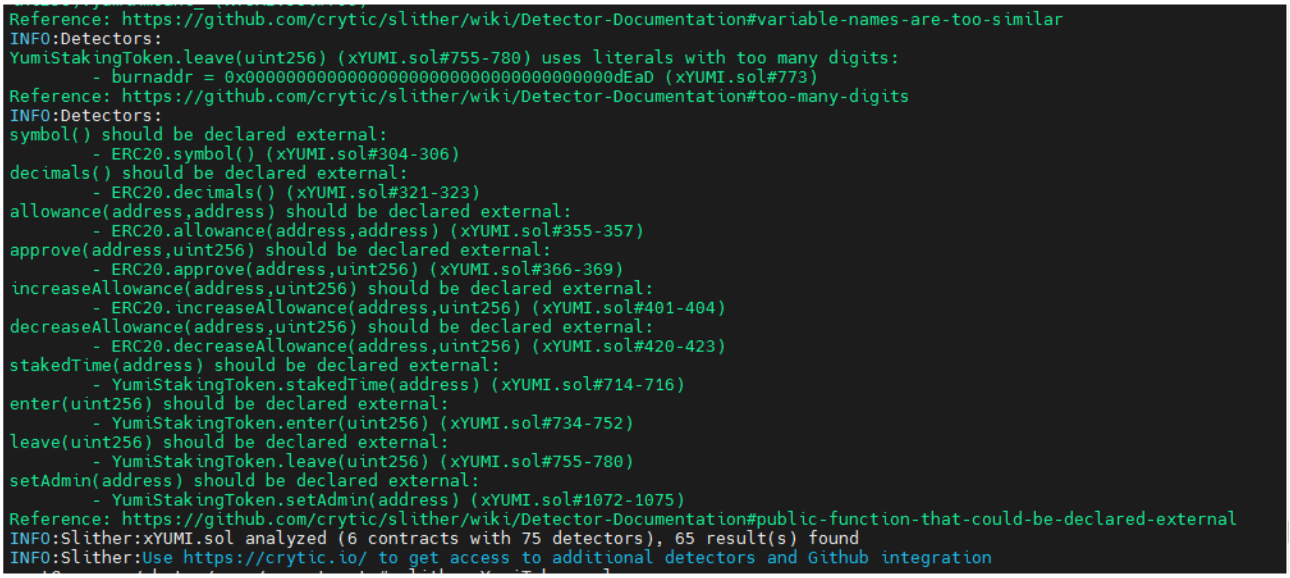


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Pair.sol#296)" inContext (Pair.sol#290-299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable YumiswapPair.swap(uint256,uint256,address,bytes).balance0Adjusted (Pair.sol#649) is too similar to YumiswapPair.swa
p(uint256,uint256,address,bytes).balance1Adjusted (Pair.sol#650)
Variable YumiswapPair.price0CumulativeLast (Pair.sol#495) is too similar to YumiswapPair.price1CumulativeLast (Pair.sol#496)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Pair.sol#337-340)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Pair.sol#346-350)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Pair.sol analyzed (12 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 >> xYUMI.sol

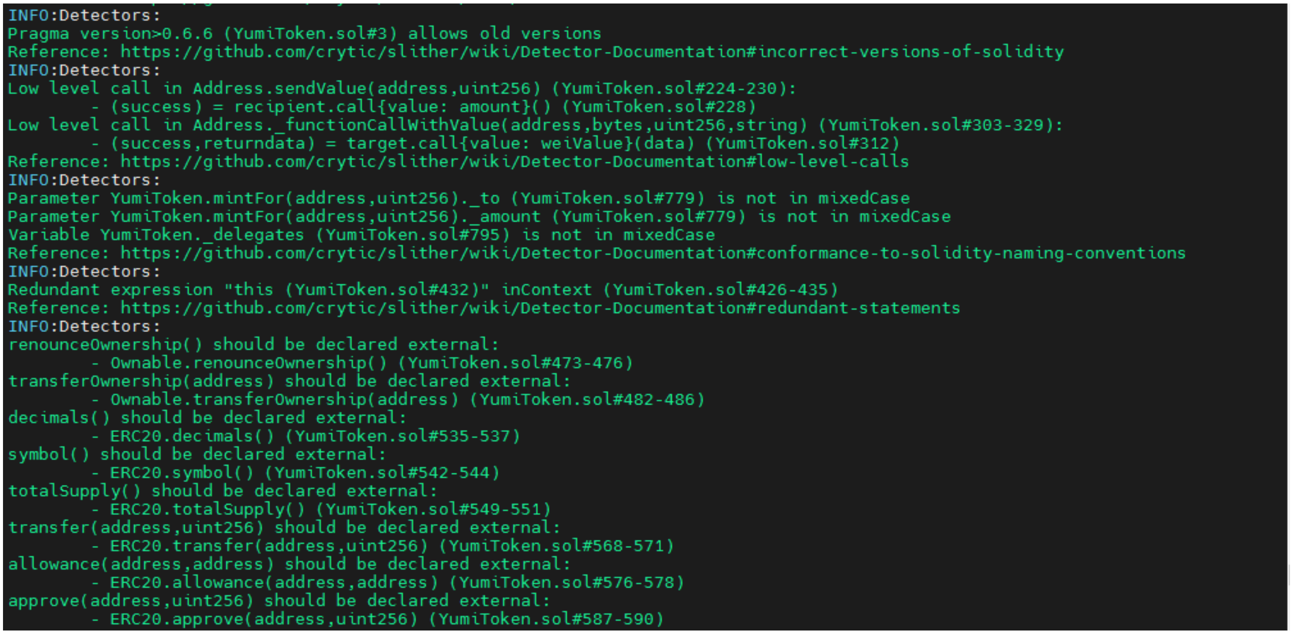


```
INFO:Detectors:
ERC20.constructor(string,string).name (xYUMI.sol#287) shadows:
        - ERC20.name() (xYUMI.sol#296-298) (function)
ERC20.constructor(string,string).symbol (xYUMI.sol#287) shadows:
        - ERC20.symbol() (xYUMI.sol#304-306) (function)
YumiStakingToken.leave(uint256).burn (xYUMI.sol#774) shadows:
        - YumiStakingToken.burn(address,uint256) (xYUMI.sol#843-846) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
YumiStakingToken.setDelayToWithdraw(uint256) (xYUMI.sol#728-731) should emit an event for:
        - delayToWithdraw = second (xYUMI.sol#730)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
YumiStakingToken.setAdmin(address)._admin (xYUMI.sol#1072) lacks a zero-check on :
                - admin = _admin (xYUMI.sol#1074)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in YumiStakingToken.enter(uint256) (xYUMI.sol#734-752):
        External calls:
        - yumi.transferFrom(msg.sender,address(this),_amount) (xYUMI.sol#749)
        State variables written after the call(s):
        - _stakedTime[msg.sender] = block.timestamp (xYUMI.sol#751)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
YumiStakingToken.canWithdraw(address) (xYUMI.sol#718-726) uses timestamp for comparisons
        Dangerous comparisons:
        - _stakedTime[account] == 0 (xYUMI.sol#719)
        - _stakedTime[account] + delayToWithdraw < block.timestamp (xYUMI.sol#722)
YumiStakingToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (xYUMI.sol#915-947) uses timestamp for compari
sons
```

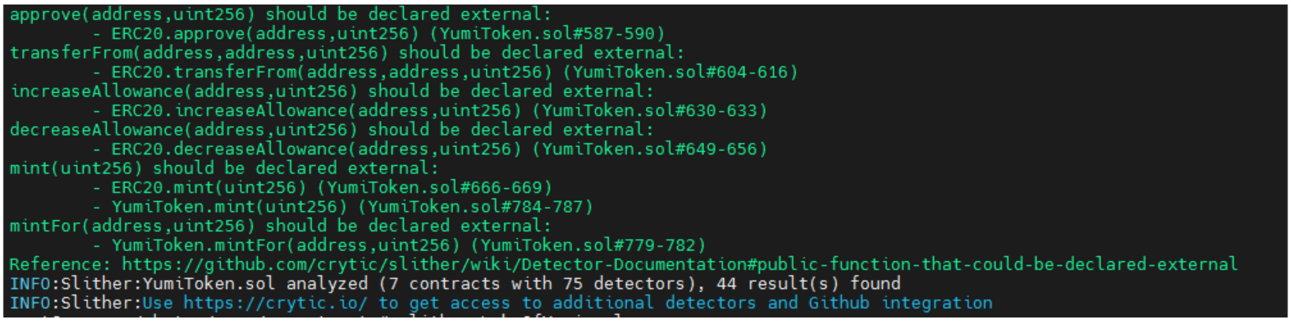


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
YumiStakingToken.leave(uint256) (xYUMI.sol#755-780) uses literals with too many digits:
        - burnaddr = 0x000000000000000000000000000000000000dEaD (xYUMI.sol#773)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
symbol() should be declared external:
        - ERC20.symbol() (xYUMI.sol#304-306)
decimals() should be declared external:
        - ERC20.decimals() (xYUMI.sol#321-323)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (xYUMI.sol#355-357)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (xYUMI.sol#366-369)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (xYUMI.sol#401-404)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (xYUMI.sol#420-423)
stakedTime(address) should be declared external:
        - YumiStakingToken.stakedTime(address) (xYUMI.sol#714-716)
enter(uint256) should be declared external:
        - YumiStakingToken.enter(uint256) (xYUMI.sol#734-752)
leave(uint256) should be declared external:
        - YumiStakingToken.leave(uint256) (xYUMI.sol#755-780)
setAdmin(address) should be declared external:
        - YumiStakingToken.setAdmin(address) (xYUMI.sol#1072-1075)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:xYUMI.sol analyzed (6 contracts with 75 detectors), 65 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> YumiToken.sol

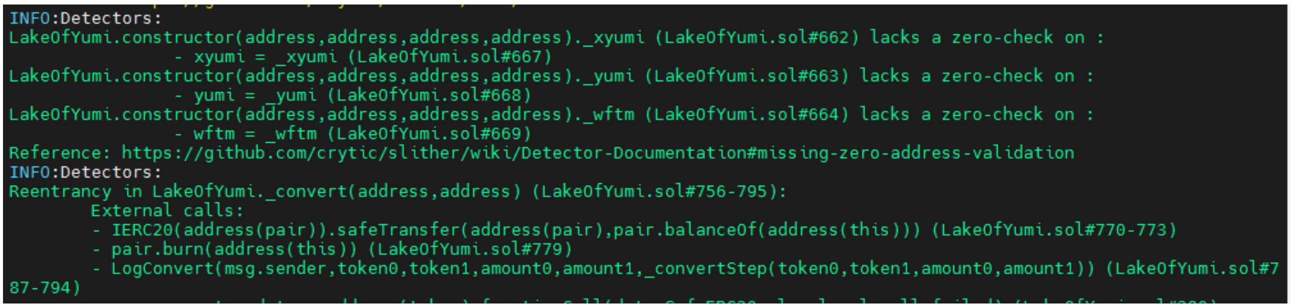


```
INFO:Detectors:
Pragma version>0.6.6 (YumiToken.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (YumiToken.sol#224-230):
        - (success) = recipient.call{value: amount}() (YumiToken.sol#228)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (YumiToken.sol#303-329):
        - (success,returndata) = target.call{value: weiValue}(data) (YumiToken.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter YumiToken.mintFor(address,uint256)._to (YumiToken.sol#779) is not in mixedCase
Parameter YumiToken.mintFor(address,uint256)._amount (YumiToken.sol#779) is not in mixedCase
Variable YumiToken._delegates (YumiToken.sol#795) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (YumiToken.sol#432)" inContext (YumiToken.sol#426-435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (YumiToken.sol#473-476)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (YumiToken.sol#482-486)
decimals() should be declared external:
        - ERC20.decimals() (YumiToken.sol#535-537)
symbol() should be declared external:
        - ERC20.symbol() (YumiToken.sol#542-544)
totalSupply() should be declared external:
        - ERC20.totalSupply() (YumiToken.sol#549-551)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (YumiToken.sol#568-571)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (YumiToken.sol#576-578)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (YumiToken.sol#587-590)
```

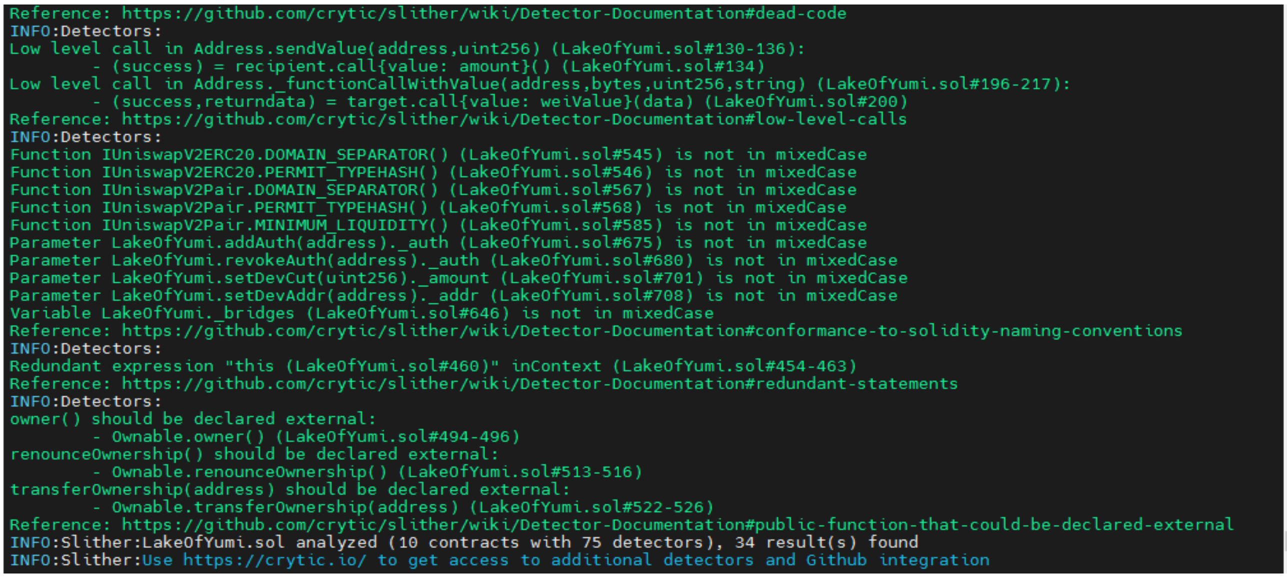


```
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (YumiToken.sol#587-590)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (YumiToken.sol#604-616)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (YumiToken.sol#630-633)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (YumiToken.sol#649-656)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (YumiToken.sol#666-669)
        - YumiToken.mint(uint256) (YumiToken.sol#784-787)
mintFor(address,uint256) should be declared external:
        - YumiToken.mintFor(address,uint256) (YumiToken.sol#779-782)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YumiToken.sol analyzed (7 contracts with 75 detectors), 44 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> LakeOfYumi.sol

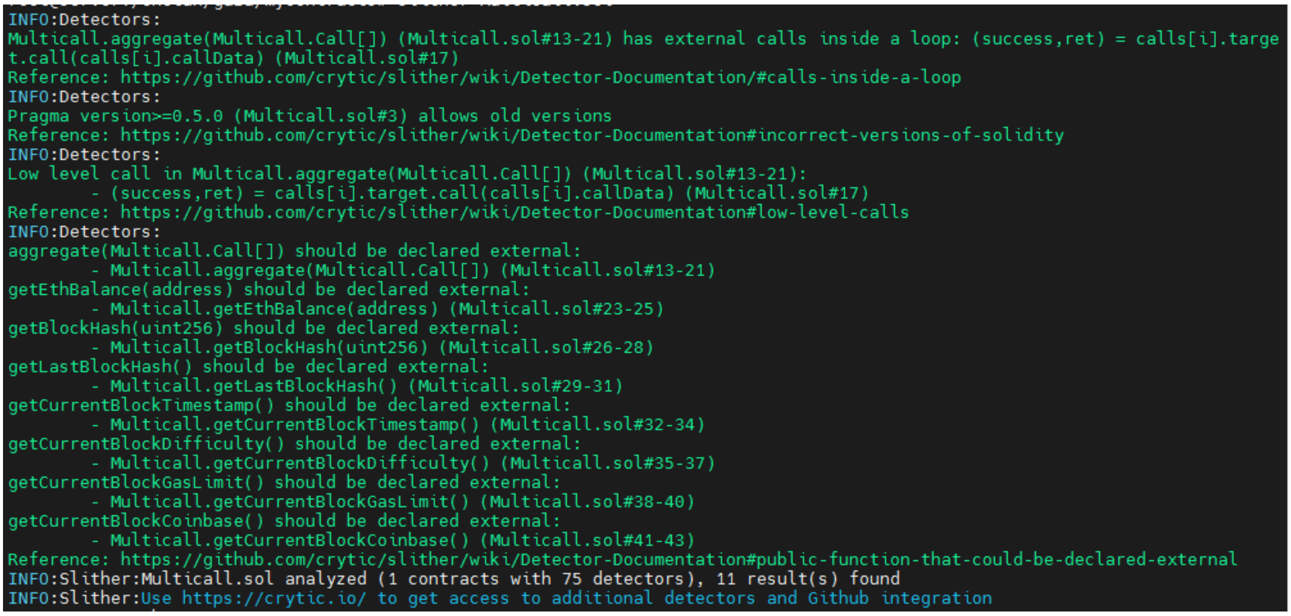


```
INFO:Detectors:
LakeOfYumi.constructor(address,address,address,address)._xyumi (LakeOfYumi.sol#662) lacks a zero-check on :
                - xyumi = _xyumi (LakeOfYumi.sol#667)
LakeOfYumi.constructor(address,address,address,address)._yumi (LakeOfYumi.sol#663) lacks a zero-check on :
                - yumi = _yumi (LakeOfYumi.sol#668)
LakeOfYumi.constructor(address,address,address,address)._wftm (LakeOfYumi.sol#664) lacks a zero-check on :
                - wftm = _wftm (LakeOfYumi.sol#669)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in LakeOfYumi._convert(address,address) (LakeOfYumi.sol#756-795):
        External calls:
        - IERC20(address(pair)).safeTransfer(address(pair),pair.balanceOf(address(this))) (LakeOfYumi.sol#770-773)
        - pair.burn(address(this)) (LakeOfYumi.sol#779)
        - LogConvert(msg.sender,token0,token1,amount0,amount1,_convertStep(token0,token1,amount0,amount1)) (LakeOfYumi.sol#7
87-794)
```



```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (LakeOfYumi.sol#130-136):
        - (success) = recipient.call{value: amount}() (LakeOfYumi.sol#134)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (LakeOfYumi.sol#196-217):
        - (success,returndata) = target.call{value: weiValue}(data) (LakeOfYumi.sol#200)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2ERC20.DOMAIN_SEPARATOR() (LakeOfYumi.sol#545) is not in mixedCase
Function IUniswapV2ERC20.PERMIT_TYPEHASH() (LakeOfYumi.sol#546) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (LakeOfYumi.sol#567) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (LakeOfYumi.sol#568) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (LakeOfYumi.sol#585) is not in mixedCase
Parameter LakeOfYumi.addAuth(address)._auth (LakeOfYumi.sol#675) is not in mixedCase
Parameter LakeOfYumi.revokeAuth(address)._auth (LakeOfYumi.sol#680) is not in mixedCase
Parameter LakeOfYumi.setDevCut(uint256)._amount (LakeOfYumi.sol#701) is not in mixedCase
Parameter LakeOfYumi.setDevAddr(address)._addr (LakeOfYumi.sol#708) is not in mixedCase
Variable LakeOfYumi._bridges (LakeOfYumi.sol#646) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (LakeOfYumi.sol#460)" inContext (LakeOfYumi.sol#454-463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (LakeOfYumi.sol#494-496)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (LakeOfYumi.sol#513-516)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (LakeOfYumi.sol#522-526)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LakeOfYumi.sol analyzed (10 contracts with 75 detectors), 34 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

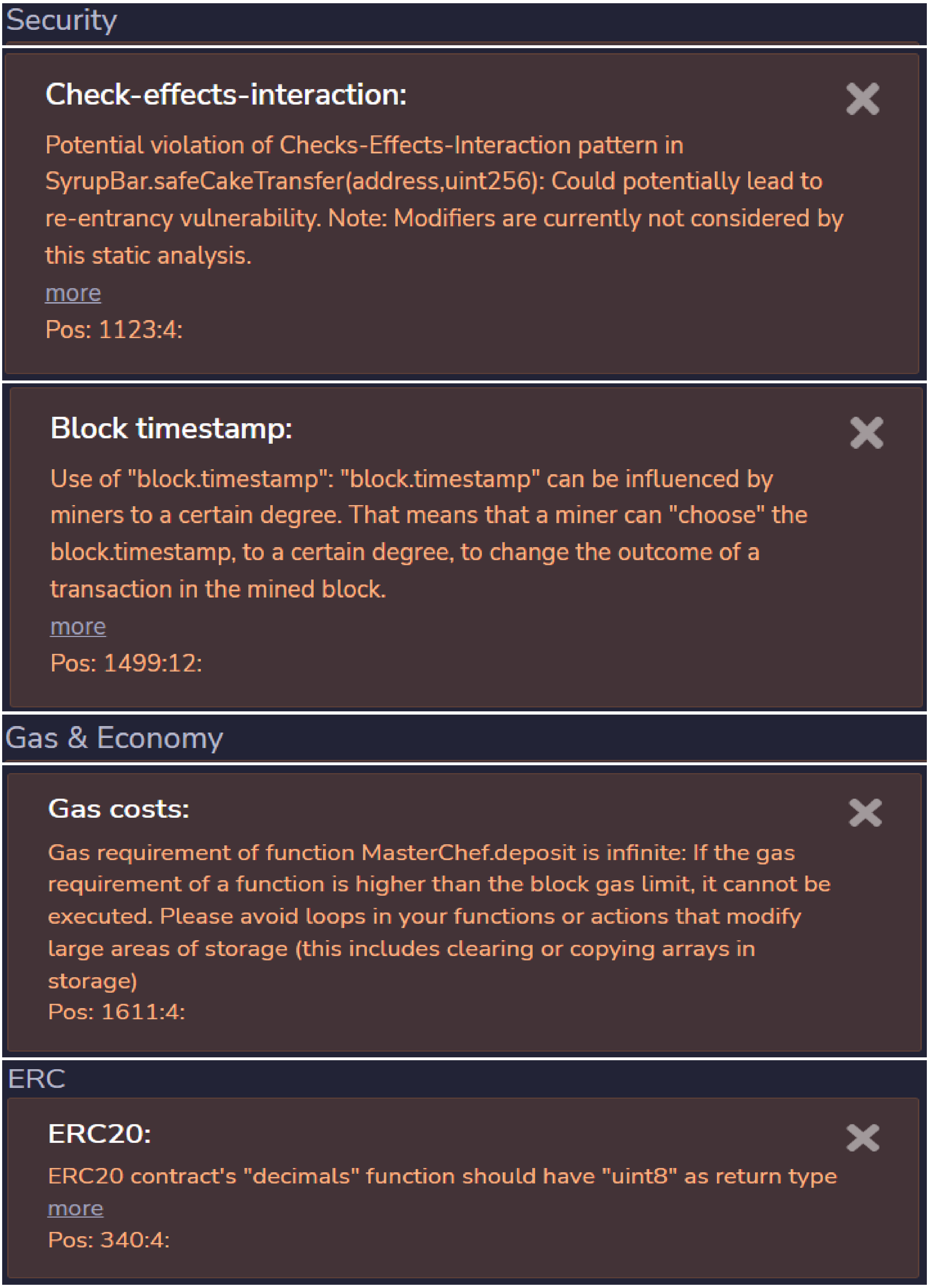
Slither log >> Multicall.sol



```
INFO:Detectors:
Multicall.aggregate(Multicall.Call[]) (Multicall.sol#13-21) has external calls inside a loop: (success,ret) = calls[i].targe
t.call(calls[i].callData) (Multicall.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Pragma version>=0.5.0 (Multicall.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Multicall.aggregate(Multicall.Call[]) (Multicall.sol#13-21):
        - (success,ret) = calls[i].target.call(calls[i].callData) (Multicall.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
aggregate(Multicall.Call[]) should be declared external:
        - Multicall.aggregate(Multicall.Call[]) (Multicall.sol#13-21)
getEthBalance(address) should be declared external:
        - Multicall.getEthBalance(address) (Multicall.sol#23-25)
getBlockHash(uint256) should be declared external:
        - Multicall.getBlockHash(uint256) (Multicall.sol#26-28)
getLastBlockHash() should be declared external:
        - Multicall.getLastBlockHash() (Multicall.sol#29-31)
getCurrentBlockTimestamp() should be declared external:
        - Multicall.getCurrentBlockTimestamp() (Multicall.sol#32-34)
getCurrentBlockDifficulty() should be declared external:
        - Multicall.getCurrentBlockDifficulty() (Multicall.sol#35-37)
getCurrentBlockGasLimit() should be declared external:
        - Multicall.getCurrentBlockGasLimit() (Multicall.sol#38-40)
getCurrentBlockCoinbase() should be declared external:
        - Multicall.getCurrentBlockCoinbase() (Multicall.sol#41-43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Multicall.sol analyzed (1 contracts with 75 detectors), 11 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 SyrupBar.safeCakeTransfer(address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1123:4:

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: 1499:12:

Gas & Economy

Gas costs:

Gas requirement of function MasterChef.deposit 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: 1611:4:

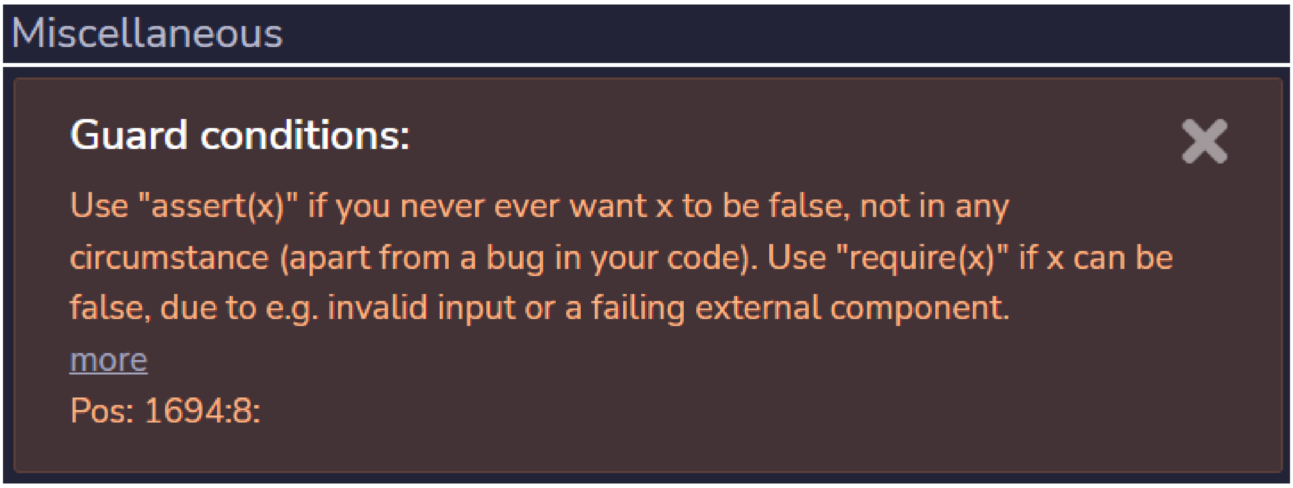
ERC

ERC20:

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

more

Pos: 340:4:



Miscellaneous

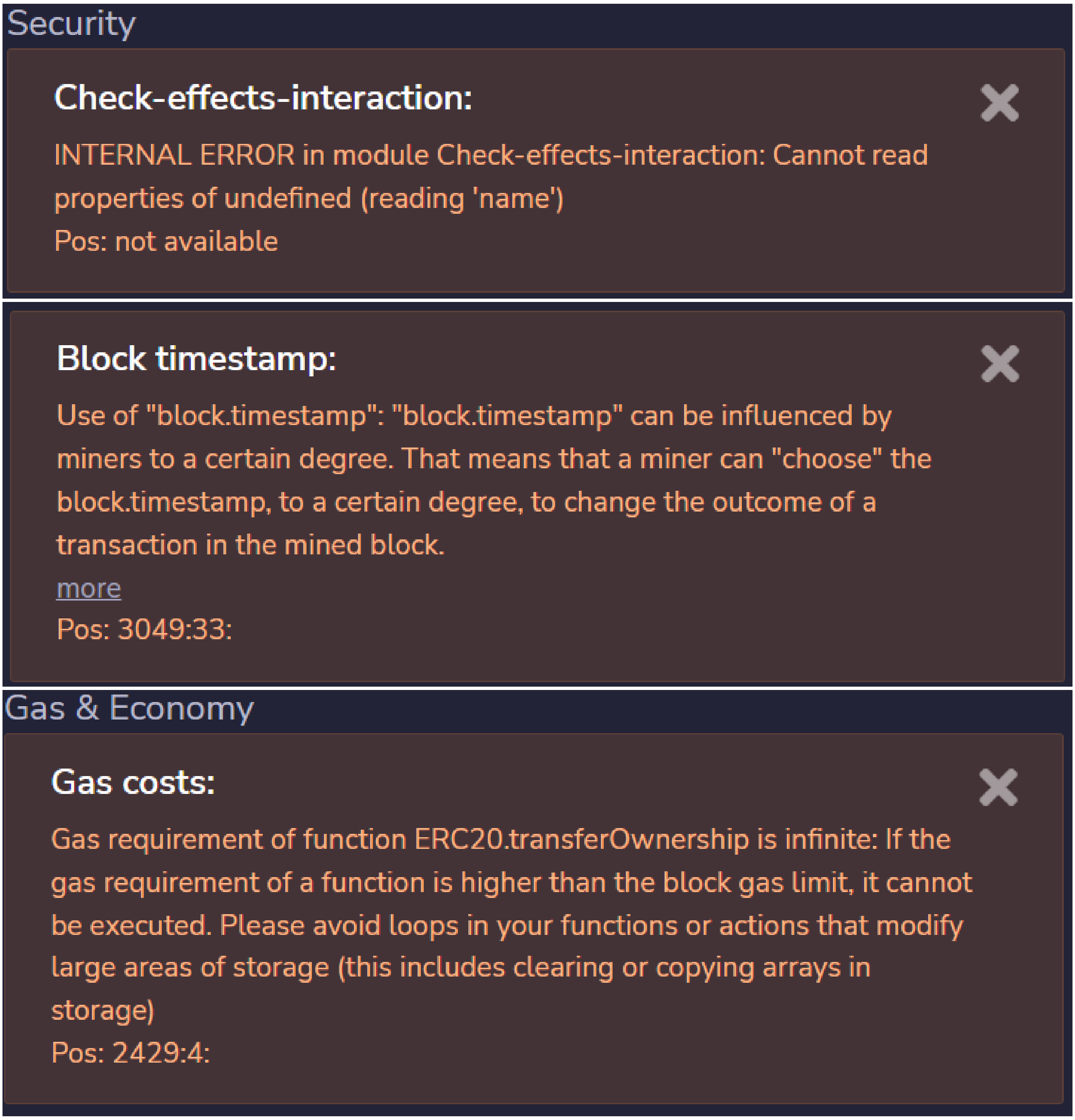
Guard conditions:

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

more

Pos: 1694:8:

SwapMining.sol



Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')

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

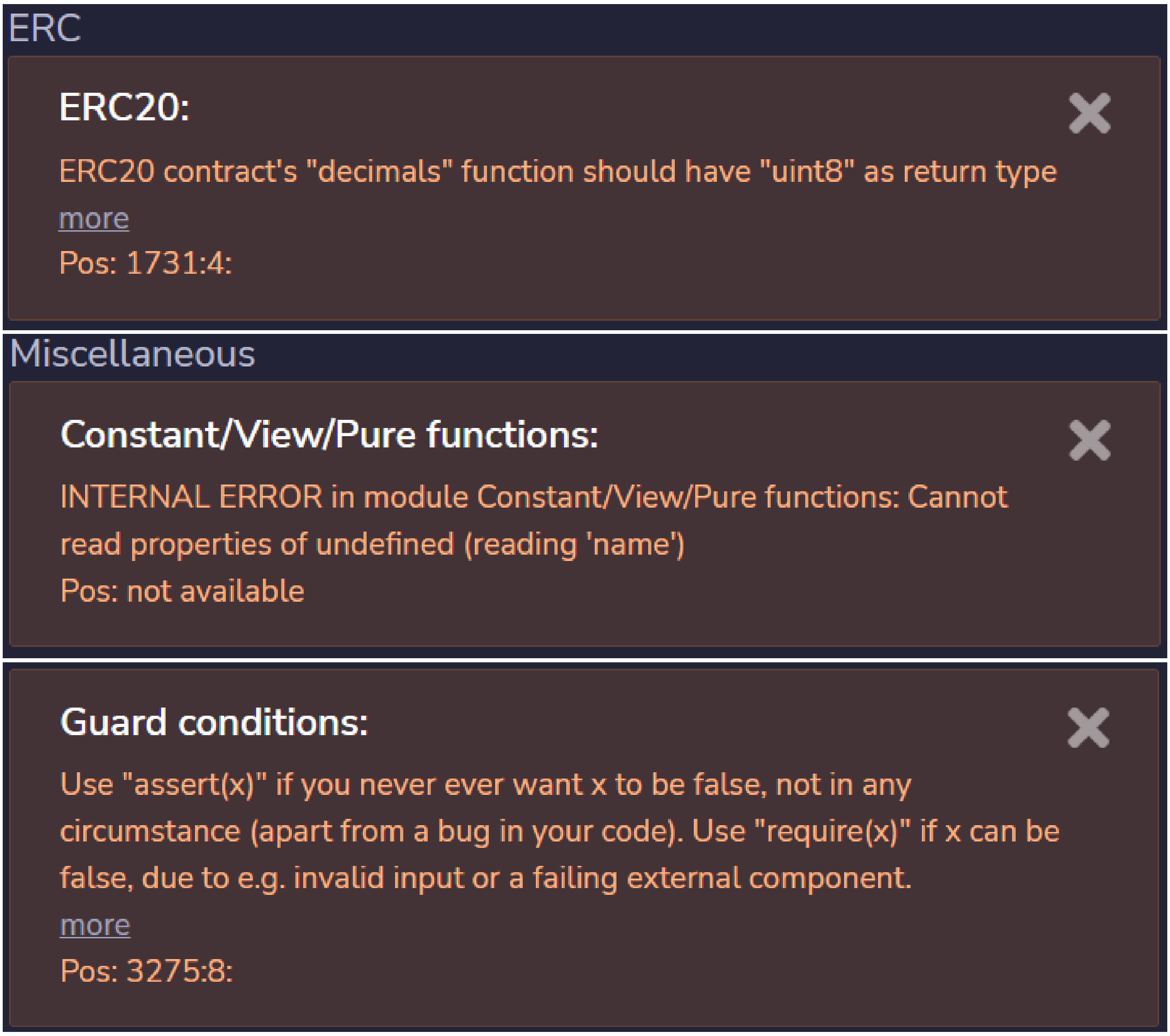
Pos: 3049:33:

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: 2429:4:



ERC

ERC20:

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

more

Pos: 1731:4:

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')

Pos: not available

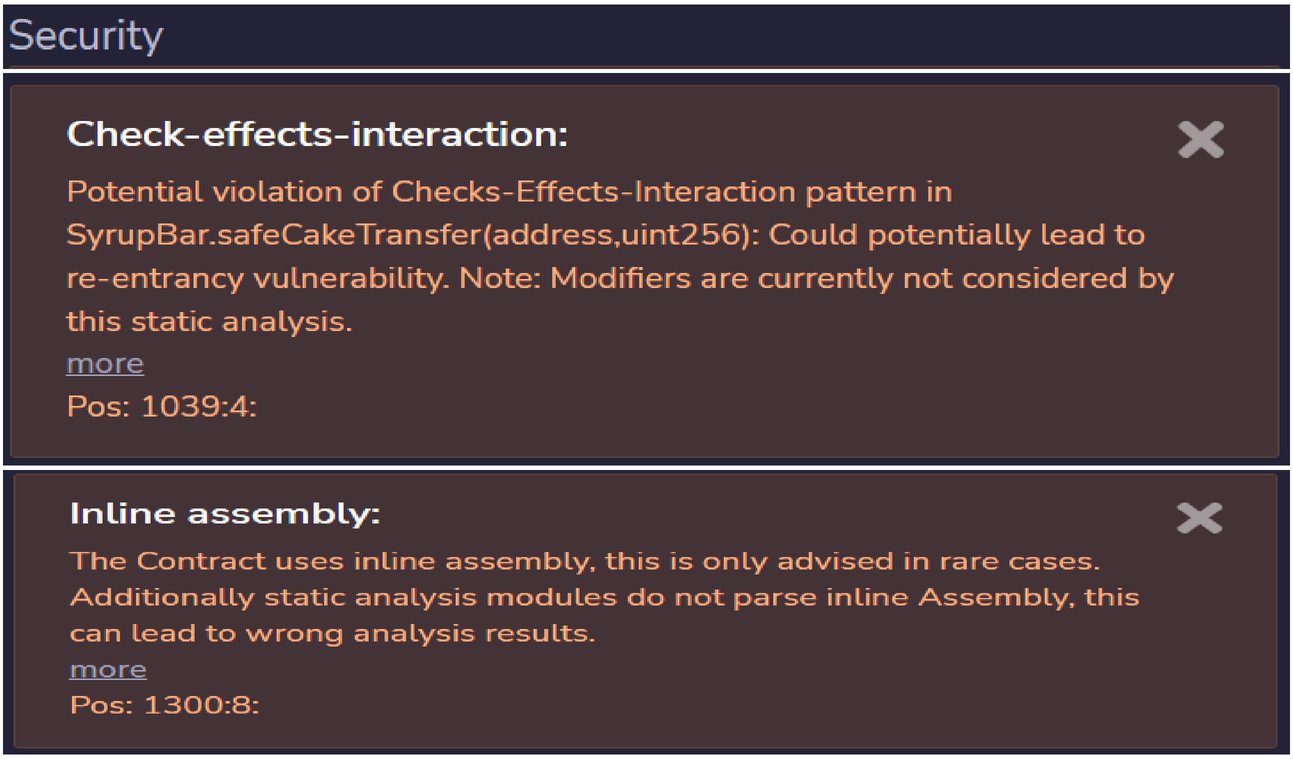
Guard conditions:

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: 3275:8:

SyrupBar.sol



Security

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

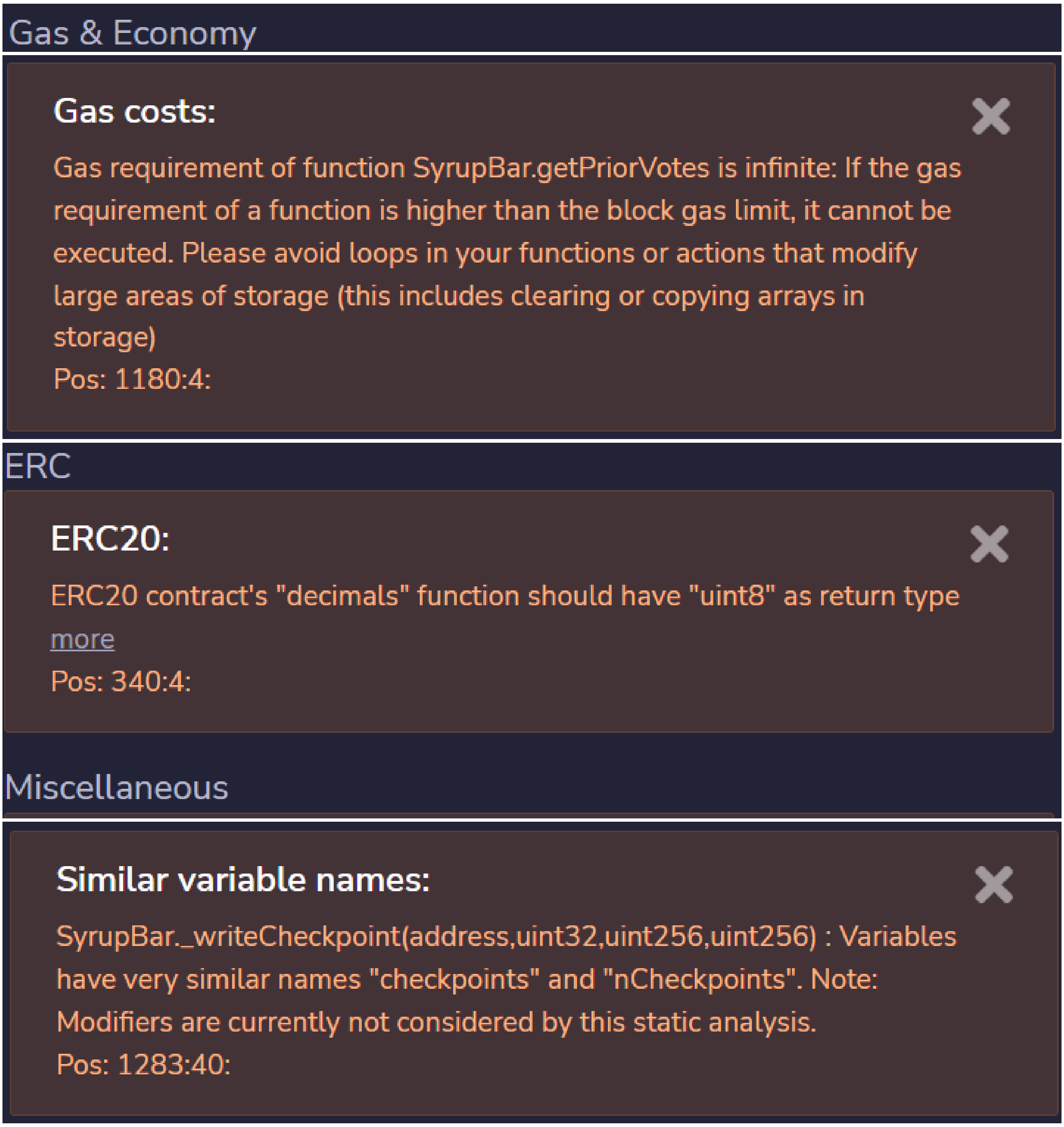
Pos: 1039:4:

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: 1300:8:



Gas & Economy

Gas costs:

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

Pos: 1180:4:

ERC

ERC20:

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

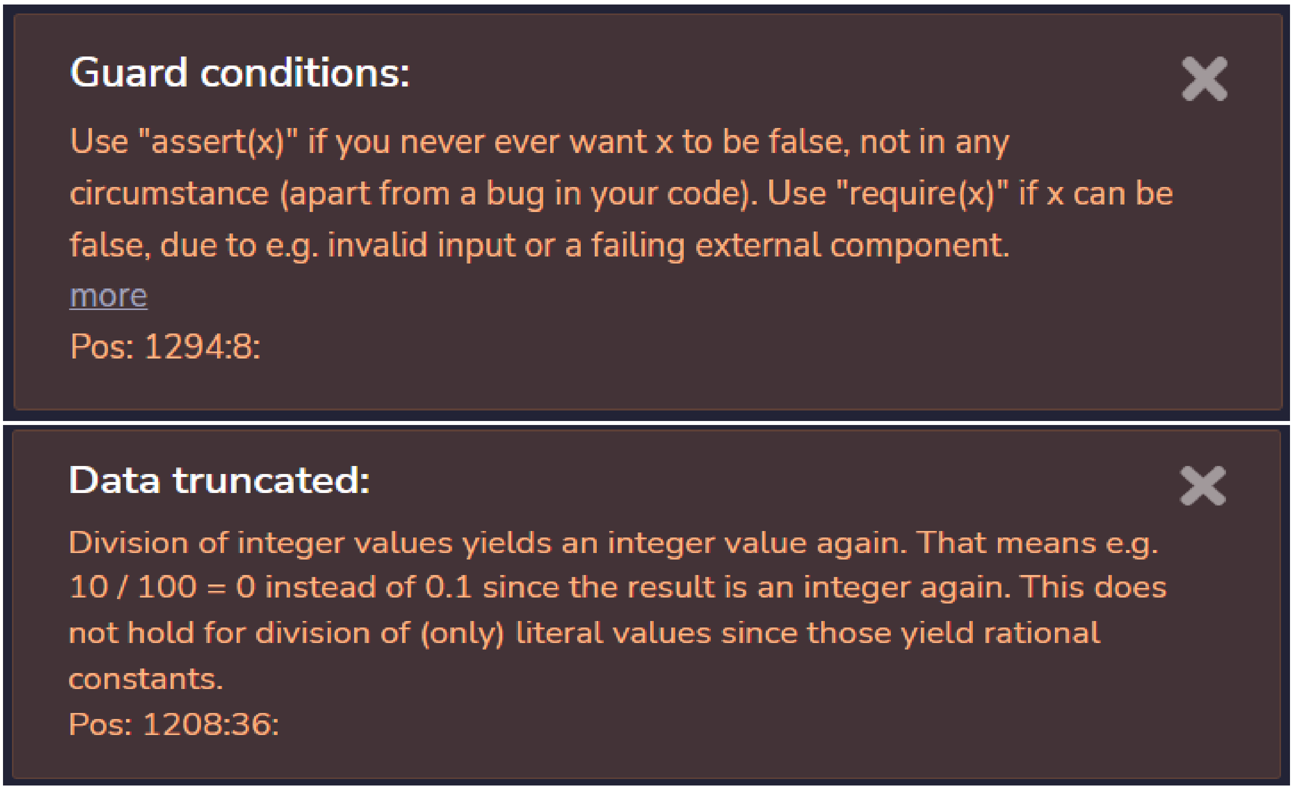
Pos: 340:4:

Miscellaneous

Similar variable names:

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

Pos: 1283:40:



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

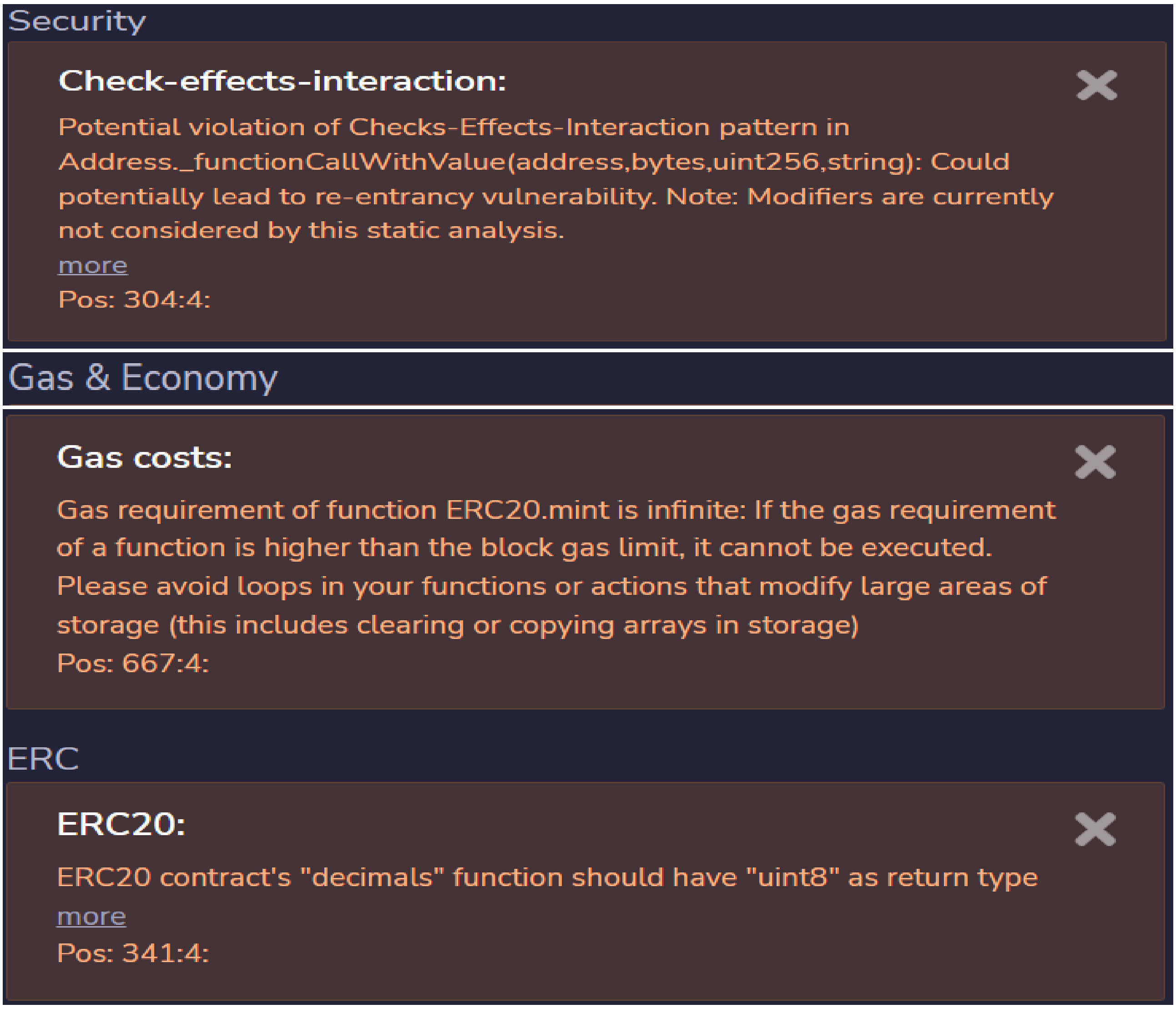
Pos: 1294:8:

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: 1208:36:

MockToken.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: 304:4:

Gas & Economy

Gas costs:

Gas requirement of function ERC20.mint 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: 667:4:

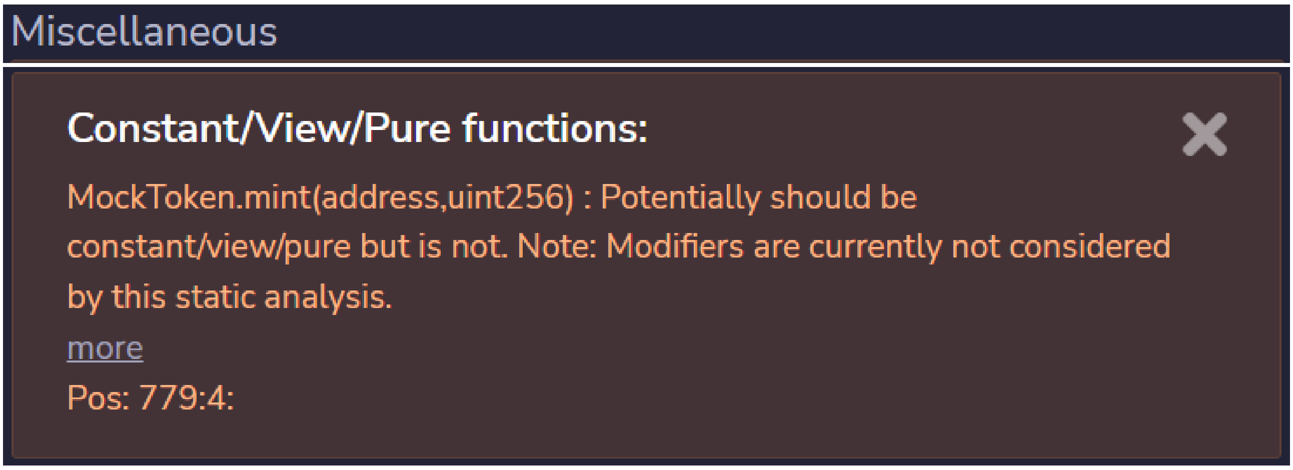
ERC

ERC20:

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

more

Pos: 341:4:



Miscellaneous

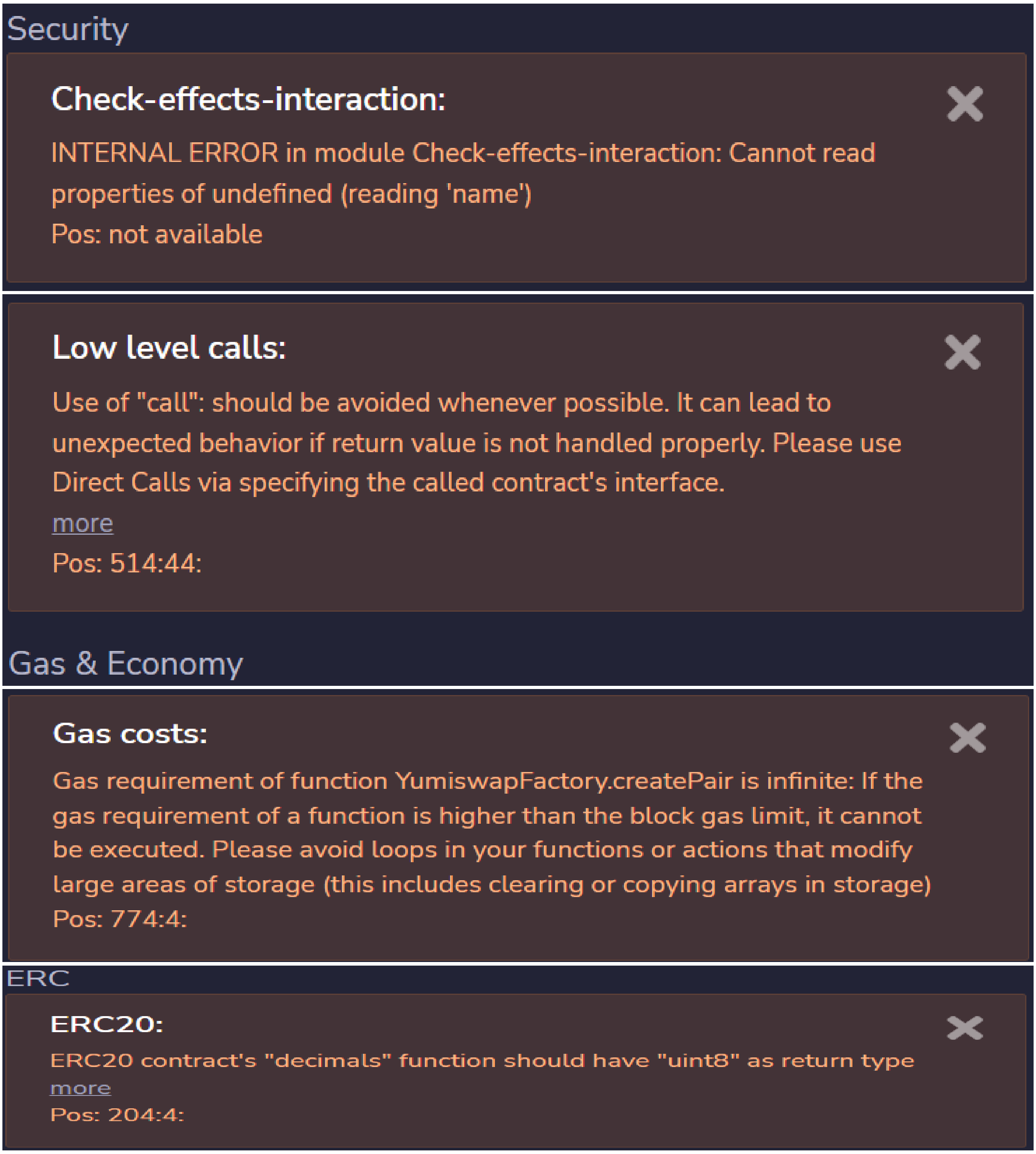
Constant/View/Pure functions:

MockToken.mint(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 779:4:

Factory.sol



Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')

Pos: not available

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: 514:44:

Gas & Economy

Gas costs:

Gas requirement of function YumiswapFactory.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: 774:4:

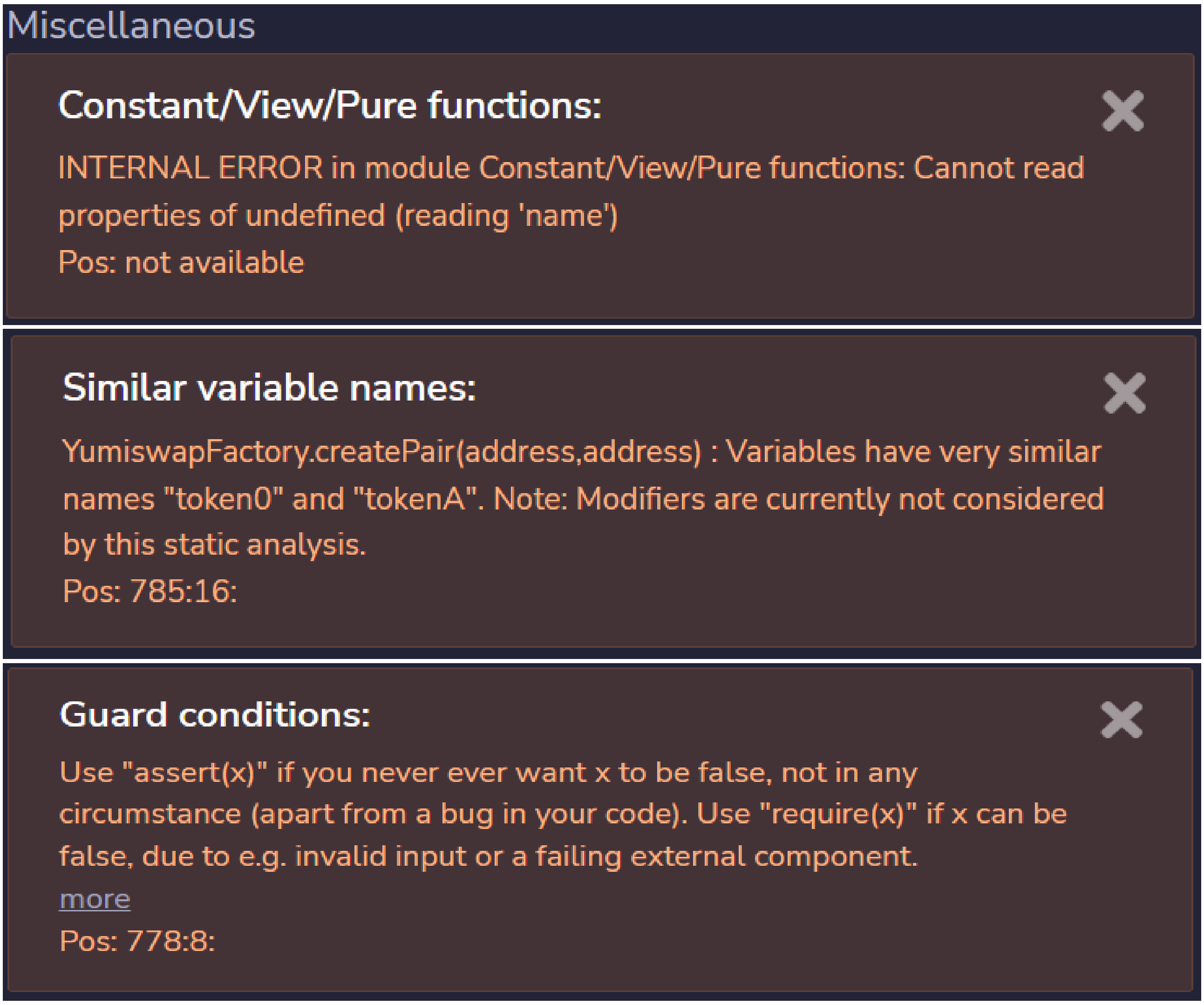
ERC

ERC20:

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

more

Pos: 204:4:



Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')
Pos: not available

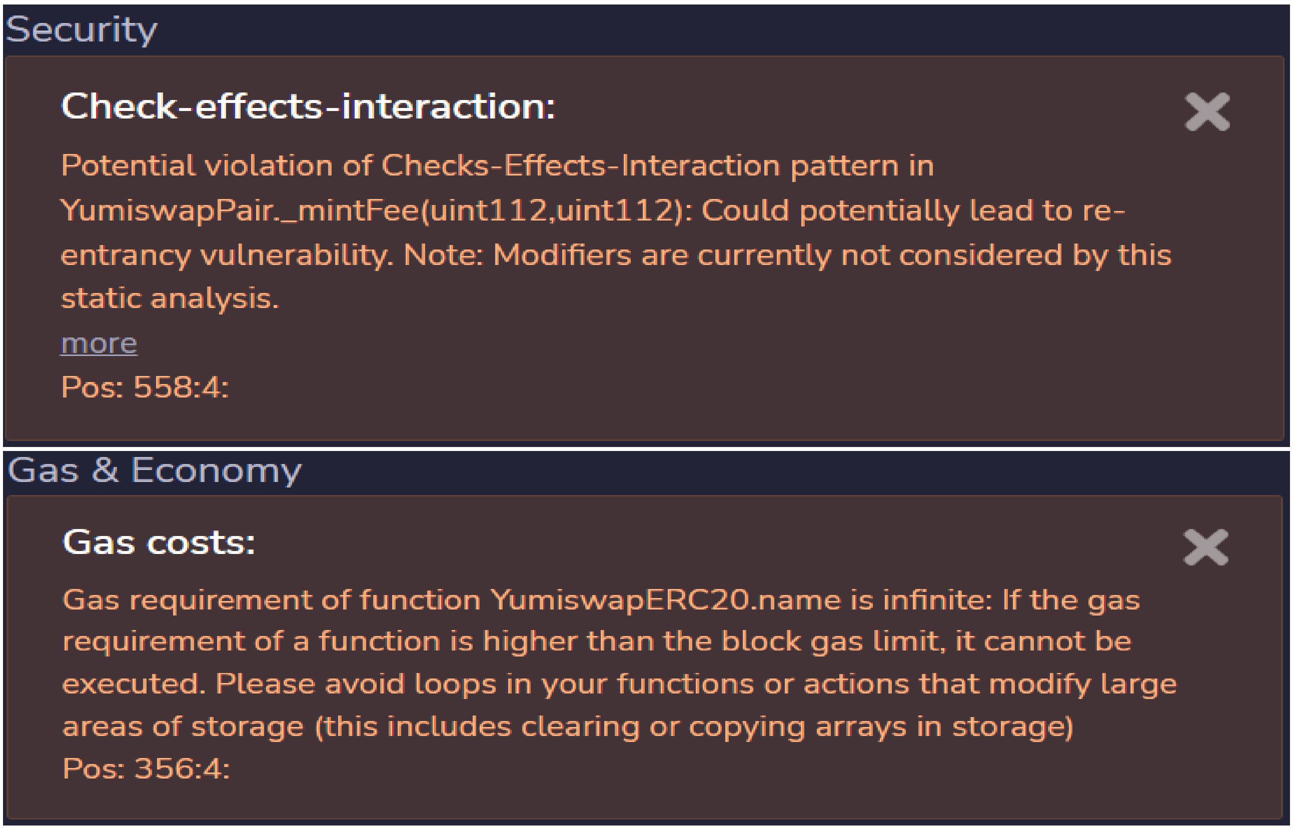
Similar variable names:

YumiswapFactory.createPair(address,address) : Variables have very similar names "token0" and "tokenA". Note: Modifiers are currently not considered by this static analysis.
Pos: 785:16:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 778:8:

Pair.sol



Security

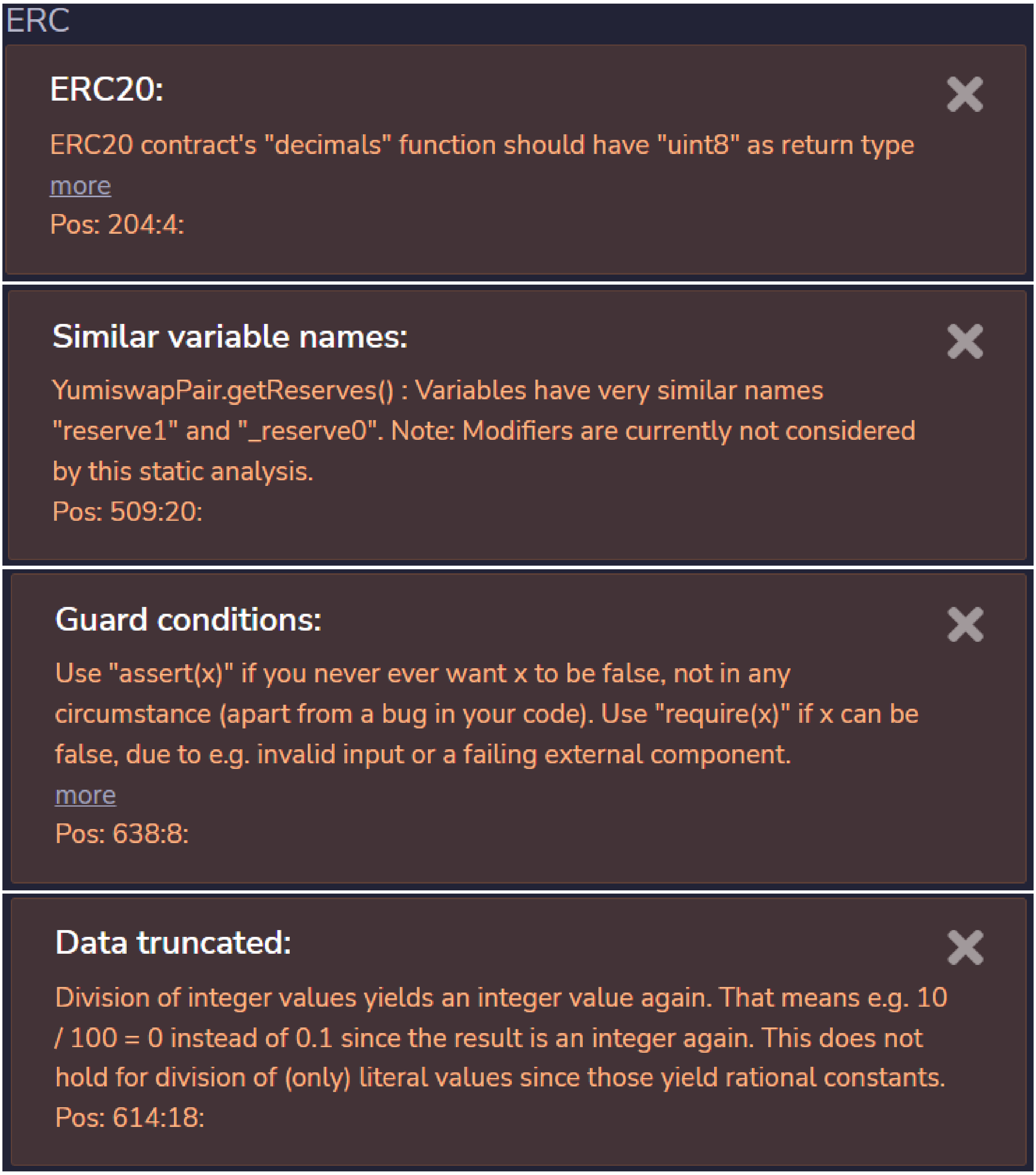
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in YumiswapPair._mintFee(uint112,uint112): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 558:4:

Gas & Economy

Gas costs:

Gas requirement of function YumiswapERC20.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 356:4:



ERC

ERC20:

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

more

Pos: 204:4:

Similar variable names:

YumiswapPair.getReserves() : Variables have very similar names "reserve1" and "_reserve0". Note: Modifiers are currently not considered by this static analysis.

Pos: 509:20:

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

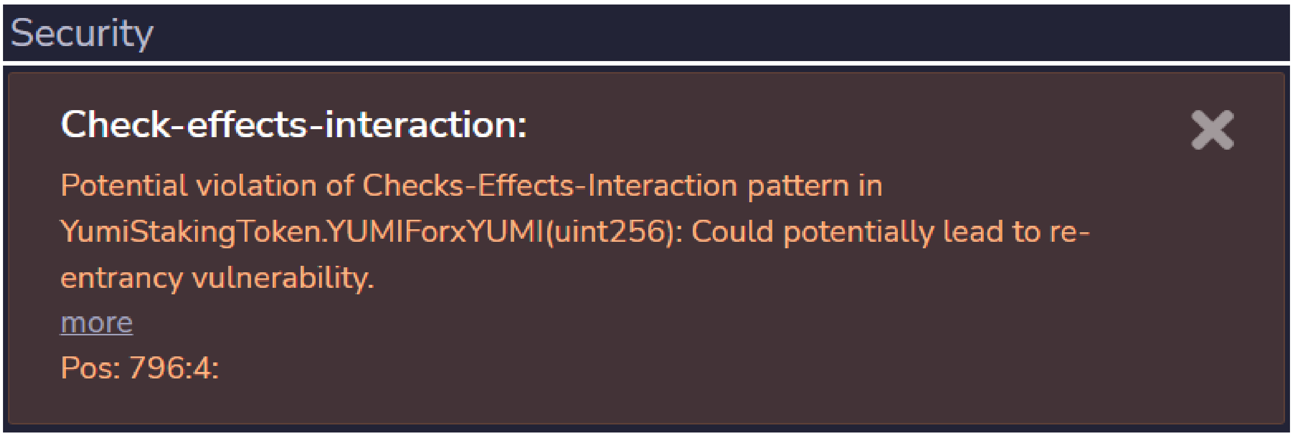
Pos: 638:8:

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: 614:18:

xYUMI.sol



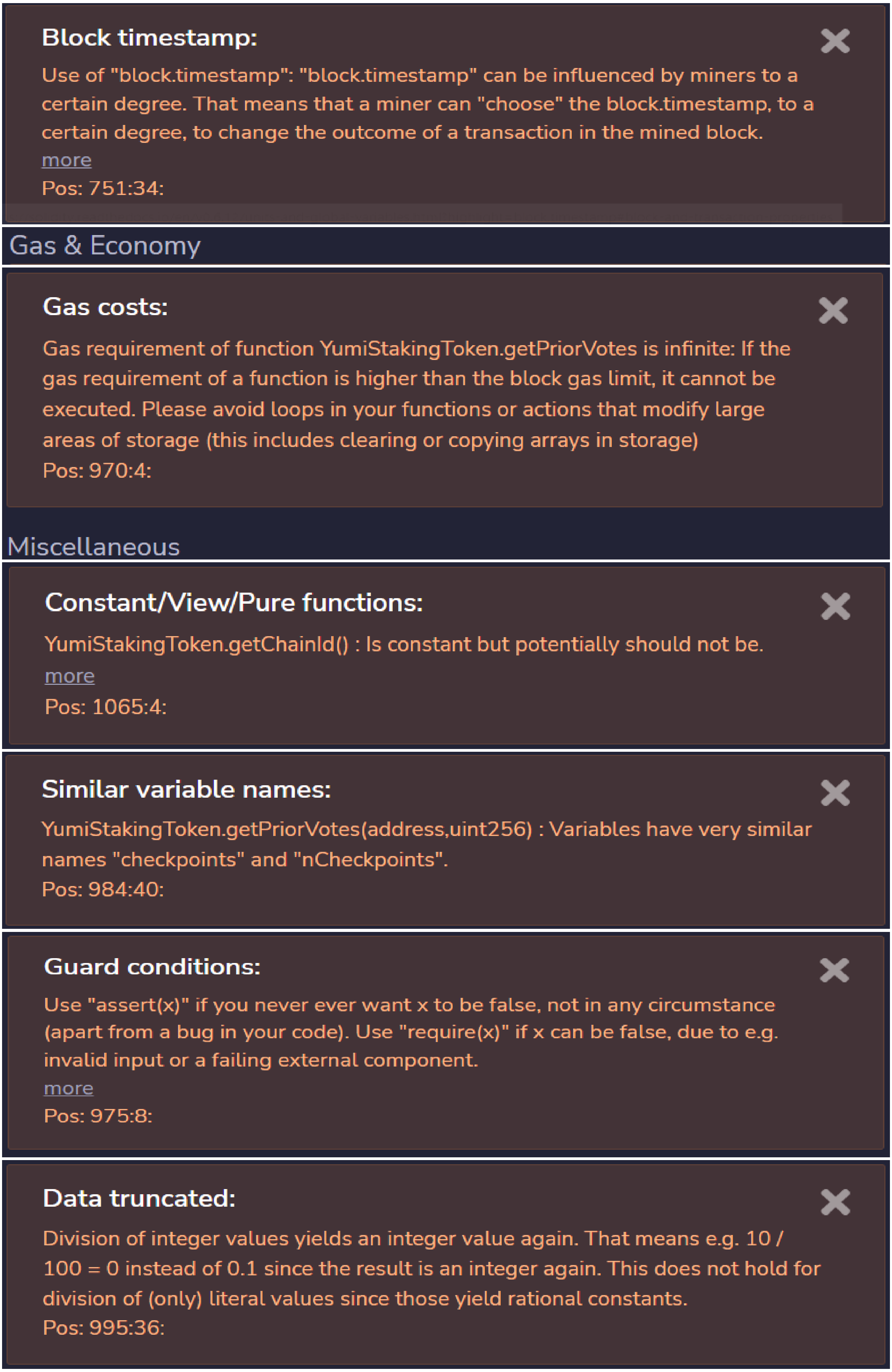
Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in YumiStakingToken.YUMIForxYUMI(uint256): Could potentially lead to re-entrancy vulnerability.

more

Pos: 796:4:



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: 751:34:

Gas & Economy

Gas costs:

Gas requirement of function YumiStakingToken.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: 970:4:

Miscellaneous

Constant/View/Pure functions:

YumiStakingToken.getChainId() : Is constant but potentially should not be.
more
Pos: 1065:4:

Similar variable names:

YumiStakingToken.getPriorVotes(address,uint256) : Variables have very similar names "checkpoints" and "nCheckpoints".
Pos: 984:40:

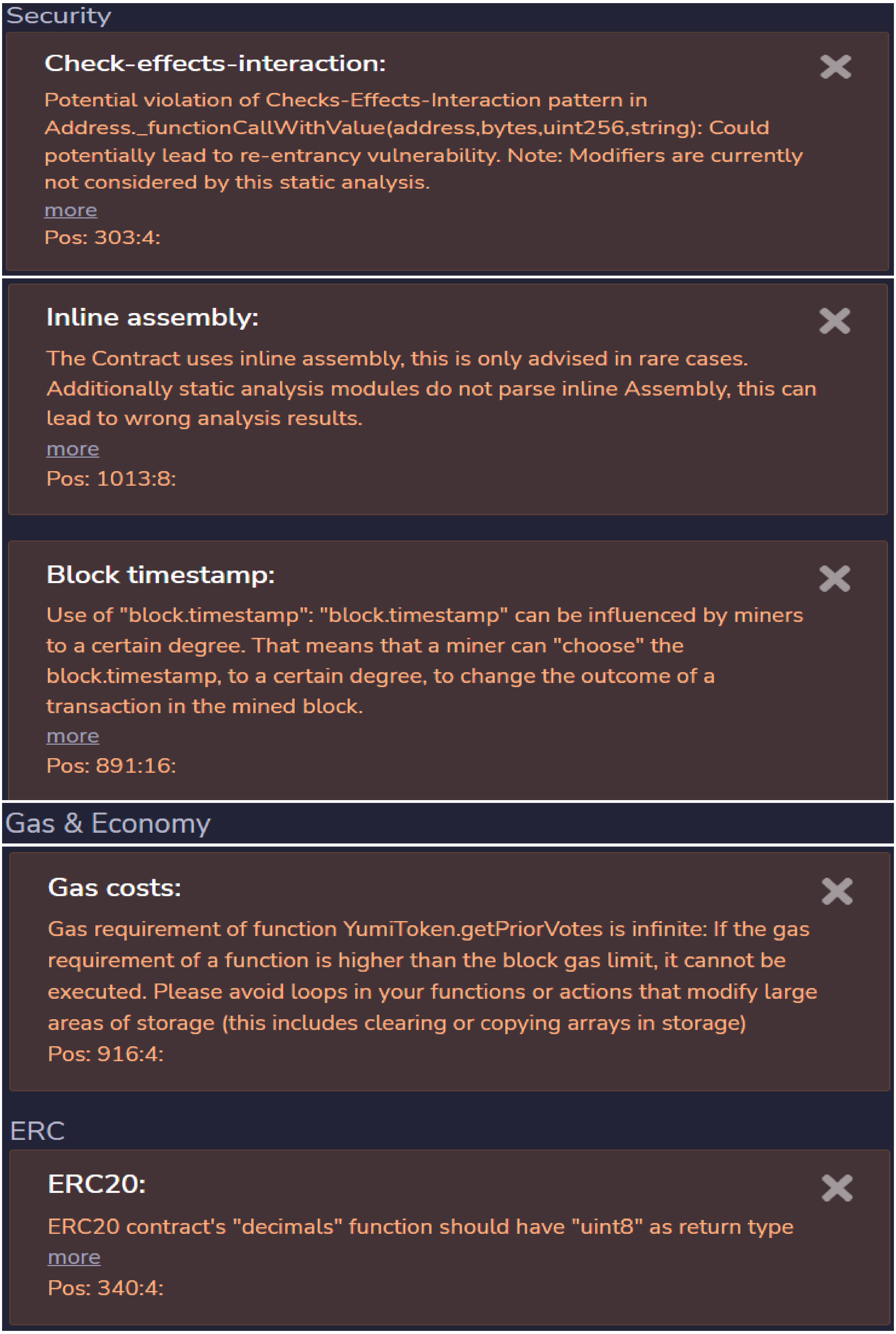
Guard conditions:

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: 975:8:

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: 995:36:

YumiToken.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: 303:4:

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: 1013:8:

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: 891:16:

Gas & Economy

Gas costs:

Gas requirement of function YumiToken.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: 916:4:

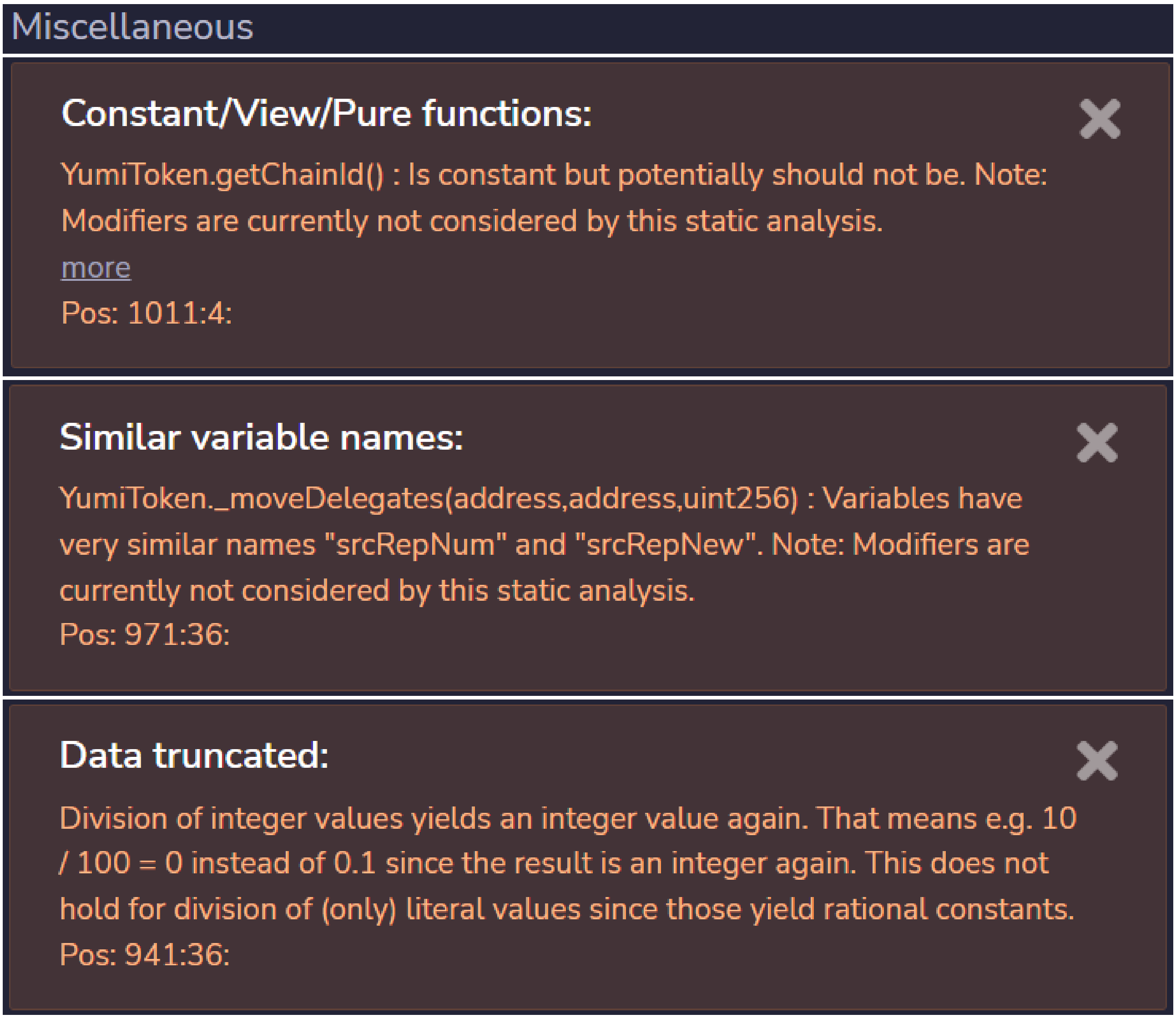
ERC

ERC20:

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

more

Pos: 340:4:



Miscellaneous

Constant/View/Pure functions: ✖

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

more

Pos: 1011:4:

Similar variable names: ✖

YumiToken._moveDelegates(address,address,uint256) : Variables have very similar names "srcRepNum" and "srcRepNew". Note: Modifiers are currently not considered by this static analysis.

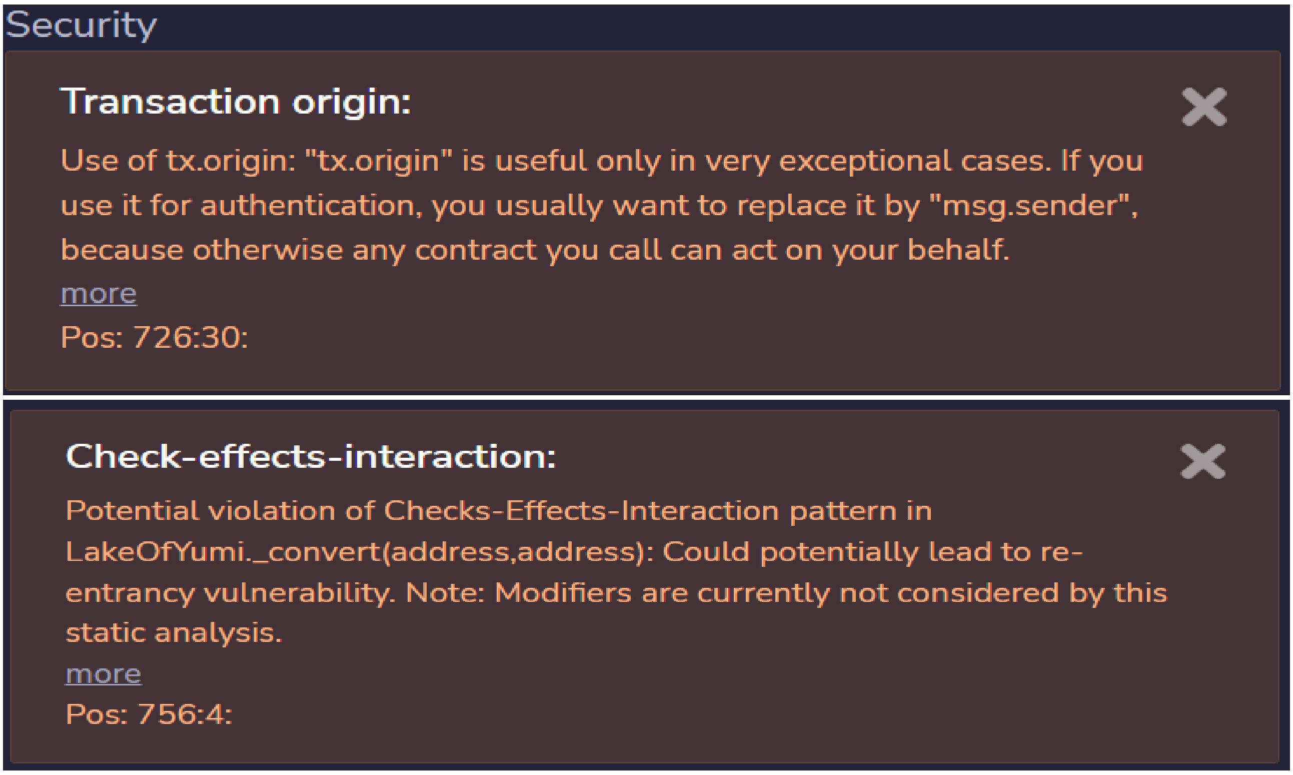
Pos: 971:36:

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: 941:36:

LakeOfYumi.sol



Security

Transaction origin: ✖

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

more

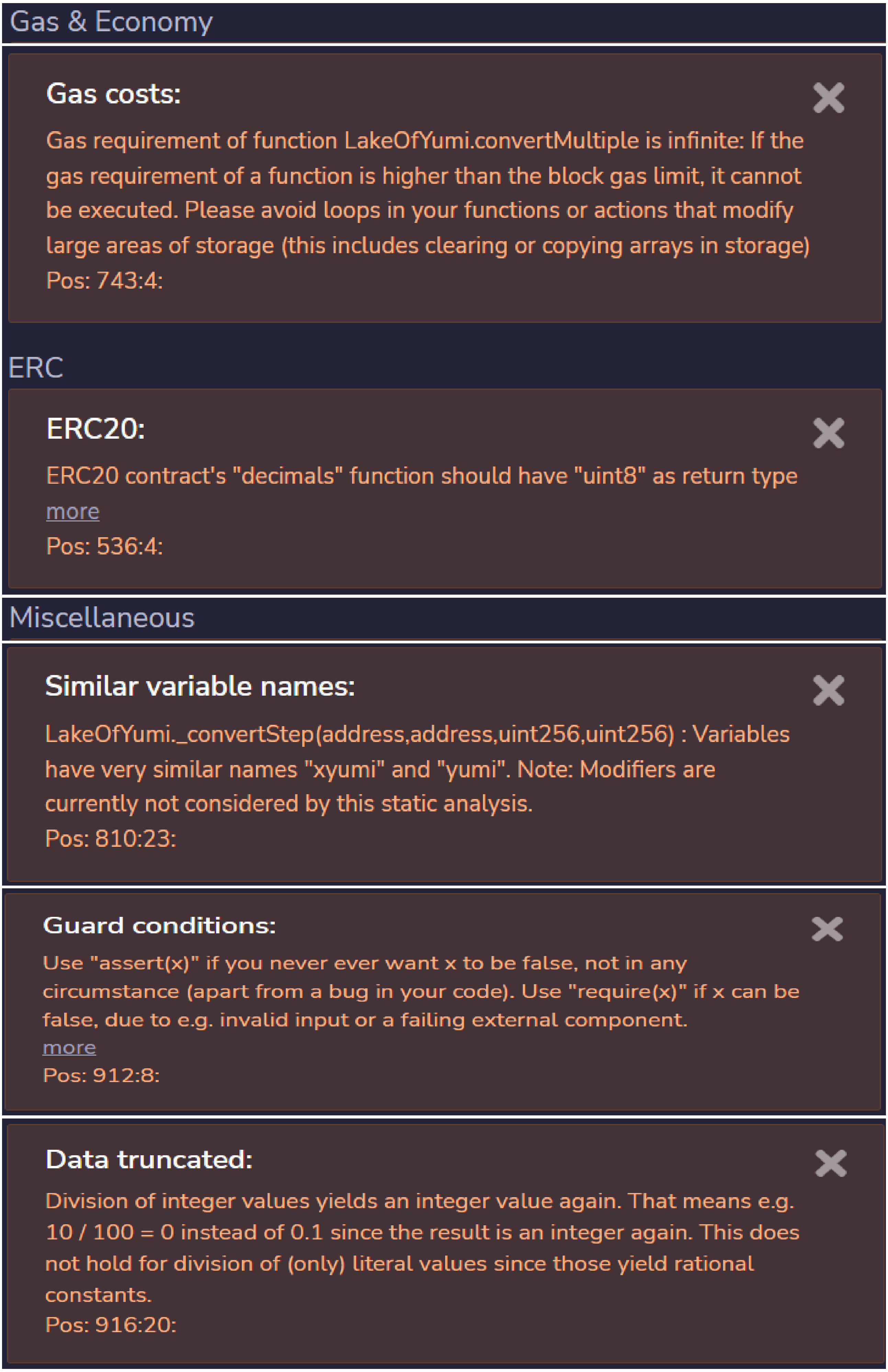
Pos: 726:30:

Check-effects-interaction: ✖

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

more

Pos: 756:4:



Gas & Economy

Gas costs:

Gas requirement of function LakeOfYumi.convertMultiple 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: 743:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 536:4:

Miscellaneous

Similar variable names:

LakeOfYumi._convertStep(address,address,uint256,uint256) : Variables have very similar names "xyumi" and "yumi". Note: Modifiers are currently not considered by this static analysis.
Pos: 810:23:

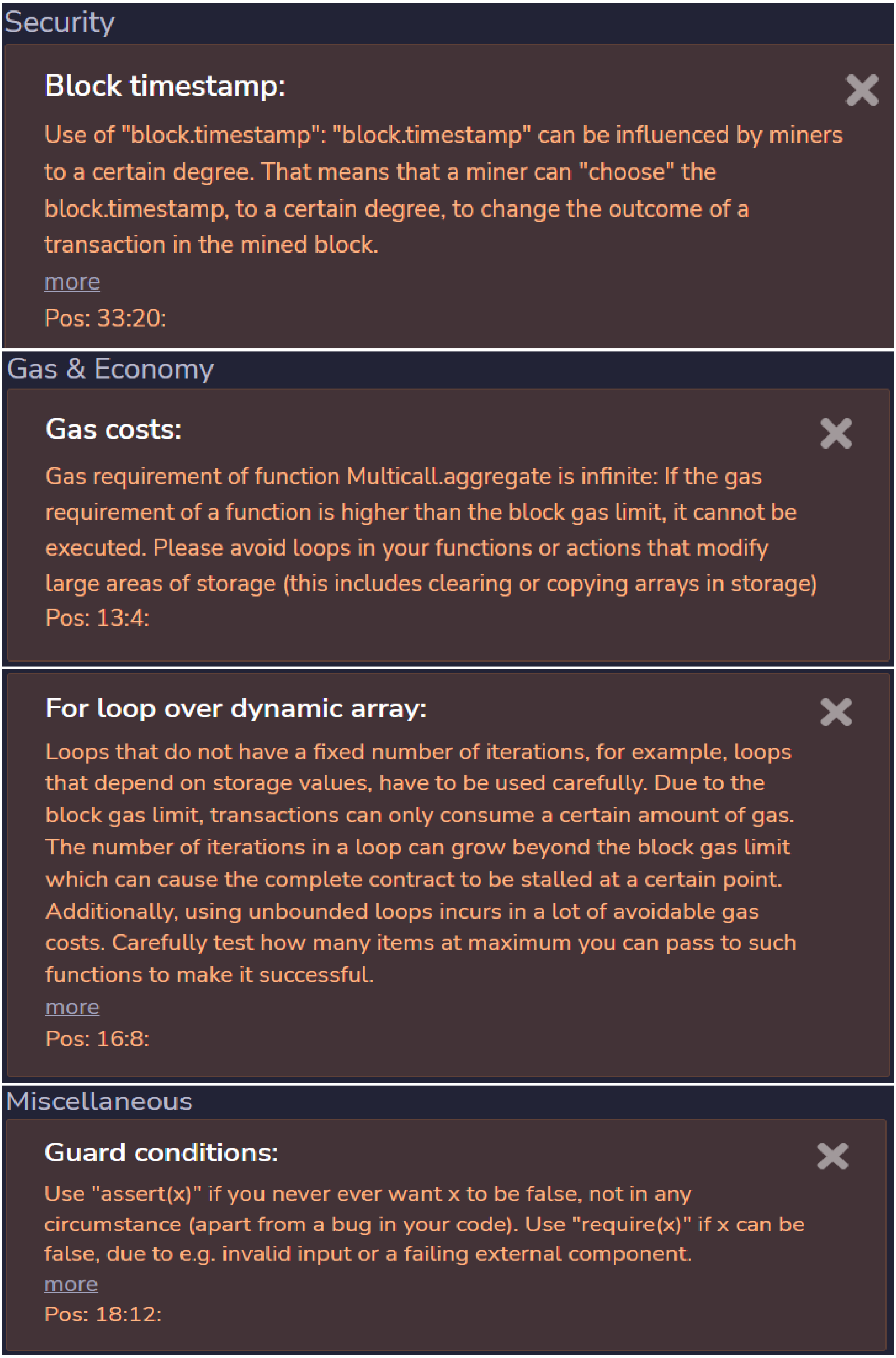
Guard conditions:

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: 912:8:

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: 916:20:

Multicall.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: 33:20:

Gas & Economy

Gas costs:

Gas requirement of function Multicall.aggregate 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: 13:4:

For loop over dynamic array:

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

more

Pos: 16:8:

Miscellaneous

Guard conditions:

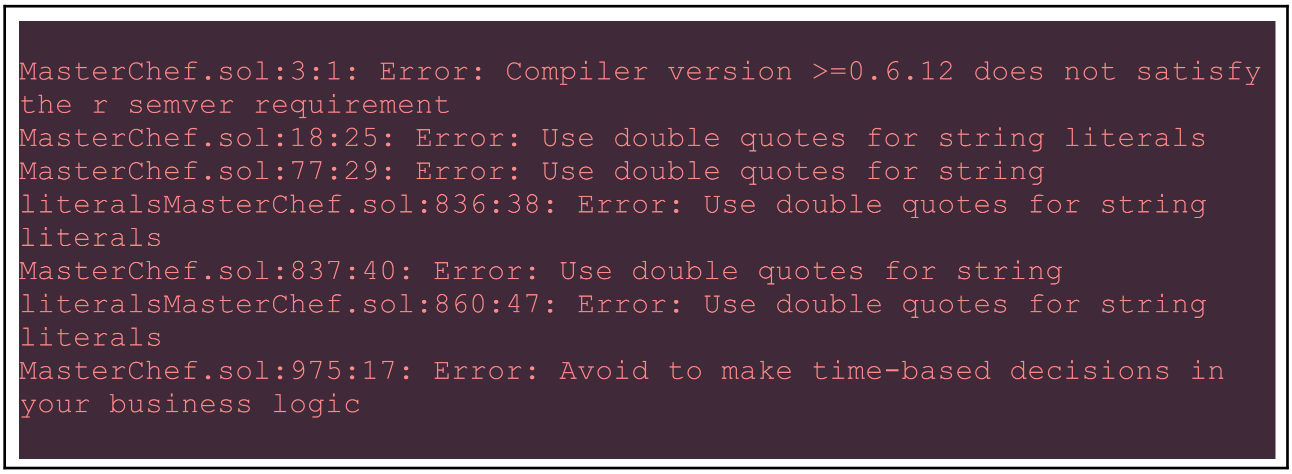
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 18:12:

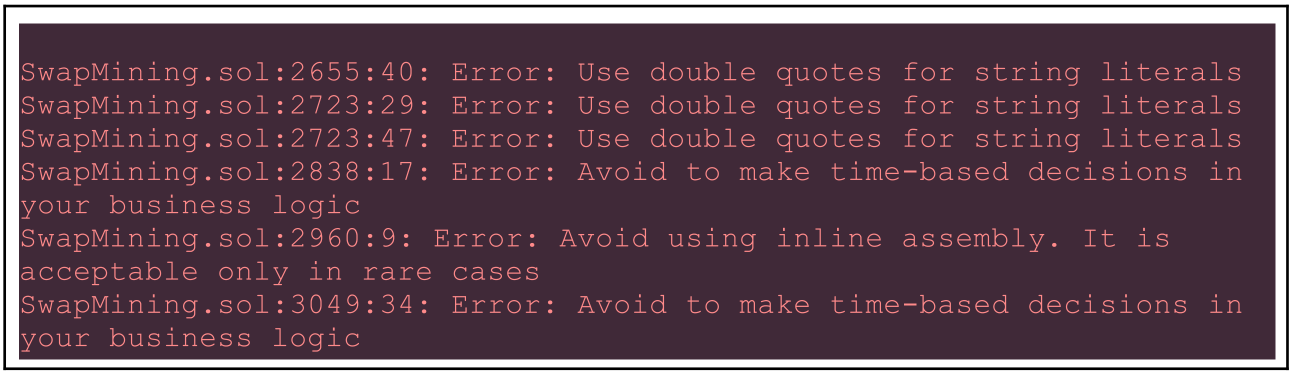
Solhint Linter

MasterChef.sol



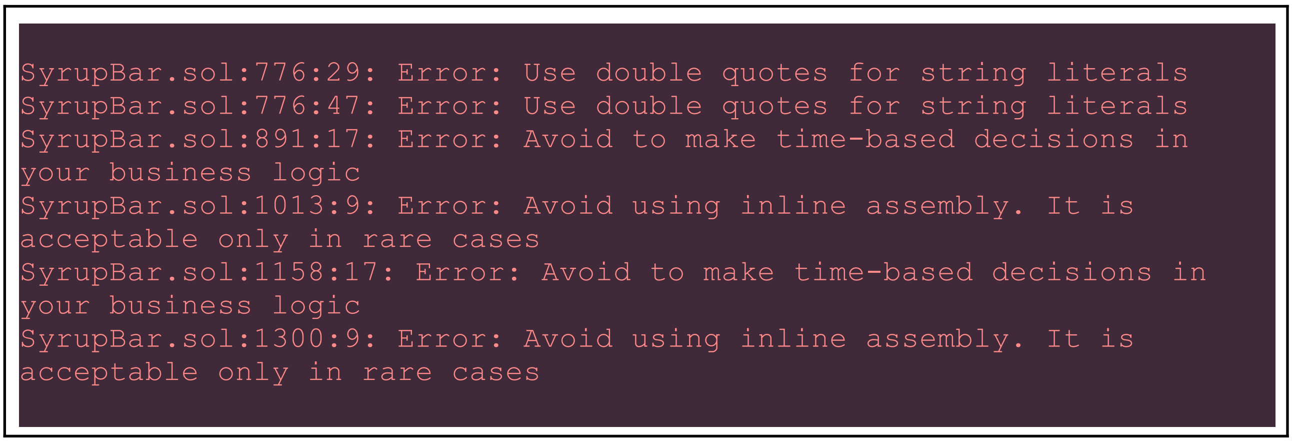
```
MasterChef.sol:3:1: Error: Compiler version >=0.6.12 does not satisfy
the r semver requirement
MasterChef.sol:18:25: Error: Use double quotes for string literals
MasterChef.sol:77:29: Error: Use double quotes for string
literalsMasterChef.sol:836:38: Error: Use double quotes for string
literals
MasterChef.sol:837:40: Error: Use double quotes for string
literalsMasterChef.sol:860:47: Error: Use double quotes for string
literals
MasterChef.sol:975:17: Error: Avoid to make time-based decisions in
your business logic
```

SwapMining.sol



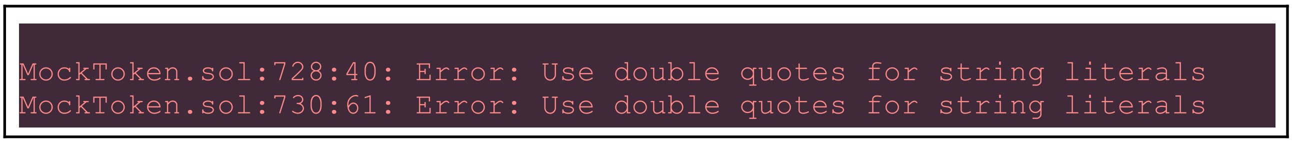
```
SwapMining.sol:2655:40: Error: Use double quotes for string literals
SwapMining.sol:2723:29: Error: Use double quotes for string literals
SwapMining.sol:2723:47: Error: Use double quotes for string literals
SwapMining.sol:2838:17: Error: Avoid to make time-based decisions in
your business logic
SwapMining.sol:2960:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
SwapMining.sol:3049:34: Error: Avoid to make time-based decisions in
your business logic
```

SyrupBar.sol

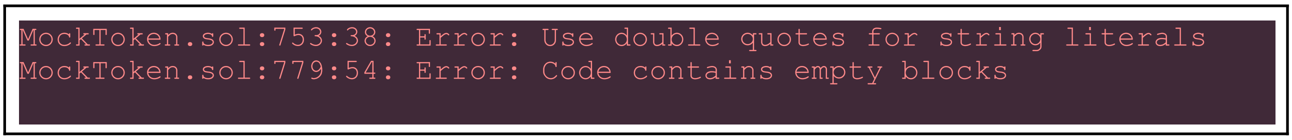


```
SyrupBar.sol:776:29: Error: Use double quotes for string literals
SyrupBar.sol:776:47: Error: Use double quotes for string literals
SyrupBar.sol:891:17: Error: Avoid to make time-based decisions in
your business logic
SyrupBar.sol:1013:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
SyrupBar.sol:1158:17: Error: Avoid to make time-based decisions in
your business logic
SyrupBar.sol:1300:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

MockToken.sol

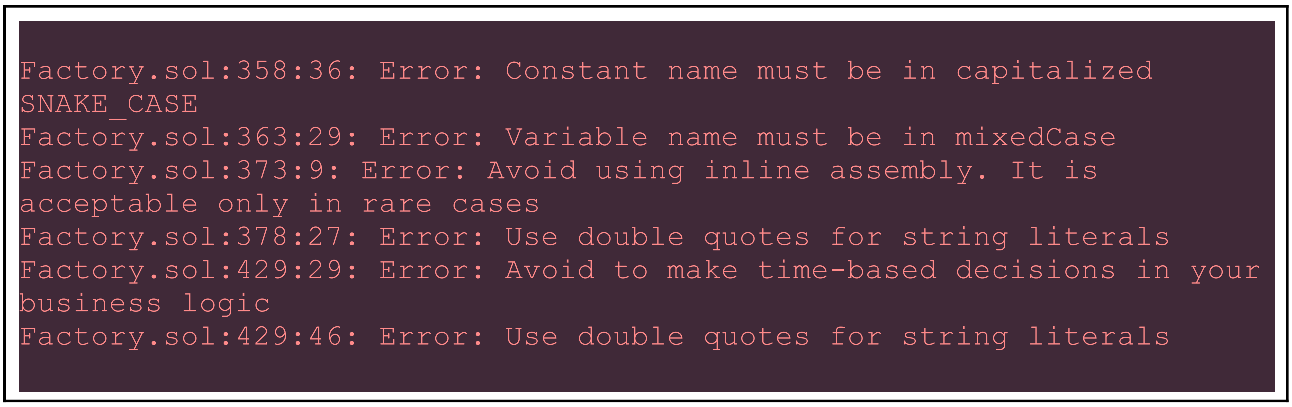


```
MockToken.sol:728:40: Error: Use double quotes for string literals
MockToken.sol:730:61: Error: Use double quotes for string literals
```



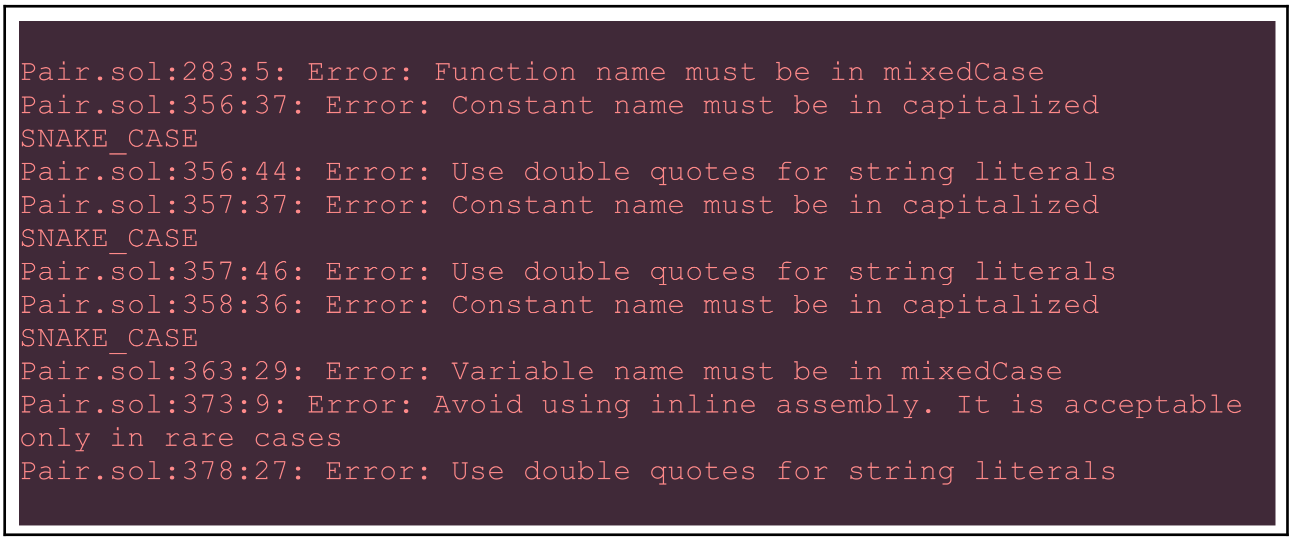
```
MockToken.sol:753:38: Error: Use double quotes for string literals
MockToken.sol:779:54: Error: Code contains empty blocks
```

Factory.sol



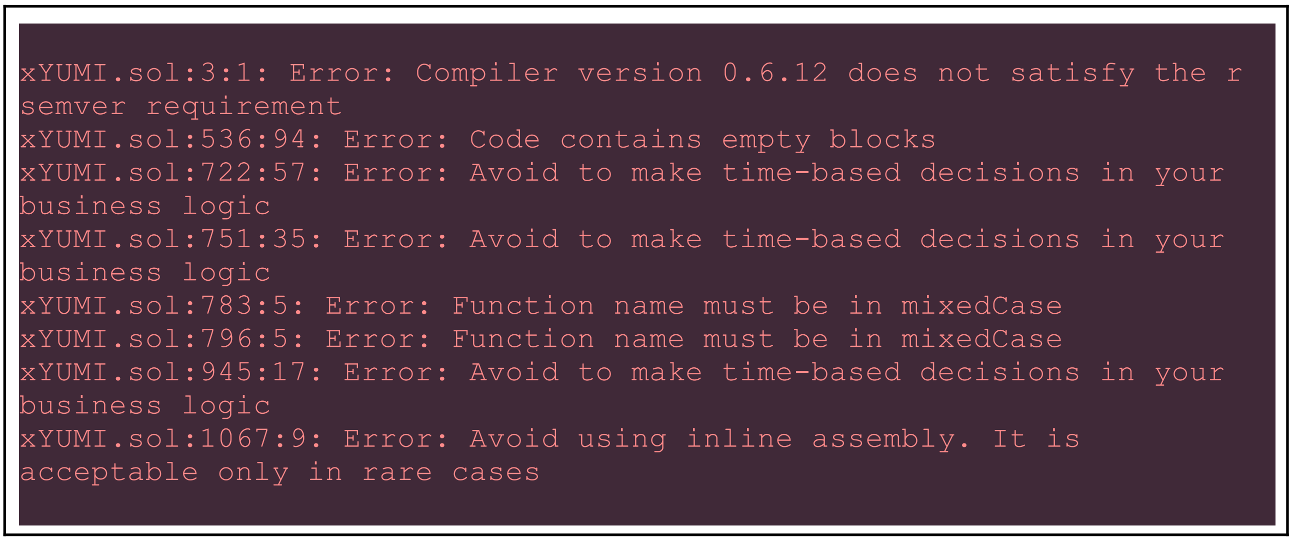
```
Factory.sol:358:36: Error: Constant name must be in capitalized
SNAKE_CASE
Factory.sol:363:29: Error: Variable name must be in mixedCase
Factory.sol:373:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Factory.sol:378:27: Error: Use double quotes for string literals
Factory.sol:429:29: Error: Avoid to make time-based decisions in your
business logic
Factory.sol:429:46: Error: Use double quotes for string literals
```

Pair.sol



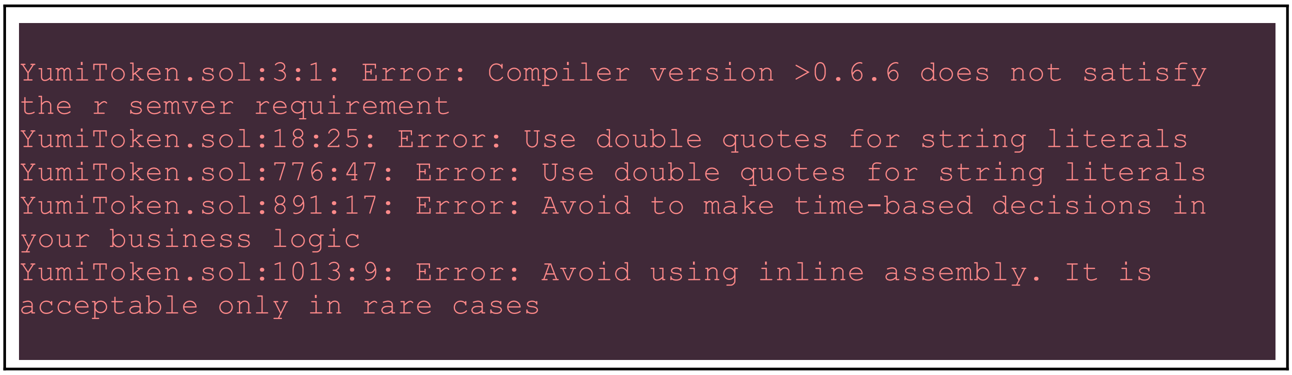
```
Pair.sol:283:5: Error: Function name must be in mixedCase
Pair.sol:356:37: Error: Constant name must be in capitalized
SNAKE_CASE
Pair.sol:356:44: Error: Use double quotes for string literals
Pair.sol:357:37: Error: Constant name must be in capitalized
SNAKE_CASE
Pair.sol:357:46: Error: Use double quotes for string literals
Pair.sol:358:36: Error: Constant name must be in capitalized
SNAKE_CASE
Pair.sol:363:29: Error: Variable name must be in mixedCase
Pair.sol:373:9: Error: Avoid using inline assembly. It is acceptable
only in rare cases
Pair.sol:378:27: Error: Use double quotes for string literals
```

xYUMI.sol



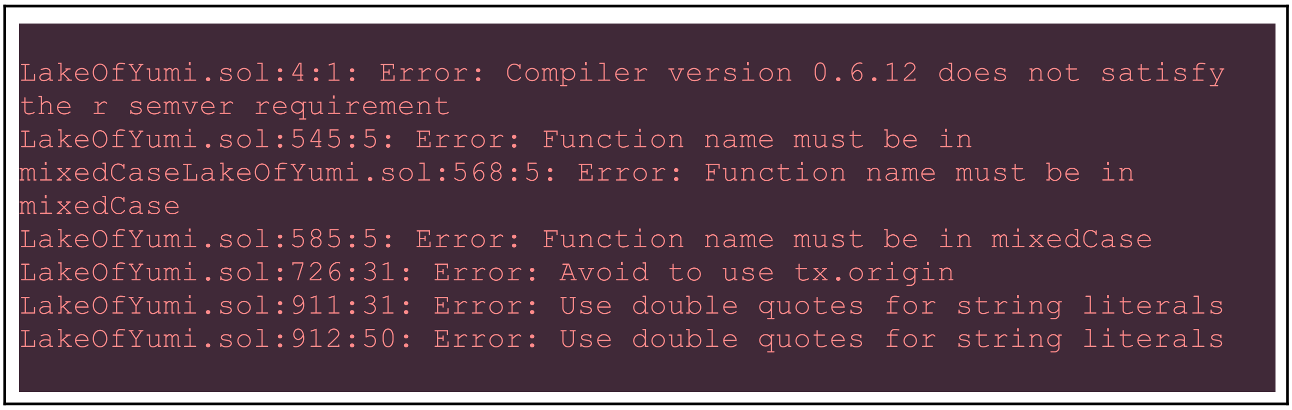
```
xYUMI.sol:3:1: Error: Compiler version 0.6.12 does not satisfy the r
semver requirement
xYUMI.sol:536:94: Error: Code contains empty blocks
xYUMI.sol:722:57: Error: Avoid to make time-based decisions in your
business logic
xYUMI.sol:751:35: Error: Avoid to make time-based decisions in your
business logic
xYUMI.sol:783:5: Error: Function name must be in mixedCase
xYUMI.sol:796:5: Error: Function name must be in mixedCase
xYUMI.sol:945:17: Error: Avoid to make time-based decisions in your
business logic
xYUMI.sol:1067:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

YumiToken.sol



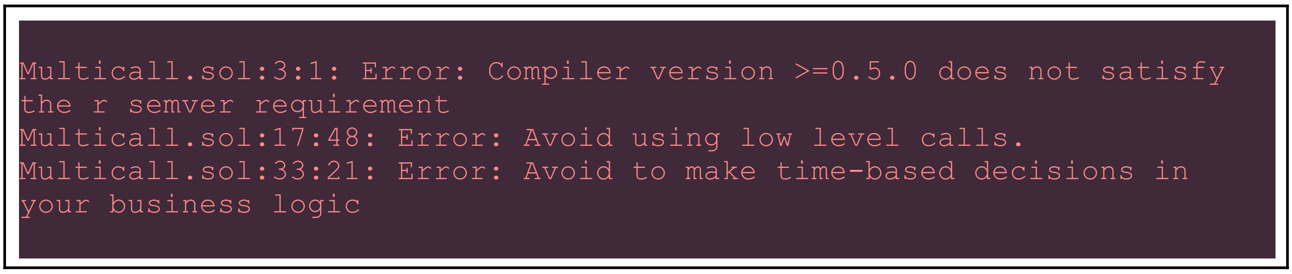
```
YumiToken.sol:3:1: Error: Compiler version >0.6.6 does not satisfy
the r semver requirement
YumiToken.sol:18:25: Error: Use double quotes for string literals
YumiToken.sol:776:47: Error: Use double quotes for string literals
YumiToken.sol:891:17: Error: Avoid to make time-based decisions in
your business logic
YumiToken.sol:1013:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

LakeOfYumi.sol



```
LakeOfYumi.sol:4:1: Error: Compiler version 0.6.12 does not satisfy
the r semver requirement
LakeOfYumi.sol:545:5: Error: Function name must be in
mixedCaseLakeOfYumi.sol:568:5: Error: Function name must be in
mixedCase
LakeOfYumi.sol:585:5: Error: Function name must be in mixedCase
LakeOfYumi.sol:726:31: Error: Avoid to use tx.origin
LakeOfYumi.sol:911:31: Error: Use double quotes for string literals
LakeOfYumi.sol:912:50: Error: Use double quotes for string literals
```

Multicall.sol



```
Multicall.sol:3:1: Error: Compiler version >=0.5.0 does not satisfy
the r semver requirement
Multicall.sol:17:48: Error: Avoid using low level calls.
Multicall.sol:33:21: Error: Avoid to make time-based decisions in
your business logic
```

Software analysis result:

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

EtherAuthority

Email: audit@EtherAuthority.io