Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Sugar Swap Protocol
Website: Sugar Swap
Platform: zkSync era Network
Language: Solidity
Date: April 26th, 2023

Table of contents

Email: audit@EtherAuthority.io

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.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by SugarSwap to perform the Security audit of the Sugar 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 26th, 2023.

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

- SugarSwap is a decentralized exchange (DEX) that is specifically designed to cater to the needs of crypto-native users who want to trade, earn rewards, and participate in gaming activities.
- As a DEX, SugarSwap operates on the zkSync era network, which offers faster transaction times and lower fees compared to the Ethereum mainnet.
- SugarSwap Contracts handle multiple contracts, and all contracts have different functions.
    - SyrupBar: SyrupBar used for SUGAR staking.
    - MasterChef: MasterChef is the master of Sugar, which will be transferred to a governance smart contract once distributed.
    - SugarStakingToken: SugarStakingToken is the place where sugar's live to create xSUGAR. This contract handles swapping to and from xSUGAR, SugarSwap's staking token.
    - Multicall: Aggregate results from multiple read-only function calls.
- Sugar Swap Contracts have functions like add a new pair and LPs, withdraw, deposit, convert, mint, burn, leave, swap, skim, enter, etc.

Audit scope

Name	Code Review and Security Analysis Report for Sugar Swap Protocol Smart Contracts
Platform	**zkSync era Network / Solidity**
File 1	Greeter.sol
File 1 MD5 Hash	C60AF4A99CE474FCC4797277C4F95E88
File 2	MasterChef.sol
File 2 MD5 Hash	3181D61377A7B522A69239AB890DB0ED
File 3	NFTController.sol
File 3 MD5 Hash	6AAE550160948A4C6E4028309D9CC9DA
File 4	SmartChef.sol
File 4 MD5 Hash	9DC1FE3609527A0398A2FB4563F306EC
File 5	SwapMining.sol
File 5 MD5 Hash	AB50C6C5A21D581F67D19ECC45103176
File 6	SyrupBar.sol
File 6 MD5 Hash	690BF6A147D29EB0300E4CEAE99DE9F3
File 7	SugarswapFactory.sol
File 7 MD5 Hash	1A6FCC03EB60E423A55F62A488D88851
File 8	SugarswapPair.sol
File 8 MD5 Hash	0B300D596064EB7DF2886FBA7A058B54
File 9	SugarswapRouter.sol
File 9 MD5 Hash	7685E7074F58D33B6DD8B535361AF48C
File 10	SugarToken.sol
File 10 MD5 Hash	13920529702545E1CABF7BE3021BEC7F
File 11	SugarStakingToken.sol
File 11 MD5 Hash	8ABF25872B916B11159F17A869279B36

File 12	IDO.sol
File 12 MD5 Hash	9300A16BA156F1AC011E0B574C54C58B
File 13	LakeOfSugar.sol
File 13 MD5 Hash	A547B3F3732D52D549FFB51A9494F5C9
File 14	Multicall.sol
File 14 MD5 Hash	B31A5401C236F10109672BC3D903C9DA
File 15	WETH9.sol
File 15 MD5 Hash	93741B992586D0B856AE852DDB678B38
File 16	ERC20.sol
File 16 MD5 Hash	3E9E55F05CF95A414E0CE704EF99E6F1
File 17	Oracle.sol
File 17 MD5 Hash	D7FF8878125A049FBF8B4D86DC5F0EB5
Audit Date	April 26th, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 Greeter.sol • Set a greeting string memory.	**YES, This is valid.**
File 2 MasterChef.sol • Bonus multiplier: 1 • The SUGAR token max total supply 28,382,400. • NFT Boost Rate: 1% • MasterChef is the master of Sugar. **Owner has control over following functions:** • Set a multiplier number. • Add a new lp to the pool. • Update the given pool's SUGAR allocation point. • Changes cake token reward per second. • Set a NftBoost rate value. • Update trademining contract address. **Other Specifications:** • MasterChef is the master of Sugar, which is ownable and has tremendous power. It will be transferred to a governance smart contract once SUGAR it is sufficiently distributed and the community can govern itself.	**YES, This is valid.**
File 3 NFTController.sol **Owner has control over following functions:** • Set a Whitelisted address value. • Set a Default Boost rate value. • Set a Boost rate value.	**YES, This is valid.**
File 4 Oracle.sol	**YES, This is valid.**

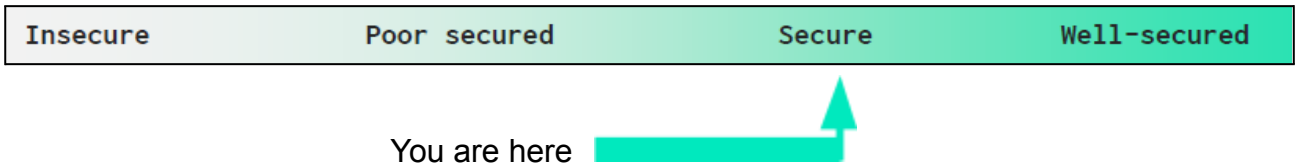
• Cycle: 1800 • Oracle can update token addresses.	
File 5 SmartChef.sol **<u>Owner has control over following functions:</u>** • :Set a stop reward time. • Withdraw DepositFee to buy back and burn. • Withdraw emergency reward. • Rescues random funds stuck. • Rescues random BNB funds stuck.	**YES, This is valid.**
File 6 SwapMining.sol **<u>Owner has control over following functions:</u>** • Add a Pair address. • Update a Pair address. • Set the number of sugar produced by each second. • Only tokens in the whitelist can be mined MDX. • Remove whitelisted addresses. • Set a halving period. • Set a router address. • Set an oracle address. • Add a Blacklist address • Remove a Blacklist address.	**YES, This is valid.**
File 7 SyrupBar.sol • Name: SugarSwapBar Token • Symbol: SYRUP • SyrupBar used for SUGAR staking. **<u>Owner has control over following functions:</u>** • Creates `_amount` token to `_to` by the owner (MasterChef).	**YES, This is valid.**

• Burn Token By The Owner. • Safe cake transfer.	
File 8 Factory.sol **<u>Owner has control over following functions:</u>** • Set a fee address. • Set a Fee to the setter address.	**YES, This is valid.**
File 9 Pair.sol • Minimum Liquidity: 1000 **<u>Owner has control over following functions:</u>** • Initialize once by the factory at time of deployment by the owner.	**YES, This is valid.**
File 10 Router.sol **<u>Owner has control over following functions:</u>** • Set a swap mining address.	**YES, This is valid.**
File 11 ERC20.sol • Decimals: 18 **<u>Owner has control over following functions:</u>** • Owner can create `amount` tokens and assign them to Owner, increasing the total supply.	**YES, This is valid.**
File 12 SugarToken.sol • Name: SugarSwap Token • Symbol: SUGAR • Decimals: 18 • SugarToken with Governance. **<u>Owner has control over following functions:</u>** • Owner can create `amount` tokens and assign them to Owner, increasing the total supply.	**YES, This is valid.**
File 13 SugarStakingToken.sol • Name: Sugar Staking Token • Symbol: xSUGAR	**YES, This is valid.**

• Decimals: 18 **Other Specifications:** • SugarStakingToken is the place where sugar's live to create xSUGAR. • This contract handles swapping to and from xSUGAR, SugarSwap's staking token. **Owner has control over following functions:** • Set a delay to withdraw time. • Update admin address by the previous admin.	
File 14 IDO.sol **Owner has control over following functions:** • Set an offering amount. • Set a raising amount. • Set a deposit limit per wallet. • User addresses included in the whitelist. • Final amount withdrawal.	**YES, This is valid.**
File 15 LakeOfSugar.sol **Owner has control over following functions:** • Add an auth address. • Revoke an auth address. • Set a Bridge address. • Set a developer address. • Set a developer cut amount.	**YES, This is valid.**
File 16 Multicall.sol • Multicall - Aggregate results from multiple read-only function calls.	**YES, This is valid.**
File 17 WETH9.sol • Name: Wrapped Ether • Symbol: WETH • Decimals: 18	**YES, This is valid.**

Audit Summary

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



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

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

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 17 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 Sugar 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 Sugar Swap Protocol.

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

Some code parts are not well commented on smart contracts. We suggest using Ethereum's NatSpec style for the commenting.

Documentation

We were given a Sugar Swap Protocol smart contract code in the form of a file. The hash of that code is mentioned above in the table.

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

Another source of information was its official website: https://defillama.com/protocol/sugar-swap 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

Greeter.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	greet	read	Passed	No Issue
3	setGreeting	write	Passed	No Issue

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	getBoost	read	Passed	No Issue
7	getSlots	read	Passed	No Issue
8	getTokenIds	read	Passed	No Issue
9	updateMultiplier	write	access only Owner	No Issue
10	poolLength	external	Passed	No Issue
11	nonDuplicatedLP	modifier	Passed	No Issue
12	add	write	Critical operation lacks event log	Refer Audit Findings
13	set	write	Critical operation lacks event log	Refer Audit Findings
14	depositNFT	write	Passed	No Issue
15	withdrawNFT	write	Passed	No Issue
16	getMultiplier	read	Passed	No Issue
17	pendingCake	external	Passed	No Issue
18	massUpdatePools	write	Passed	No Issue
19	updatePool	write	Critical operation lacks event log	Refer Audit Findings
20	deposit	write	Passed	No Issue
21	withdraw	write	Passed	No Issue
22	emergencyWithdraw	write	Passed	No Issue
23	safeCakeTransfer	internal	Passed	No Issue
24	setCakePerSecond	external	access only Owner	No Issue
25	setNftController	write	access only Owner	No Issue
26	setNftBoostRate	write	access only Owner	No Issue
27	setDevaddr	write	Passed	No Issue
28	setReserveaddr	write	Passed	No Issue

29	setMiningaddr	external	access only Owner	No Issue

NFTController.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	getBoostRate	external	Passed	No Issue
7	setWhitelist	external	access only Owner	No Issue
8	setDefaultBoostRate	external	access only Owner	No Issue
9	setBoostRate	external	access only Owner	No Issue

Oracle.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	update	external	Passed	No Issue
3	computeAmountOut	write	Passed	No Issue
4	consult	external	Passed	No Issue

SmartChef.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	stopReward	write	access only Owner	No Issue
7	getMultiplier	read	Passed	No Issue
8	pendingReward	external	Passed	No Issue
9	updatePool	write	Passed	No Issue
10	massUpdatePools	write	Passed	No Issue
11	deposit	write	Passed	No Issue
12	withdraw	write	Passed	No Issue
13	emergencyWithdraw	write	Passed	No Issue

14	emergencyRewardWithdraw	write	access only Owner	No Issue
15	withdrawDepositFee	write	access only Owner	No Issue
16	inCaseTokensGetStuck	external	access only Owner	No Issue
17	inCaseBNBGetStuck	external	access only Owner	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	setSugarswapPerSecond	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	getSugarReward	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	safeSugarTransfer	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	read	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	mint	write	access only Owner	No Issue
15	_transfer	internal	Passed	No Issue
16	_mint	internal	Passed	No Issue
17	_burn	internal	Passed	No Issue
18	_approve	internal	Passed	No Issue
19	_burnFrom	internal	Passed	No Issue
20	mint	write	access only Owner	No Issue
21	burn	write	access only Owner	No Issue
22	safeCakeTransfer	write	access only Owner	No Issue
23	delegates	external	Passed	No Issue
24	delegate	external	Passed	No Issue
25	delegateBySig	external	Passed	No Issue
26	getCurrentVotes	external	Passed	No Issue
27	getPriorVotes	external	Passed	No Issue
28	_delegate	internal	Passed	No Issue
29	_moveDelegates	internal	Passed	No Issue
30	_writeCheckpoint	internal	Passed	No Issue
31	safe32	internal	Passed	No Issue
32	getChainId	internal	Passed	No Issue

SugarswapFactory.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

SugarswapPair.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	lock	modifier	Passed	No Issue
11	getReserves	read	Passed	No Issue
12	_safeTransfer	write	Passed	No Issue
13	initialize	external	Passed	No Issue
14	_update	write	Passed	No Issue
15	_mintFee	write	Passed	No Issue
16	mint	external	lock	No Issue
17	burn	external	lock	No Issue
18	swap	external	lock	No Issue
19	skim	external	lock	No Issue
20	sync	external	lock	No Issue

SugarswapRouter.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	ensure	modifier	Passed	No Issue
7	setSwapMining	write	access only Owner	No Issue
8	receive	external	Passed	No Issue
9	_addLiquidity	internal	Passed	No Issue
10	addLiquidity	external	Passed	No Issue
11	addLiquidityETH	external	Passed	No Issue
12	removeLiquidity	write	Passed	No Issue
13	removeLiquidityETH	write	Passed	No Issue
14	removeLiquidityWithPermit	external	Passed	No Issue
15	removeLiquidityETHWithPermit	external	Passed	No Issue
16	removeLiquidityETHSupportingFeeOnTransferTokens	write	Passed	No Issue
17	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	external	Passed	No Issue
18	_swap	internal	Passed	No Issue
19	swapExactTokensForTokens	external	Passed	No Issue
20	swapTokensForExactTokens	external	Passed	No Issue
21	swapExactETHForTokens	external	Passed	No Issue
22	swapTokensForExactETH	external	Passed	No Issue
23	swapExactTokensForETH	external	Passed	No Issue
24	swapETHForExactTokens	external	Passed	No Issue
25	_swapSupportingFeeOnTransferTokens	internal	Passed	No Issue
26	swapExactTokensForTokensSupportingFeeOnTransferTokens	external	Passed	No Issue
27	swapExactETHForTokensSupportingFeeOnTransferTokens	external	Passed	No Issue
28	quote	write	Passed	No Issue
29	getAmountOut	write	Passed	No Issue
30	getAmountIn	write	Passed	No Issue
31	getAmountsOut	read	Passed	No Issue
32	getAmountsIn	read	Passed	No Issue

ERC20.sol

Functions

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

6	totalSupply	read	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	transfer	write	Passed	No Issue
9	allowance	read	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	mint	write	access only Owner	No Issue
15	_transfer	internal	Passed	No Issue
16	_mint	internal	Passed	No Issue
17	_burn	internal	Passed	No Issue
18	_approve	internal	Passed	No Issue
19	_burnFrom	internal	Passed	No Issue
20	owner	read	Passed	No Issue
21	onlyOwner	modifier	Passed	No Issue
22	renounceOwnership	write	access only Owner	No Issue
23	transferOwnership	write	access only Owner	No Issue

SugarToken.sol

Functions

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

SugarStakingToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getOwner	external	Passed	No Issue
3	name	read	Passed	No Issue
4	decimals	read	Passed	No Issue
5	symbol	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	transfer	write	Passed	No Issue
9	allowance	read	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	mint	write	access only Owner	No Issue
15	_transfer	internal	Passed	No Issue
16	_mint	internal	Passed	No Issue
17	_burn	internal	Passed	No Issue
18	_approve	internal	Passed	No Issue
19	_burnFrom	internal	Passed	No Issue
20	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	SUGARBalance	external	Passed	No Issue
26	xSUGARForSUGAR	external	Passed	No Issue
27	SUGARForxSUGAR	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

IDO.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	nonReentrant	modifier	Passed	No Issue
3	onlyAdmin	modifier	Passed	No Issue
4	setOfferingAmount	write	access only Admin	No Issue
5	setRaisingAmount	write	access only Admin	No Issue
6	setDepositLimitPerWallet	write	access only Admin	No Issue
7	deposit	write	Passed	No Issue
8	harvest	write	Passed	No Issue
9	hasHarvest	external	Passed	No Issue
10	getUserAllocation	read	Passed	No Issue
11	getOfferingAmount	read	Passed	No Issue
12	getRefundingAmount	read	Passed	No Issue
13	isWhitelisted	read	Passed	No Issue
14	getAddressListLength	external	Passed	No Issue
15	includeToWhiteList	write	access only Admin	No Issue
16	finalWithdraw	write	access only Admin	No Issue
17	setLiquidityIsCreated	write	access only Admin	No Issue

LakeOfSugar.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	external	Passed	No Issue
7	revokeAuth	external	access only Owner	No Issue
8	addAuth	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	write	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	_toSUGAR	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

WETH9.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	deposit	write	Passed	No Issue
3	withdraw	write	Passed	No Issue
4	totalSupply	read	Passed	No Issue
5	approve	write	Passed	No Issue
6	transfer	write	Passed	No Issue
7	transferFrom	write	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens 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.

Email: audit@EtherAuthority.io

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

SugarStakingToken.sol

- enter.
- leave

SwapMining.sol

- addPair
- setPair
- mint
- ownerWithdraw
- takerWithdraw

Resolution: Write an event log for listed events.

Very Low / Informational / Best practices:

(1) Use the latest solidity version: **- NFTController.sol, SwapMining.sol, xSUGAR.sol, WETH9.sol, LakeOfSugar.sol, IDO.sol, Ownable.sol, EnumerableSet.sol, Context.sol, IERC721.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:

MasterChef.sol

- updateMultiplier: Multiplier number can be updated by the owner.
- add: Add a new lp to the pool by the owner.
- set: Update the given pool's SUGAR allocation point by the owner.
- setCakePerSecond: Changes cake token reward per second by the owner.
- setNftController: NftController address can be set by the owner.
- setNftBoostRate: NftBoost rate can be set by the owner.
- setDevaddr: Update dev address by the previous dev address.
- setReserveaddr: Update reserve address by the previous reserve address.
- setMiningaddr: Update trademing contract address can be set by the owner.

NFTController.sol

- setWhitelist: Whitelisted address values can be set by the owner.
- setDefaultBoostRate: Default Boost rate value can be set by the owner.
- setBoostRate: Boost rate value can be set by the owner.

SmartChef.sol

- stopReward: Stop reward time can be set by the owner.
- emergencyRewardWithdraw: Withdraw emergency reward by the owner.
- withdrawDepositFee: Withdraw DepositFee to buy back and burn by the owner.
- inCaseTokensGetStuck: Rescues random funds stuck by the owner.
- inCaseBNBGetStuck: Rescues random BNB funds stuck by the owner.

SwapMining.sol

- addPair: Add a Pair address by the owner.
- setPair: Update a Pair address by the owner.
- setSugarswapPerSecond: Set the number of sugar produced by each second by the owner.
- addWhitelist: Only tokens in the whitelist can be mined MDX by the owner.
- delWhitelist: Remove whitelisted addresses by the owner.
- setHalvingPeriod: Halving Period can be set by the owner.
- setRouter: Router address can be set by the owner.
- setOracle: Oracle address can be set by the owner.
- swap: Swap mining by the router owner.
- ownerWithdraw: Withdraw tokens by the owner.
- addBlacklist: Add a Blacklist address by the owner.
- removeBlacklist: Remove a Blacklist address by the owner.

SyrupBar.sol

- mint: Creates `_amount` token to `_to` by the owner (MasterChef).
- burn: burn token by the owner.
- safeCakeTransfer: Safe cake transfer by the owner.

Factory.sol

- setFeeTo: Fees to address can be set by the owner.
- setFeeToSetter: Fee to setter address can be set by the owner.

Pair.sol

- initialize: Initialize once by the factory at time of deployment by the owner.

Router.sol

- setSwapMining: Swap mining address can be set by the owner.

ERC20.sol

- mint: Owner can create `amount` tokens and assign them to Owner, increasing the total supply.

SugarToken.sol

- mintFor: Owner can create `amount` tokens and assign them to Owner, increasing the total supply.
- mint: Owner can create `amount` tokens and assign them to Owner, increasing the total supply.

xSUGAR.sol

- setDelayToWithdraw: Delay to withdraw time can be set by the owner.
- setAdmin: Update admin address by the previous admin.

IDO.sol

- setOfferingAmount: Offering amount can be set by the admin.
- setRaisingAmount: Raising amount can be set by the admin.
- setDepositLimitPerWallet: Deposit limit per wallet can be set by the admin.
- includeToWhiteList: User addresses included in whitelist by the admin.
- finalWithdraw: Final withdrawal by the admin.

LakeOfSugar.sol

- addAuth: Add an auth address by the owner.

- revokeAuth: Revoke an auth address by the owner.
- setAnyAuth: Setting anyAuth to true allows anyone to call functions protected by only Auth.
- setBridge: Bridge address can be set by the owner.
- setDevCut: Developer cut amount can be set by the owner.
- setDevAddr: Developer address can be set by the owner.
- convert: The onlyEOA modifier prevents this being done with a flash loan.
- convertMultiple: The onlyEOA modifier prevents this being done with a flash loan multiple.

Ownable.sol

- renounceOwnership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transferOwnership: Current owner can transfer ownership of the contract to a new account.

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

Conclusion

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

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

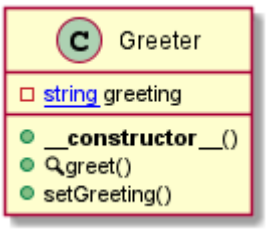
Technical Disclaimer

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

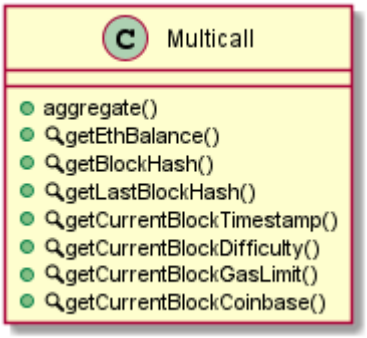
Appendix

Code Flow Diagram - Sugar Swap Protocol

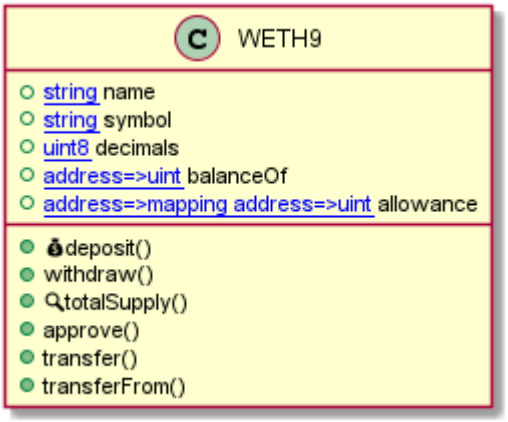
Greeter Diagram



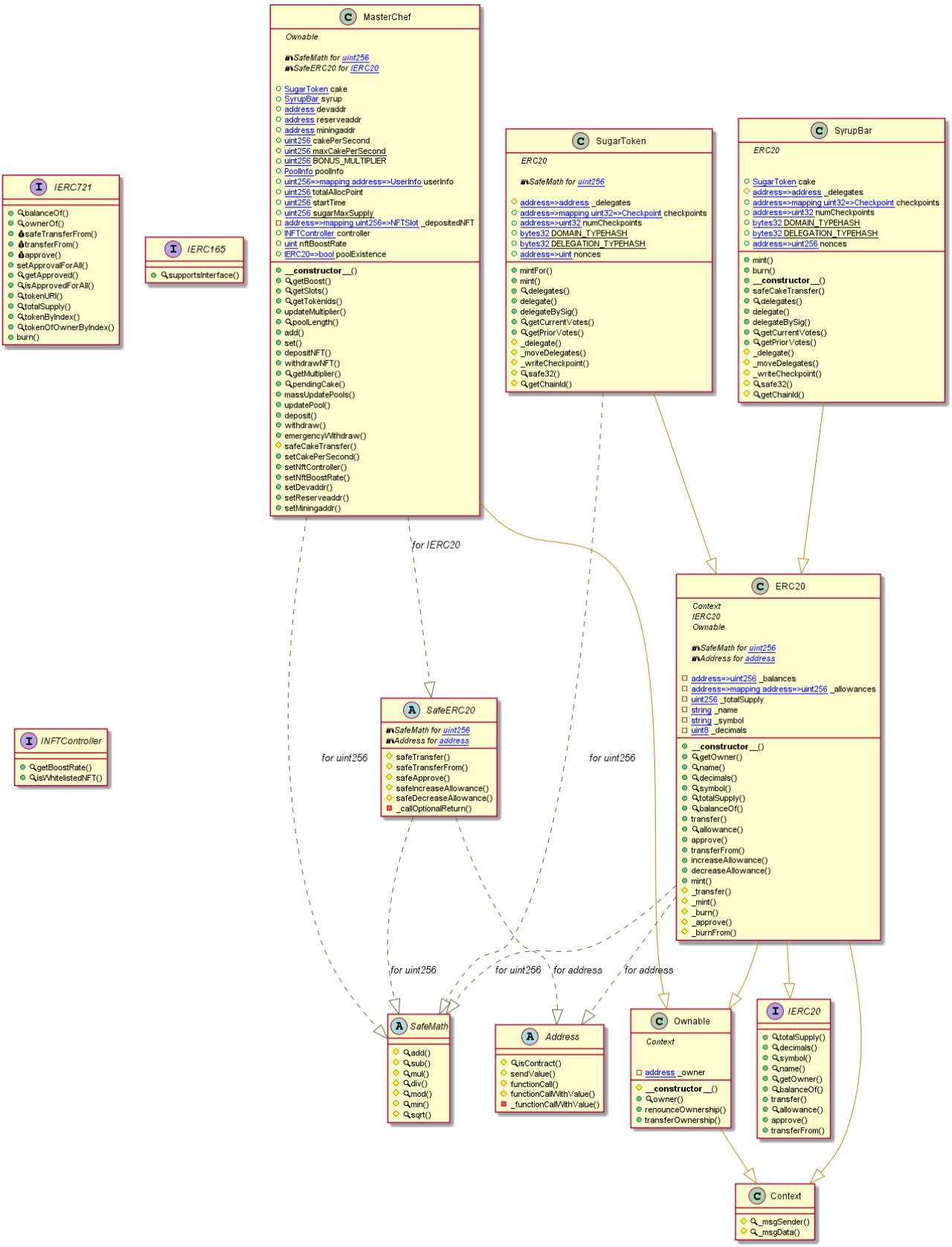
Multicall Diagram



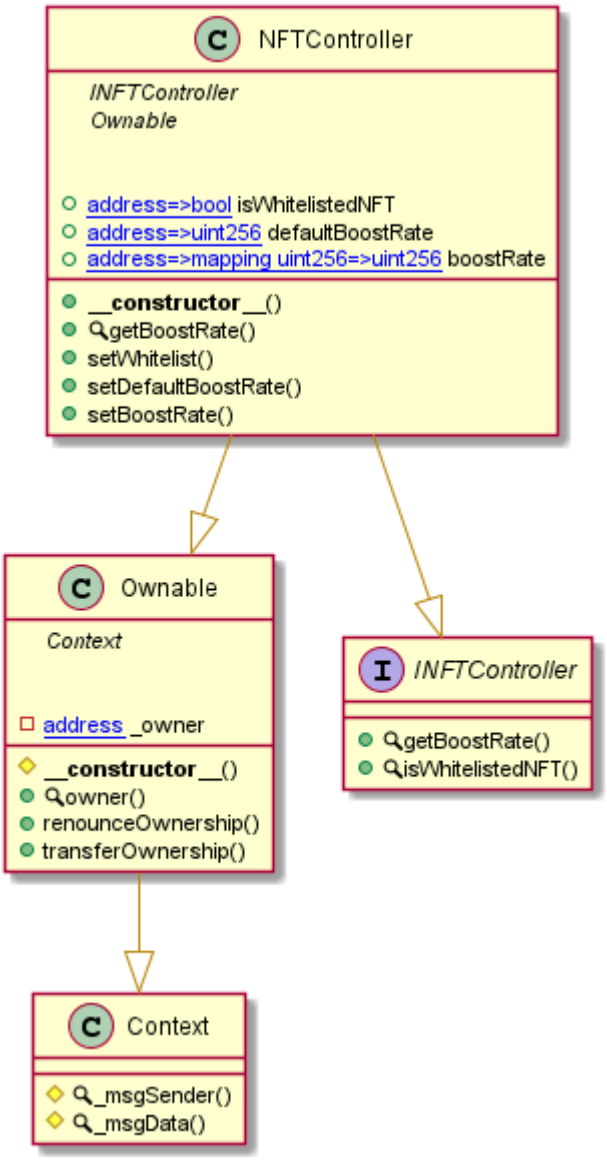
WETH9 Diagram



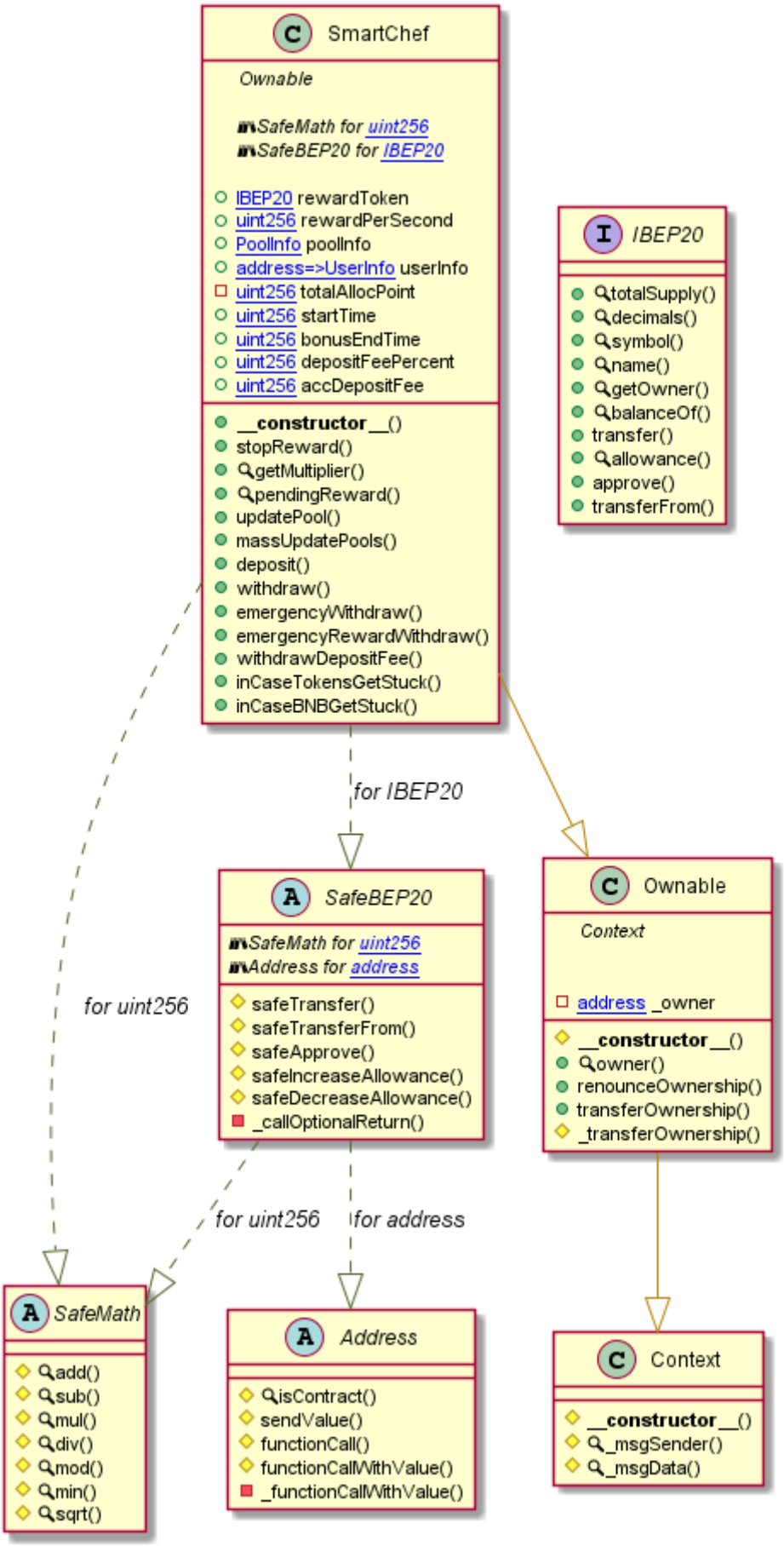
MasterChef Diagram



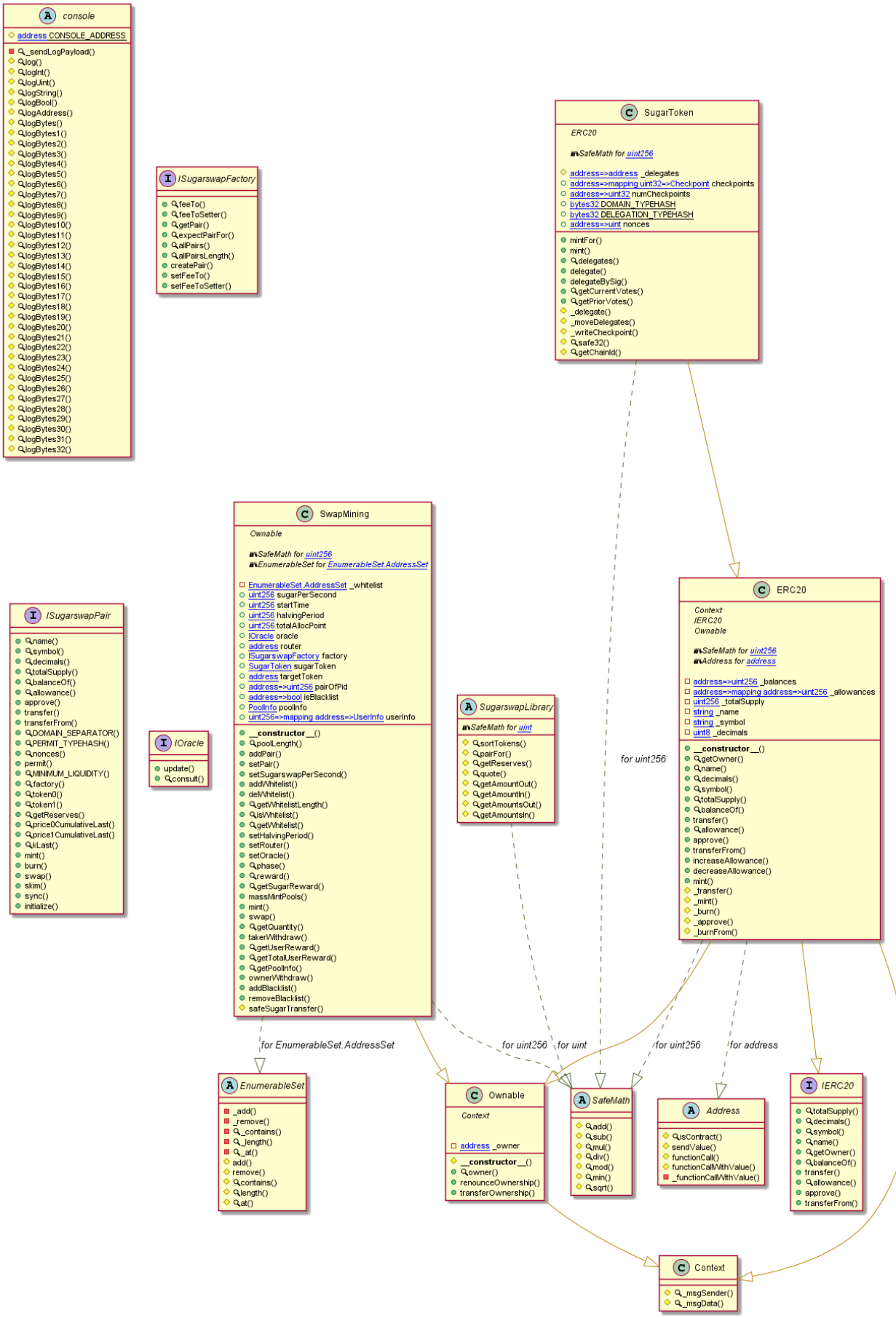
NFTController Diagram



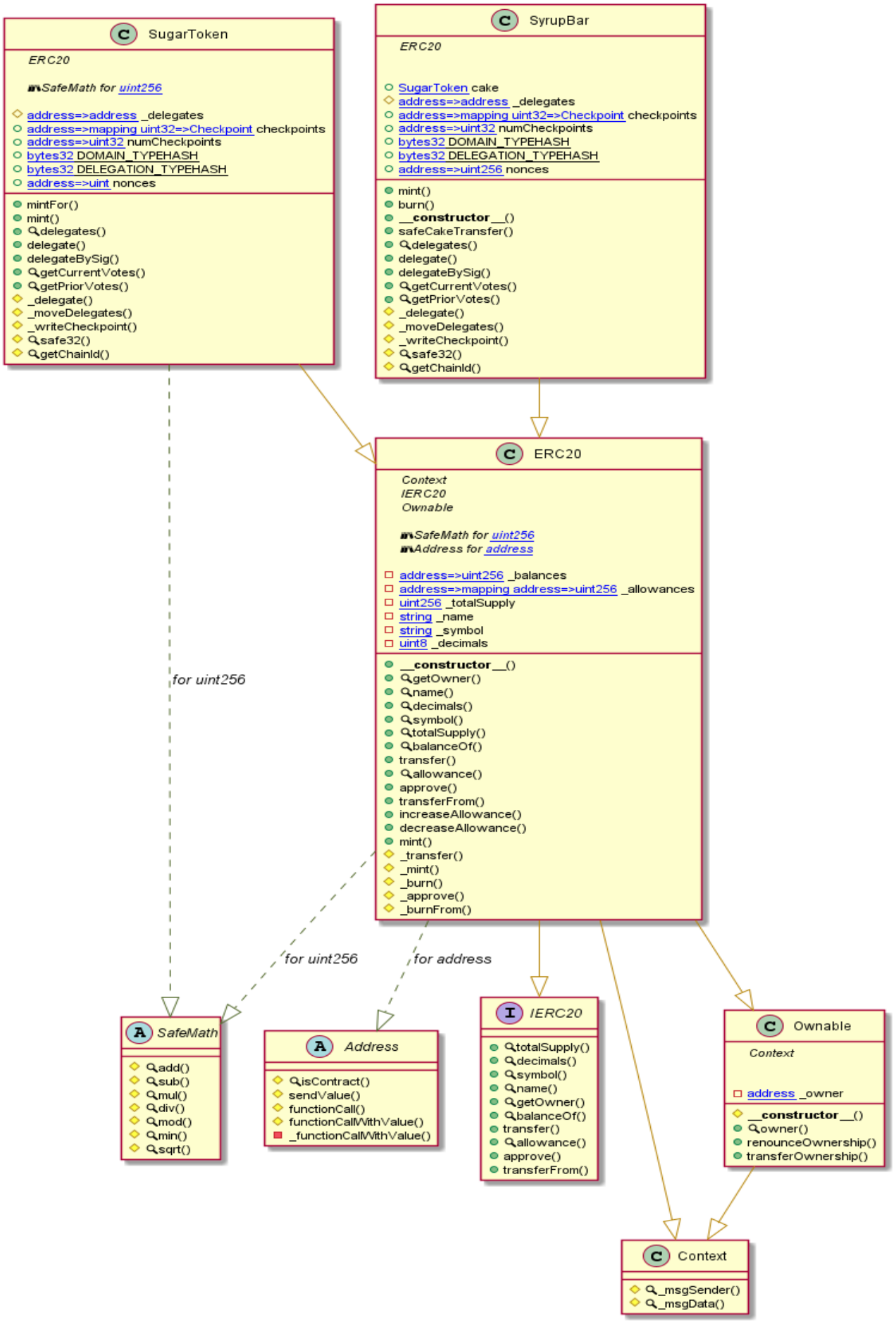
SmartChef Diagram



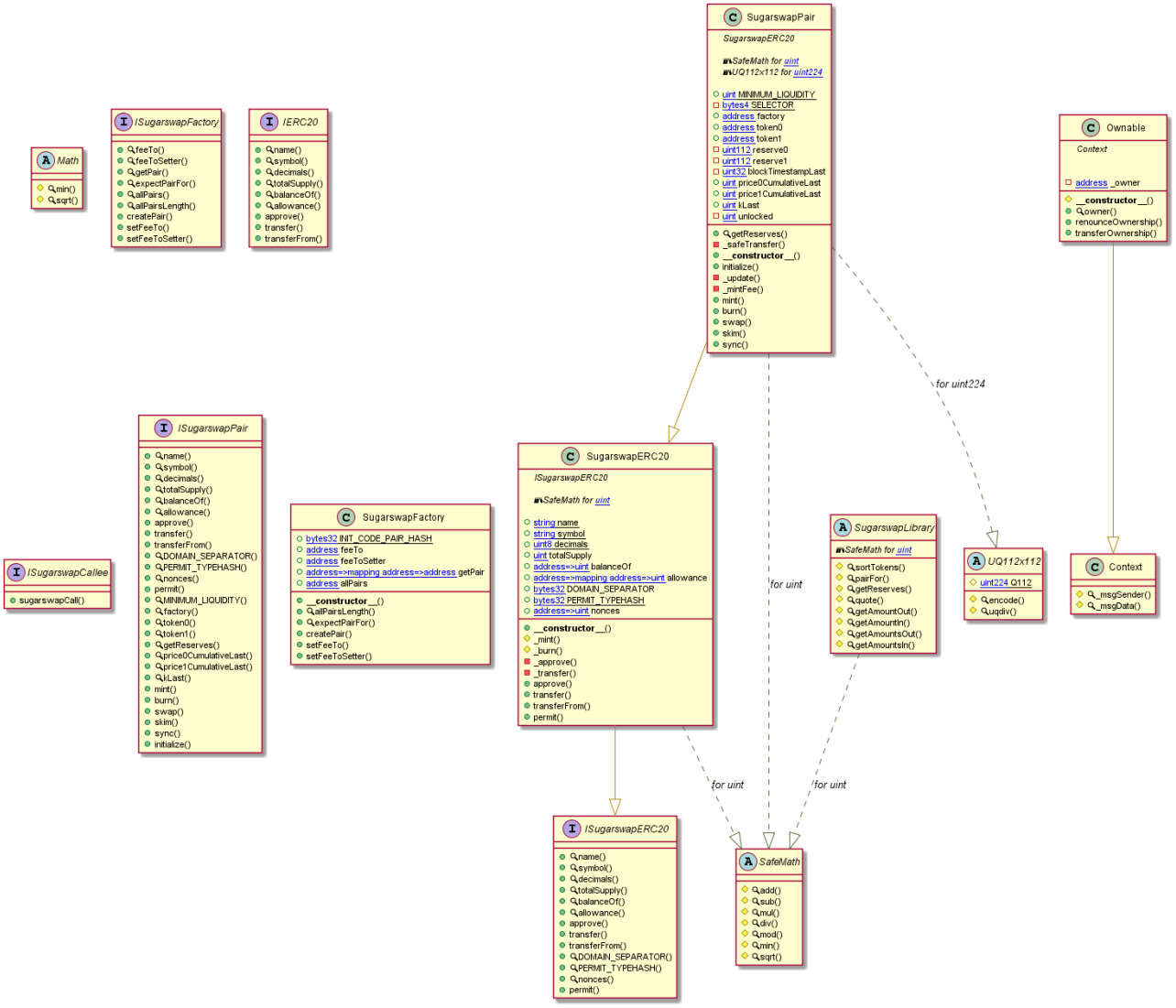
SwapMining Diagram



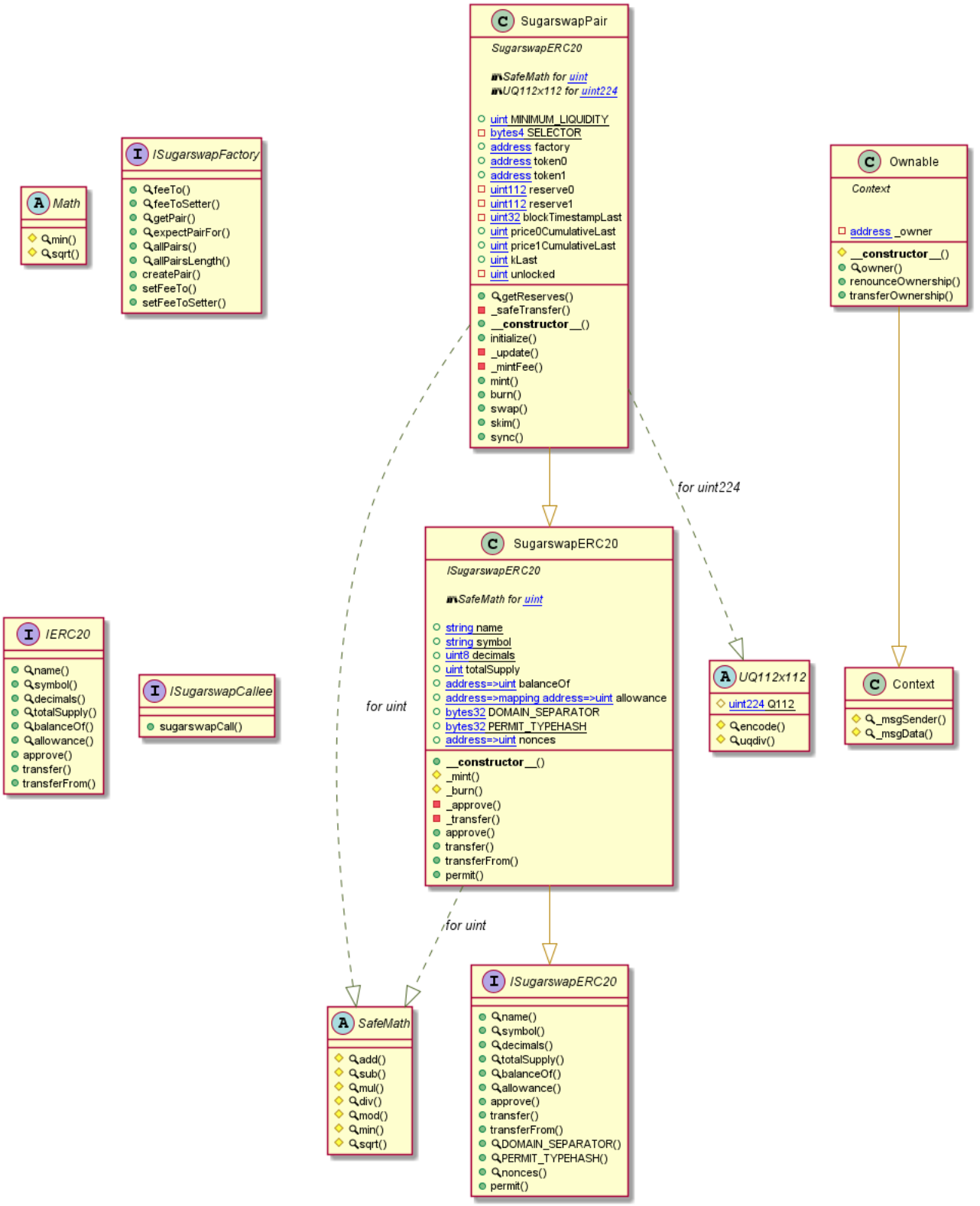
SyrupBar Diagram



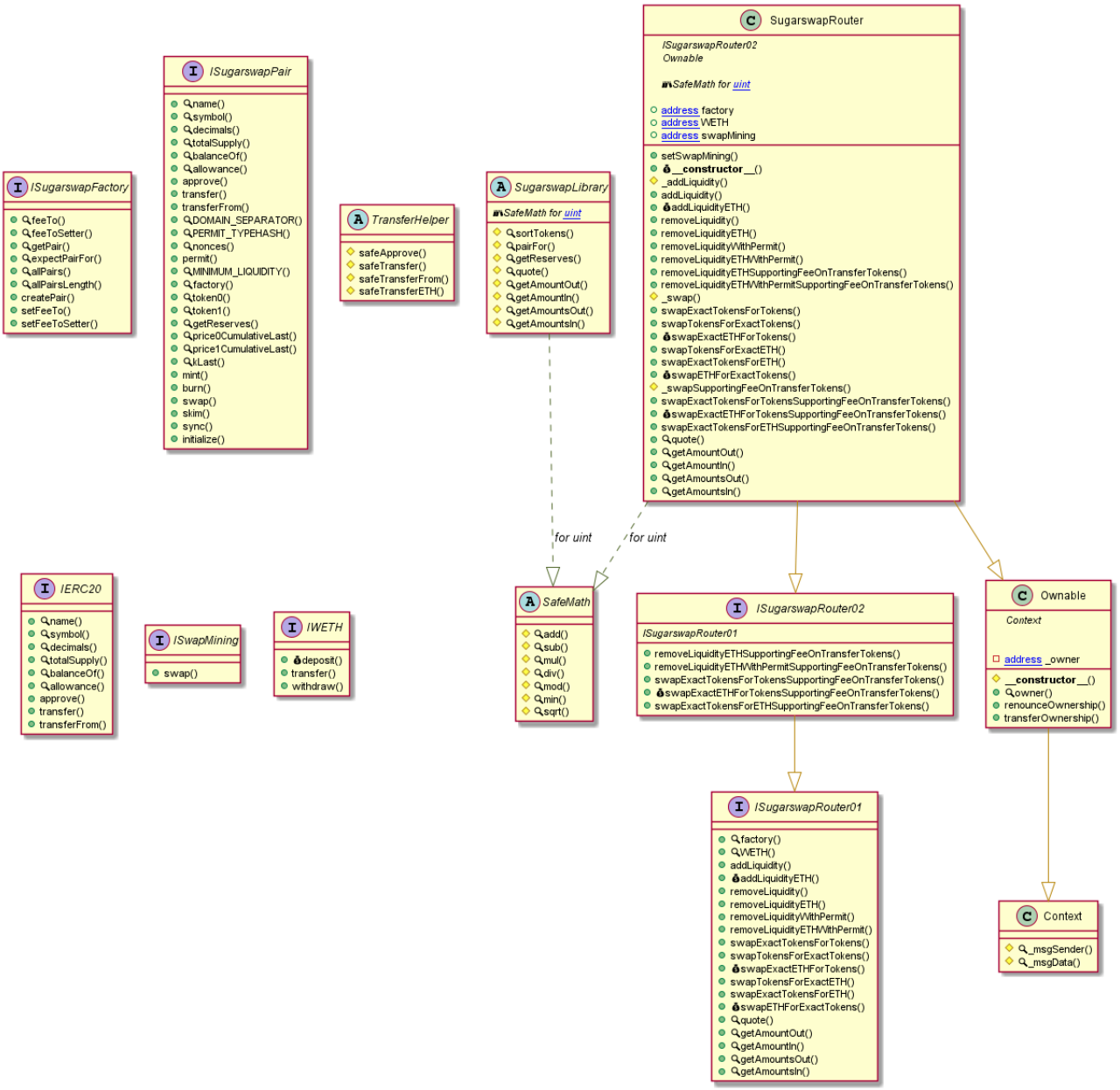
SugarswapFactory Diagram



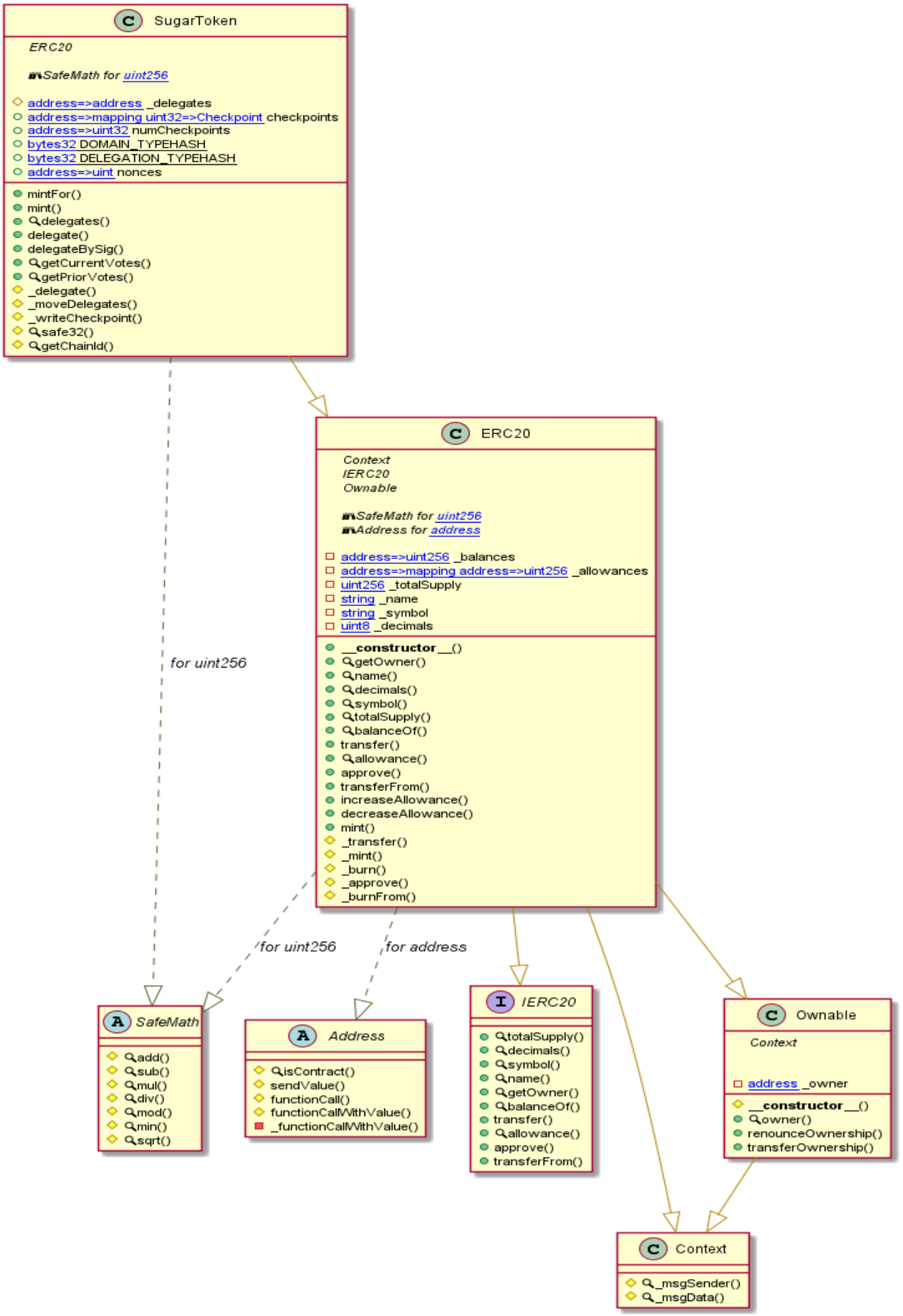
SugarswapPair Diagram



SugarswapRouter Diagram

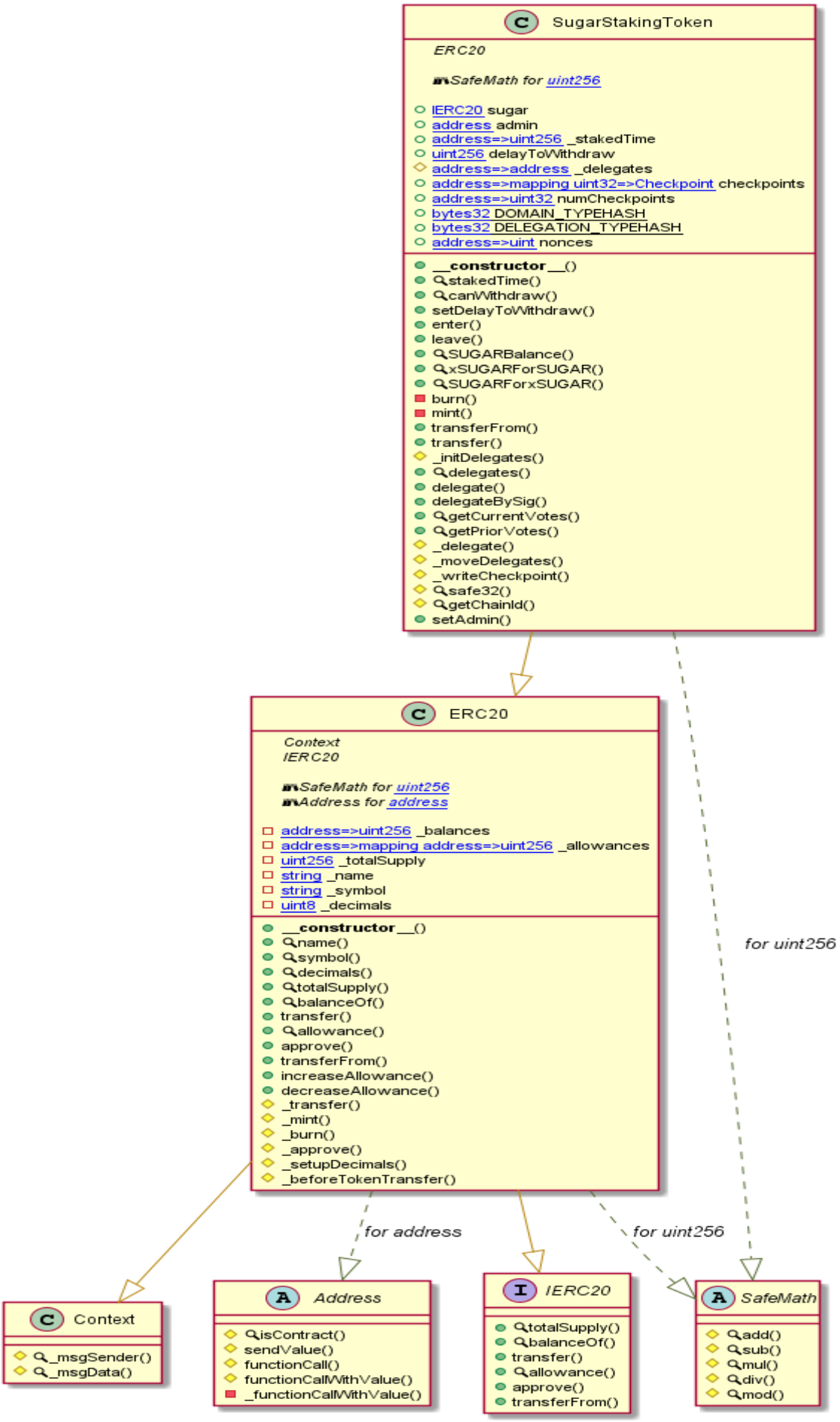


SugarToken Diagram



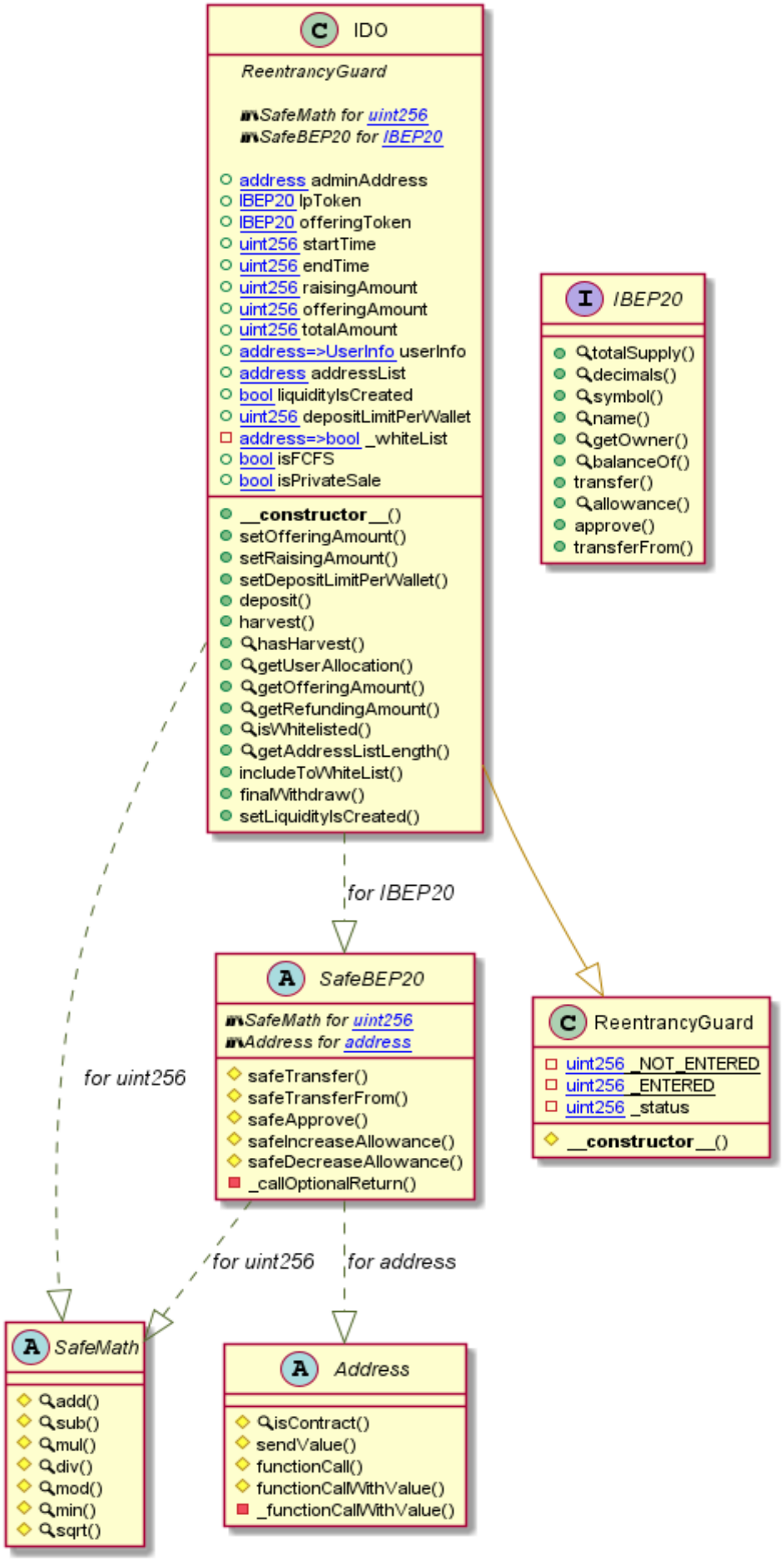
Email: audit@EtherAuthority.io

SugarStakingToken Diagram

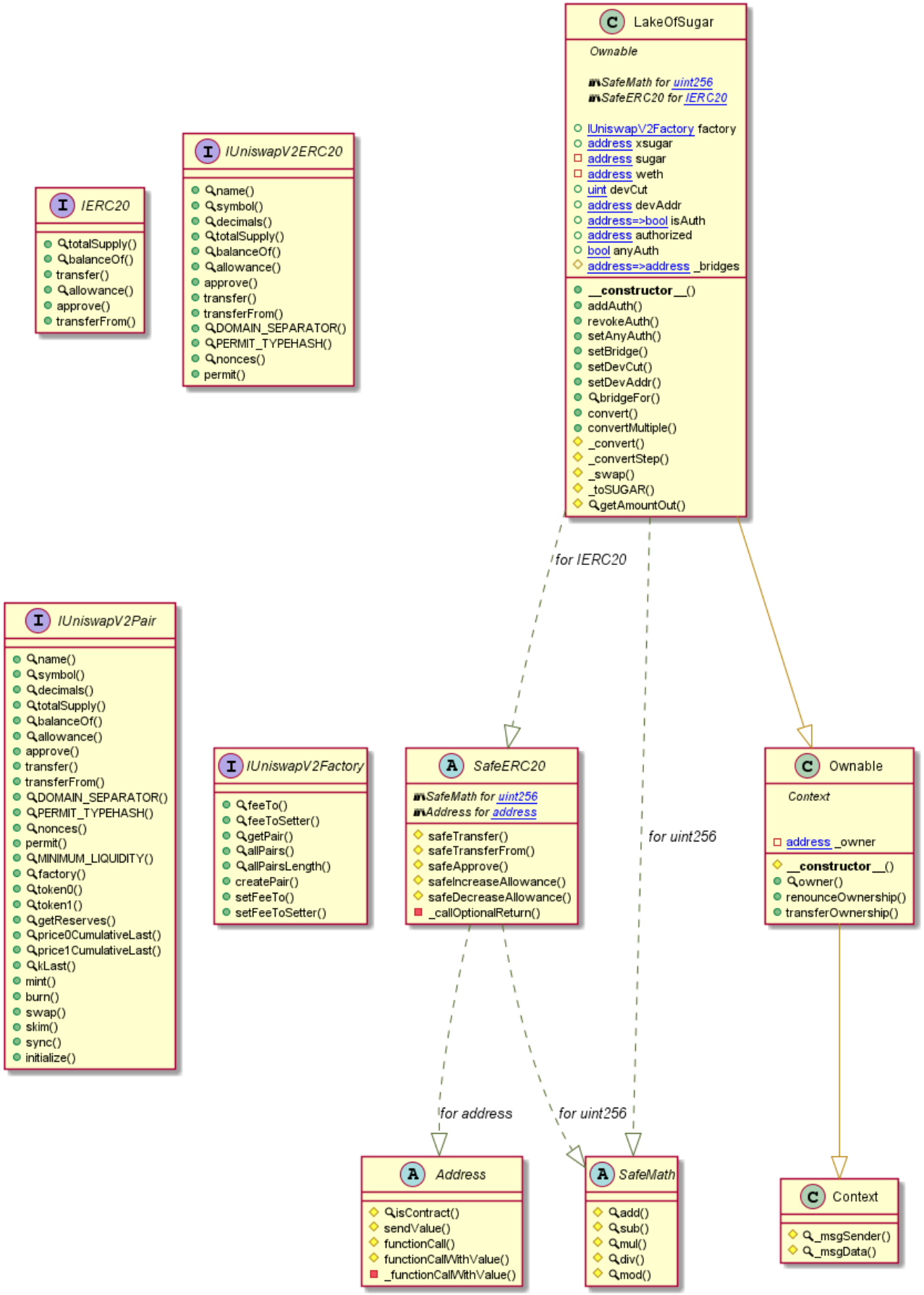


Email: audit@EtherAuthority.io

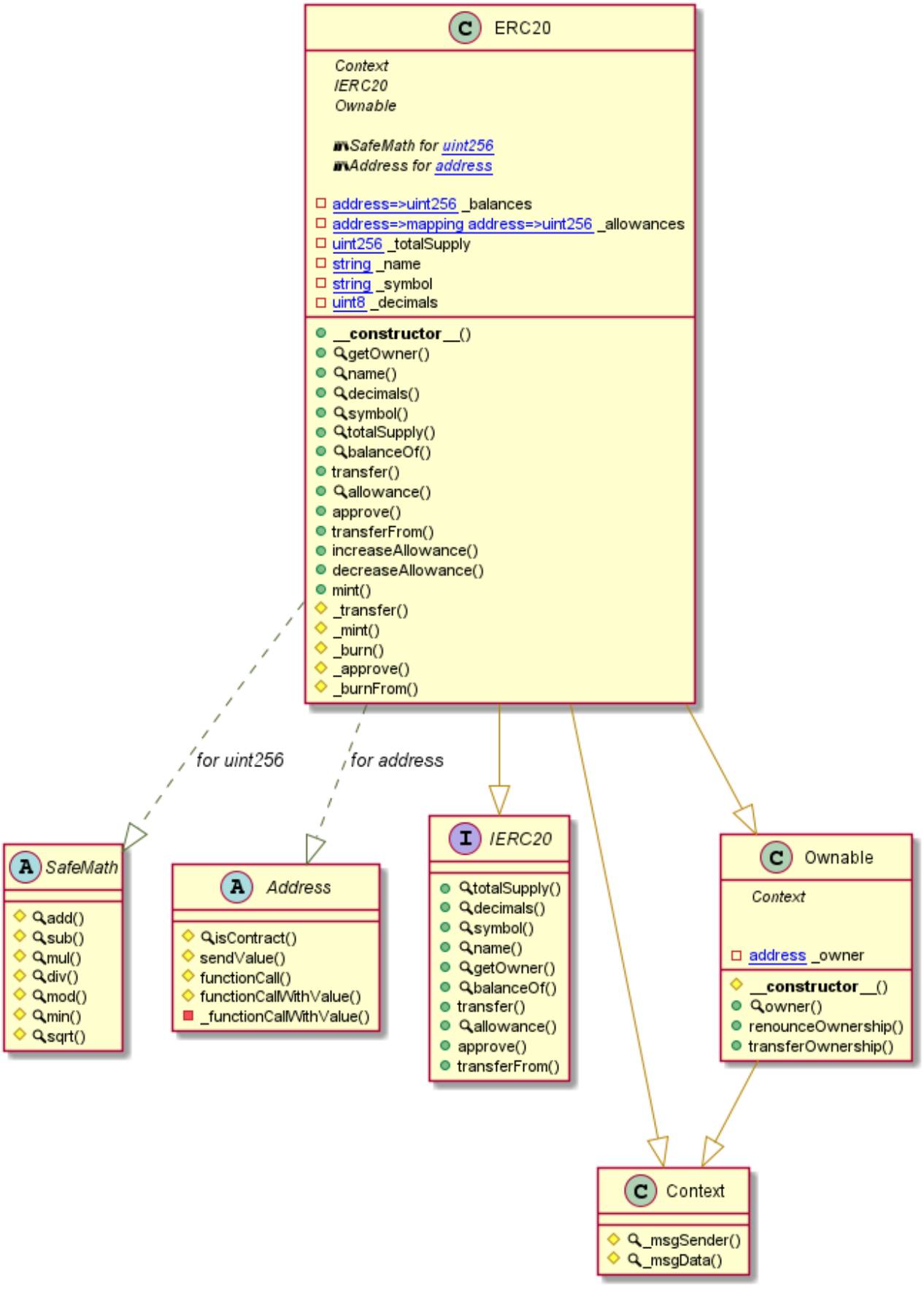
IDO Diagram



LakeOfSugar Diagram



ERC20 Diagram



Slither Results Log

Slither log >> Greeter.sol

```
Pragma version>0.6.6 (Greeter.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter Greeter.setGreeting(string)._greeting (Greeter.sol#15) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
Greeter.sol analyzed (1 contracts with 84 detectors), 2 result(s) found
```

Slither log >> MasterChef.sol

```
MasterChef.updateMultiplier(uint256) (MasterChef.sol#1016-1018) should emit an event for:
        - BONUS_MULTIPLIER = multiplierNumber (MasterChef.sol#1017)
MasterChef.add(uint256,IERC20,bool) (MasterChef.sol#1029-1049) should emit an event for:
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (MasterChef.sol#1039)
MasterChef.set(uint256,uint256,bool) (MasterChef.sol#1051-1063) should emit an event for:
        - totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint) (MasterChef.sol#1059-1061)
MasterChef.setCakePerSecond(uint256) (MasterChef.sol#1248-1254) should emit an event for:
        - cakePerSecond = _cakePerSecond (MasterChef.sol#1253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
```

```
MasterChef.constructor(SugarToken,SyrupBar,address,address,address,uint256,uint256)._devaddr (MasterChef.sol#979) lacks a zero
-check on :
                - devaddr = _devaddr (MasterChef.sol#987)
MasterChef.constructor(SugarToken,SyrupBar,address,address,address,uint256,uint256)._reserveaddr (MasterChef.sol#980) lacks a
zero-check on :
                - reserveaddr = _reserveaddr (MasterChef.sol#988)
MasterChef.constructor(SugarToken,SyrupBar,address,address,address,uint256,uint256)._miningaddr (MasterChef.sol#981) lacks a z
ero-check on :
                - miningaddr = _miningaddr (MasterChef.sol#989)
MasterChef.setDevaddr(address)._addr (MasterChef.sol#1267) lacks a zero-check on :
                - devaddr = _addr (MasterChef.sol#1269)
MasterChef.setReserveaddr(address)._addr (MasterChef.sol#1272) lacks a zero-check on :
                - reserveaddr = _addr (MasterChef.sol#1274)
MasterChef.setMiningaddr(address)._addr (MasterChef.sol#1277) lacks a zero-check on :
                - miningaddr = _addr (MasterChef.sol#1278)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
MasterChef.updatePool(uint256) (MasterChef.sol#1156-1182) has external calls inside a loop: lpSupply = pool.lpToken.balanceOf(
address(this)) (MasterChef.sol#1161)
MasterChef.getMultiplier(uint256,uint256) (MasterChef.sol#1114-1124) has external calls inside a loop: cake.totalSupply() >= s
ugarMaxSupply (MasterChef.sol#1119)
MasterChef.updatePool(uint256) (MasterChef.sol#1156-1182) has external calls inside a loop: cake.mintFor(devaddr,cakeReward.mu
l(10).div(100)) (MasterChef.sol#1173)
MasterChef.updatePool(uint256) (MasterChef.sol#1156-1182) has external calls inside a loop: cake.mintFor(reserveaddr,cakeRewar
d.mul(3).div(100)) (MasterChef.sol#1174)
MasterChef.updatePool(uint256) (MasterChef.sol#1156-1182) has external calls inside a loop: cake.mintFor(miningaddr,cakeReward
.mul(2).div(100)) (MasterChef.sol#1175)
MasterChef.updatePool(uint256) (MasterChef.sol#1156-1182) has external calls inside a loop: cake.mintFor(address(syrup),cakeRe
ward.mul(80).div(100)) (MasterChef.sol#1177)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
MasterChef.nonDuplicatedLP(IERC20) (MasterChef.sol#1025-1028) compares to a boolean constant:
        -require(bool,string)(poolExistence[_lpToken] == false,nonDuplicated: Duplicated LPToken) (MasterChef.sol#1026)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Redundant expression "this (MasterChef.sol#194)" inContext (MasterChef.sol#188-197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

MasterChef.cake (MasterChef.sol#941) should be immutable
MasterChef.startTime (MasterChef.sol#957) should be immutable
MasterChef.syrup (MasterChef.sol#942) should be immutable
SyrupBar.cake (MasterChef.sol#583) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
MasterChef.sol analyzed (13 contracts with 84 detectors), 124 result(s) found
```

Slither log >> NFTController.sol

```
Context._msgData() (NFTController.sol#10-13) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Redundant expression "this (NFTController.sol#11)" inContext (NFTController.sol#5-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
NFTController.sol analyzed (4 contracts with 84 detectors), 2 result(s) found
```

Slither log >> Oracle.sol

```
Oracle.constructor(address).factory_ (Oracle.sol#433) lacks a zero-check on :
                - factory = factory_ (Oracle.sol#434)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

SugarswapOracleLibrary.currentCumulativePrices(address) (Oracle.sol#396-414) uses timestamp for comparisons
        Dangerous comparisons:
        - blockTimestampLast != blockTimestamp (Oracle.sol#405)
Oracle.update(address,address) (Oracle.sol#438-451) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(timeElapsed >= CYCLE,MDEXOracle: PERIOD_NOT_ELAPSED) (Oracle.sol#446)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Pragma version>=0.6.6 (Oracle.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function ISugarswapPair.DOMAIN_SEPARATOR() (Oracle.sol#39) is not in mixedCase
Function ISugarswapPair.PERMIT_TYPEHASH() (Oracle.sol#40) is not in mixedCase
Function ISugarswapPair.MINIMUM_LIQUIDITY() (Oracle.sol#57) is not in mixedCase
Struct FixedPoint.uq112x112 (Oracle.sol#333-335) is not in CapWords
Struct FixedPoint.uq144x112 (Oracle.sol#339-341) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable SugarswapOracleLibrary.currentCumulativePrices(address).price0Cumulative (Oracle.sol#398) is too similar to Sugarswap
OracleLibrary.currentCumulativePrices(address).price1Cumulative (Oracle.sol#398)
Variable Oracle.update(address,address).price0Cumulative (Oracle.sol#447) is too similar to Oracle.update(address,address).pri
ce1Cumulative (Oracle.sol#447)
Variable Oracle.consult(address,uint256,address).price0Cumulative (Oracle.sol#470) is too similar to Oracle.update(address,add
ress).price1Cumulative (Oracle.sol#447)
Variable Oracle.consult(address,uint256,address).price0Cumulative (Oracle.sol#470) is too similar to Oracle.consult(address,ui
nt256,address).price1Cumulative (Oracle.sol#470)
Variable Oracle.update(address,address).price0Cumulative (Oracle.sol#447) is too similar to Oracle.consult(address,uint256,add
ress).price1Cumulative (Oracle.sol#470)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
Oracle.sol analyzed (7 contracts with 84 detectors), 38 result(s) found
```

Slither log >> SmartChef.sol

```
SmartChef.updatePool(uint256) (SmartChef.sol#766-780) has external calls inside a loop: lpSupply = pool.lpToken.balanceOf(addr
ess(this)) (SmartChef.sol#771)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
SmartChef.getMultiplier(uint256,uint256) (SmartChef.sol#741-749) uses timestamp for comparisons
        Dangerous comparisons:
        - _to <= bonusEndTime (SmartChef.sol#742)
        - _from >= bonusEndTime (SmartChef.sol#744)
SmartChef.pendingReward(address) (SmartChef.sol#752-763) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime && lpSupply != 0 (SmartChef.sol#757)
SmartChef.updatePool(uint256) (SmartChef.sol#766-780) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= pool.lastRewardTime (SmartChef.sol#768)
SmartChef.massUpdatePools() (SmartChef.sol#783-788) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (SmartChef.sol#785)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Redundant expression "this (SmartChef.sol#576)" inContext (SmartChef.sol#566-579)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

SmartChef.depositFeePercent (SmartChef.sol#700) should be immutable
SmartChef.rewardPerSecond (SmartChef.sol#687) should be immutable
SmartChef.rewardToken (SmartChef.sol#682) should be immutable
SmartChef.startTime (SmartChef.sol#696) should be immutable
SmartChef.totalAllocPoint (SmartChef.sol#694) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
SmartChef.sol analyzed (7 contracts with 84 detectors), 57 result(s) found
```

Slither log >> SwapMining.sol

```
SwapMining.setRouter(address) (SwapMining.sol#3112-3115) should emit an event for:
        - router = newRouter (SwapMining.sol#3114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

SwapMining.addPair(uint256,address,bool) (SwapMining.sol#3051-3067) should emit an event for:
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (SwapMining.sol#3057)
SwapMining.setPair(uint256,uint256,bool) (SwapMining.sol#3070-3076) should emit an event for:
        - totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint) (SwapMining.sol#3074)
SwapMining.setSugarswapPerSecond(uint256) (SwapMining.sol#3079-3082) should emit an event for:
        - sugarPerSecond = _newPerSecond (SwapMining.sol#3081)
SwapMining.setHalvingPeriod(uint256) (SwapMining.sol#3108-3110) should emit an event for:
        - halvingPeriod = _period (SwapMining.sol#3109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

SwapMining.constructor(SugarToken,ISugarswapFactory,IOracle,address,address,uint256,uint256)._targetToken (SwapMining.sol#3011
) lacks a zero-check on :
                - targetToken = _targetToken (SwapMining.sol#3023)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
SwapMining.massMintPools() (SwapMining.sol#3166-3171) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (SwapMining.sol#3168)
SwapMining.mint(uint256) (SwapMining.sol#3173-3188) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= pool.lastRewardTime (SwapMining.sol#3175)
        - blockReward <= 0 (SwapMining.sol#3179)
SwapMining.swap(address,address,address,uint256) (SwapMining.sol#3196-3233) uses timestamp for comparisons
        Dangerous comparisons:
        - poolLength() <= 0 (SwapMining.sol#3201)
SwapMining.takerWithdraw() (SwapMining.sol#3256-3278) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (SwapMining.sol#3259)
SwapMining.getUserReward(uint256,address) (SwapMining.sol#3281-3293) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_pid <= poolInfo.length - 1,SwapMining: Not find this pool) (SwapMining.sol#3282)
        - user.quantity > 0 (SwapMining.sol#3286)
SwapMining.getTotalUserReward(address) (SwapMining.sol#3296-3315) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (SwapMining.sol#3301)
        - user.quantity > 0 (SwapMining.sol#3305)
SwapMining.getPoolInfo(uint256) (SwapMining.sol#3318-3329) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_pid <= poolInfo.length - 1,SwapMining: Not find this pool) (SwapMining.sol#3319)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Redundant expression "this (SwapMining.sol#435)" inContext (SwapMining.sol#429-438)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements


SwapMining.factory (SwapMining.sol#2996) should be immutable
SwapMining.startTime (SwapMining.sol#2987) should be immutable
SwapMining.sugarToken (SwapMining.sol#2998) should be immutable
SwapMining.targetToken (SwapMining.sol#3000) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
SwapMining.sol analyzed (14 contracts with 84 detectors), 482 result(s) found
```

Slither log >> SyrupBar.sol

```
SugarToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#856-897) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (SyrupBar.sol#895)
SyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#1119-1158) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (SyrupBar.sol#1156)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
SugarToken.getChainId() (SyrupBar.sol#1015-1019) uses assembly
        - INLINE ASM (SyrupBar.sol#1017)
SyrupBar.getChainId() (SyrupBar.sol#1296-1302) uses assembly
        - INLINE ASM (SyrupBar.sol#1298-1300)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version>=0.4.0 (SyrupBar.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (SyrupBar.sol#225-231):
        - (success) = recipient.call{value: amount}() (SyrupBar.sol#229)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#304-330):
        - (success,returndata) = target.call{value: weiValue}(data) (SyrupBar.sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter SugarToken.mintFor(address,uint256)._to (SyrupBar.sol#783) is not in mixedCase
Parameter SugarToken.mintFor(address,uint256)._amount (SyrupBar.sol#783) is not in mixedCase
Variable SugarToken._delegates (SyrupBar.sol#799) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._to (SyrupBar.sol#1025) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._amount (SyrupBar.sol#1025) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._from (SyrupBar.sol#1030) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._amount (SyrupBar.sol#1030) is not in mixedCase
Parameter SyrupBar.safeCakeTransfer(address,uint256)._to (SyrupBar.sol#1043) is not in mixedCase
Parameter SyrupBar.safeCakeTransfer(address,uint256)._amount (SyrupBar.sol#1043) is not in mixedCase
Variable SyrupBar._delegates (SyrupBar.sol#1053) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (SyrupBar.sol#435)" inContext (SyrupBar.sol#429-438)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

SyrupBar.cake (SyrupBar.sol#1036) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
SyrupBar.sol analyzed (8 contracts with 84 detectors), 43 result(s) found
```

Slither log >> SugarswapFactory.sol

```
SugarswapPair.initialize(address,address)._token0 (SugarswapFactory.sol#484) lacks a zero-check on :
                - token0 = _token0 (SugarswapFactory.sol#486)
SugarswapPair.initialize(address,address)._token1 (SugarswapFactory.sol#484) lacks a zero-check on :
                - token1 = _token1 (SugarswapFactory.sol#487)
SugarswapFactory.constructor(address)._feeToSetter (SugarswapFactory.sol#766) lacks a zero-check on :
                - feeToSetter = _feeToSetter (SugarswapFactory.sol#767)
SugarswapFactory.setFeeTo(address)._feeTo (SugarswapFactory.sol#795) lacks a zero-check on :
                - feeTo = _feeTo (SugarswapFactory.sol#797)
SugarswapFactory.setFeeToSetter(address)._feeToSetter (SugarswapFactory.sol#800) lacks a zero-check on :
                - feeToSetter = _feeToSetter (SugarswapFactory.sol#802)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
SugarswapERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (SugarswapFactory.sol#377-389) uses timestamp for
 comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Sugarswap: EXPIRED) (SugarswapFactory.sol#378)
SugarswapPair._update(uint256,uint256,uint112,uint112) (SugarswapFactory.sol#491-504) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (SugarswapFactory.sol#495)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

SugarswapERC20.constructor() (SugarswapFactory.sol#320-334) uses assembly
        - INLINE ASM (SugarswapFactory.sol#322-324)
SugarswapFactory.createPair(address,address) (SugarswapFactory.sol#778-793) uses assembly
        - INLINE ASM (SugarswapFactory.sol#785-787)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Low level call in SugarswapPair._safeTransfer(address,address,uint256) (SugarswapFactory.sol#462-465):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (SugarswapFactory.sol#463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Variable SugarswapPair.swap(uint256,uint256,address,bytes).balance0Adjusted (SugarswapFactory.sol#598) is too similar to Sugar
swapPair.swap(uint256,uint256,address,bytes).balance1Adjusted (SugarswapFactory.sol#599)
Variable SugarswapPair.price0CumulativeLast (SugarswapFactory.sol#444) is too similar to SugarswapPair.price1CumulativeLast (S
ugarswapFactory.sol#445)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

SugarswapFactory.createPair(address,address) (SugarswapFactory.sol#778-793) uses literals with too many digits:
        - bytecode = type()(SugarswapPair).creationCode (SugarswapFactory.sol#783)
SugarswapFactory.slitherConstructorConstantVariables() (SugarswapFactory.sol#755-805) uses literals with too many digits:
        - INIT_CODE_PAIR_HASH = keccak256(bytes)(abi.encodePacked(type()(SugarswapPair).creationCode)) (SugarswapFactory.sol#7
56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

SugarswapERC20.DOMAIN_SEPARATOR (SugarswapFactory.sol#312) should be immutable
SugarswapPair.factory (SugarswapFactory.sol#436) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
SugarswapFactory.sol analyzed (14 contracts with 84 detectors), 55 result(s) found
```

Slither log >> SugarswapPair.sol

```
SugarswapPair.initialize(address,address)._token0 (SugarswapPair.sol#483) lacks a zero-check on :
                - token0 = _token0 (SugarswapPair.sol#485)
SugarswapPair.initialize(address,address)._token1 (SugarswapPair.sol#483) lacks a zero-check on :
                - token1 = _token1 (SugarswapPair.sol#486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Pragma version>=0.6.6 (SugarswapPair.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in SugarswapPair._safeTransfer(address,address,uint256) (SugarswapPair.sol#461-464):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (SugarswapPair.sol#462)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function ISugarswapERC20.DOMAIN_SEPARATOR() (SugarswapPair.sol#231) is not in mixedCase
Function ISugarswapERC20.PERMIT_TYPEHASH() (SugarswapPair.sol#232) is not in mixedCase
Variable SugarswapERC20.DOMAIN_SEPARATOR (SugarswapPair.sol#311) is not in mixedCase
Parameter SugarswapPair.initialize(address,address)._token0 (SugarswapPair.sol#483) is not in mixedCase
Parameter SugarswapPair.initialize(address,address)._token1 (SugarswapPair.sol#483) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (SugarswapPair.sol#245)" inContext (SugarswapPair.sol#239-248)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable SugarswapPair.swap(uint256,uint256,address,bytes).balance0Adjusted (SugarswapPair.sol#597) is too similar to Sugarswa
pPair.swap(uint256,uint256,address,bytes).balance1Adjusted (SugarswapPair.sol#598)
Variable SugarswapPair.price0CumulativeLast (SugarswapPair.sol#443) is too similar to SugarswapPair.price1CumulativeLast (Suga
rswapPair.sol#444)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

SugarswapERC20.DOMAIN_SEPARATOR (SugarswapPair.sol#311) should be immutable
SugarswapPair.factory (SugarswapPair.sol#435) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
SugarswapPair.sol analyzed (11 contracts with 84 detectors), 33 result(s) found
```

Slither log >> SugarswapRouter.sol

```
SugarswapRouter.setSwapMining(address)._swapMininng (SugarswapRouter.sol#593) lacks a zero-check on :
                - swapMining = _swapMininng (SugarswapRouter.sol#594)
SugarswapRouter.constructor(address,address)._factory (SugarswapRouter.sol#597) lacks a zero-check on :
                - factory = _factory (SugarswapRouter.sol#598)
SugarswapRouter.constructor(address,address)._WETH (SugarswapRouter.sol#597) lacks a zero-check on :
                - WETH = _WETH (SugarswapRouter.sol#599)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Pragma version>=0.6.6 (SugarswapRouter.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in TransferHelper.safeApprove(address,address,uint256) (SugarswapRouter.sol#332-336):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (SugarswapRouter.sol#334)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (SugarswapRouter.sol#338-342):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (SugarswapRouter.sol#340)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (SugarswapRouter.sol#344-348):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (SugarswapRouter.sol#346)
Low level call in TransferHelper.safeTransferETH(address,uint256) (SugarswapRouter.sol#350-353):
        - (success) = to.call{value: value}(new bytes(0)) (SugarswapRouter.sol#351)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Function ISugarswapPair.DOMAIN_SEPARATOR() (SugarswapRouter.sol#39) is not in mixedCase
Function ISugarswapPair.PERMIT_TYPEHASH() (SugarswapRouter.sol#40) is not in mixedCase
Function ISugarswapPair.MINIMUM_LIQUIDITY() (SugarswapRouter.sol#57) is not in mixedCase
Function ISugarswapRouter01.WETH() (SugarswapRouter.sol#358) is not in mixedCase
Parameter SugarswapRouter.setSwapMining(address)._swapMininng (SugarswapRouter.sol#593) is not in mixedCase
Variable SugarswapRouter.WETH (SugarswapRouter.sol#585) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (SugarswapRouter.sol#524)" inContext (SugarswapRouter.sol#518-527)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

```
Variable SugarswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (SugarswapRouter.sol#610) is too similar to ISugarswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (SugarswapRouter.sol#364)
Variable SugarswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (SugarswapRouter.sol#638) is too similar to ISugarswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (SugarswapRouter.sol#364)
Variable SugarswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (SugarswapRouter.sol#638) is too similar to SugarswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (SugarswapRouter.sol#611)
Variable SugarswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (SugarswapRouter.sol#628) is too similar to SugarswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (SugarswapRouter.sol#623)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
SugarswapRouter.sol analyzed (13 contracts with 84 detectors), 40 result(s) found
```

Slither log >> SugarToken.sol

```
SugarToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SugarToken.sol#855-896) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (SugarToken.sol#894)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Pragma version>=0.4.0 (SugarToken.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (SugarToken.sol#225-231):
        - (success) = recipient.call{value: amount}() (SugarToken.sol#229)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (SugarToken.sol#304-330):
        - (success,returndata) = target.call{value: weiValue}(data) (SugarToken.sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter SugarToken.mintFor(address,uint256)._to (SugarToken.sol#782) is not in mixedCase
Parameter SugarToken.mintFor(address,uint256)._amount (SugarToken.sol#782) is not in mixedCase
Variable SugarToken._delegates (SugarToken.sol#798) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (SugarToken.sol#435)" inContext (SugarToken.sol#429-438)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
SugarToken.sol analyzed (7 contracts with 84 detectors), 31 result(s) found
```

Slither log >> SugarStakingToken.sol

```
SugarStakingToken.setDelayToWithdraw(uint256) (SugarStakingToken.sol#728-731) should emit an event for:
        - delayToWithdraw = second (SugarStakingToken.sol#730)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

SugarStakingToken.setAdmin(address)._admin (SugarStakingToken.sol#1072) lacks a zero-check on :
                - admin = _admin (SugarStakingToken.sol#1074)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SugarStakingToken.enter(uint256) (SugarStakingToken.sol#734-752):
        External calls:
        - sugar.transferFrom(msg.sender,address(this),_amount) (SugarStakingToken.sol#749)
        State variables written after the call(s):
        - _stakedTime[msg.sender] = block.timestamp (SugarStakingToken.sol#751)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
SugarStakingToken.canWithdraw(address) (SugarStakingToken.sol#718-726) uses timestamp for comparisons
        Dangerous comparisons:
        - _stakedTime[account] == 0 (SugarStakingToken.sol#719)
        - _stakedTime[account] + delayToWithdraw < block.timestamp (SugarStakingToken.sol#722)
SugarStakingToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SugarStakingToken.sol#915-947) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,SUGAR::delegateBySig: signature expired) (SugarStakingToken.sol#945)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
SugarStakingToken.sugar (SugarStakingToken.sol#699) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
SugarStakingToken.sol analyzed (6 contracts with 84 detectors), 55 result(s) found
```

Slither log >> IDO.sol

```
IDO.setOfferingAmount(uint256) (IDO.sol#675-678) should emit an event for:
        - offeringAmount = _offerAmount (IDO.sol#677)
IDO.setRaisingAmount(uint256) (IDO.sol#680-683) should emit an event for:
        - raisingAmount = _raisingAmount (IDO.sol#682)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

IDO.constructor(IBEP20,IBEP20,uint256,uint256,uint256,uint256,uint256,address,bool,bool)._adminAddress (IDO.sol#653) lacks a zero-check on :
                - adminAddress = _adminAddress (IDO.sol#665)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Reentrancy in IDO.deposit(uint256) (IDO.sol#690-708):
        External calls:
        - lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (IDO.sol#695)
        State variables written after the call(s):
        - addressList.push(address(msg.sender)) (IDO.sol#697)
        - totalAmount = totalAmount.add(_amount) (IDO.sol#703)
        - userInfo[msg.sender].amount = userInfo[msg.sender].amount.add(_amount) (IDO.sol#699)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in IDO.deposit(uint256) (IDO.sol#690-708):
        External calls:
        - lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (IDO.sol#695)
        Event emitted after the call(s):
        - Deposit(msg.sender,_amount) (IDO.sol#707)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

IDO.setOfferingAmount(uint256) (IDO.sol#675-678) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < startTime,no) (IDO.sol#676)
IDO.setDepositLimitPerWallet(uint256) (IDO.sol#685-688) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < startTime,no) (IDO.sol#686)
IDO.deposit(uint256) (IDO.sol#690-708) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > startTime && block.timestamp < endTime,not ido time) (IDO.sol#691)
IDO.harvest() (IDO.sol#710-723) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > endTime,not harvest time) (IDO.sol#711)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
IDO.deposit(uint256) (IDO.sol#690-708) compares to a boolean constant:
        -require(bool,string)(isPrivateSale == false || isWhitelisted(msg.sender),not whitelisted) (IDO.sol#693)
IDO.deposit(uint256) (IDO.sol#690-708) compares to a boolean constant:
        -(isFCFS == true) && (totalAmount >= raisingAmount) (IDO.sol#704)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
IDO.adminAddress (IDO.sol#613) should be immutable
IDO.isFCFS (IDO.sol#638) should be immutable
IDO.isPrivateSale (IDO.sol#639) should be immutable
IDO.lpToken (IDO.sol#615) should be immutable
IDO.offeringToken (IDO.sol#617) should be immutable
IDO.startTime (IDO.sol#619) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
IDO.sol analyzed (6 contracts with 84 detectors), 46 result(s) found
```

Slither log >> LakeOfSugar.sol

```
LakeOfSugar.constructor(address,address,address,address)._xsugar (LakeOfSugar.sol#662) lacks a zero-check on :
                - xsugar = _xsugar (LakeOfSugar.sol#667)
LakeOfSugar.constructor(address,address,address,address)._sugar (LakeOfSugar.sol#663) lacks a zero-check on :
                - sugar = _sugar (LakeOfSugar.sol#668)
LakeOfSugar.constructor(address,address,address,address)._weth (LakeOfSugar.sol#664) lacks a zero-check on :
                - weth = _weth (LakeOfSugar.sol#669)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Low level call in Address.sendValue(address,uint256) (LakeOfSugar.sol#130-136):
        - (success) = recipient.call{value: amount}() (LakeOfSugar.sol#134)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (LakeOfSugar.sol#196-217):
        - (success,returndata) = target.call{value: weiValue}(data) (LakeOfSugar.sol#200)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Function IUniswapV2ERC20.DOMAIN_SEPARATOR() (LakeOfSugar.sol#545) is not in mixedCase
Function IUniswapV2ERC20.PERMIT_TYPEHASH() (LakeOfSugar.sol#546) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (LakeOfSugar.sol#567) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (LakeOfSugar.sol#568) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (LakeOfSugar.sol#585) is not in mixedCase
Parameter LakeOfSugar.addAuth(address)._auth (LakeOfSugar.sol#675) is not in mixedCase
Parameter LakeOfSugar.revokeAuth(address)._auth (LakeOfSugar.sol#680) is not in mixedCase
Parameter LakeOfSugar.setDevCut(uint256)._amount (LakeOfSugar.sol#701) is not in mixedCase
Parameter LakeOfSugar.setDevAddr(address)._addr (LakeOfSugar.sol#708) is not in mixedCase
Variable LakeOfSugar._bridges (LakeOfSugar.sol#646) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (LakeOfSugar.sol#460)" inContext (LakeOfSugar.sol#454-463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
LakeOfSugar.sol analyzed (10 contracts with 84 detectors), 46 result(s) found
```

Slither log >> Multicall.sol

```
Multicall.aggregate(Multicall.Call[]) (Multicall.sol#13-21) has external calls inside a loop: (success,ret) = calls[i].target.
call(calls[i].callData) (Multicall.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Pragma version>=0.5.0 (Multicall.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

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
Multicall.sol analyzed (1 contracts with 84 detectors), 3 result(s) found
```

Slither log >> WETH9.sol

```
Reentrancy in WETH9.withdraw(uint256) (WETH9.sol#24-31):
        External calls:
        - (success) = msg.sender.call{value: wad}() (WETH9.sol#28)
        Event emitted after the call(s):
        - Withdrawal(msg.sender,wad) (WETH9.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Low level call in WETH9.withdraw(uint256) (WETH9.sol#24-31):
        - (success) = msg.sender.call{value: wad}() (WETH9.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

WETH9.decimals (WETH9.sol#7) should be constant
WETH9.name (WETH9.sol#5) should be constant
WETH9.symbol (WETH9.sol#6) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
WETH9.sol analyzed (1 contracts with 84 detectors), 5 result(s) found
```

Slither log >> ERC20.sol

```
ERC20.allowance(address,address).owner (ERC20.sol#579) shadows:
        - Ownable.owner() (ERC20.sol#456-458) (function)
ERC20._approve(address,address,uint256).owner (ERC20.sol#751) shadows:
        - Ownable.owner() (ERC20.sol#456-458) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
Pragma version>=0.4.0 (ERC20.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (ERC20.sol#225-231):
        - (success) = recipient.call{value: amount}() (ERC20.sol#229)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (ERC20.sol#304-330):
        - (success,returndata) = target.call{value: weiValue}(data) (ERC20.sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Redundant expression "this (ERC20.sol#435)" inContext (ERC20.sol#429-438)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

ERC20._decimals (ERC20.sol#504) should be immutable
ERC20._name (ERC20.sol#502) should be immutable
ERC20._symbol (ERC20.sol#503) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
ERC20.sol analyzed (6 contracts with 84 detectors), 29 result(s) found
```

Solidity Static Analysis

Greeter.sol

Gas & Economy

Gas costs:

Gas requirement of function Greeter.setGreeting 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: 15:4:

MasterChef.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in MasterChef.depositNFT(address,uint256,uint256,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 203:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in MasterChef.depositNFT(address,uint256,uint256,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 203:4:

Gas & Economy

Gas costs:

Gas requirement of function MasterChef.massUpdatePools is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 290:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 14:4:

Miscellaneous

Constant/View/Pure functions:

MasterChef.getBoost(address,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 128:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 204:8:

NFTController.sol

Miscellaneous

Similar variable names:

NFTController.getBoostRate(address,uint256) : Variables have very similar names "token" and "tokenId". Note: Modifiers are currently not considered by this static analysis.
Pos: 17:30:

Oracle.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: 148:27:

Gas & Economy

Gas costs:

Gas requirement of function Oracle.update 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: 117:4:

Miscellaneous

Similar variable names:

Oracle.update(address,address) : Variables have very similar names "price0Cumulative" and "price1Cumulative".
Pos: 126:9:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 79:8:

SmartChef.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: 423:4:

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: 777:30:

Gas & Economy

Gas costs:

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

Pos: 781:4:

Miscellaneous

Email: audit@EtherAuthority.io

Constant/View/Pure functions:

SmartChef.emergencyRewardWithdraw(uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 845:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 878:8:

SwapMining.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: 310:38:

Gas & Economy

Gas costs:

Gas requirement of function SwapMining.massMintPools 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: 207:4:

Gas costs:

Gas requirement of function SwapMining.getTotalUserReward 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: 337:4:

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: 140:8:

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: 30:4:

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: 149:16:

Gas & Economy

Gas costs:

Gas requirement of function SyrupBar.getPriorVotes is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 148:4:

Miscellaneous

Similar variable names:

SyrupBar.getCurrentVotes(address) : Variables have very similar names "checkpoints" and "nCheckpoints". Note: Modifiers are currently not considered by this static analysis.
Pos: 161:12:

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: 141:8:

Factory.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: 223:39:

Gas & Economy

Gas costs:

Gas requirement of function SugarswapFactory.expectPairFor 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: 27:4:

Miscellaneous

Similar variable names:

SugarswapFactory.createPair(address,address) : Variables have very similar names "token0" and "tokenA". Note: Modifiers are currently not considered by this static analysis.
Pos: 33:9:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 117:8:

Pair.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SugarswapPair._mintFee(uint112,uint112): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 237:4:

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: 223:39:

Gas & Economy

Gas costs:

Gas requirement of function SugarswapPair.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: 258:4:

Miscellaneous

Similar variable names:

SugarswapPair._update(uint256,uint256,uint112,uint112) : Variables have very similar names "reserve1" and "_reserve0". Note: Modifiers are currently not considered by this static analysis.
Pos: 233:28:

Router.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: 203:28:

Gas & Economy

Gas costs:

Gas requirement of function SugarswapRouter.swapExactTokensForTokens 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: 415:4:

Miscellaneous

Similar variable names:

SugarswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256
: Variables have very similar names "reserveA" and "reserveB". Note: Modifiers are currently not considered by this static analysis.
Pos: 233:24:

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: 203:8:

SugarToken.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: 123:16:

Gas & Economy

Gas costs:

Gas requirement of function SugarToken.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: 148:4:

Miscellaneous

Constant/View/Pure functions:

SugarToken.getChainId() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 243:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 272:8:

SugarStakingToken.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: 751:34:

Gas & Economy

Gas costs:

Gas requirement of function SugarStakingToken.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

Similar variable names:

SugarStakingToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes3
: Variables have very similar names "_delegates" and "delegatee".
Pos: 946:36:

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: 729:8:

IDO.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: 705:18:

Gas & Economy

Gas costs:

Gas requirement of function IDO.includeToWhiteList 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: 764:2:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 765:4:

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: 772:4:

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in LakeOfSugar._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 LakeOfSugar.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:

Miscellaneous

Similar variable names:

LakeOfSugar.(address,address,address,address) : Variables have very similar names "xsugar" and "sugar". Note: Modifiers are currently not considered by this static analysis.

Pos: 668:8:

Guard conditions:

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

more

Pos: 369:8:

Security

Block hash:

Use of "blockhash": "blockhash(uint blockNumber)" is used to access the last 256 block hashes. A miner computes the block hash by "summing up" the information in the current block mined. By "summing up" the information cleverly, a miner can try to influence the outcome of a transaction in the current block. This is especially easy if there are only a small number of equally likely outcomes.
Pos: 30:20:

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:

WETH9.sol

Security

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: 42:27:

Gas & Economy

Gas costs:

Gas requirement of function WETH9.withdraw 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: 38:4:

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: 68:12:

ERC20.sol

Security

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: 143:50:

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: 63:4:

Miscellaneous

Similar variable names:

ERC20.(string,string) : Variables have very similar names "_name" and "name_". Note: Modifiers are currently not considered by this static analysis.
Pos: 58:16:

Solhint Linter

Greeter.sol

```
Greeter.sol:2:1: Error: Compiler version >0.6.6 does not satisfy the
r semver requirement
```

MasterChef.sol

```
MasterChef.sol:3:1: Error: Compiler version 0.6.12 does not satisfy
the r semver requirement
MasterChef.sol:12:8: Error: Use double quotes for string literals
MasterChef.sol:75:29: Error: Constant name must be in capitalized
SNAKE_CASE
MasterChef.sol:78:20: Error: Variable name must be in mixedCase
MasterChef.sol:90:29: Error: Constant name must be in capitalized
SNAKE_CASE: Error: Avoid to make time-based decisions in your
business logic
MasterChef.sol:305:35: Error: Avoid to make time-based decisions in
your business logic
MasterChef.sol:308:65: Error: Avoid to make time-based decisions in
your business logic
MasterChef.sol:331:31: Error: Avoid to make time-based decisions in
your business logic
```

NFTController.sol

```
NFTController.sol:3:1: Error: Compiler version 0.6.12 does not
satisfy the r semver requirement
NFTController.sol:13:26: Error: Code contains empty blocks
```

Oracle.sol

```
Oracle.sol:3:1: Error: Compiler version >=0.6.6 does not satisfy the
r semver requirement
Oracle.sol:12:5: Error: Contract name must be in CamelCase
Oracle.sol:18:5: Error: Contract name must be in CamelCase
Oracle.sol:36:25: Error: Use double quotes for string literals
Oracle.sol:71:23: Error: Avoid to make time-based decisions in your
business logic
Oracle.sol:125:39: Error: Use double quotes for string literals
Oracle.sol:127:33: Error: Avoid to make time-based decisions in your
```

```
business logic
Oracle.sol:148:28: Error: Avoid to make time-based decisions in your
business logic
```

SmartChef.sol

```
SmartChef.sol:1:1: Error: Compiler version >=0.6.0 does not satisfy
the r semver requirement
SmartChef.sol:146:26: Error: Use double quotes for string literals
SmartChef.sol:192:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
SmartChef.sol:292:1: Error: Compiler version ^0.6.2 does not satisfy
the r semver requirement
SmartChef.sol:419:49: Error: Use double quotes for string literals
SmartChef.sol:429:37: Error: Use double quotes for string literals
SmartChef.sol:454:1: Error: Compiler version ^0.6.0 does not satisfy
the r semver requirement
SmartChef.sol:545:53: Error: Use double quotes for string literals
SmartChef.sol:552:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
SmartChef.sol:567:28: Error: Code contains empty blocks
SmartChef.sol:581:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
SmartChef.sol:621:41: Error: Use double quotes for string literals
SmartChef.sol:649:41: Error: Use double quotes for string literals
SmartChef.sol:657:1: Error: Compiler version 0.6.12 does not satisfy
the r semver requirement
SmartChef.sol:774:65: Error: Avoid to make time-based decisions in
your business logic
SmartChef.sol:777:31: Error: Avoid to make time-based decisions in
your business logic
SmartChef.sol:846:65: Error: Use double quotes for string literals
SmartChef.sol:852:36: Error: Use double quotes for string literals
SmartChef.sol:877:28: Error: Avoid using low level calls.
```

SwapMining.sol

```
SwapMining.sol:3:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
SwapMining.sol:12:8: Error: Use double quotes for string literals
SwapMining.sol:97:34: Error: Avoid to make time-based decisions in
your business logic
SwapMining.sol:227:31: Error: Avoid to make time-based decisions in
your business logic
SwapMining.sol:272:31: Error: Avoid to make time-based decisions in
your business logic
SwapMining.sol:310:39: Error: Avoid to make time-based decisions in
your business logic
```

SyrupBar.sol

```
SyrupBar.sol:3:1: Error: Compiler version >=0.6.12 does not satisfy
the r semver requirement
SyrupBar.sol:149:17: Error: Avoid to make time-based decisions in
your business logic
SyrupBar.sol:291:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

Factory.sol

```
Factory.sol:3:1: Error: Compiler version >=0.5.16 does not satisfy
the r semver requirement
Factory.sol:34:39: Error: Use double quotes for string literals
Factory.sol:35:56: Error: Use double quotes for string literals
Factory.sol:38:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Factory.sol:49:44: Error: Use double quotes for string literals
Factory.sol:54:44: Error: Use double quotes for string literals
```

Pair.sol

```
Pair.sol:3:1: Error: Compiler version >=0.6.6 does not satisfy the r
semver requirement
Pair.sol:37:36: Error: Constant name must be in capitalized
SNAKE_CASE
Pair.sol:42:29: Error: Variable name must be in mixedCase
Pair.sol:52:9: Error: Avoid using inline assembly. It is acceptable
only in rare cases
Pair.sol:59:33: Error: Use double quotes for string literals
Pair.sol:108:29: Error: Avoid to make time-based decisions in your
business logic
Pair.sol:317:49: Error: Use double quotes for string literals
Pair.sol:326:49: Error: Use double quotes for string literals
Pair.sol:330:104: Error: Use double quotes for string literals
```

Router.sol

```
Router.sol:3:1: Error: Compiler version >=0.6.6 does not satisfy the
r semver requirement
Router.sol:5:8: Error: Use double quotes for string literals
Router.sol:199:39: Error: Variable name must be in mixedCase
Router.sol:203:29: Error: Avoid to make time-based decisions in your
business logic
Router.sol:203:46: Error: Use double quotes for string literals
Router.sol:211:35: Error: Variable name must be in mixedCase
Router.sol:500:34: Error: Use double quotes for string literals
```

```
Router.sol:502:42: Error: Use double quotes for string literals
Router.sol:547:13: Error: Use double quotes for string literals
Router.sol:562:34: Error: Use double quotes for string literals
Router.sol:570:13: Error: Use double quotes for string literals
Router.sol:585:48: Error: Use double quotes for string literals
Router.sol:591:44: Error: Use double quotes for string literals
```

SugarToken.sol

```
SugarToken.sol:3:1: Error: Compiler version >0.6.6 does not satisfy
the r semver requirement
SugarToken.sol:8:30: Error: Use double quotes for string literals
SugarToken.sol:8:49: Error: Use double quotes for string literals
SugarToken.sol:123:17: Error: Avoid to make time-based decisions in
your business logic
SugarToken.sol:245:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

XSUGER.sol

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

IDO.sol

```
IDO.sol:3:1: Error: Compiler version 0.6.12 does not satisfy the r
semver requirement
IDO.sol:686:43: Error: Use double quotes for string literals
IDO.sol:691:14: Error: Avoid to make time-based decisions in your
business logic
IDO.sol:691:45: Error: Avoid to make time-based decisions in your
business logic
IDO.sol:701:70: Error: Use double quotes for string literals
IDO.sol:705:19: Error: Avoid to make time-based decisions in your
business logic
IDO.sol:711:14: Error: Avoid to make time-based decisions in your
business logic
IDO.sol:771:60: Error: Use double quotes for string literals
```

```
IDO.sol:772:69: Error: Use double quotes for string literals
```

LakeOfSugar.sol

```
LakeOfSugar.sol:4:1: Error: Compiler version 0.6.12 does not satisfy
the r semver requirement
LakeOfSugar.sol:568:5: Error: Function name must be in mixedCase
LakeOfSugar.sol:585:5: Error: Function name must be in mixedCase
LakeOfSugar.sol:726:31: Error: Avoid to use tx.origin
LakeOfSugar.sol:916:31: Error: Use double quotes for string literals
LakeOfSugar.sol:917:50: 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
```

WETH9.sol

```
WETH9.sol:16:1: Error: Compiler version =0.6.12 does not satisfy the
r semver requirement
WETH9.sol:42:28: Error: Avoid using low level calls.
```

ERC20.sol

```
ERC20.sol:3:1: Error: Compiler version >=0.4.0 does not satisfy the r
semver requirement
ERC20.sol:297:38: Error: Use double quotes for string literals
ERC20.sol:298:40: Error: Use double quotes for string literals
ERC20.sol:315:60: Error: Use double quotes for string literals
```

Software analysis result:

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

Email: audit@EtherAuthority.io