Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: TheIronMan Finance
Website: theironman.finance
Platform: Core Chain
Language: Solidity
Date: March 13th, 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 TheIronMan Finance to perform the Security audit of the TheIronMan Finance Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on March 13th, 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

- TheIronMan Finance is a project that aims to create a synthetic protocol and support a synthetic asset market.
- TheIronMan Finance uses a Collateral Ratio (CR) for minting and redeeming TheIronMan's synths. There will be a minimum CR set by governance for respective synths. The CR is used by the minting and redeeming functions. It is displayed as a percentage. The ratio shows what percentage of IMT is needed in order to mint or redeem a CORE token.
- In the case of COREX trading above the value of CORE, a user can mint the synth token through the protocol, using the value of 1 CORE token for 1 synth token, and then sell the synth token on the open market. Over a short time period the profitable arb opportunity will subside—the prices will converge.
- TheIronMan Finance contract inherits the ERC20, SafeERC20, Context, Ownable, ReentrancyGuard, Address, IERC20, Math, SafeMath, Initializable standard smart contracts from the OpenZeppelin library.
- TheIronMan Finance contract inherits the IUniswapV2Pair, IUniswapV2Router02 standard smart contracts from the uniswap library.
- These OpenZeppelin and uniswap contracts are considered community-audited and time-tested, and hence are not part of the audit scope.

Audit scope

Name	Code Review and Security Analysis Report for TheIronMan Finance Protocol Smart Contracts
Platform	**Core Chain / Solidity**
File 1	Pool.sol
File 1 MD5 Hash	5A2A00BB08B8E6D864762B7923234D83
File 2	SwapStrategyPOL.sol
File 2 MD5 Hash	3AE7E63D48701C411ABB283789C1437F
File 3	StratReduceReserveLP.sol
File 3 MD5 Hash	16E6A30B5CAEDE87A5F4A5BFF827D22F
File 4	DaoChef.sol
File 4 MD5 Hash	E12C4E0BDCB405DD0DB61CCF7173ED06
File 5	DaoStaking.sol
File 5 MD5 Hash	420E6CBD2A617CF1A889C9FB3748A035
File 6	DaoZapMMSwap.sol
File 6 MD5 Hash	8C94DD4015FAD718D7A77F614090C88E
File 7	NFTController.sol
File 7 MD5 Hash	7B517FFAE5E28C8D3B7020747FFA8659
File 8	DevFund.sol
File 8 MD5 Hash	E8F0D50D10EA81F373B23544A020892B
File 9	EcosystemFund.sol
File 9 MD5 Hash	252B8F6FCBABD3E4A205363E4245D000
File 10	Fund.sol
File 10 MD5 Hash	47370A0301A3BBA40747C7FFD8A18E6B
File 11	Reserve.sol
File 11 MD5 Hash	FEF578E09DAA9039E6B42E44190E2245
File 12	MasterOracle.sol

File 12 MD5 Hash	26FFB8A6EB84AABF384A830DB4572C0A
File 13	UniswapPairOracle.sol
File 13 MD5 Hash	37801A23DE6F4571ADD278A4A062C1D5
File 14	XToken.sol
File 14 MD5 Hash	83382FC411F2E4462B30C55D6F62A2DD
File 15	YToken.sol
File 15 MD5 Hash	FFA9BDAB9AEE9D07DB46CB3A23A34696
File 16	IMT.sol
File 16 MD5 Hash	1B73A648455049564C87A16513A14E13
File 17	COREX.sol
File 17 MD5 Hash	1A564457EF100782F814814C3BAEA787
File 18	DaoTreasury.sol
File 18 MD5 Hash	0B9AE0B9E415F68B36F7B1047E21B02C
File 19	StratRecollateralize.sol
File 19 MD5 Hash	C02B3F40E26D074FB153BAC73AD35F92
Audit Date	March 13th,2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 Pool.sol • Refresh Cooldown: 1 hour • Ratio StepUp: 0.2% • Ratio StepDown: 0.2% • Price Target: 1 • Price Band: 0.004 (CR will be adjusted if XToken > 1.004 MUSD or if XToken < 0.996 MUSD) • Min Collateral Ratio: 95% • yTokenSlippage: 20% • Redemption Fee: 10% • Redemption Fee Maximum: 0.9% • Minting Fee: 0.5% • Minting Fee Maximum:0.5%	**YES, This is valid. Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**

File 2 SwapStrategyPOL.sol
• Swap Slippage: 20%
• Swapper: Protocol Owned Liquidity can execute functionality like:
  ○ Add liquidity for YToken/WETH LP.
  ○ Transfer LP to Treasury.
  ○ Swap 50% of WETH to YToken.
	YES, This is valid. Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.

File 3 DaoChef.sol • Maximum Reward: 10 token per second. • A new LP can be added to the pool by the owner. • A pool's reward allocation point and IRewarder contract can be updated by the owner.	**YES, This is valid.**
File 4 DaoStaking.sol • DaoStaking is based on EPS's & Geist's multi fee distribution.	**YES, This is valid.**

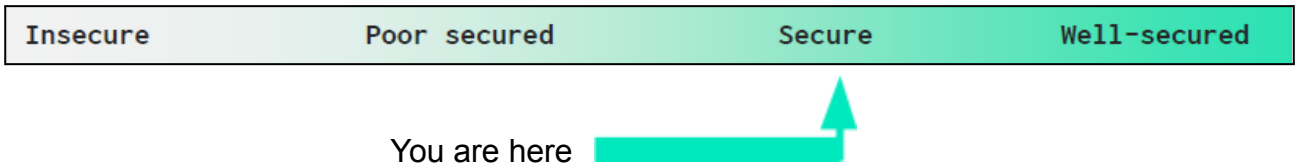
• Grouped Duration: 1 day • Rewards Duration: 1 week • Lock Duration: 4 weeks • Team Rewards: 20%	
File 5 DaoZapMMSwap.sol • DaoZap is a ZapperFi's simplified version of zapper contract which will: 1. use ETH to swap to target token 2. make LP between ETH and target token 3. add into DaoChef farm	**YES, This is valid.**
File 6 NFTController.sol • Default Boost Rate: 1%	**YES, This is valid.**
File 7 Fund.sol • Owner can transfer amounts.	**YES, This is valid.**
File 8 DevFund.sol • Allocation: 5% • Vesting Duration: 2 Years	**YES, This is valid.**
File 9 Reserve.sol • Owner can set the pool address. • Owner can remove the pool address.	**YES, This is valid.**
File 10 EcosystemFund.sol • Allocation: 15% • Vesting Duration: 2 Years	**YES, This is valid.**
File 11 MasterOracle.sol • MasterOracle has functions like: getXTokenPrice, getYTokenPrice, getYTokenTWAP, etc.	**YES, This is valid.**
File 12 UniswapPairOracle.sol • Period: 60-minute TWAP (Time-Weighted Average	**YES, This is valid.**

Price) • Maximum Period: 48 Hours • Minimum Period: 10 Minutes • Leniency: 12 Hours	
File 13 XToken.sol • Owner can set the minter address for XToken(only once). • Owner can remove the minter address from XToken. • Owner can mint a new XToken.	**YES, This is valid.**
File 14 YToken.sol • The YToken contract inherits the ERC20Burnable standard smart contracts from the OpenZeppelin library.	**YES, This is valid.**
File 15 IMT.sol • Decimals: 18 • Total Supply: 5 Million • Owner can set openTrading's true status.	**YES, This is valid.**
File 16 COREX.sol • 100 will be minted at genesis for liquid pool seeding.	**YES, This is valid.**
File 17 DaoTreasury.sol • DaoTreasury is to store the reserve of TheIronMan Protocol. • These contracts will have a whitelist of strategy contracts which can request funding from Reserve. • These strategy contracts can be used to Allocate fee, Convert reserve to Protocol Owned Liquidity, Recollateralize, etc.	**YES, This is valid.**
File 18 StratRecollateralize.sol	**YES, This is valid.**

• Owner can recollateralize the minting pool.	
File 19 StratReduceReserveLP.sol • Owner can remove liquidity, buy back YToken and burn.	**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 2 low and some very low level issues.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

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

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 19 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 TheIronMan Finance Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the TheIronMan Finance Protocol.

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

Code parts are well commented on smart contracts.

Documentation

We were given a TheIronMan Finance 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 **well** commented. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

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

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

Pool.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	_checkOwner	internal	Passed	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	nonReentrant	modifier	Passed	No Issue
9	_nonReentrantBefore	write	Passed	No Issue
10	_nonReentrantAfter	write	Passed	No Issue
11	_reentrancyGuardEntered	internal	Passed	No Issue
12	info	external	Passed	No Issue
13	usableCollateralBalance	read	Passed	No Issue
14	calcMint	read	Passed	No Issue
15	calcRedeem	read	Passed	No Issue
16	calcExcessCollateralBalance	read	Passed	No Issue
17	refreshCollateralRatio	read	Passed	No Issue
18	mint	external	Passed	No Issue
19	redeem	external	Passed	No Issue
20	collect	external	Passed	No Issue
21	recollateralize	external	Passed	No Issue
22	checkPriceFluctuation	internal	Passed	No Issue
23	toggle	write	access only Owner	No Issue
24	setCollateralRatioOptions	write	access only Owner	No Issue
25	toggleCollateralRatio	write	access only Owner	No Issue
26	setFees	write	access only Owner	No Issue
27	setMinCollateralRatio	external	access only Owner	No Issue
28	reduceExcessCollateral	external	access only Owner	No Issue
29	setSwapStrategy	external	access only Owner	No Issue
30	setOracle	external	access only Owner	No Issue
31	setYTokenSlippage	external	access only Owner	No Issue
32	setTreasury	external	access only Owner	No Issue
33	transferToTreasury	internal	Passed	No Issue

SwapStrategyPOL.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	_checkOwner	internal	Passed	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	lpBalance	read	Passed	No Issue
9	execute	external	Passed	No Issue
10	swap	internal	Passed	No Issue
11	addLiquidity	internal	Passed	No Issue
12	cleanDust	external	access only Owner	No Issue
13	changeSlippage	external	access only Owner	No Issue
14	calculateSwapInAmount	internal	Passed	No Issue

DaoChef.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	_checkOwner	internal	Passed	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	poolLength	read	Passed	No Issue
9	pendingReward	external	Passed	No Issue
10	updatePool	write	Passed	No Issue
11	massUpdatePools	write	Infinite loop	Refer Audit Findings
12	deposit	write	Passed	No Issue
13	withdraw	write	Passed	No Issue
14	harvest	write	Passed	No Issue
15	withdrawAndHarvest	write	Passed	No Issue
16	emergencyWithdraw	write	Passed	No Issue
17	harvestAllRewards	external	Infinite loop	Refer Audit Findings
18	checkPoolDuplicate	internal	Infinite loop	Refer Audit Findings
19	add	write	access only Owner	No Issue
20	set	write	access only Owner	No Issue

21	setRewardPerSecond	write	access only Owner	No Issue
22	setRewardMinter	external	Passed	No Issue
23	getBoost	read	Passed	No Issue
24	getSlots	read	Passed	No Issue
25	getTokenIds	read	Passed	No Issue
26	depositNFT	write	Passed	No Issue
27	withdrawNFT	write	Passed	No Issue
28	setNftController	write	Function input parameters lack of check	Refer Audit Findings
29	setNftBoostRate	write	access only Owner	No Issue

DaoStaking.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	_checkOwner	internal	Passed	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	nonReentrant	modifier	Passed	No Issue
9	_nonReentrantBefore	write	Passed	No Issue
10	_nonReentrantAfter	write	Passed	No Issue
11	_reentrancyGuardEntered	internal	Passed	No Issue
12	addReward	write	Function input parameters lack of check	Refer Audit Findings
13	approveRewardDistributor	external	Function input parameters lack of check	Refer Audit Findings
14	_rewardPerToken	internal	Passed	No Issue
15	_earned	internal	Passed	No Issue
16	lastTimeRewardApplicable	read	Passed	No Issue
17	rewardPerToken	external	Passed	No Issue
18	getRewardForDuration	external	Passed	No Issue
19	claimableRewards	external	Passed	No Issue
20	totalBalance	external	Passed	No Issue
21	unlockedBalance	external	Passed	No Issue
22	earnedBalances	external	Passed	No Issue
23	lockedBalances	external	Passed	No Issue
24	withdrawableBalance	read	Passed	No Issue
25	stake	external	Passed	No Issue

26	mint	external	Function input parameters lack of check	Refer Audit Findings
27	withdraw	write	Passed	No Issue
28	getReward	write	Passed	No Issue
29	emergencyWithdraw	external	Critical operation lacks event log	Refer Audit Findings
30	withdrawExpiredLocks	external	Passed	No Issue
31	_notifyReward	internal	Passed	No Issue
32	notifyRewardAmount	external	Function input parameters lack of check	Refer Audit Findings
33	recoverERC20	external	access only Owner	No Issue
34	updateReward	modifier	Passed	No Issue
35	receive	external	Passed	No Issue
36	setTeamWalletAddress	external	Passed	No Issue
37	setTeamRewardPercent	external	Passed	No Issue

DaoZapMMSwap.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	_checkOwner	internal	Passed	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	nonReentrant	modifier	Passed	No Issue
9	_nonReentrantBefore	write	Passed	No Issue
10	_nonReentrantAfter	write	Passed	No Issue
11	_reentrancyGuardEntered	internal	Passed	No Issue
12	zap	external	Passed	No Issue
13	receive	external	Passed	No Issue
14	swap	internal	access only Owner	No Issue
15	doSwapETH	internal	Passed	No Issue
16	approveToken	internal	Passed	No Issue
17	calculateSwapInAmount	internal	Passed	No Issue
18	addZap	external	access only Owner	No Issue
19	removeZap	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	_checkOwner	internal	Passed	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	initialize	write	initializer	No Issue
9	initializer	modifier	Passed	No Issue
10	reinitializer	modifier	Passed	No Issue
11	onlyInitializing	modifier	Passed	No Issue
12	_disableInitializers	internal	Passed	No Issue
13	_getInitializedVersion	internal	Passed	No Issue
14	_isInitializing	internal	Passed	No Issue
15	getBoostRate	external	Passed	No Issue
16	setWhitelist	external	access only Owner	No Issue
17	setDefaultBoostRate	external	access only Owner	No Issue
18	setBoostRate	external	access only Owner	No Issue

Fund.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	_checkOwner	internal	Passed	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	nonReentrant	modifier	Passed	No Issue
9	_nonReentrantBefore	write	Passed	No Issue
10	_nonReentrantAfter	write	Passed	No Issue
11	_reentrancyGuardEntered	internal	Passed	No Issue
12	initialize	external	initializer	No Issue
13	allocation	read	Passed	No Issue
14	vestingStart	read	Passed	No Issue
15	vestingDuration	read	Passed	No Issue
16	currentBalance	read	Passed	No Issue
17	vestedBalance	read	Passed	No Issue
18	claimable	read	Passed	No Issue
19	transfer	external	access only Owner	No Issue

DevFund.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	external	Passed	No Issue
3	allocation	read	Passed	No Issue
4	vestingStart	read	Passed	No Issue
5	vestingDuration	read	Passed	No Issue
6	currentBalance	read	Passed	No Issue
7	vestedBalance	read	Passed	No Issue
8	claimable	read	Passed	No Issue
9	transfer	external	access only Owner	No Issue
10	allocation	write	Passed	No Issue
11	vestingStart	write	Passed	No Issue
12	vestingDuration	write	Passed	No Issue

Reserve.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	setRewarder	external	Passed	No Issue
4	setPool	external	access only Owner	No Issue
5	removePool	external	access only Owner	No Issue
6	transfer	external	Passed	No Issue
7	nonReentrant	modifier	Passed	No Issue
8	_nonReentrantBefore	write	Passed	No Issue
9	_nonReentrantAfter	write	Passed	No Issue
10	_reentrancyGuardEntered	internal	Passed	No Issue
11	owner	read	Passed	No Issue
12	onlyOwner	modifier	Passed	No Issue
13	renounceOwnership	write	access only Owner	No Issue
14	_checkOwner	internal	Passed	No Issue
15	transferOwnership	write	access only Owner	No Issue
16	_transferOwnership	internal	Passed	No Issue

EcosystemFund.sol

Functions

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

4	vestingStart	read	Passed	No Issue
5	vestingDuration	read	Passed	No Issue
6	currentBalance	read	Passed	No Issue
7	vestedBalance	read	Passed	No Issue
8	claimable	read	Passed	No Issue
9	transfer	external	access only Owner	No Issue
10	allocation	write	Passed	No Issue
11	vestingStart	write	Passed	No Issue
12	vestingDuration	write	Passed	No Issue

MasterOracle.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	_checkOwner	internal	Passed	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	getXTokenPrice	read	Passed	No Issue
9	getYTokenPrice	read	Passed	No Issue
10	getXTokenTWAP	read	Passed	No Issue
11	getYTokenTWAP	read	Passed	No Issue

UniswapPairOracle.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	setPeriod	external	access only Owner	No Issue
3	update	external	Passed	No Issue
4	twap	external	Passed	No Issue
5	spot	external	Passed	No Issue
6	currentBlockTimestamp	internal	Passed	No Issue
7	currentCumulativePrices	internal	Passed	No Issue

XToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyMinter	modifier	Passed	No Issue
3	setMinter	external	access only Owner	No Issue

4	removeMinter	external	access only Owner	No Issue
5	mint	external	access only Minter	No Issue
6	burn	write	Passed	No Issue
7	burnFrom	write	Passed	No Issue

YToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	burn	write	Passed	No Issue
3	burnFrom	write	Passed	No Issue

IMT.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	OpenTrade	external	Passed	No Issue
3	includeToWhitelist	write	Passed	No Issue
4	excludeFromWhitlist	write	Passed	No Issue
5	_transfer	write	Passed	No Issue

COREX.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	OpenTrade	external	Passed	No Issue
3	includeToWhitelist	write	Passed	No Issue
4	excludeFromWhitlist	write	Passed	No Issue
5	_transfer	internal	Passed	No Issue

StratRecollateralize.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	recollateralize	external	access only Owner	No Issue
3	receive	external	Passed	No Issue
4	owner	read	Passed	No Issue
5	onlyOwner	modifier	Passed	No Issue
6	renounceOwnership	write	access only Owner	No Issue
7	_checkOwner	internal	Passed	No Issue

8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue

StratReduceReserveLP.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	_checkOwner	internal	Passed	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	reduceReserve	external	access only Owner	No Issue
9	swap	internal	Passed	No Issue

DaoTreasury.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	_checkOwner	internal	Passed	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	balanceOf	read	Passed	No Issue
9	requestFund	external	Passed	No Issue
10	addStrategy	external	access only Owner	No Issue
11	removeStrategy	external	access only Owner	No Issue
12	allocateFee	external	access only Owner	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) Function input parameters lack of check:
Variable validation is not performed in below functions:

DaoChef.sol

- setNftController = _controller

DaoStaking.sol

- addReward =_rewardsToken
- approveRewardDistributor = _rewardsToken ,_distributor
- mint = user
- notifyRewardAmount = _rewardsToken

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

(2) Critical operation lacks event log: **DaoStaking.sol**
Missing event log for:

- emergencyWithdraw

Resolution: Write an event log for listed events.

Very Low / Informational / Best practices:

(1) Infinite loop: **DaoChef.sol**

In harvestAllRewards,checkPoolDuplicate ,massUpdatePools functions for loop does not have an upper length limit , which costs more gas.

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

(2) Division before multiplication: **DaoStaking.sol**

Solidity being resource constraint language, dividing any amount and then multiplying will cause discrepancy in the outcome. Therefore always multiply the amount first and then divide it.

Resolution: Consider ordering multiplication before division.

(3) Owner can drain all ERC20 tokens: **DaoStaking.sol**

The function recoverERC20() will allow the owner to withdraw all the ERC20 tokens. This would create trust issues in the users.

Resolution: If these are desired features, then please ignore this point.

(4) Hardcoded Wrapped Token: **WethUtils.sol**

```
library WethUtils {
    using SafeERC20 for IWETH;
    using SafeERC20 for IERC20;

    IWETH public constant weth = IWETH(0xc86c7C0eFbd6A49B35E8714C5f59D99De09A225b); // WCORE
```

In WethUtils.sol file, weth public constant variable has a hardcoded token address which is of KAVA network.

Resolution: We suggest changing wrapped token addresses before deploying contracts.

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:

Pool.sol

- toggle: Owner can turn on / off minting and redemption.
- setCollateralRatioOptions: Owner can configure variables related to Collateral Ratio.
- toggleCollateralRatio: Owner can pause or unpause collateral ratio updates.
- setFees: Owner can set the protocol fees.
- setMinCollateralRatio: Owner can set the minimum Collateral Ratio.
- reduceExcessCollateral: Owner can transfer the excess balance of WETH to FeeReserve.
- setSwapStrategy: Owner can set the address of Swapper utils.
- setOracle: Owner can set new oracle address.
- setYTokenSlippage: Owner can set yTokenSlipage.

SwapStrategyPOL.sol

- cleanDust: Owner can clean dust.
- changeSlippage: Owner can change slippage value.

DaoChef.sol

- add: Owner can add a new LP to the pool.
- set: Owner can update the given pool's reward allocation point and `IRewarder` contract
- setRewardPerSecond: Owner can set the reward per second to be distributed.
- setRewardMinter: Owner can set the address of rewardMinter.
- depositNFT: Owner can check if User does not have the specified NFT.
- setNftController: Owner can set Nft Controller address.
- setNftBoostRate: Owner can set Nft Boost Rate.

DaoStaking.sol

- addReward: Owner can add a new reward token to be distributed to stakers.
- approveRewardDistributor: Owner can modify approval for an address to call notifyRewardAmount.
- recoverERC20: Owner can be added to support recovering LP Rewards from other systems such as BAL to be distributed to holders.
- setTeamWalletAddress: Owner can set team wallet address.
- setTeamRewardPercent: Owner can set team reward percentage.

DaoZapMMSwap.sol

- addZap: Owner can add new zap configuration.
- removeZap: Owner can Deactivate a Zap configuration.

NFTController.sol

- setWhitelist: Owner can set whitelist addresses.
- setDefaultBoostRate: Owner can set default BoostRate value 1%.
- setBoostRate: Owner can set BoostRate value 1%.

Fund.sol

- transfer: Owner can transfer amounts.

Reserve.sol

- setPool: Owner can set pool address.
- removePool: Owner can remove pool address.

UniswapPairOracle.sol

- setPeriod: Owner can set the period.

XToken.sol

- setMinter: Owner can set minter address for XToken.
- removeMinter: Owner can remove minter address from XToken.

IMT.sol

- OpenTrade: Owner can set openTrading true status.

- includeToWhitelist: Owner can include address to Whitelist.
- excludeFromWhitlist: Owner can exclude address from Whitelist.

COREX.sol

- OpenTrade: Owner can set openTrading true status.
- includeToWhitelist: Owner can include address to Whitelist.
- excludeFromWhitlist: Owner can exclude address from Whitelist.

DaoTreasury.sol

- addStrategy: Owner can add new strategy.
- removeStrategy: Owner can remove the current strategy.
- allocateFee: Owner can allocate protocol's fee to stakers.

StratRecollateralize.sol

- recollateralize: Owner can recollateralize the minting pool.

StratReduceReserveLP.sol

- reduceReserve: Owner can remove liquidity, buy back YToken and burn.

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 had observed some low severity 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.

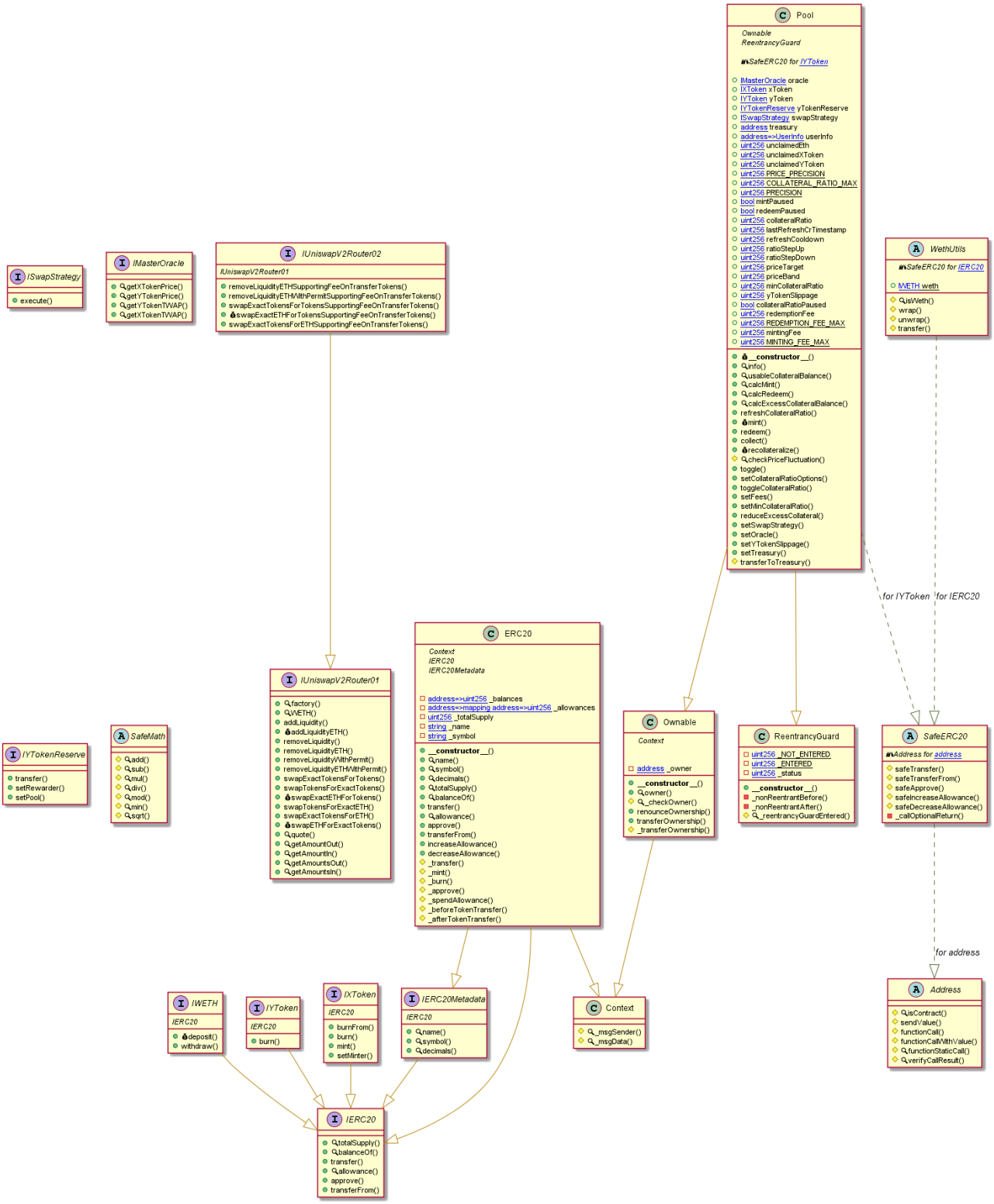
Technical Disclaimer

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.

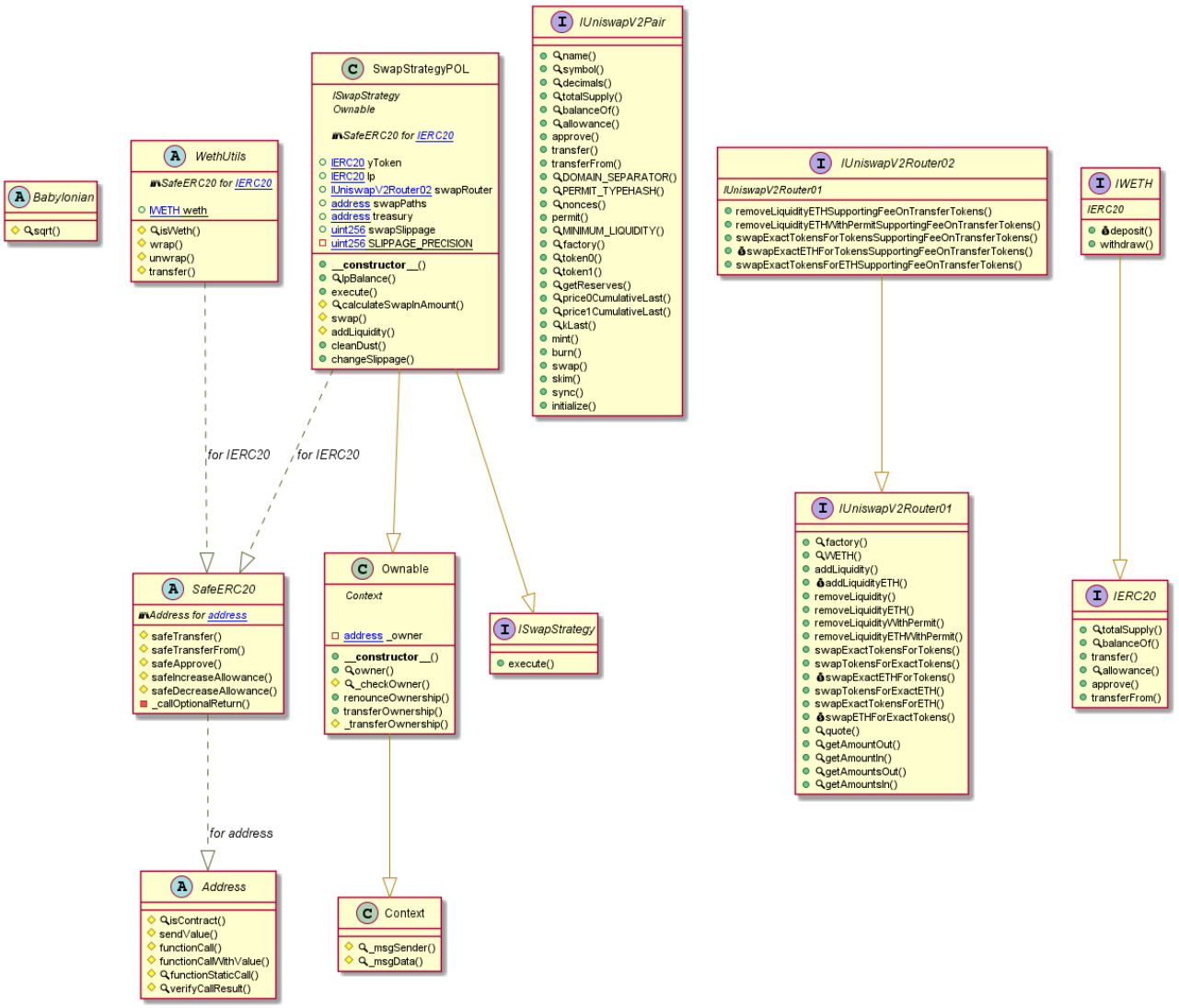
Appendix

Code Flow Diagram - TheIronMan Finance

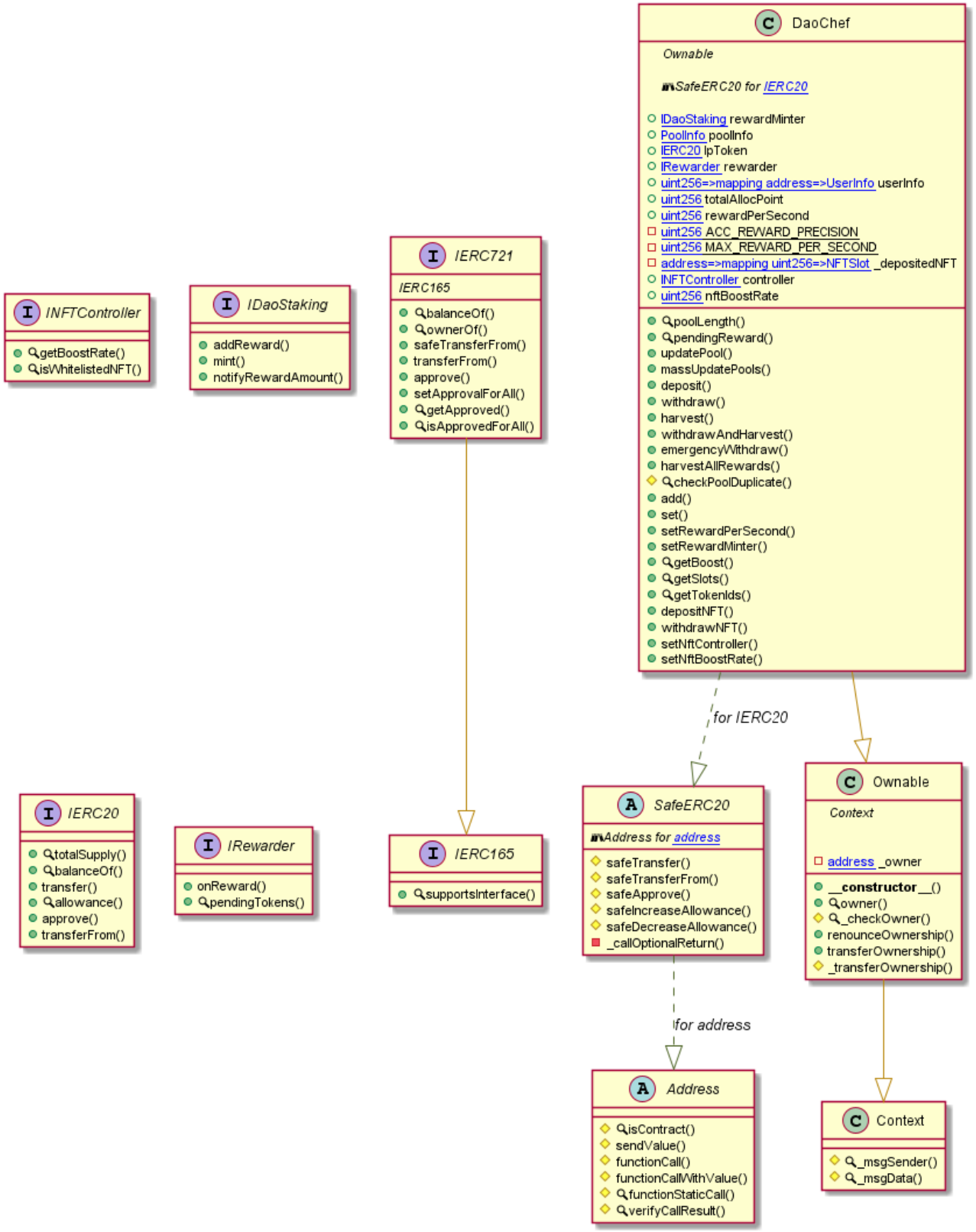
Pool Diagram



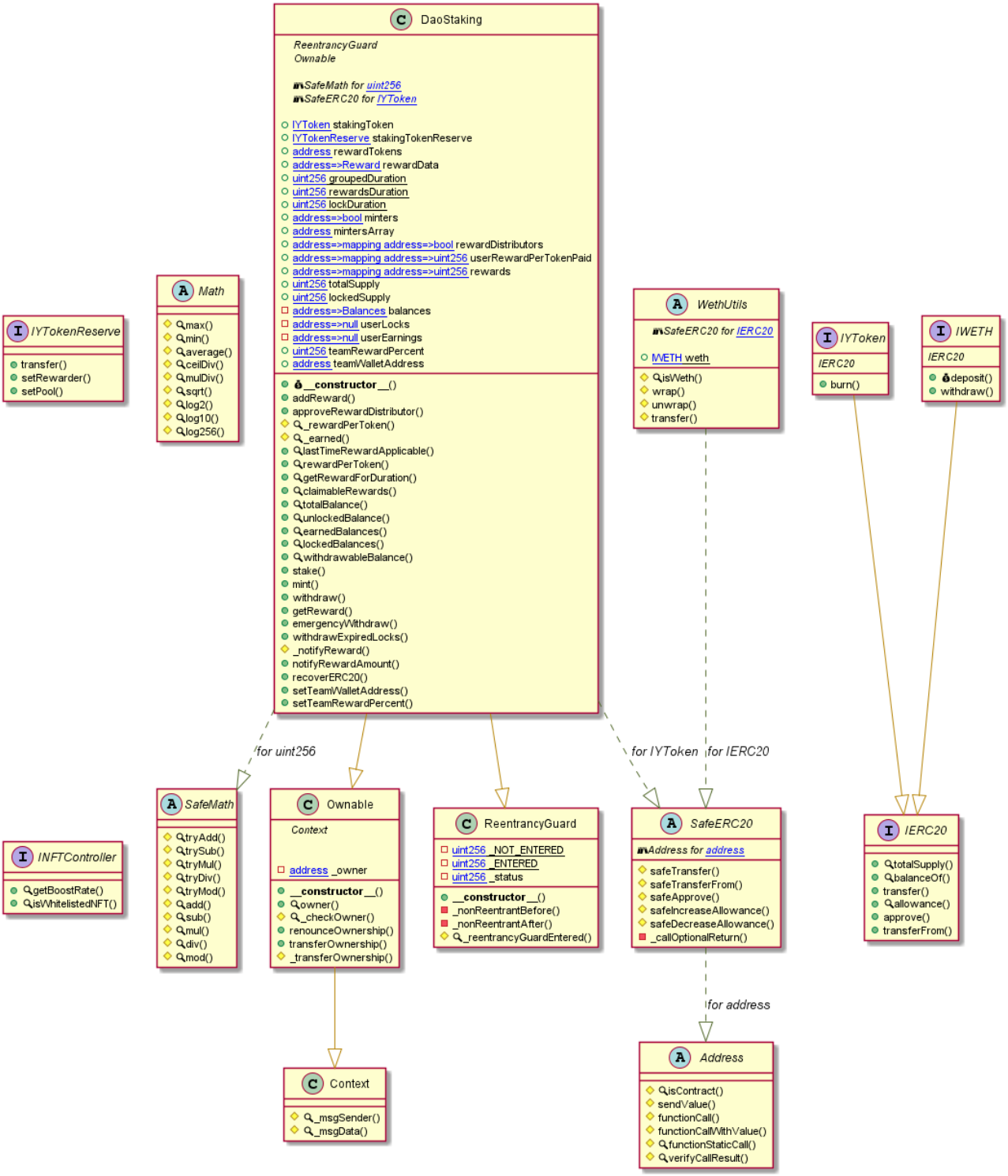
SwapStrategyPOL Diagram



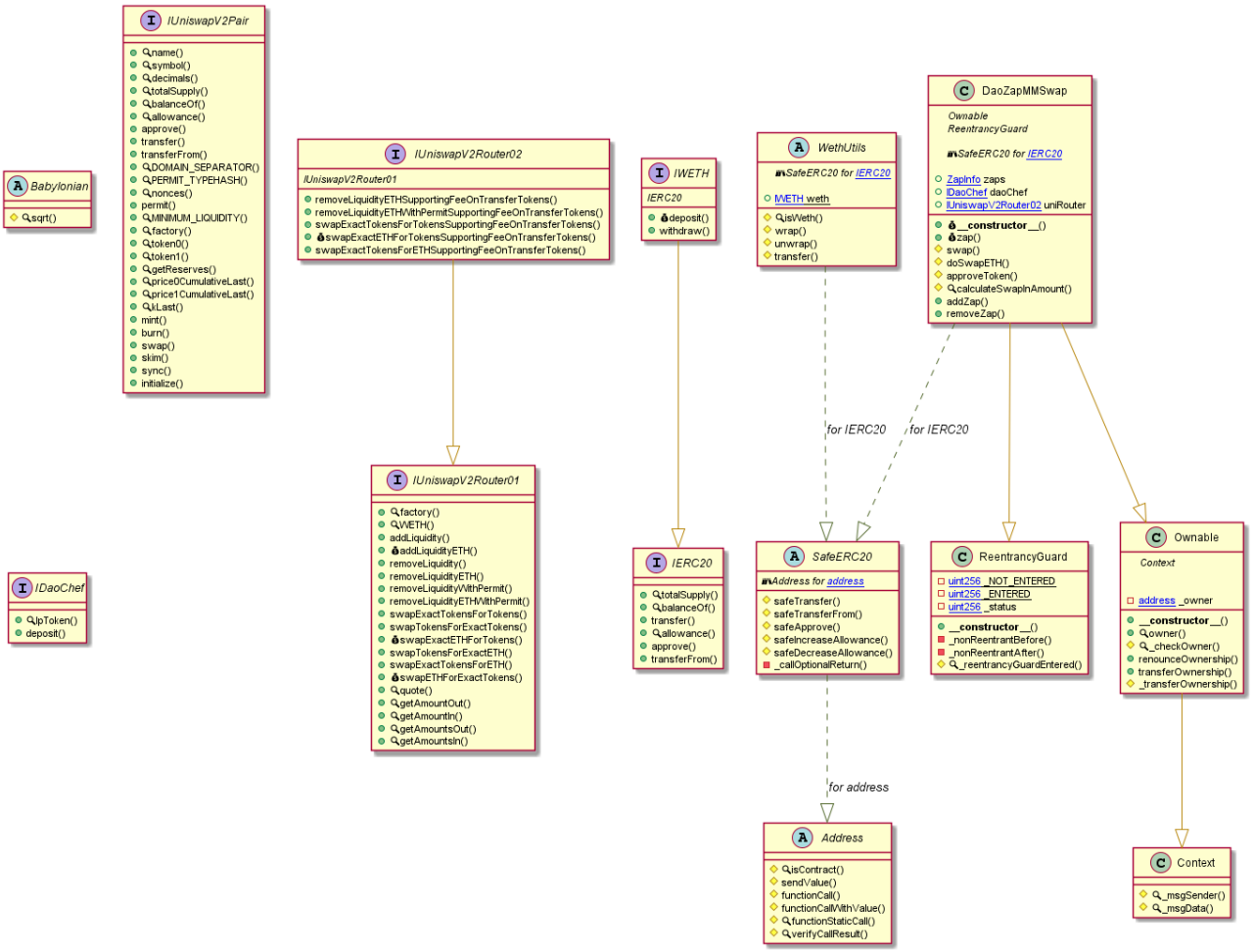
DaoChef Diagram



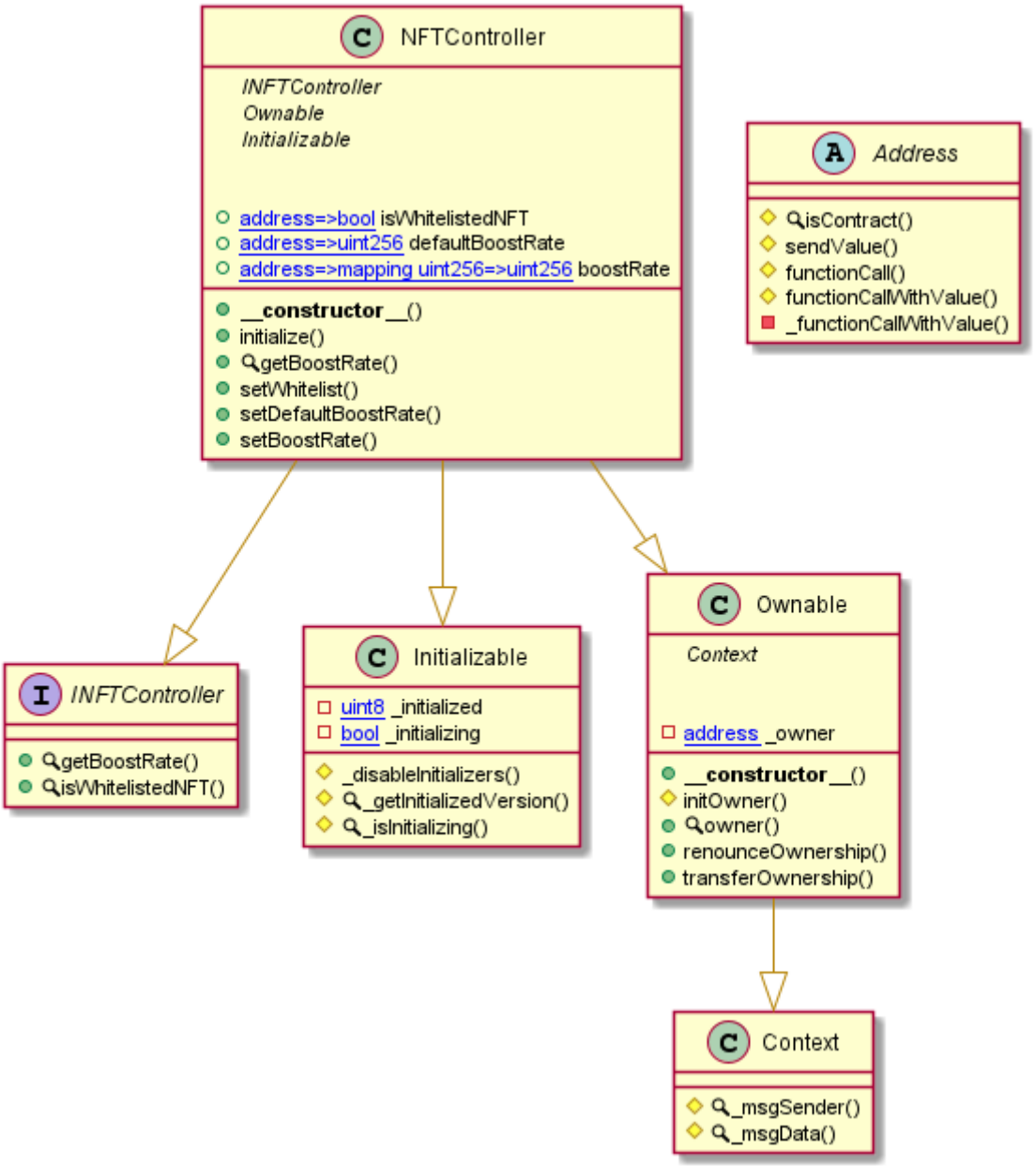
DaoStaking Diagram



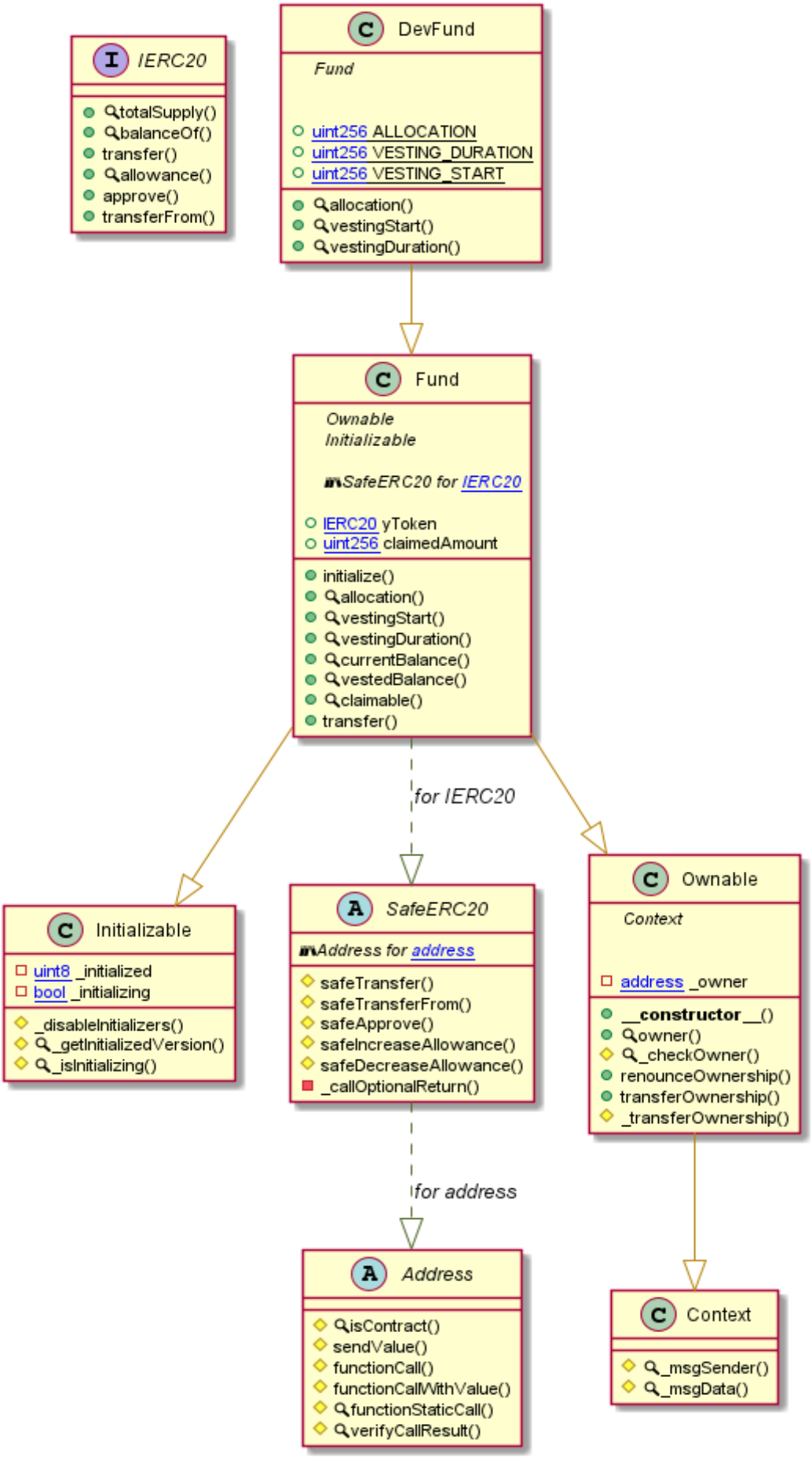
DaoZapMMSwap Diagram



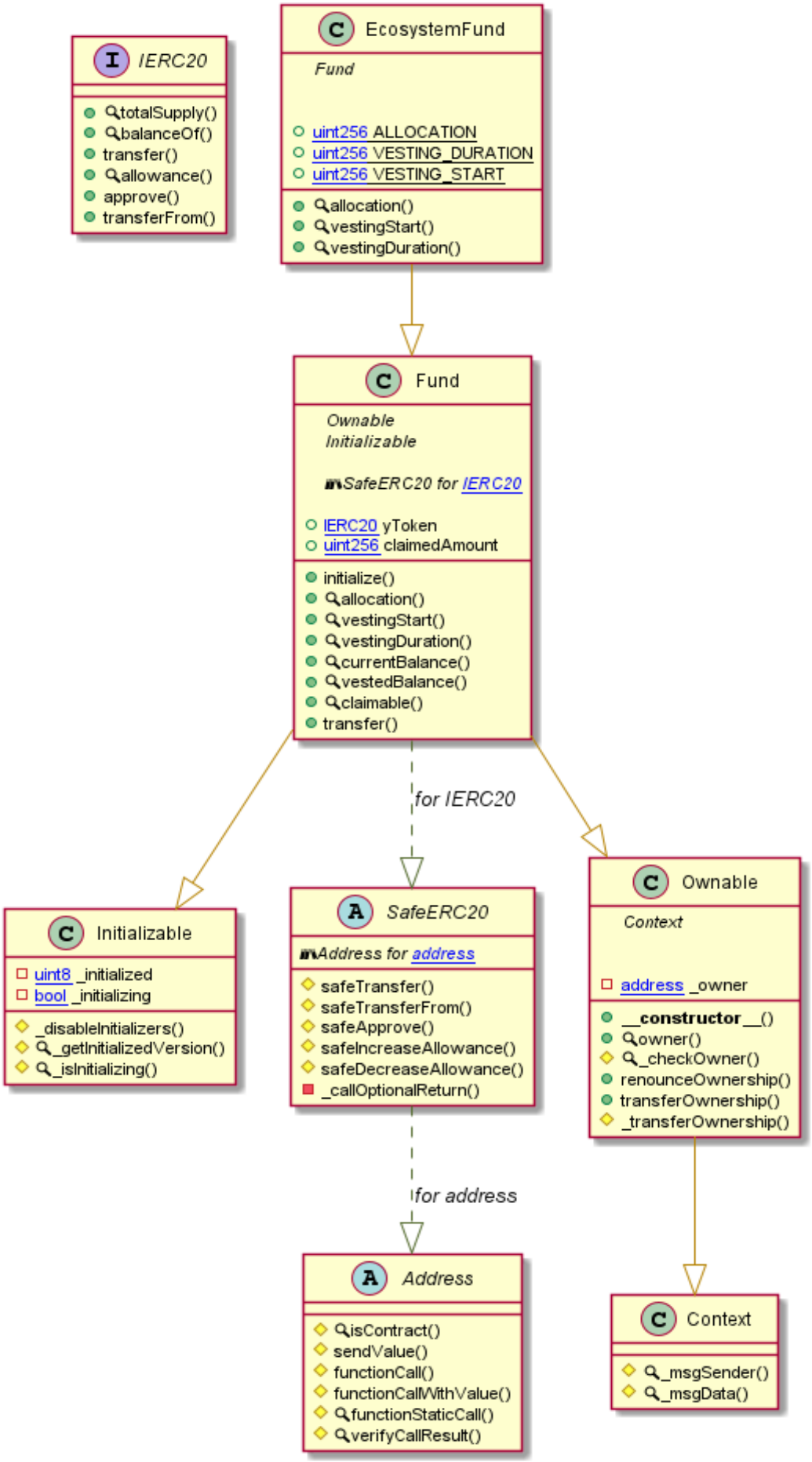
NFTController Diagram



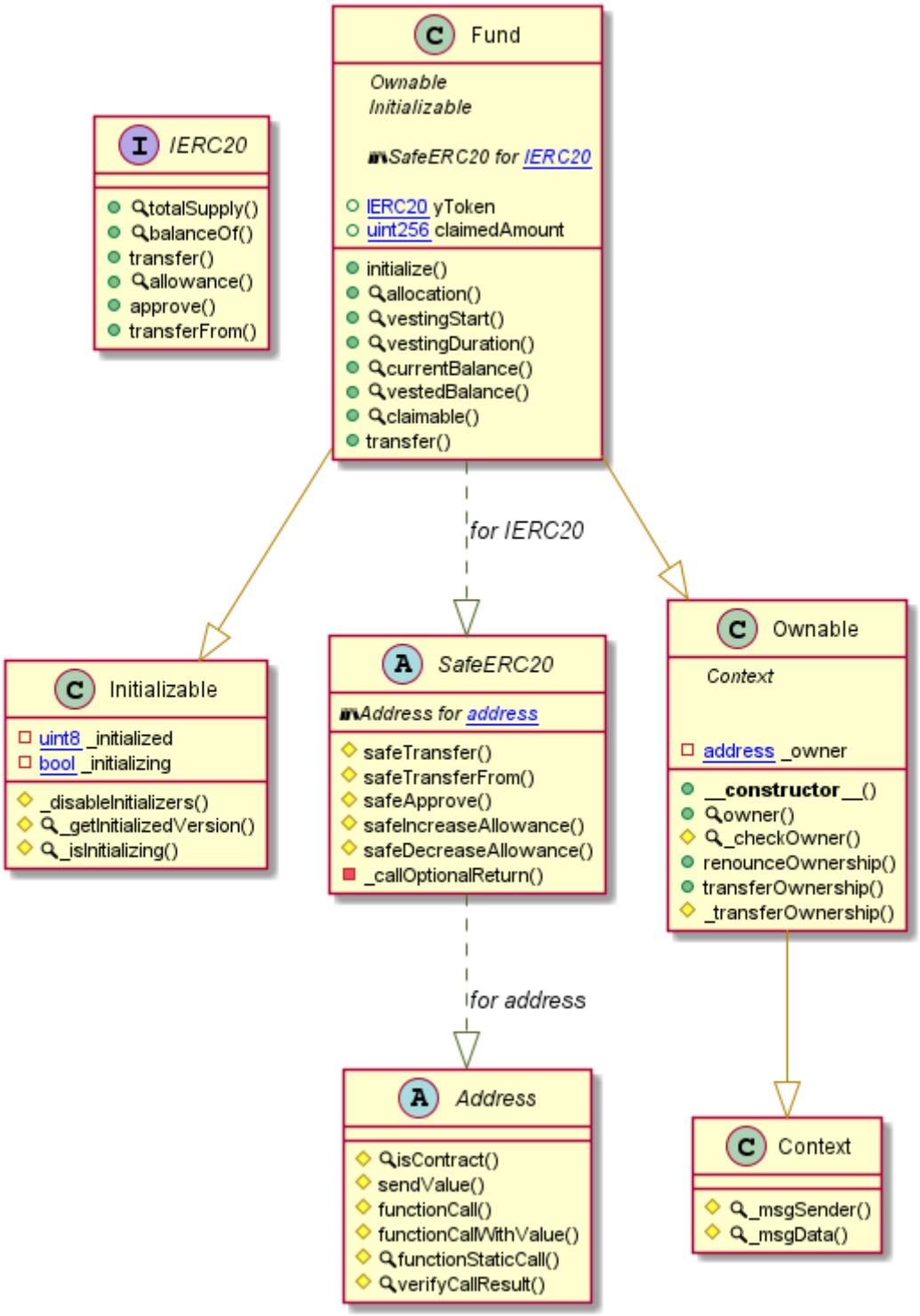
DevFund Diagram



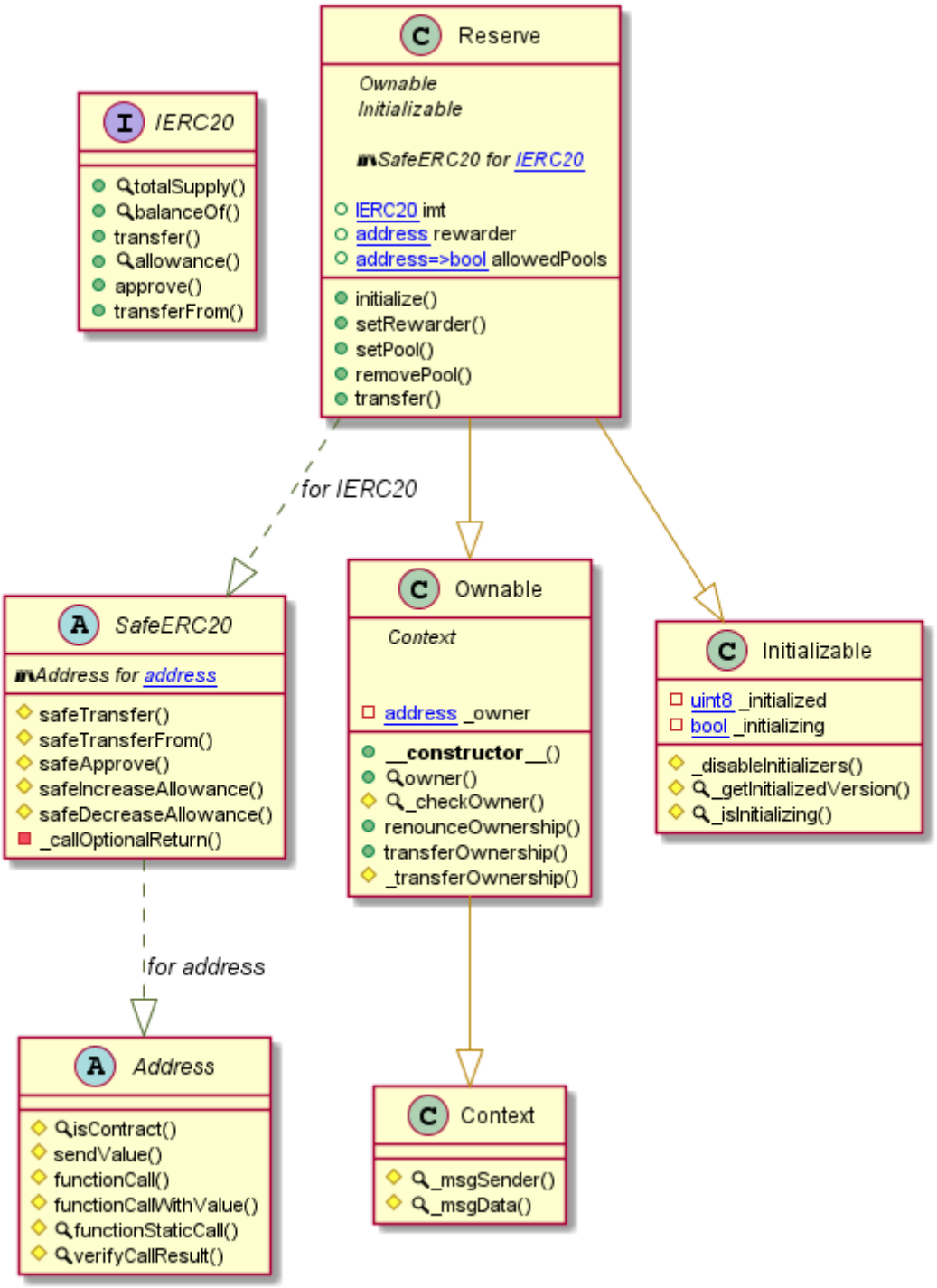
EcosystemFund Diagram



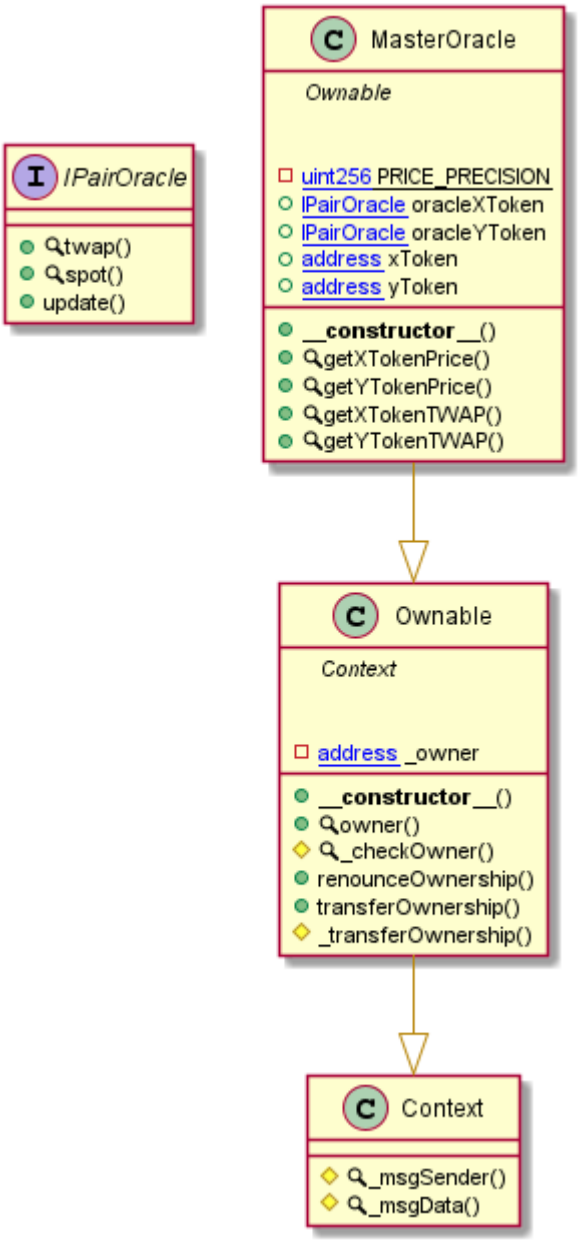
Fund Diagram



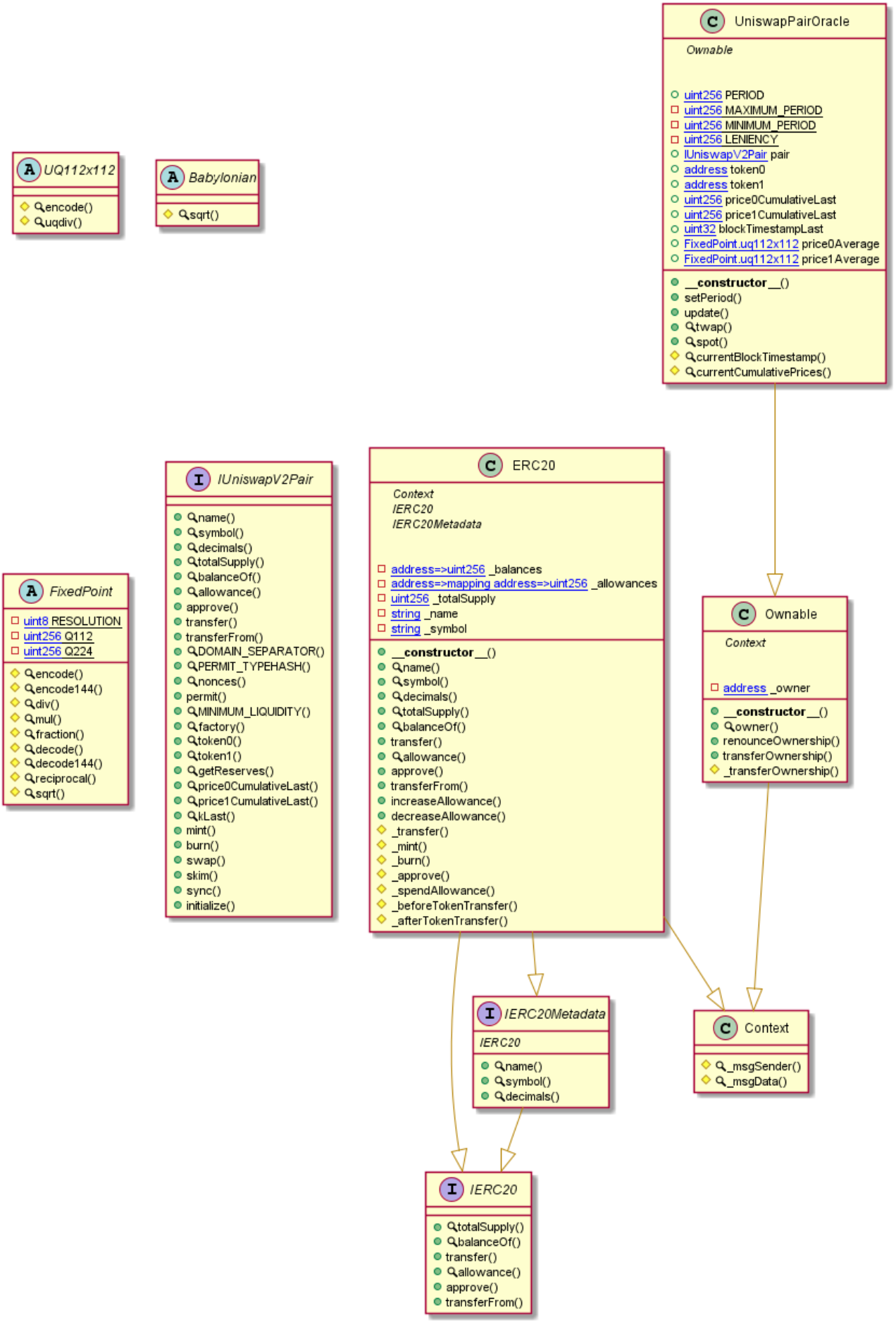
Reserve Diagram



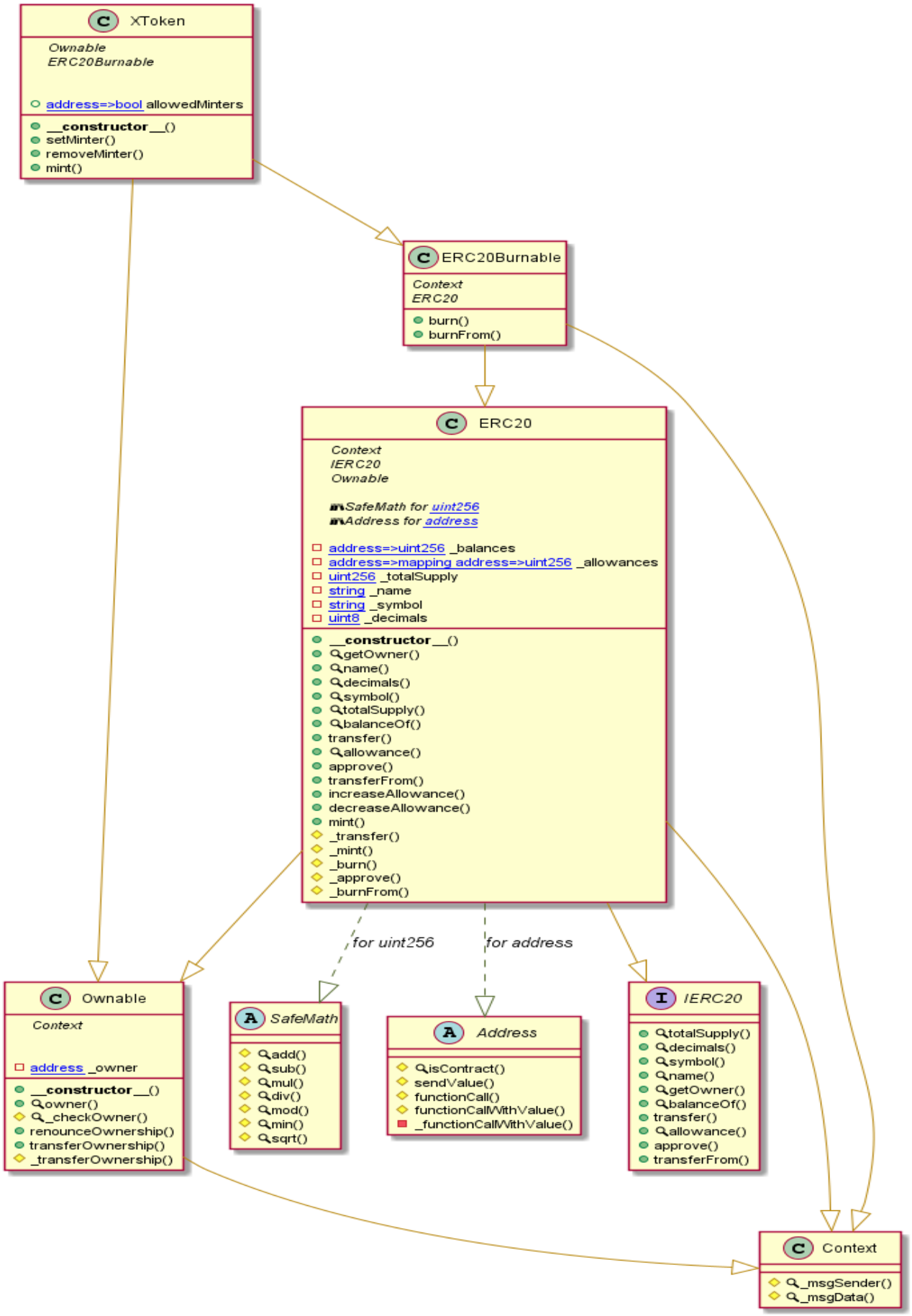
MasterOracle Diagram



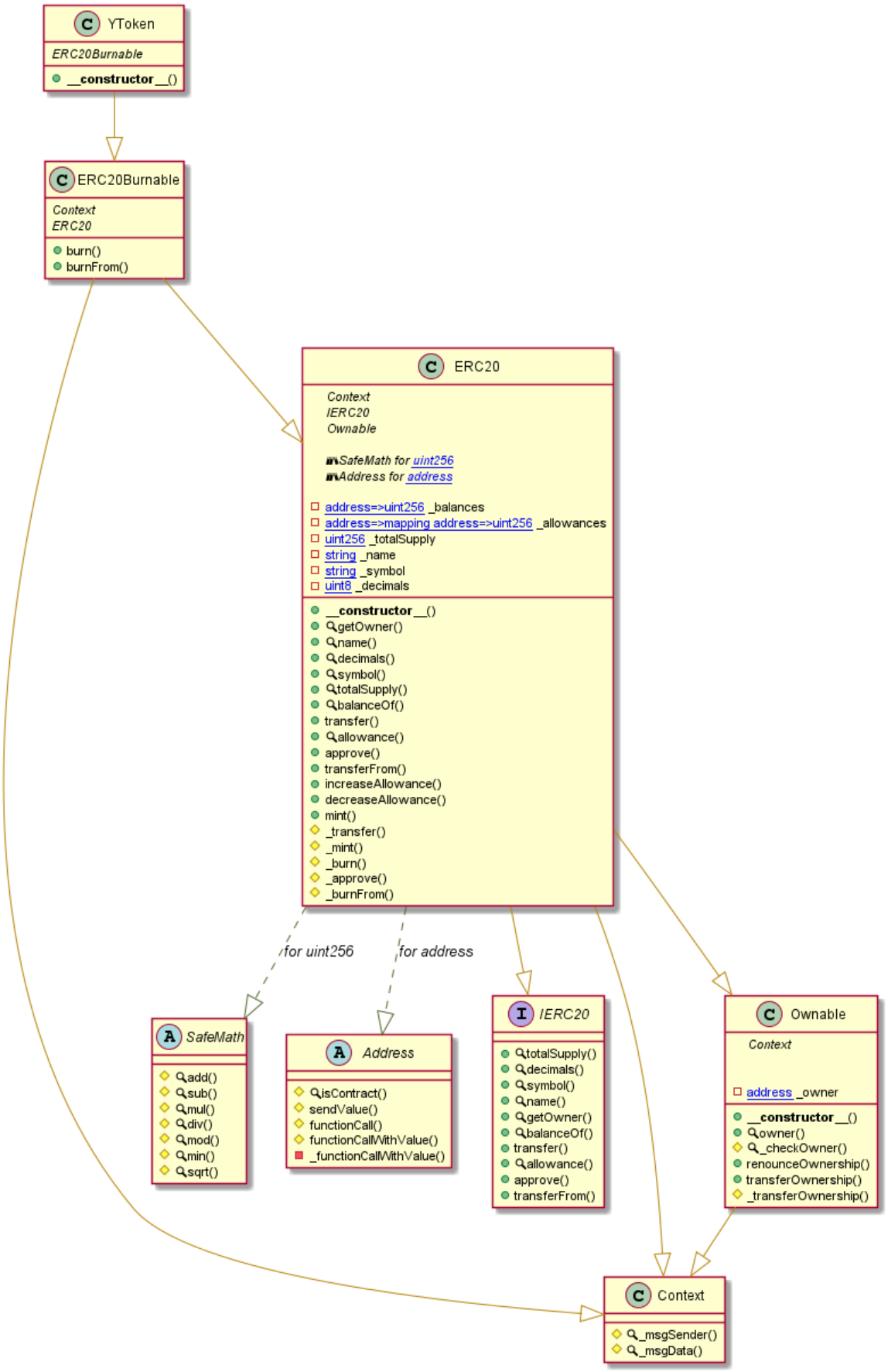
UniswapPairOracle Diagram



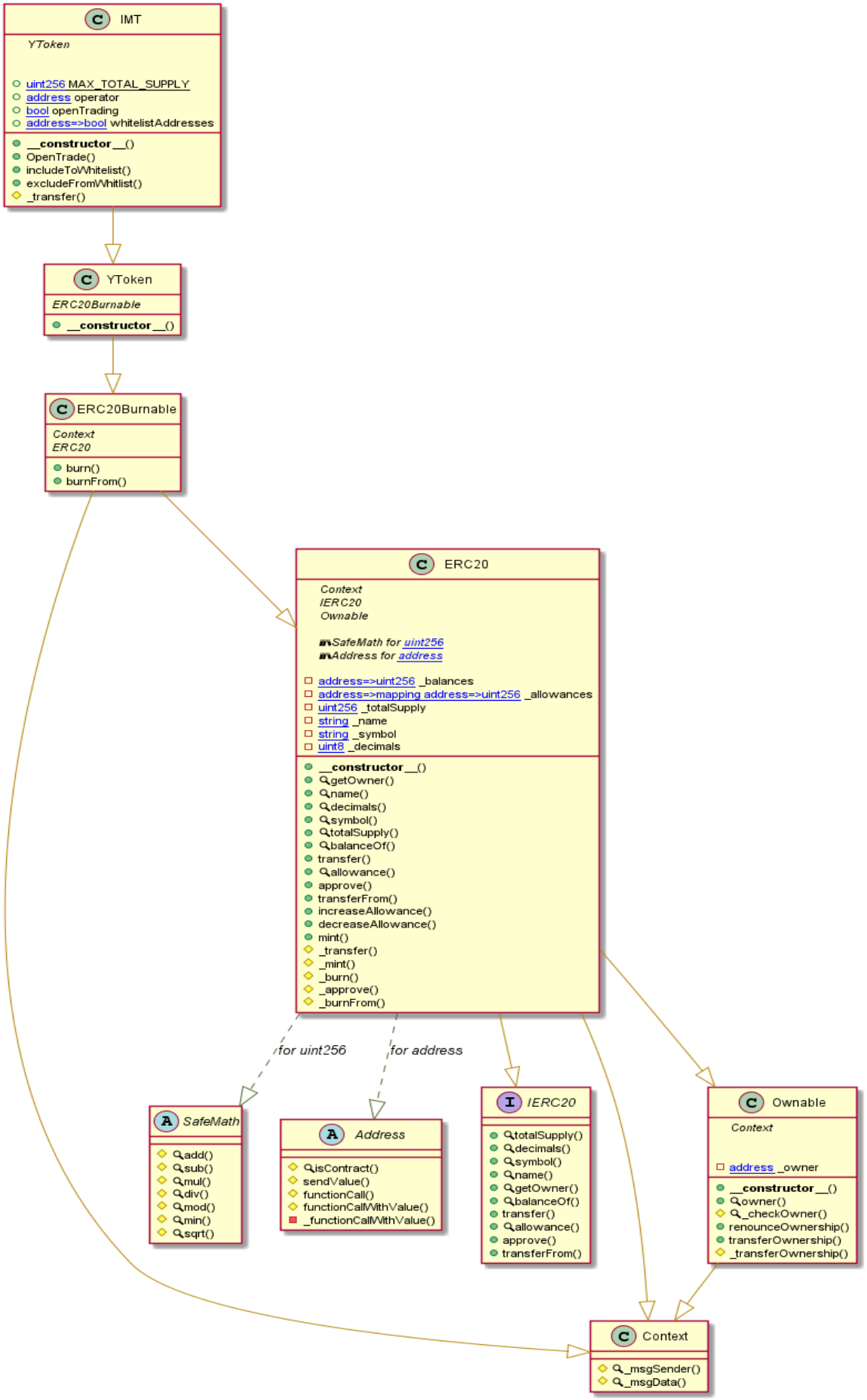
XToken Diagram



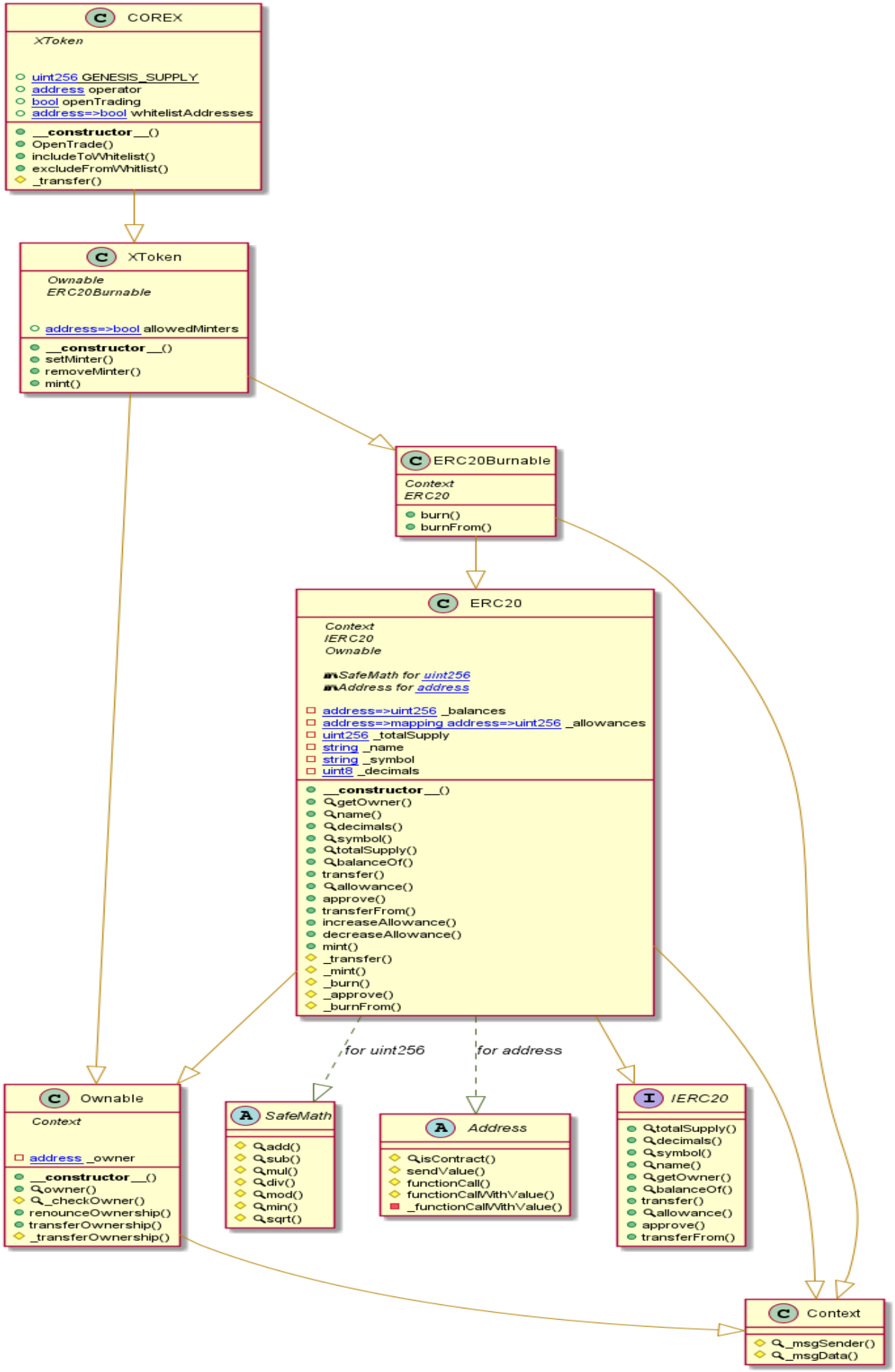
YToken Diagram



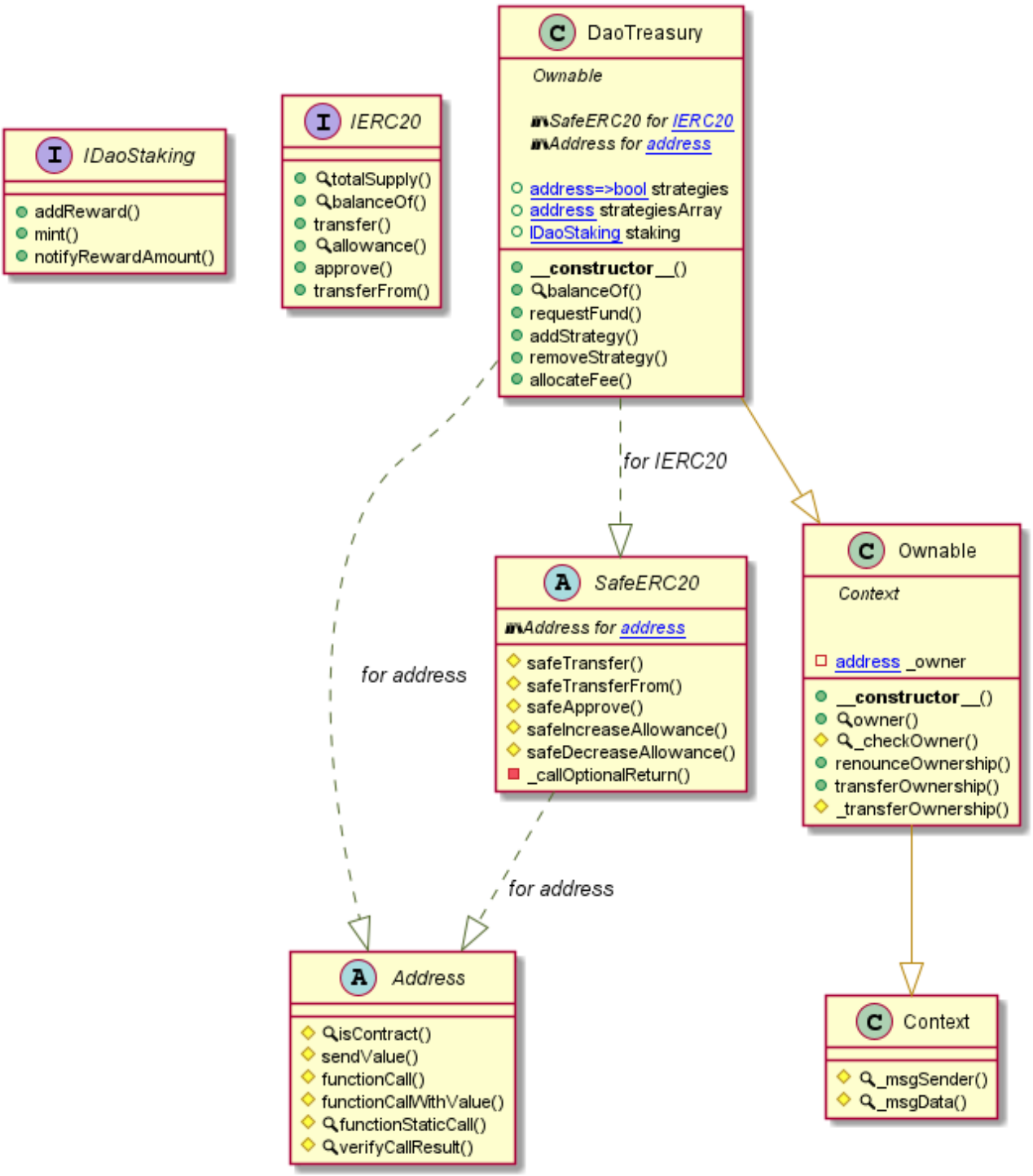
IMT Diagram



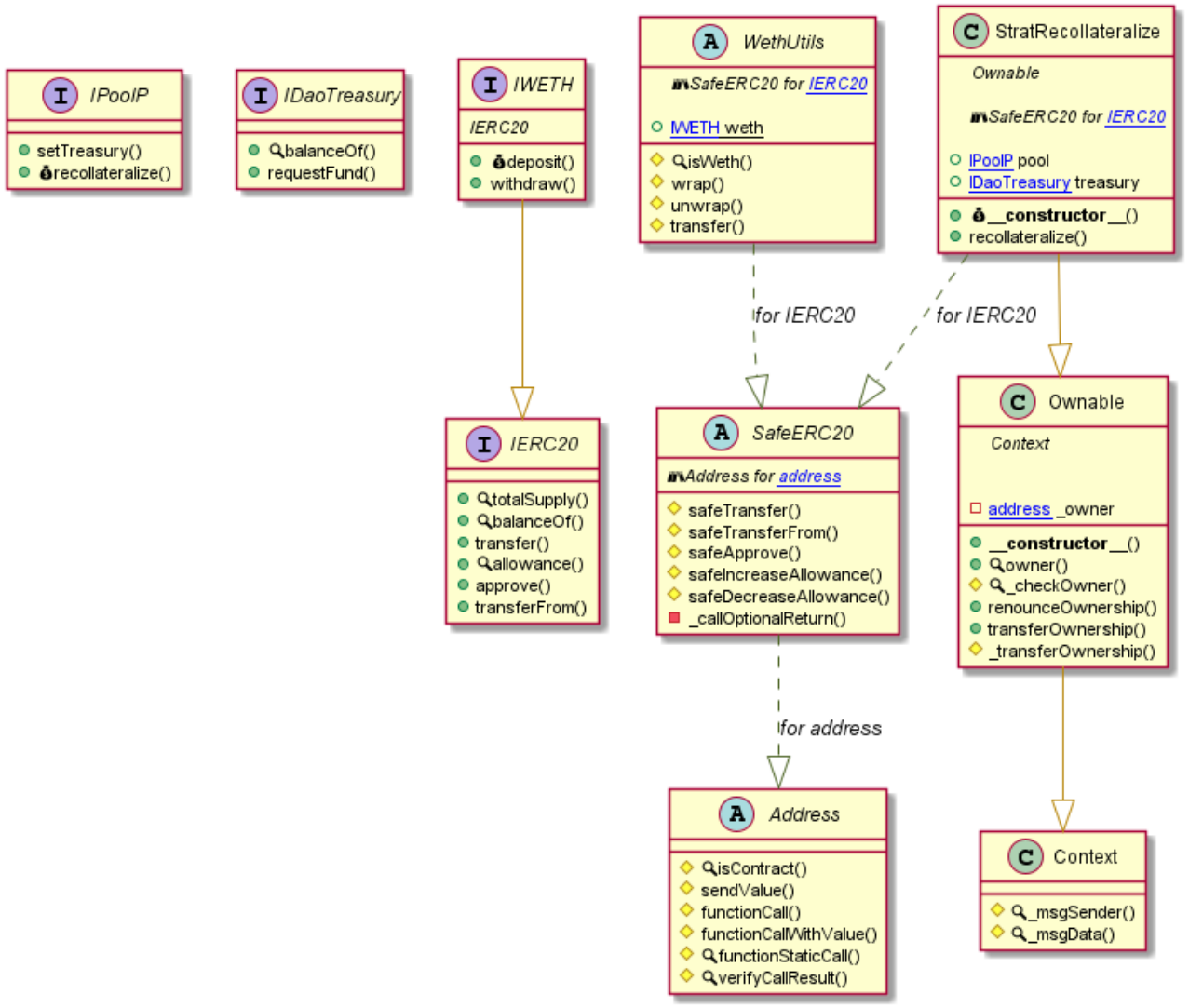
COREX Diagram



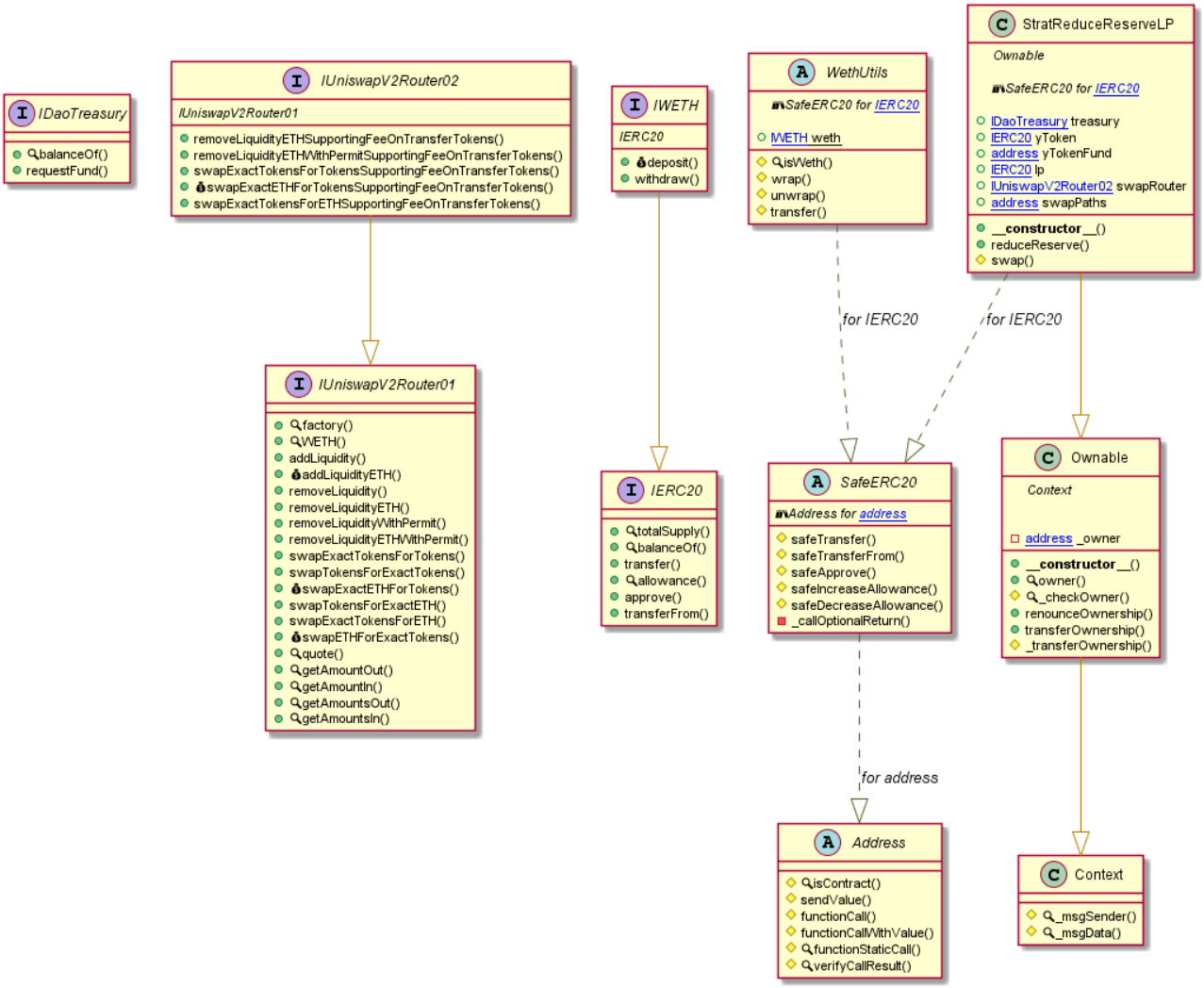
DaoTreasury Diagram



StratRecollateralize Diagram



StratReduceReserveLP Diagram



Slither Results Log

Slither log >> Pool.sol

```
Pool.setTreasury(address)._treasury (Pool.sol#1582) lacks a zero-check on :
                - treasury = _treasury (Pool.sol#1584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Pool.mint(uint256) (Pool.sol#1354-1378):
        External calls:
        - WethUtils.wrap(_ethIn) (Pool.sol#1362)
        - IERC20(WethUtils.weth).safeIncreaseAllowance(address(swapStrategy),_wethSwapIn) (Pool.sol#1365)
        - swapStrategy.execute(_wethSwapIn,_yTokenOutTwap) (Pool.sol#1366)
        State variables written after the call(s):
        - unclaimedXToken = unclaimedXToken + _xTokenOut (Pool.sol#1371)
        - userInfo[_sender].xTokenBalance = userInfo[_sender].xTokenBalance + _xTokenOut (Pool.sol#1370)
        - userInfo[_sender].lastAction = block.number (Pool.sol#1374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Pool.recollateralize() (Pool.sol#1468-1473):
        External calls:
        - WethUtils.wrap(_amount) (Pool.sol#1471)
        Event emitted after the call(s):
        - Recollateralized(msg.sender,_amount) (Pool.sol#1472)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Pool.refreshCollateralRatio() (Pool.sol#1321-1347) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp - lastRefreshCrTimestamp >= refreshCooldown,Pool::refreshCollateralRatio: Must
wait for the refresh cooldown since last refresh) (Pool.sol#1323)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Pool.refreshCollateralRatio() (Pool.sol#1321-1347) compares to a boolean constant:
        -require(bool,string)(collateralRatioPaused == false,Pool::refreshCollateralRatio: Collateral Ratio has been paused) (
Pool.sol#1322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Pragma version0.8.4 (Pool.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Pool.sol#310-315):
        - (success) = recipient.call{value: amount}() (Pool.sol#313)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Pool.sol#378-389):
        - (success,returndata) = target.call{value: value}(data) (Pool.sol#387)
Low level call in Address.functionStaticCall(address,bytes,string) (Pool.sol#407-416):
        - (success,returndata) = target.staticcall(data) (Pool.sol#414)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IUniswapV2Router01.WETH() (Pool.sol#33) is not in mixedCase
Constant WethUtils.weth (Pool.sol#562) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Pool.calcMint(uint256)._ethIn (Pool.sol#1252) is not in mixedCase
Parameter Pool.calcRedeem(uint256)._xTokenIn (Pool.sol#1277) is not in mixedCase
Parameter Pool.mint(uint256)._minXTokenOut (Pool.sol#1354) is not in mixedCase
Parameter Pool.redeem(uint256,uint256,uint256)._xTokenIn (Pool.sol#1381) is not in mixedCase
Parameter Pool.redeem(uint256,uint256,uint256)._minYTokenOut (Pool.sol#1382) is not in mixedCase
Parameter Pool.redeem(uint256,uint256,uint256)._minEthOut (Pool.sol#1383) is not in mixedCase
Parameter Pool.checkPriceFluctuation(uint256,uint256)._yAmountSpotPrice (Pool.sol#1475) is not in mixedCase
Parameter Pool.checkPriceFluctuation(uint256,uint256)._yAmountTwap (Pool.sol#1475) is not in mixedCase
Parameter Pool.toggle(bool,bool)._mintPaused (Pool.sol#1495) is not in mixedCase
Parameter Pool.toggle(bool,bool)._redeemPaused (Pool.sol#1495) is not in mixedCase
Parameter Pool.setCollateralRatioOptions(uint256,uint256,uint256,uint256)._ratioStepUp (Pool.sol#1507) is not in mixedCase
Parameter Pool.setCollateralRatioOptions(uint256,uint256,uint256,uint256)._ratioStepDown (Pool.sol#1508) is not in mixedCase
Parameter Pool.setCollateralRatioOptions(uint256,uint256,uint256,uint256)._priceBand (Pool.sol#1509) is not in mixedCase
Parameter Pool.setCollateralRatioOptions(uint256,uint256,uint256,uint256)._refreshCooldown (Pool.sol#1510) is not in mixedCase
Parameter Pool.toggleCollateralRatio(bool)._collateralRatioPaused (Pool.sol#1521) is not in mixedCase
Parameter Pool.setFees(uint256,uint256)._mintingFee (Pool.sol#1531) is not in mixedCase
Parameter Pool.setFees(uint256,uint256)._redemptionFee (Pool.sol#1531) is not in mixedCase
Parameter Pool.setMinCollateralRatio(uint256)._minCollateralRatio (Pool.sol#1541) is not in mixedCase
Parameter Pool.reduceExcessCollateral(uint256)._amount (Pool.sol#1549) is not in mixedCase
Parameter Pool.setSwapStrategy(ISwapStrategy)._swapStrategy (Pool.sol#1559) is not in mixedCase
Parameter Pool.setOracle(IMasterOracle)._oracle (Pool.sol#1567) is not in mixedCase
```

```
Parameter Pool.setSwapStrategy(ISwapStrategy)._swapStrategy (Pool.sol#1559) is not in mixedCase
Parameter Pool.setOracle(IMasterOracle)._oracle (Pool.sol#1567) is not in mixedCase
Parameter Pool.setYTokenSlippage(uint256)._slippage (Pool.sol#1574) is not in mixedCase
Parameter Pool.setTreasury(address)._treasury (Pool.sol#1582) is not in mixedCase
Parameter Pool.transferToTreasury(uint256)._amount (Pool.sol#1589) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Pool.setYTokenSlippage(uint256) (Pool.sol#1574-1578) uses literals with too many digits:
        - require(bool,string)(_slippage <= 300000,Pool::setYTokenSlippage: yTokenSlippage cannot be more than 30%) (Pool.sol#
1575)
Pool.slitherConstructorVariables() (Pool.sol#1144-1610) uses literals with too many digits:
        - yTokenSlippage = 200000 (Pool.sol#1191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Pool.priceTarget (Pool.sol#1188) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

ERC20._name (Pool.sol#820) should be immutable
ERC20._symbol (Pool.sol#821) should be immutable
Pool.xToken (Pool.sol#1159) should be immutable
Pool.yToken (Pool.sol#1160) should be immutable
Pool.yTokenReserve (Pool.sol#1161) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Pool.sol analyzed (19 contracts with 84 detectors), 81 result(s) found
```

Slither log >> SwapStrategyPOL.sol

```
Pragma version0.8.4 (SwapStrategyPOL.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (SwapStrategyPOL.sol#340-345):
        - (success) = recipient.call{value: amount}() (SwapStrategyPOL.sol#343)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (SwapStrategyPOL.sol#408-419):
        - (success,returndata) = target.call{value: value}(data) (SwapStrategyPOL.sol#417)
Low level call in Address.functionStaticCall(address,bytes,string) (SwapStrategyPOL.sol#437-446):
        - (success,returndata) = target.staticcall(data) (SwapStrategyPOL.sol#444)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IUniswapV2Pair.DOMAIN_SEPARATOR() (SwapStrategyPOL.sol#42) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (SwapStrategyPOL.sol#43) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (SwapStrategyPOL.sol#60) is not in mixedCase
Function IUniswapV2Router01.WETH() (SwapStrategyPOL.sol#80) is not in mixedCase
Constant WethUtils.weth (SwapStrategyPOL.sol#598) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter SwapStrategyPOL.execute(uint256,uint256)._wethIn (SwapStrategyPOL.sol#736) is not in mixedCase
Parameter SwapStrategyPOL.execute(uint256,uint256)._yTokenOut (SwapStrategyPOL.sol#736) is not in mixedCase
Parameter SwapStrategyPOL.calculateSwapInAmount(uint256,uint256)._reserveIn (SwapStrategyPOL.sol#762) is not in mixedCase
Parameter SwapStrategyPOL.calculateSwapInAmount(uint256,uint256)._tokenIn (SwapStrategyPOL.sol#762) is not in mixedCase
Parameter SwapStrategyPOL.swap(uint256,uint256)._wethToSwap (SwapStrategyPOL.sol#771) is not in mixedCase
Parameter SwapStrategyPOL.swap(uint256,uint256)._minYTokenOut (SwapStrategyPOL.sol#771) is not in mixedCase
Parameter SwapStrategyPOL.changeSlippage(uint256)._newSlippage (SwapStrategyPOL.sol#808) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
SwapStrategyPOL.slitherConstructorVariables() (SwapStrategyPOL.sol#700-818) uses literals with too many digits:
        - swapSlippage = 200000 (SwapStrategyPOL.sol#708)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
SwapStrategyPOL.sol analyzed (13 contracts with 84 detectors), 37 result(s) found
```

Slither log >> DaoChef.sol

```
DaoChef.pendingReward(uint256,address) (DaoChef.sol#585-596) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime && lpSupply != 0 (DaoChef.sol#590)
DaoChef.updatePool(uint256) (DaoChef.sol#601-614) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime (DaoChef.sol#603)
DaoChef.massUpdatePools() (DaoChef.sol#617-622) uses timestamp for comparisons
        Dangerous comparisons:
        - i < len (DaoChef.sol#619)
DaoChef.harvestAllRewards(address) (DaoChef.sol#760-765) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (DaoChef.sol#762)
DaoChef.checkPoolDuplicate(IERC20) (DaoChef.sol#769-774) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (DaoChef.sol#771)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Parameter DaoChef.getSlots(address,uint256)._account (DaoChef.sol#845) is not in mixedCase
Parameter DaoChef.getSlots(address,uint256)._pid (DaoChef.sol#845) is not in mixedCase
Parameter DaoChef.getTokenIds(address,uint256)._account (DaoChef.sol#858) is not in mixedCase
Parameter DaoChef.getTokenIds(address,uint256)._pid (DaoChef.sol#858) is not in mixedCase
Parameter DaoChef.depositNFT(address,uint256,uint256,uint256)._nft (DaoChef.sol#872) is not in mixedCase
Parameter DaoChef.depositNFT(address,uint256,uint256,uint256)._tokenId (DaoChef.sol#873) is not in mixedCase
Parameter DaoChef.depositNFT(address,uint256,uint256,uint256)._slotIndex (DaoChef.sol#874) is not in mixedCase
Parameter DaoChef.depositNFT(address,uint256,uint256,uint256)._pid (DaoChef.sol#875) is not in mixedCase
Parameter DaoChef.withdrawNFT(uint256,uint256)._slotIndex (DaoChef.sol#894) is not in mixedCase
Parameter DaoChef.withdrawNFT(uint256,uint256)._pid (DaoChef.sol#894) is not in mixedCase
Parameter DaoChef.setNftController(address)._controller (DaoChef.sol#908) is not in mixedCase
Parameter DaoChef.setNftBoostRate(uint256)._rate (DaoChef.sol#913) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (DaoChef.sol#462)" inContext (DaoChef.sol#456-465)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
DaoChef.sol analyzed (11 contracts with 84 detectors), 61 result(s) found
```

Slither log >> DaoStaking.sol

```
Pragma version0.8.4 (DaoStaking.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (DaoStaking.sol#678-683):
        - (success) = recipient.call{value: amount}() (DaoStaking.sol#681)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (DaoStaking.sol#746-757):
        - (success,returndata) = target.call{value: value}(data) (DaoStaking.sol#755)
Low level call in Address.functionStaticCall(address,bytes,string) (DaoStaking.sol#775-784):
        - (success,returndata) = target.staticcall(data) (DaoStaking.sol#782)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Constant WethUtils.weth (DaoStaking.sol#936) is not in UPPER_CASE_WITH_UNDERSCORES
Event DaoStakingupdatedTeamRewardPercent(uint256) (DaoStaking.sol#1580) is not in CapWords
Event DaoStakingupdatedTeamWalletAddress(address) (DaoStaking.sol#1581) is not in CapWords
Parameter DaoStaking.addReward(address,address)._rewardsToken (DaoStaking.sol#1178) is not in mixedCase
Parameter DaoStaking.addReward(address,address)._distributor (DaoStaking.sol#1178) is not in mixedCase
Parameter DaoStaking.approveRewardDistributor(address,address,bool)._rewardsToken (DaoStaking.sol#1190) is not in mixedCase
Parameter DaoStaking.approveRewardDistributor(address,address,bool)._distributor (DaoStaking.sol#1191) is not in mixedCase
Parameter DaoStaking.approveRewardDistributor(address,address,bool)._approved (DaoStaking.sol#1192) is not in mixedCase
Parameter DaoStaking.lastTimeRewardApplicable(address)._rewardsToken (DaoStaking.sol#1220) is not in mixedCase
Parameter DaoStaking.rewardPerToken(address)._rewardsToken (DaoStaking.sol#1224) is not in mixedCase
Parameter DaoStaking.getRewardForDuration(address)._rewardsToken (DaoStaking.sol#1229) is not in mixedCase
Parameter DaoStaking.notifyRewardAmount(address,uint256)._rewardsToken (DaoStaking.sol#1501) is not in mixedCase
Parameter DaoStaking.setTeamWalletAddress(address)._newWalletAddress (DaoStaking.sol#1524) is not in mixedCase
Parameter DaoStaking.setTeamRewardPercent(uint256)._percent (DaoStaking.sol#1531) is not in mixedCase
Constant DaoStaking.groupedDuration (DaoStaking.sol#1121) is not in UPPER_CASE_WITH_UNDERSCORES
Constant DaoStaking.rewardsDuration (DaoStaking.sol#1124) is not in UPPER_CASE_WITH_UNDERSCORES
Constant DaoStaking.lockDuration (DaoStaking.sol#1127) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (DaoStaking.sol#962)" inContext (DaoStaking.sol#956-965)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

```
DaoStaking.stakingToken (DaoStaking.sol#1116) should be immutable
DaoStaking.stakingTokenReserve (DaoStaking.sol#1117) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
DaoStaking.sol analyzed (14 contracts with 84 detectors), 105 result(s) found
```

Slither log >> DaoZapMMSwap.sol

```
DaoZapMMSwap.zap(uint256,uint256,uint256,bool) (DaoZapMMSwap.sol#789-839) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_liquidity >= _minLiquidity,Zap::zap: Slippage. < minLiquidity) (DaoZapMMSwap.sol#822)
        - _tokenAmtToAddLP > _tokenAmtUsedInLp (DaoZapMMSwap.sol#830)
        - _ethAmtToAddLP > _ethAmtUsedInLp (DaoZapMMSwap.sol#833)
DaoZapMMSwap.doSwapETH(address,address,uint256) (DaoZapMMSwap.sol#874-887) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_tokenReceived > 0,Zap::doSwapETH: Error Swapping Tokens 2) (DaoZapMMSwap.sol#886)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Parameter DaoZapMMSwap.addZap(address,address,uint256)._token0 (DaoZapMMSwap.sol#922) is not in mixedCase
Parameter DaoZapMMSwap.addZap(address,address,uint256)._token (DaoZapMMSwap.sol#923) is not in mixedCase
Parameter DaoZapMMSwap.addZap(address,address,uint256)._pid (DaoZapMMSwap.sol#924) is not in mixedCase
Parameter DaoZapMMSwap.removeZap(uint256)._zapId (DaoZapMMSwap.sol#935) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (DaoZapMMSwap.sol#629)" inContext (DaoZapMMSwap.sol#623-632)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (DaoZ
apMMSwap.sol#80) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uin
t256).amountBDesired (DaoZapMMSwap.sol#81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

DaoZapMMSwap.daoChef (DaoZapMMSwap.sol#772) should be immutable
DaoZapMMSwap.uniRouter (DaoZapMMSwap.sol#773) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
DaoZapMMSwap.sol analyzed (14 contracts with 84 detectors), 47 result(s) found
```

Slither log >> NFTController.sol

```
NFTController.initialize(address).owner (NFTController.sol#250) shadows:
        - Ownable.owner() (NFTController.sol#221-223) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
Pragma version0.8.4 (NFTController.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (NFTController.sol#22-27):
        - (success) = recipient.call{value: amount}() (NFTController.sol#25)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (NFTController.sol#59-81):
        - (success,returndata) = target.call{value: weiValue}(data) (NFTController.sol#67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Redundant expression "this (NFTController.sol#201)" inContext (NFTController.sol#195-204)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
NFTController.sol analyzed (6 contracts with 84 detectors), 18 result(s) found
```

Slither log >> DevFund.sol

```
Fund.vestedBalance() (DevFund.sol#594-605) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= _start (DevFund.sol#598)
        - block.timestamp > _start + _duration (DevFund.sol#601)
Fund.transfer(address,uint256) (DevFund.sol#612-619) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(amount <= claimable(),Fund::transfer: > vestedAmount) (DevFund.sol#615)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Pragma version0.8.4 (DevFund.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (DevFund.sol#133-138):
        - (success) = recipient.call{value: amount}() (DevFund.sol#136)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (DevFund.sol#201-212):
        - (success,returndata) = target.call{value: value}(data) (DevFund.sol#210)
Low level call in Address.functionStaticCall(address,bytes,string) (DevFund.sol#230-239):
        - (success,returndata) = target.staticcall(data) (DevFund.sol#237)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter Fund.initialize(address)._yToken (DevFund.sol#577) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (DevFund.sol#498)" inContext (DevFund.sol#492-501)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
DevFund.sol analyzed (8 contracts with 84 detectors), 24 result(s) found
```

Slither log >> EcosystemFund.sol

```
Pragma version0.8.4 (EcosystemFund.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (EcosystemFund.sol#133-138):
        - (success) = recipient.call{value: amount}() (EcosystemFund.sol#136)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (EcosystemFund.sol#201-212):
        - (success,returndata) = target.call{value: value}(data) (EcosystemFund.sol#210)
Low level call in Address.functionStaticCall(address,bytes,string) (EcosystemFund.sol#230-239):
        - (success,returndata) = target.staticcall(data) (EcosystemFund.sol#237)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter Fund.initialize(address)._yToken (EcosystemFund.sol#577) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (EcosystemFund.sol#498)" inContext (EcosystemFund.sol#492-501)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
EcosystemFund.sol analyzed (8 contracts with 84 detectors), 24 result(s) found
```

Slither log >> Fund.sol

```
Fund.transfer(address,uint256) (Fund.sol#612-619) should emit an event for:
        - claimedAmount = claimedAmount + amount (Fund.sol#617)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Fund.vestedBalance() (Fund.sol#594-605) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= _start (Fund.sol#598)
        - block.timestamp > _start + _duration (Fund.sol#601)
Fund.transfer(address,uint256) (Fund.sol#612-619) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(amount <= claimable(),Fund::transfer: > vestedAmount) (Fund.sol#615)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Parameter Fund.initialize(address)._yToken (Fund.sol#577) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Fund.sol#498)" inContext (Fund.sol#492-501)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Fund (Fund.sol#570-621) does not implement functions:
        - Fund.allocation() (Fund.sol#584)
        - Fund.vestingDuration() (Fund.sol#588)
        - Fund.vestingStart() (Fund.sol#586)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
Fund.sol analyzed (7 contracts with 84 detectors), 26 result(s) found
```

Slither log >> Reserve.sol

```
Reserve.setRewarder(address)._rewarder (Reserve.sol#584) lacks a zero-check on :
                - rewarder = _rewarder (Reserve.sol#586)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Pragma version0.8.4 (Reserve.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Reserve.sol#131-136):
        - (success) = recipient.call{value: amount}() (Reserve.sol#134)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Reserve.sol#199-210):
        - (success,returndata) = target.call{value: value}(data) (Reserve.sol#208)
Low level call in Address.functionStaticCall(address,bytes,string) (Reserve.sol#228-237):
        - (success,returndata) = target.staticcall(data) (Reserve.sol#235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter Reserve.initialize(address)._imt (Reserve.sol#577) is not in mixedCase
Parameter Reserve.setRewarder(address)._rewarder (Reserve.sol#584) is not in mixedCase
Parameter Reserve.setPool(address)._pool (Reserve.sol#590) is not in mixedCase
Parameter Reserve.removePool(address)._pool (Reserve.sol#596) is not in mixedCase
Parameter Reserve.transfer(address,uint256)._to (Reserve.sol#602) is not in mixedCase
Parameter Reserve.transfer(address,uint256)._amount (Reserve.sol#602) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Reserve.sol#386)" inContext (Reserve.sol#380-389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
Reserve.sol analyzed (7 contracts with 84 detectors), 30 result(s) found
```

Slither log >> MasterOracle.sol

```
Context._msgData() (MasterOracle.sol#19-22) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.4 (MasterOracle.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Redundant expression "this (MasterOracle.sol#20)" inContext (MasterOracle.sol#14-23)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable MasterOracle.constructor(address,address,address,address)._oracleXToken (MasterOracle.sol#102) is too similar to Mast
erOracle.constructor(address,address,address,address)._oracleYToken (MasterOracle.sol#103)
Variable MasterOracle.oracleXToken (MasterOracle.sol#93) is too similar to MasterOracle.oracleYToken (MasterOracle.sol#94)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

MasterOracle.oracleXToken (MasterOracle.sol#93) should be immutable
MasterOracle.oracleYToken (MasterOracle.sol#94) should be immutable
MasterOracle.xToken (MasterOracle.sol#96) should be immutable
MasterOracle.yToken (MasterOracle.sol#97) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
MasterOracle.sol analyzed (4 contracts with 84 detectors), 10 result(s) found
```

Slither log >> UniswapPairOracle.sol

```
Variable UniswapPairOracle.spot(address,uint256)._token0MissingDecimals (UniswapPairOracle.sol#820) is too similar to UniswapP
airOracle.spot(address,uint256)._token1MissingDecimals (UniswapPairOracle.sol#821)
Variable UniswapPairOracle.twap(address,uint256)._token0MissingDecimals (UniswapPairOracle.sol#799) is too similar to UniswapP
airOracle.twap(address,uint256)._token1MissingDecimals (UniswapPairOracle.sol#800)
Variable UniswapPairOracle.twap(address,uint256)._token0MissingDecimals (UniswapPairOracle.sol#799) is too similar to UniswapP
airOracle.spot(address,uint256)._token1MissingDecimals (UniswapPairOracle.sol#821)
Variable UniswapPairOracle.spot(address,uint256)._token0MissingDecimals (UniswapPairOracle.sol#820) is too similar to UniswapP
airOracle.twap(address,uint256)._token1MissingDecimals (UniswapPairOracle.sol#800)
Variable UniswapPairOracle.currentCumulativePrices(address).price0Cumulative (UniswapPairOracle.sol#840) is too similar to Uni
swapPairOracle.update().price1Cumulative (UniswapPairOracle.sol#771)
Variable UniswapPairOracle.currentCumulativePrices(address).price0Cumulative (UniswapPairOracle.sol#840) is too similar to Uni
swapPairOracle.currentCumulativePrices(address).price1Cumulative (UniswapPairOracle.sol#841)
Variable UniswapPairOracle.update().price0Cumulative (UniswapPairOracle.sol#770) is too similar to UniswapPairOracle.update().
price1Cumulative (UniswapPairOracle.sol#771)
Variable UniswapPairOracle.update().price0Cumulative (UniswapPairOracle.sol#770) is too similar to UniswapPairOracle.currentCu
mulativePrices(address).price1Cumulative (UniswapPairOracle.sol#841)
Variable UniswapPairOracle.price0CumulativeLast (UniswapPairOracle.sol#742) is too similar to UniswapPairOracle.price1Cumulati
veLast (UniswapPairOracle.sol#743)
Variable UniswapPairOracle.price0Average (UniswapPairOracle.sol#745) is too similar to UniswapPairOracle.price1Average (Uniswa
pPairOracle.sol#746)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

ERC20 (UniswapPairOracle.sol#430-727) does not implement functions:
        - IERC20Metadata.decimals() (UniswapPairOracle.sol#351)
        - IERC20.getOwner() (UniswapPairOracle.sol#317)
        - IERC20Metadata.name() (UniswapPairOracle.sol#341)
        - IERC20Metadata.symbol() (UniswapPairOracle.sol#346)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

ERC20._name (UniswapPairOracle.sol#437) should be immutable
ERC20._symbol (UniswapPairOracle.sol#438) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
UniswapPairOracle.sol analyzed (12 contracts with 84 detectors), 61 result(s) found
```

Slither log >> XToken.sol

```
XToken.setMinter(address) (XToken.sol#835-838) compares to a boolean constant:
        -require(bool,string)(allowedMinters[_minter] == false,XToken::setMinter: ALREADY_ALLOWED) (XToken.sol#836)
XToken.removeMinter(address) (XToken.sol#840-843) compares to a boolean constant:
        -require(bool,string)(allowedMinters[_minter] == true,XToken::setMinter: NOT_ALLOWED) (XToken.sol#841)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Pragma version0.8.4 (XToken.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

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

Parameter XToken.setMinter(address)._minter (XToken.sol#835) is not in mixedCase
Parameter XToken.removeMinter(address)._minter (XToken.sol#840) is not in mixedCase
Parameter XToken.mint(address,uint256)._address (XToken.sol#848) is not in mixedCase
Parameter XToken.mint(address,uint256)._amount (XToken.sol#848) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (XToken.sol#433)" inContext (XToken.sol#427-436)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
XToken.sol analyzed (8 contracts with 84 detectors), 33 result(s) found
```

Slither log >> YToken.sol

```
YToken.constructor(string,string)._name (YToken.sol#817) shadows:
        - ERC20._name (YToken.sol#512) (state variable)
YToken.constructor(string,string)._symbol (YToken.sol#817) shadows:
        - ERC20._symbol (YToken.sol#513) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
Pragma version0.8.4 (YToken.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (YToken.sol#224-230):
        - (success) = recipient.call{value: amount}() (YToken.sol#228)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (YToken.sol#303-329):
        - (success,returndata) = target.call{value: weiValue}(data) (YToken.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Redundant expression "this (YToken.sol#432)" inContext (YToken.sol#426-435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
YToken.sol analyzed (8 contracts with 84 detectors), 27 result(s) found
```

Slither log >> IMT.sol

```
IMT.constructor(string,string,address,address,address)._name (IMT.sol#829) shadows:
        - ERC20._name (IMT.sol#512) (state variable)
IMT.constructor(string,string,address,address,address)._symbol (IMT.sol#830) shadows:
        - ERC20._symbol (IMT.sol#513) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
Pragma version0.8.4 (IMT.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (IMT.sol#224-230):
        - (success) = recipient.call{value: amount}() (IMT.sol#228)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (IMT.sol#303-329):
        - (success,returndata) = target.call{value: weiValue}(data) (IMT.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IMT.OpenTrade() (IMT.sol#843-847) is not in mixedCase
Parameter IMT.includeToWhitelist(address)._address (IMT.sol#849) is not in mixedCase
Parameter IMT.excludeFromWhitlist(address)._address (IMT.sol#856) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (IMT.sol#432)" inContext (IMT.sol#426-435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

IMT.operator (IMT.sol#824) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
IMT.sol analyzed (9 contracts with 84 detectors), 33 result(s) found
```

Slither log >> COREX.sol

```
Pragma version0.8.4 (COREX.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

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

Parameter XToken.setMinter(address)._minter (COREX.sol#835) is not in mixedCase
Parameter XToken.removeMinter(address)._minter (COREX.sol#840) is not in mixedCase
Parameter XToken.mint(address,uint256)._address (COREX.sol#848) is not in mixedCase
Parameter XToken.mint(address,uint256)._amount (COREX.sol#848) is not in mixedCase
Function COREX.OpenTrade() (COREX.sol#866-870) is not in mixedCase
Parameter COREX.includeToWhitelist(address)._address (COREX.sol#872) is not in mixedCase
Parameter COREX.excludeFromWhitlist(address)._address (COREX.sol#879) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (COREX.sol#433)" inContext (COREX.sol#427-436)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

COREX.operator (COREX.sol#857) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
COREX.sol analyzed (9 contracts with 84 detectors), 39 result(s) found
```

Slither log >> DaoTreasury.sol

```
Reentrancy in DaoTreasury.allocateFee(address,uint256) (DaoTreasury.sol#535-541):
        External calls:
        - IERC20(_token).safeIncreaseAllowance(address(staking),_amount) (DaoTreasury.sol#538)
        - staking.notifyRewardAmount(_token,_amount) (DaoTreasury.sol#539)
        Event emitted after the call(s):
        - TokenRewardAllocated(_token,_amount) (DaoTreasury.sol#540)
Reentrancy in DaoTreasury.requestFund(address,uint256) (DaoTreasury.sol#499-504):
        External calls:
        - IERC20(_token).safeIncreaseAllowance(msg.sender,_amount) (DaoTreasury.sol#502)
        Event emitted after the call(s):
        - FundRequested(msg.sender,_amount) (DaoTreasury.sol#503)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
Pragma version0.8.4 (DaoTreasury.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (DaoTreasury.sol#141-146):
        - (success) = recipient.call{value: amount}() (DaoTreasury.sol#144)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (DaoTreasury.sol#209-220):
        - (success,returndata) = target.call{value: value}(data) (DaoTreasury.sol#218)
Low level call in Address.functionStaticCall(address,bytes,string) (DaoTreasury.sol#238-247):
        - (success,returndata) = target.staticcall(data) (DaoTreasury.sol#245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter DaoTreasury.balanceOf(address)._token (DaoTreasury.sol#490) is not in mixedCase
Parameter DaoTreasury.requestFund(address,uint256)._token (DaoTreasury.sol#499) is not in mixedCase
Parameter DaoTreasury.requestFund(address,uint256)._amount (DaoTreasury.sol#499) is not in mixedCase
Parameter DaoTreasury.addStrategy(address)._strategy (DaoTreasury.sol#508) is not in mixedCase
Parameter DaoTreasury.removeStrategy(address)._strategy (DaoTreasury.sol#518) is not in mixedCase
Parameter DaoTreasury.allocateFee(address,uint256)._token (DaoTreasury.sol#535) is not in mixedCase
Parameter DaoTreasury.allocateFee(address,uint256)._amount (DaoTreasury.sol#535) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (DaoTreasury.sol#396)" inContext (DaoTreasury.sol#390-399)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

DaoTreasury.staking (DaoTreasury.sol#479) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
DaoTreasury.sol analyzed (7 contracts with 84 detectors), 27 result(s) found
```

Slither log >> StratRecollateralize.sol

```
Reentrancy in StratRecollateralize.recollateralize(uint256) (StratRecollateralize.sol#512-522):
        External calls:
        - treasury.requestFund(address(WethUtils.weth),_amount) (StratRecollateralize.sol#516)
        - IERC20(WethUtils.weth).safeTransferFrom(address(treasury),address(this),_amount) (StratRecollateralize.sol#517)
        - WethUtils.unwrap(_amount) (StratRecollateralize.sol#518)
        - pool.recollateralize{value: _amount}() (StratRecollateralize.sol#519)
        External calls sending eth:
        - pool.recollateralize{value: _amount}() (StratRecollateralize.sol#519)
        Event emitted after the call(s):
        - Recollateralized(_amount) (StratRecollateralize.sol#521)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
Pragma version0.8.4 (StratRecollateralize.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (StratRecollateralize.sol#150-155):
        - (success) = recipient.call{value: amount}() (StratRecollateralize.sol#153)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (StratRecollateralize.sol#218-229):
        - (success,returndata) = target.call{value: value}(data) (StratRecollateralize.sol#227)
Low level call in Address.functionStaticCall(address,bytes,string) (StratRecollateralize.sol#247-256):
        - (success,returndata) = target.staticcall(data) (StratRecollateralize.sol#254)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Constant WethUtils.weth (StratRecollateralize.sol#403) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter StratRecollateralize.recollateralize(uint256)._amount (StratRecollateralize.sol#512) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (StratRecollateralize.sol#429)" inContext (StratRecollateralize.sol#423-432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
StratRecollateralize.sol analyzed (10 contracts with 84 detectors), 24 result(s) found
```

Slither log >> StratReduceReserveLP.sol

```
StratReduceReserveLP.constructor(IERC20,address,IERC20,IUniswapV2Router02,address[],IDaoTreasury)._yTokenFund (StratReduceRese
rveLP.sol#639) lacks a zero-check on :
                - yTokenFund = _yTokenFund (StratReduceReserveLP.sol#647)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Pragma version0.8.4 (StratReduceReserveLP.sol#3) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (StratReduceReserveLP.sol#271-276):
        - (success) = recipient.call{value: amount}() (StratReduceReserveLP.sol#274)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (StratReduceReserveLP.sol#339-350):
        - (success,returndata) = target.call{value: value}(data) (StratReduceReserveLP.sol#348)
Low level call in Address.functionStaticCall(address,bytes,string) (StratReduceReserveLP.sol#368-377):
        - (success,returndata) = target.staticcall(data) (StratReduceReserveLP.sol#375)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IUniswapV2Router01.WETH() (StratReduceReserveLP.sol#14) is not in mixedCase
Constant WethUtils.weth (StratReduceReserveLP.sol#529) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter StratReduceReserveLP.reduceReserve(uint256,uint256)._amount (StratReduceReserveLP.sol#657) is not in mixedCase
Parameter StratReduceReserveLP.reduceReserve(uint256,uint256)._minYTokenAmount (StratReduceReserveLP.sol#657) is not in mixedC
ase
Parameter StratReduceReserveLP.swap(uint256,uint256)._wethToSwap (StratReduceReserveLP.sol#681) is not in mixedCase
Parameter StratReduceReserveLP.swap(uint256,uint256)._minYTokenOut (StratReduceReserveLP.sol#681) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (StratReduceReserveLP.sol#556)" inContext (StratReduceReserveLP.sol#550-559)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Stra
tReduceReserveLP.sol#19) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,add
ress,uint256).amountBDesired (StratReduceReserveLP.sol#20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
```

```
StratReduceReserveLP.lp (StratReduceReserveLP.sol#633) should be immutable
StratReduceReserveLP.swapRouter (StratReduceReserveLP.sol#634) should be immutable
StratReduceReserveLP.treasury (StratReduceReserveLP.sol#630) should be immutable
StratReduceReserveLP.yToken (StratReduceReserveLP.sol#631) should be immutable
StratReduceReserveLP.yTokenFund (StratReduceReserveLP.sol#632) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
StratReduceReserveLP.sol analyzed (11 contracts with 84 detectors), 35 result(s) found
```

Solidity Static Analysis

Pool.sol

Security

Check-effects-interaction:

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

more

Pos: 196:4:

Gas & Economy

Gas costs:

Gas requirement of function Pool.refreshCollateralRatio 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: 196:4:

Miscellaneous

Constant/View/Pure functions:

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

more

Pos: 6:4:

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: 465:8:

SwapStrategyPOL.sol

Security

Check-effects-interaction:

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

more

Pos: 55:4:

Gas & Economy

Gas costs:

Gas requirement of function SwapStrategyPOL.execute 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: 55:4:

Miscellaneous

Constant/View/Pure functions:

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

more

Pos: 6:4:

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: 128:8:

DaoChef.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in DaoChef.updatePool(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 81:4:

Gas & Economy

Gas costs:

Gas requirement of function DaoChef.harvest 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: 161:4:

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: 394:8:

DaoStaking.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in DaoStaking.getReward(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 355:4:

Gas & Economy

Gas costs:

Gas requirement of function DaoStaking.earnedBalances 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: 193:4:

Delete dynamic array:

The "delete" operation when applied to a dynamically sized array in Solidity generates code to delete each of the elements contained. If the array is large, this operation can surpass the block gas limit and raise an OOG exception. Also nested dynamically sized objects can produce the same results.
more
Pos: 398:12:

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: 482:8:

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: 460:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 329:24:

DaoZapMMSwap.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: 144:94:

Gas & Economy

Gas costs:

Gas requirement of function DaoZapMMSwap.removeZap 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: 195:4:

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: 196:8:

NFTController.sol

Gas & Economy

Gas costs:

Gas requirement of function NFTController.getBoostRate 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: 55:4:

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: 37:8:

DevFund.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: 44:31:

Gas & Economy

Gas costs:

Gas requirement of function DevFund.currentBalance 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: 30:4:

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: 55:8:

EcosystemFund.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: 38:12:

Gas & Economy

Gas costs:

Gas requirement of function EcosystemFund.claimable 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: 47:4:

Miscellaneous

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 44:15:

Fund.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: 38:12:

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: 55:8:

Reserve.sol

Miscellaneous

Constant/View/Pure functions:

Reserve.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 45:4:

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: 21:8:

MasterOracle.sol

Gas & Economy

Gas costs:

Gas requirement of function MasterOracle.getXTokenPrice 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: 33:4:

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: 26:8:

UniswapPairOracle.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: 112:22:

Gas & Economy

Gas costs:

Gas requirement of function UniswapPairOracle.spot 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: 93:4:

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: 86:12:

XToken.sol

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: 33:8:

IMT.sol

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: 54:8:

COREX.sol

Gas & Economy

Gas costs:

Gas requirement of function COREX.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: 40:4:

Miscellaneous

Constant/View/Pure functions:

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

more

Pos: 39:4:

DaoTreasury.sol

Security

Check-effects-interaction:

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

more

Pos: 77:4:

Gas & Economy

Gas costs:

Gas requirement of function DaoTreasury.removeStrategy 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: 60:4:

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: 64:8:

Miscellaneous

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 62:8:

StratRecollateralize.sol

Security

Check-effects-interaction:

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

more

Pos: 26:4:

Gas & Economy

Gas costs:

Gas requirement of function StratRecollateralize.recollateralize 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: 26:4:

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: 28:8:

StratReduceReserveLP.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: 68:127:

Gas & Economy

Gas costs:

Gas requirement of function StratReduceReserveLP.reduceReserve 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: 42:4:

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: 44:8:

Solhint Linter

Pool.sol

```
Pool.sol:3:1: Error: Compiler version 0.8.4 does not satisfy the r
semver requirement
Pool.sol:19:1: Error: Contract has 24 states declarations but allowed
no more than 15
Pool.sol:77:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
Pool.sol:198:17: Error: Avoid to make time-based decisions in your
business logic
Pool.sol:220:34: Error: Avoid to make time-based decisions in your
business logic
Pool.sol:225:32: Error: Code contains empty blocks
```

SwapStrategyPOL.sol

```
SwapStrategyPOL.sol:3:1: Error: Compiler version 0.8.4 does not
satisfy the r semver requirement
SwapStrategyPOL.sol:30:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
SwapStrategyPOL.sol:83:9: Error: Variable name must be in mixedCase
SwapStrategyPOL.sol:84:9: Error: Variable name must be in mixedCase
SwapStrategyPOL.sol:92:127: Error: Avoid to make time-based decisions
in your business logic
SwapStrategyPOL.sol:116:17: Error: Avoid to make time-based decisions
in your business logic
```

DaoChef.sol

```
DaoChef.sol:3:1: Error: Compiler version 0.8.4 does not satisfy the r
semver requirement
DaoChef.sol:70:13: Error: Avoid to make time-based decisions in your
business logic
DaoChef.sol:71:28: Error: Avoid to make time-based decisions in your
business logic
DaoChef.sol:83:13: Error: Avoid to make time-based decisions in your
business logic
DaoChef.sol:86:32: Error: Avoid to make time-based decisions in your
business logic
DaoChef.sol:90:35: Error: Avoid to make time-based decisions in your
business logic
DaoChef.sol:274:73: Error: Avoid to make time-based decisions in your
business logic
```

DaoStaking.sol

```
DaoStaking.sol:3:1: Error: Compiler version 0.8.4 does not satisfy
the r semver requirement
DaoStaking.sol:15:1: Error: Contract has 16 states declarations but
allowed no more than 15DaoStaking.sol:54:29: Error: Constant name
must be in capitalized SNAKE_CASE
DaoStaking.sol:81:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
DaoStaking.sol:97:52: Error: Avoid to make time-based decisions in
your business logic
DaoStaking.sol:98:50: Error: Avoid to make time-based decisions in
your business logic
DaoStaking.sol:424:52: Error: Avoid to make time-based decisions in
your business logic
DaoStaking.sol:425:50: Error: Avoid to make time-based decisions in
your business logic
DaoStaking.sol:495:32: Error: Code contains empty blocks
DaoStaking.sol:507:5: Error: Event name must be in CamelCase
DaoStaking.sol:508:5: Error: Event name must be in CamelCase
```

DaoZapMMSwap.sol

```
DaoZapMMSwap.sol:3:1: Error: Compiler version 0.8.4 does not satisfy
the r semver requirement
DaoZapMMSwap.sol:35:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
DaoZapMMSwap.sol:80:13: Error: Avoid to make time-based decisions in
your business logic
DaoZapMMSwap.sol:102:32: Error: Code contains empty blocks
DaoZapMMSwap.sol:144:95: Error: Avoid to make time-based decisions in
your business logic
DaoZapMMSwap.sol:169:9: Error: Variable name must be in mixedCase
DaoZapMMSwap.sol:170:9: Error: Variable name must be in mixedCase
```

NFTController.sol

```
NFTController.sol:3:1: Error: Compiler version 0.8.4 does not satisfy
the r semver requirement
NFTController.sol:14:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
NFTController.sol:14:20: Error: Code contains empty blocks
NFTController.sol:48:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
NFTController.sol:48:19: Error: Code contains empty blocks
```

DevFund.sol

```
DevFund.sol:3:1: Error: Compiler version 0.8.4 does not satisfy the r
semver requirement
```

EcosystemFund.sol

```
EcosystemFund.sol:3:1: Error: Compiler version 0.8.4 does not satisfy
the r semver requirement
```

Reserve.sol

```
Reserve.sol:3:1: Error: Compiler version 0.8.4 does not satisfy the r
semver requirement
```

Fund.sol

```
Fund.sol:3:1: Error: Compiler version 0.8.4 does not satisfy the r
semver requirement
Fund.sol:38:13: Error: Avoid to make time-based decisions in your
business logic
Fund.sol:41:13: Error: Avoid to make time-based decisions in your
business logic
Fund.sol:44:32: Error: Avoid to make time-based decisions in your
business logic
```

MasterOracle.sol

```
MasterOracle.sol:3:1: Error: Compiler version 0.8.4 does not satisfy
the r semver requirement
MasterOracle.sol:17:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

UniswapPairOracle.sol

```
UniswapPairOracle.sol:54:18: Error: Parse error: missing ';' at '{'
UniswapPairOracle.sol:133:18: Error: Parse error: missing ';' at '{'
```

XToken.sol

```
XToken.sol:3:1: Error: Compiler version 0.8.4 does not satisfy the r
semver requirement
XToken.sol:21:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
XToken.sol:21:83: Error: Code contains empty blocks
```

YToken.sol

```
YToken.sol:3:1: Error: Compiler version 0.8.4 does not satisfy the r
semver requirement
YToken.sol:8:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
YToken.sol:8:83: Error: Code contains empty blocks
```

COREX.sol

```
COREX.sol:3:1: Error: Compiler version 0.8.4 does not satisfy the r
semver requirement
COREX.sol:14:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
COREX.sol:19:5: Error: Function name must be in mixedCase
```

IMT.sol

```
IMT.sol:3:1: Error: Compiler version 0.8.4 does not satisfy the r
semver requirement
IMT.sol:14:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
IMT.sol:29:5: Error: Function name must be in mixedCase
```

StratRecollateralize.sol

```
StratRecollateralize.sol:3:1: Error: Compiler version 0.8.4 does not
satisfy the r semver requirement
StratRecollateralize.sol:19:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
StratRecollateralize.sol:39:32: Error: Code contains empty blocks
```

StratReduceReserveLP.sol

```
StratReduceReserveLP.sol:3:1: Error: Compiler version 0.8.4 does not
satisfy the r semver requirement
StratReduceReserveLP.sol:22:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
StratReduceReserveLP.sol:51:108: Error: Avoid to make time-based
decisions in your business logic
StratReduceReserveLP.sol:68:128: Error: Avoid to make time-based
decisions in your business logic
```

DaoTreasury.sol

```
DaoTreasury.sol:3:1: Error: Compiler version 0.8.4 does not satisfy
the r semver requirement
DaoTreasury.sol:23:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

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