Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Sahara DAO Protocol
Website: saharadao.finance
Platform: Cronos Chain
Language: Solidity
Date: May 5th, 2022

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 Sahara DAO to perform the Security audit of the Sahara DAO 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 May 5th, 2022.

The purpose of this audit was to address the following:

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

Project Background

Sahara DAO Contracts have functions like mint, redeem, recollateralize, addLiquidity, add, set, withdraw, stake, setRewarder, getYTokenPrice, maxTotalSupply, etc. The Sahara DAO contract inherits the ERC20, SafeERC20, Ownable, ReentrancyGuard, Address, IUniswapV2Router02, SafeMath, Math, Initializable, IERC20, IUniswapV2Pair, ERC20Burnable standard smart contracts from the OpenZeppelin library. These OpenZeppelin 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 Sahara DAO Protocol Smart Contracts
Platform	**Cronos / Solidity**
File 1	Pool.sol
File 1 MD5 Hash	E39F2C63B9F2B7DF9221FAA8CCDF7C75
Updated File 1 MD5 Hash	BD5C77866255FA38D7E073BFD6A90141
File 2	SwapStrategyPOL.sol
File 2 MD5 Hash	1249FB016B7C21CAF703BC2578F27779
Updated File 2 MD5 Hash	16EB0491A7FCCCA0C0E74B589698C13C
File 3	SaharaDaoChef.sol
File 3 MD5 Hash	02321B441379C7C67FD26467057412FD
File 4	SaharaDaoStaking.sol
File 4 MD5 Hash	0A6662EB713D5C5F43F359435568E419
Updated File 4 MD5 Hash	72B0C357D32B9976F3F5BCC4A446EC2D
File 5	SaharaDaoZapMMSwap.sol
File 5 MD5 Hash	381F4A7BFEAF3253D098412BD2E9EEA0
File 6	Fund.sol
File 6 MD5 Hash	A37372AC87DD651C420E505B52A70E88
File 7	MNGDaoFund.sol
File 7 MD5 Hash	911326C418887646F57EA59F56E02BBC
File 8	MNGDevFund.sol
File 8 MD5 Hash	6657AE95F3E95CFF955BF4620F9B9730
File 9	MNGReserve.sol
File 9 MD5 Hash	1AF612E73BBAD7E84B752FE5AFCDD66E
File 10	MNGTreasuryFund.sol

File 10 MD5 Hash	BBB52629F52EA8A67CC5A6F56C4A606D
File 11	MockERC20.sol
File 11 MD5 Hash	94278D4A01D92E76EBDE914556B3A6A0
File 12	MockTreasury.sol
File 12 MD5 Hash	EAB3F68107BE7B69CAFA290A0FD6FE83
File 13	MasterOracle.sol
File 13 MD5 Hash	26FFB8A6EB84AABF384A830DB4572C0A
File 14	UniswapPairOracle.sol
File 14 MD5 Hash	37801A23DE6F4571ADD278A4A062C1D5
File 15	XToken.sol
File 15 MD5 Hash	E905290FA8FFB182588943AA4D60EAC6
File 16	YToken.sol
File 16 MD5 Hash	FFA9BDAB9AEE9D07DB46CB3A23A34696
File 17	MMFX.sol
File 17 MD5 Hash	C0CF1CBCC02763696123A46D401557D5
Updated File 17 MD5 Hash	664C0017F4BF8498B957DB667ED68580
File 18	MNG.sol
File 18 MD5 Hash	D81DB17DEBEF1FDC4B7D1AF9441E5F57
Updated File 18 MD5 Hash	98BA05DE9BE689E4DE0C775A96137717
File 19	SaharaDaoTreasury.sol
File 19 MD5 Hash	0179F91AA5432801AB18BB46B9CA3D07
File 20	StratRecollateralize.sol
File 20 MD5 Hash	BD1D0DE6A225D1268BD7BAA040B7CA3A
File 21 MD5 Hash	F1600CDDAD4A8AB6F42455417FEA97CE
File 21	StratReduceReserveLP.sol
File 21 MD5 Hash	DF023B3B9F8F23225BD08DA03ADC2255
Updated File 21 MD5 Hash	19390D3837AF290D881A7711AA462465

File 22	Timelock.sol
File 22 MD5 Hash	94F559046B7CB4335EE0F49341A23DA0
Audit Date	May 5th,2022
Revise Audit Date	May 9th,2022

Claimed Smart Contract Features

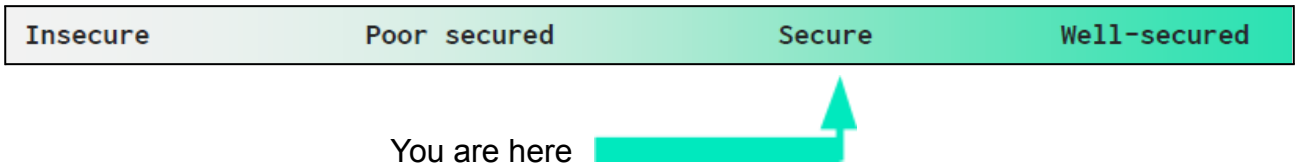
Claimed Feature Detail	Our Observation
File 1 Pool.sol • Refresh Cooldown: 1 hour • Ratio StepUp: 0.2% • Ratio StepDown: 0.1% • Price Target: 1 • Price Band: 0.005 • YToken Slippage: 20% • Redemption Fee: 0.5% • Redemption Fee Maximum: 0.9% • Minting Fee: 0.3% • Minting Fee Maximum:0.5%	**YES, This is 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%	**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 SaharaDaoChef.sol • Maximum Number Of Pools: 36 • Maximum Reward: 1 token per second	**YES, This is valid.**
File 4 SaharaDaoStaking.sol • Rewards Duration: 1 week • Lock Duration: 8 weeks • Team Rewards: 20% • Maximum Team Rewards: 20%	**YES, This is valid.**
File 5 SaharaDaoZapMMSwap.sol • SaharaDaoZapMMSwap has functions like: zap, swap, doSwapETH, etc	**YES, This is valid.**

File 6 Fund.sol • Fund has functions like: allocation, initialization, vestedBalance, claimable, etc.	**YES, This is valid.**
File 7 MNGDaoFund.sol • Allocation: 10% • Vesting Duration: 3 Years	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**
File 8 MNGDevFund.sol • Allocation: 10% • Vesting Duration: 2 Years	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**
File 9 MNGReserve.sol • MNGReserve has functions like: initialize, setRewarder, setPool, transfer.	**YES, This is valid.**
File 10 MNGTreasuryFund.sol • Allocation: 10% • Vesting Duration: 3 Years	**YES, This is valid.**
File 11 MockERC20.sol • MockERC20 has functions like: mint, decimals.	**YES, This is valid.**
File 12 MockTreasury.sol • MockTreasury has functions like: mock, info.	**YES, This is valid.**
File 13 MasterOracle.sol • MasterOracle has functions like: getXTokenPrice, getYTokenPrice,	**YES, This is valid.**

getYTokenTWAP, etc.	
File 14 UniswapPairOracle.sol • Period: 60-minute TWAP (Time-Weighted Average Price) • Maximum Period: 48 Hours • Minimum Period: 10 Minutes • Leniency: 12 Hours	**YES, This is valid.**
File 15 XToken.sol • XToken has functions like: setMinter, mint.	**YES, This is valid.**
File 16 YToken.sol • The YToken contract inherits the ERC20Burnable standard smart contracts from the OpenZeppelin library.	**YES, This is valid.**
File 17 MMFX.sol • Genesis Supply: 100	**YES, This is valid.**
File 18 MNG.sol • The MNG contract inherits the YToken contract.	**YES, This is valid.**
File 19 SaharaDaoTreasury.sol • SaharaDaoTreasury has functions like: balanceOf, requestFund, etc.	**YES, This is valid.**
File 20 StratRecollateralize.sol • StratRecollateralize has functions like: recollateralize, etc.	**YES, This is valid.**
File 21 StratReduceReserveLP.sol • StratReduceReserveLP has functions like: reduceReserve, swap.	**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 3 low and some very low level issues.

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

Technical Quick Stats

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

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 21 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 Sahara DAO 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 Sahara DAO Protocol.

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

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

Documentation

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

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

Another source of information was its official website https://saharadao.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	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	nonReentrant	modifier	Passed	No Issue
8	info	external	Passed	No Issue
9	usableCollateralBalance	read	Passed	No Issue
10	calcMint	read	Passed	No Issue
11	calcRedeem	read	Passed	No Issue
12	calcExcessCollateralBalance	read	Passed	No Issue
13	refreshCollateralRatio	read	Passed	No Issue
14	mint	external	Passed	No Issue
15	redeem	external	Passed	No Issue
16	collect	external	Passed	No Issue
17	recollateralize	external	Passed	No Issue
18	checkPriceFluctuation	internal	Passed	No Issue
19	toggle	write	access only Owner	No Issue
20	setCollateralRatioOptions	write	access only Owner	No Issue
21	toggleCollateralRatio	write	access only Owner	No Issue
22	setFees	write	access only Owner	No Issue
23	setMinCollateralRatio	external	access only Owner	No Issue
24	reduceExcessCollateral	external	access only Owner	No Issue
25	setSwapStrategy	external	access only Owner	No Issue
26	setOracle	external	access only Owner	No Issue
27	setYTokenSlippage	external	access only Owner	No Issue
28	setTreasury	external	Function access control lacks management	Refer Audit Findings
29	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	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	lpBalance	read	Passed	No Issue
8	execute	external	Passed	No Issue
9	swap	internal	Passed	No Issue
10	addLiquidity	internal	Passed	No Issue
11	cleanDust	external	access only Owner	No Issue
12	changeSlippage	external	access only Owner	No Issue

SaharaDaoChef.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	poolLength	read	Passed	No Issue
8	pendingReward	external	Passed	No Issue
9	updatePool	write	Passed	No Issue
10	massUpdatePools	write	Passed	No Issue
11	deposit	write	Passed	No Issue
12	withdraw	write	Passed	No Issue
13	harvest	write	Passed	No Issue
14	withdrawAndHarvest	write	Passed	No Issue
15	emergencyWithdraw	write	Passed	No Issue
16	harvestAllRewards	external	Passed	No Issue
17	checkPoolDuplicate	internal	Passed	No Issue
18	add	write	access only Owner	No Issue
19	set	write	access only Owner	No Issue
20	setRewardPerSecond	write	access only Owner	No Issue
21	setRewardMinter	external	Passed	No Issue

SaharaDaoStaking.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue

6	_transferOwnership	internal	Passed	No Issue
7	addReward	write	Function input parameters lack of check	Refer Audit Findings
8	approveRewardDistributor	external	Function input parameters lack of check	Refer Audit Findings
9	_rewardPerToken	internal	Passed	No Issue
10	_earned	internal	Passed	No Issue
11	lastTimeRewardApplicable	read	Passed	No Issue
12	rewardPerToken	external	Passed	No Issue
13	getRewardForDuration	external	Passed	No Issue
14	claimableRewards	external	Passed	No Issue
15	totalBalance	external	Passed	No Issue
16	unlockedBalance	external	Passed	No Issue
17	earnedBalances	external	Passed	No Issue
18	lockedBalances	external	Passed	No Issue
19	withdrawableBalance	read	Passed	No Issue
20	stake	external	Passed	No Issue
21	mint	external	Passed	No Issue
22	withdraw	write	Passed	No Issue
23	getReward	write	Passed	No Issue
24	emergencyWithdraw	external	Critical operation lacks event log	Refer Audit Findings
25	withdrawExpiredLocks	external	Critical operation lacks event log	Refer Audit Findings
26	_notifyReward	internal	Passed	No Issue
27	notifyRewardAmount	external	Passed	No Issue
28	recoverERC20	external	access only Owner	No Issue
29	updateReward	modifier	Passed	No Issue
30	receive	external	Passed	No Issue

SaharaDaoZapMMSwap.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	zap	external	Passed	No Issue
8	receive	external	Passed	No Issue
9	swap	internal	access only Owner	No Issue
10	doSwapETH	internal	Passed	No Issue
11	approveToken	internal	Passed	No Issue

12	calculateSwapInAmount	internal	Passed	No Issue
13	addZap	external	access only Owner	No Issue
14	removeZap	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	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	nonReentrant	modifier	Passed	No Issue
8	initialize	external	initializer	No Issue
9	allocation	read	Passed	No Issue
10	vestingStart	read	Passed	No Issue
11	vestingDuration	read	Passed	No Issue
12	currentBalance	read	Passed	No Issue
13	vestedBalance	read	Passed	No Issue
14	claimable	read	Passed	No Issue
15	transfer	external	access only Owner	No Issue

MNGDaoFund.sol

Functions

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

MNGDevFund.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

MNGReserve.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	_setInitializedVersion	write	Passed	No Issue
7	initialize	external	Passed	No Issue
8	setRewarder	external	Passed	No Issue
9	setPool	external	Passed	No Issue
10	transfer	external	Passed	No Issue

MNGTreasuryFund.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

MockERC20.sol

Functions

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

MockTreasury.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	mock	write	Passed	No Issue
3	info	read	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	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	getXTokenPrice	read	Passed	No Issue
8	getYTokenPrice	read	Passed	No Issue
9	getXTokenTWAP	read	Passed	No Issue
10	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	burn	write	Passed	No Issue
3	burnFrom	write	Passed	No Issue
4	onlyMinter	modifier	Passed	No Issue
5	setMinter	external	Passed	No Issue
6	mint	external	access only Minter	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
4	maxTotalSupply	internal	Passed	No Issue

MNG.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	maxTotalSupply	internal	Passed	No Issue

MMFX.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyMinter	modifier	Passed	No Issue
3	setMinter	external	Passed	No Issue
4	mint	external	Unlimited Minting	Refer Audit Findings

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

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

SaharaDaoTreasury.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue

4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	requestFund	external	Passed	No Issue
9	addStrategy	external	access only Owner	No Issue
10	removeStrategy	external	access only Owner	No Issue
11	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) Critical operation lacks event log: **SaharaDaoStaking.sol**

Missing event log for:

1. withdrawExpiredLocks
2. emergencyWithdraw.

Resolution: Write an event log for listed events.

(2) Function input parameters lack of check: **SaharaDaoStaking.sol**

Variable validation is not performed in the functions below :

1. addReward
2. approveRewardDistributor.

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

(3) Function access control lacks management: **Pool.sol**

The Treasury address is used to transfer fees. The treasury address can be set only once but anyone can execute the setTreasury function.

Resolution: The owner has to make sure to set treasury before anyone sets it.

Status: Acknowledged.

Very Low / Informational / Best practices:

(1) Unlimited Minting: **MMFX.sol**

Minter can mint unlimited tokens.

Resolution: We suggest putting a minting limit.

(2) SPDX license identifier Missing: **MockTreasury.sol**

SPDX license identifier not provided in source file.

Resolution: We suggest adding an SPDX license identifier.

(3) HardCoded address: **WethUtils.sol**

```
IWETH public constant weth = IWETH(0x5C7F8A570d578ED84E63fdFA7b1eE72dEae1AE23); //WCRO
// IERC20 public constant MMF = IERC20(0x97749c9B61F878a880DfE312d2594AE07AEd7656); //MMF token
IERC20 public constant MMF = IERC20(0xC6C2300A9bbD4181c728Ba60E7D9b738052Ae1BB); //MMF Test token
```

These addresses have been set to static addresses and cannot be changed after deploying.

Resolution: We suggest that the deployer should confirm 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:

- toggle: Pool owner can turn on / off minting and redemption.
- setCollateralRatioOptions: Pool owner can configure variables related to Collateral Ratio.
- toggleCollateralRatio: Pool owner can pause or unpause collateral ratio updates.
- setFees: Pool owners can set the protocol fees.
- setMinCollateralRatio: Pool owners can set the minimum Collateral Ratio.
- reduceExcessCollateral: Pool owners can transfer the excess balance of WETH to FeeReserve.
- setSwapStrategy: Pool owner can set the address of Swapper utils.
- setOracle: Pool owner can set new oracle address.
- setYTokenSlippage: Pool owner can set yTokenSlipage.
- cleanDust: SwapStrategyPOL owner can clean dust.
- changeSlippage: SwapStrategyPOL owner can change slippage value.
- add: SaharaDaoChef owner can add a new LP to the pool.
- set: SaharaDaoChef owner can update the given pool's reward allocation point and `IRewarder` contract
- setRewardPerSecond: SaharaDaoChef owner can set the reward per second to be distributed.
- addReward: SaharaDaoStaking can add a new reward token to be distributed to stakers.
- approveRewardDistributor: SaharaDaoStaking can modify approval for an address to call notifyRewardAmount.
- recoverERC20: SaharaDaoStaking can be added to support recovering LP Rewards from other systems such as BAL to be distributed to holders.
- setTeamWalletAddress: SaharaDaoStaking owner can set team wallet address.
- setTeamRewardPercent: SaharaDaoStaking owner can set team reward percentage.
- addZap: SaharaDaoZapMMSwap owner can add new zap configuration.

- removeZap: SaharaDaoZapMMSwap owner can Deactivate a Zap configuration.
- transfer: Fund owners can transfer amounts.
- setPeriod: UniswapPairOracle owner can set the period.
- addStrategy: SaharaDaoTreasury owner can add new strategy.
- removeStrategy: SaharaDaoTreasury owner can remove the current strategy.
- allocateFee: SaharaDaoTreasury owner can allocate protocol's fee to stakers.
- recollateralize: StratRecollateralize owner recollateralize the minting pool.
- reduceReserve: StratReduceReserveLP 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 issues in the smart contracts, but they were resolved in the revised smart contract code. **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.

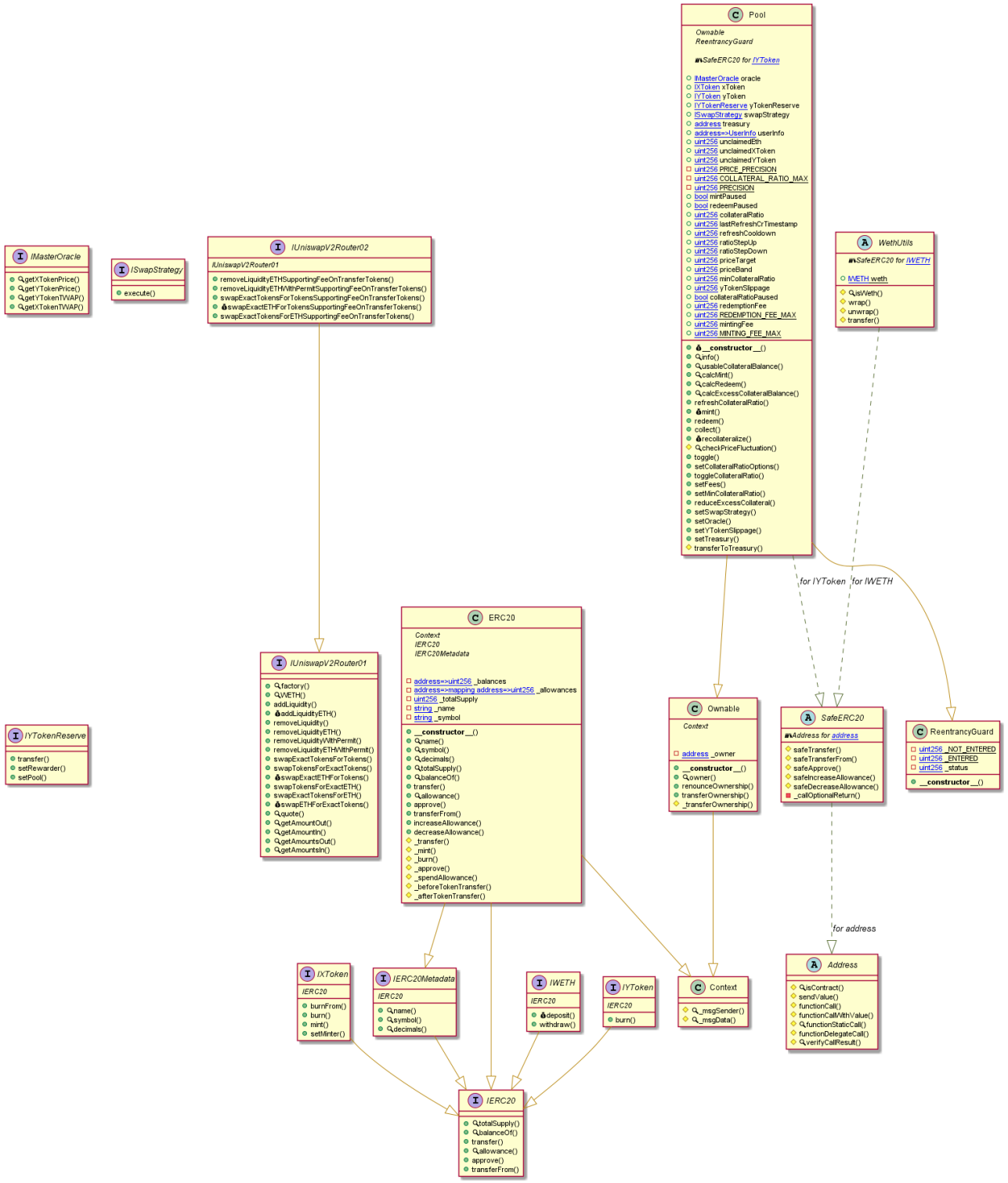
Technical Disclaimer

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.

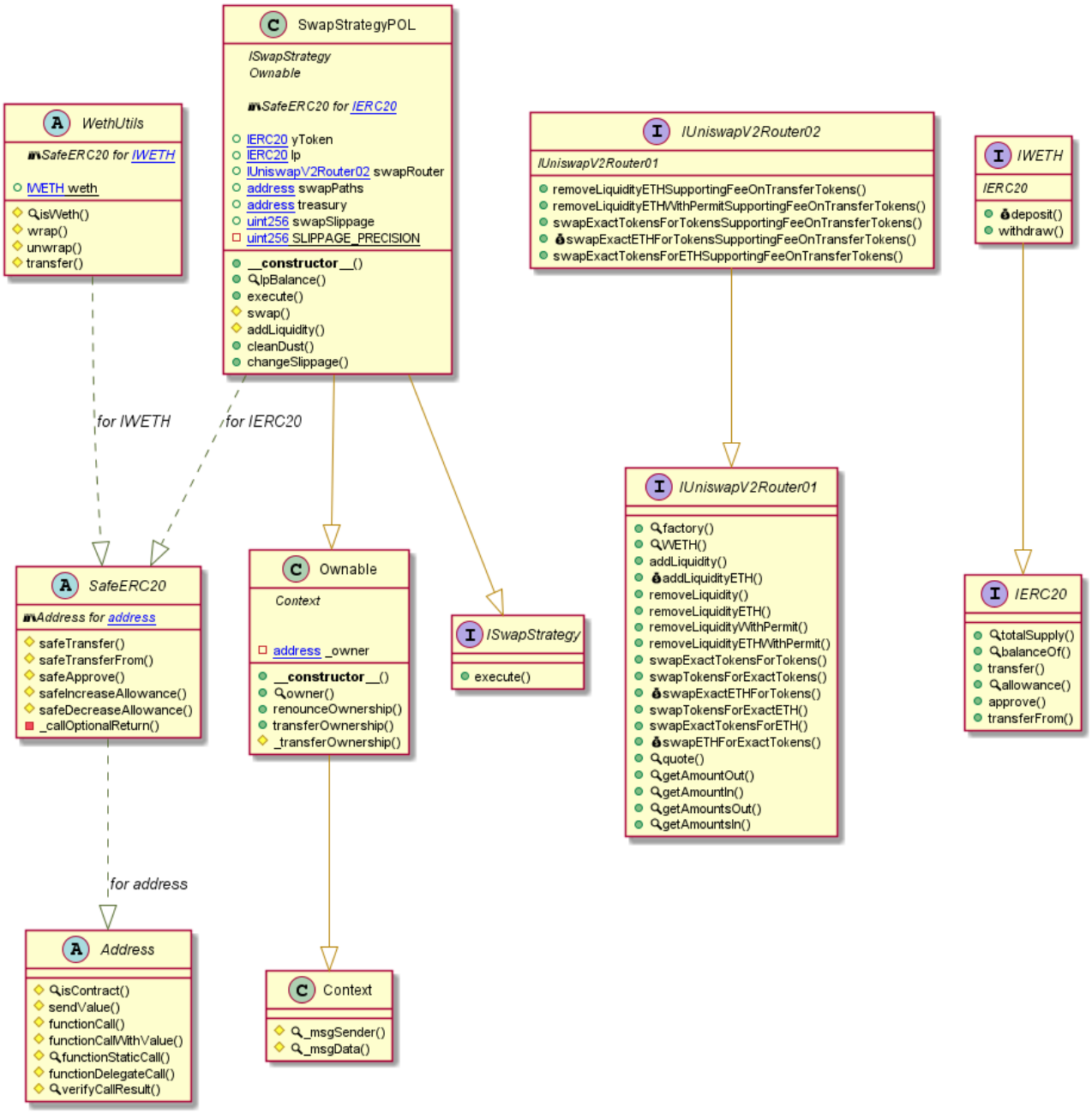
Appendix

Code Flow Diagram - Sahara DAO Protocol

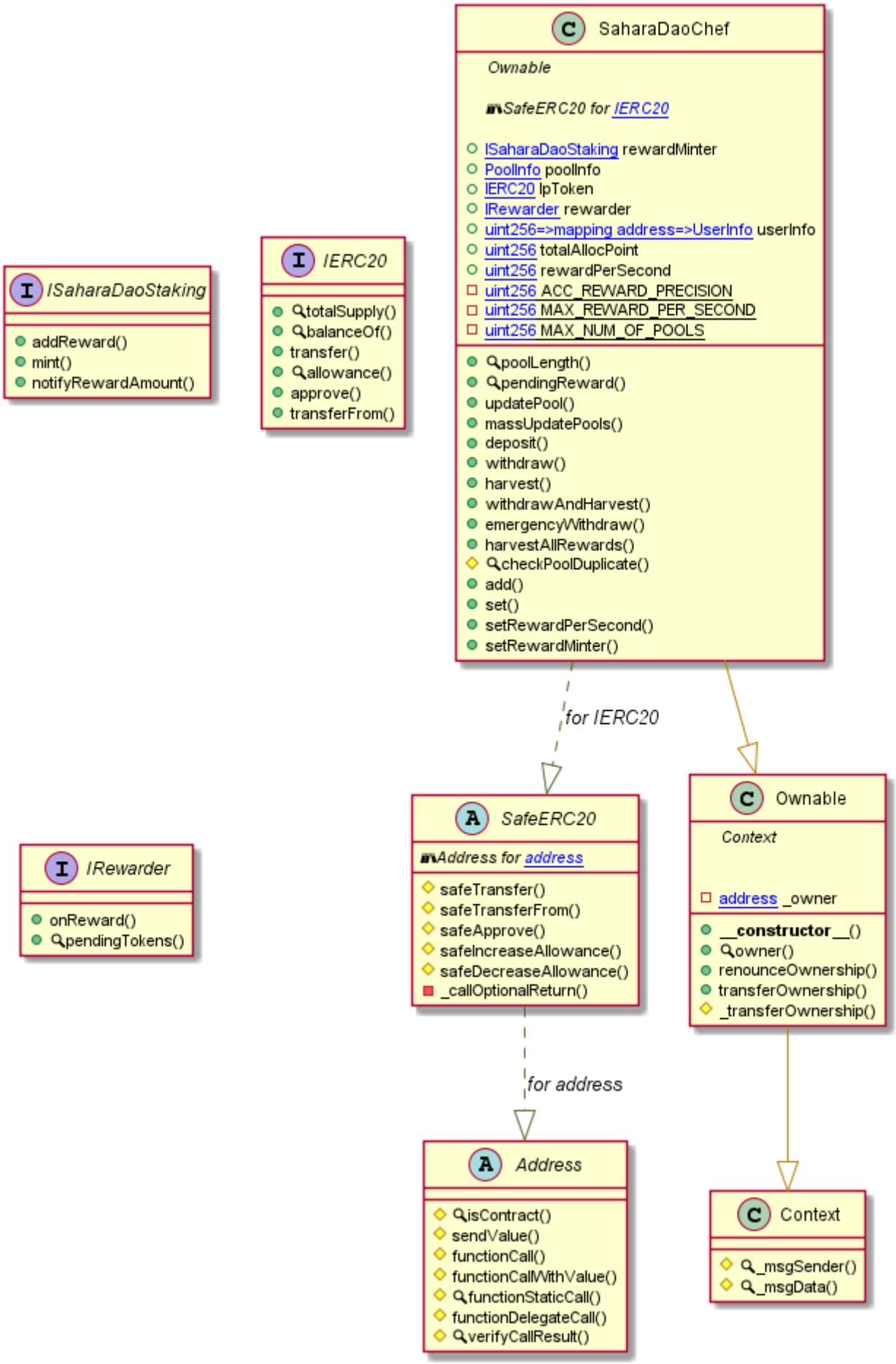
Pool Diagram



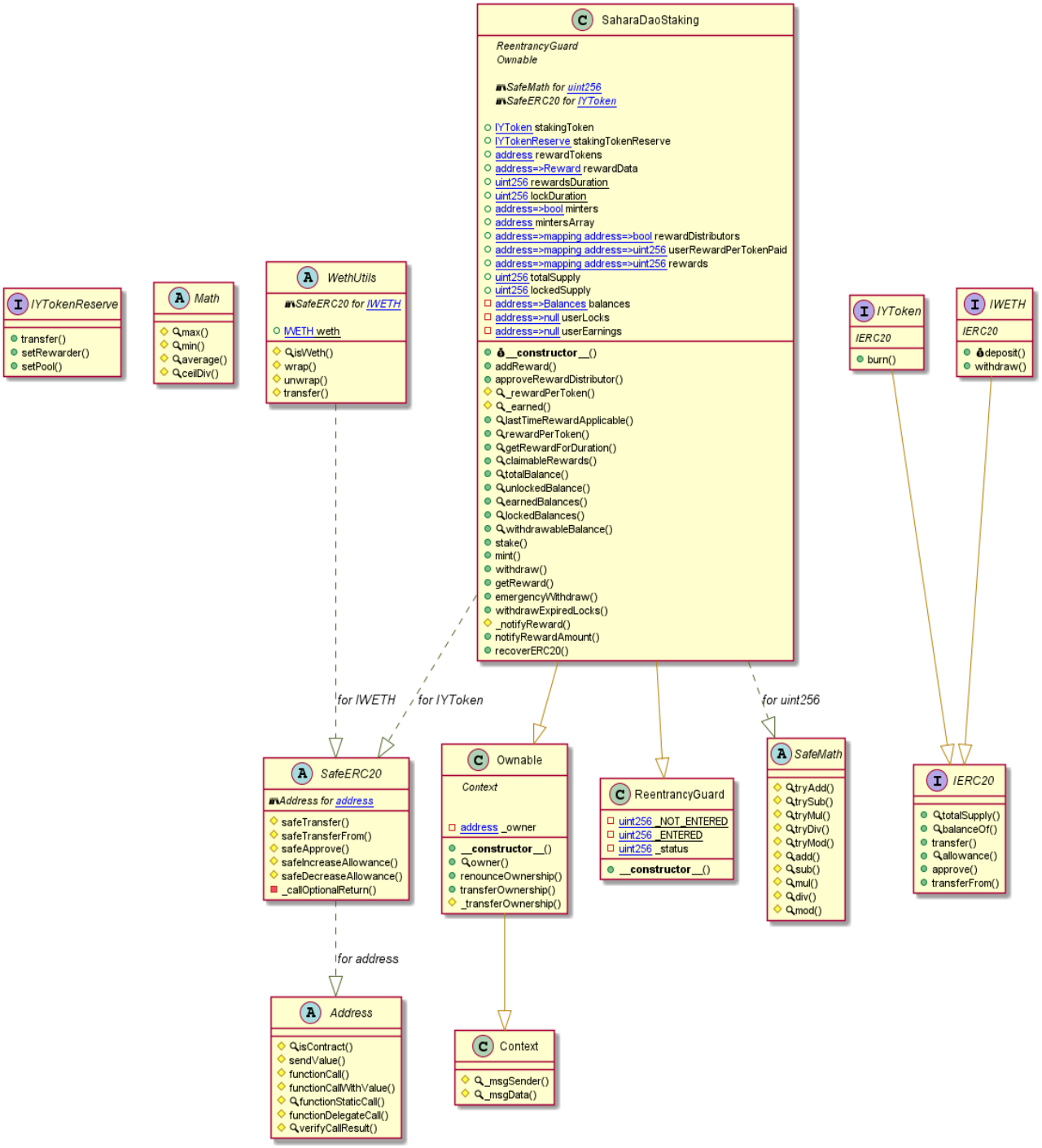
SwapStrategyPOL Diagram



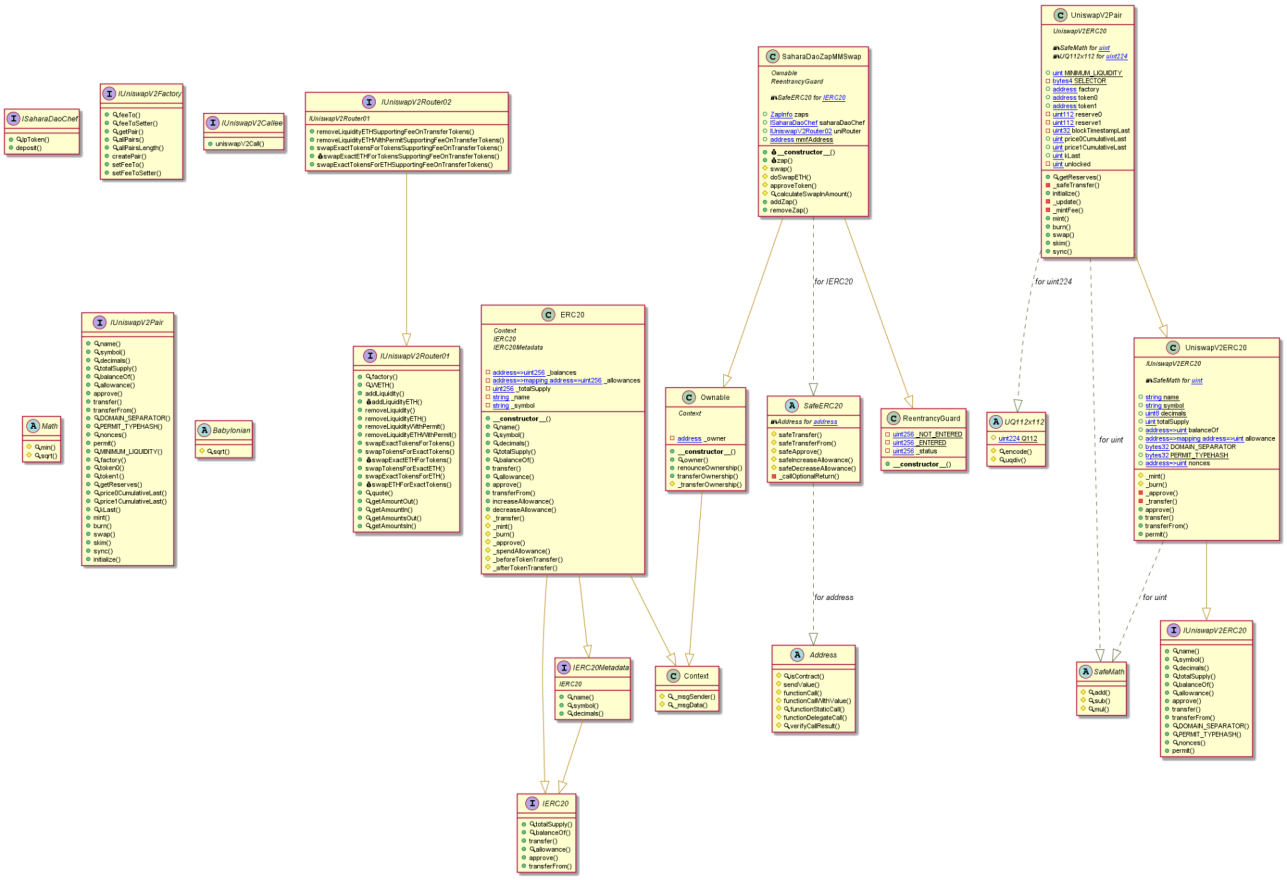
SaharaDaoChef Diagram



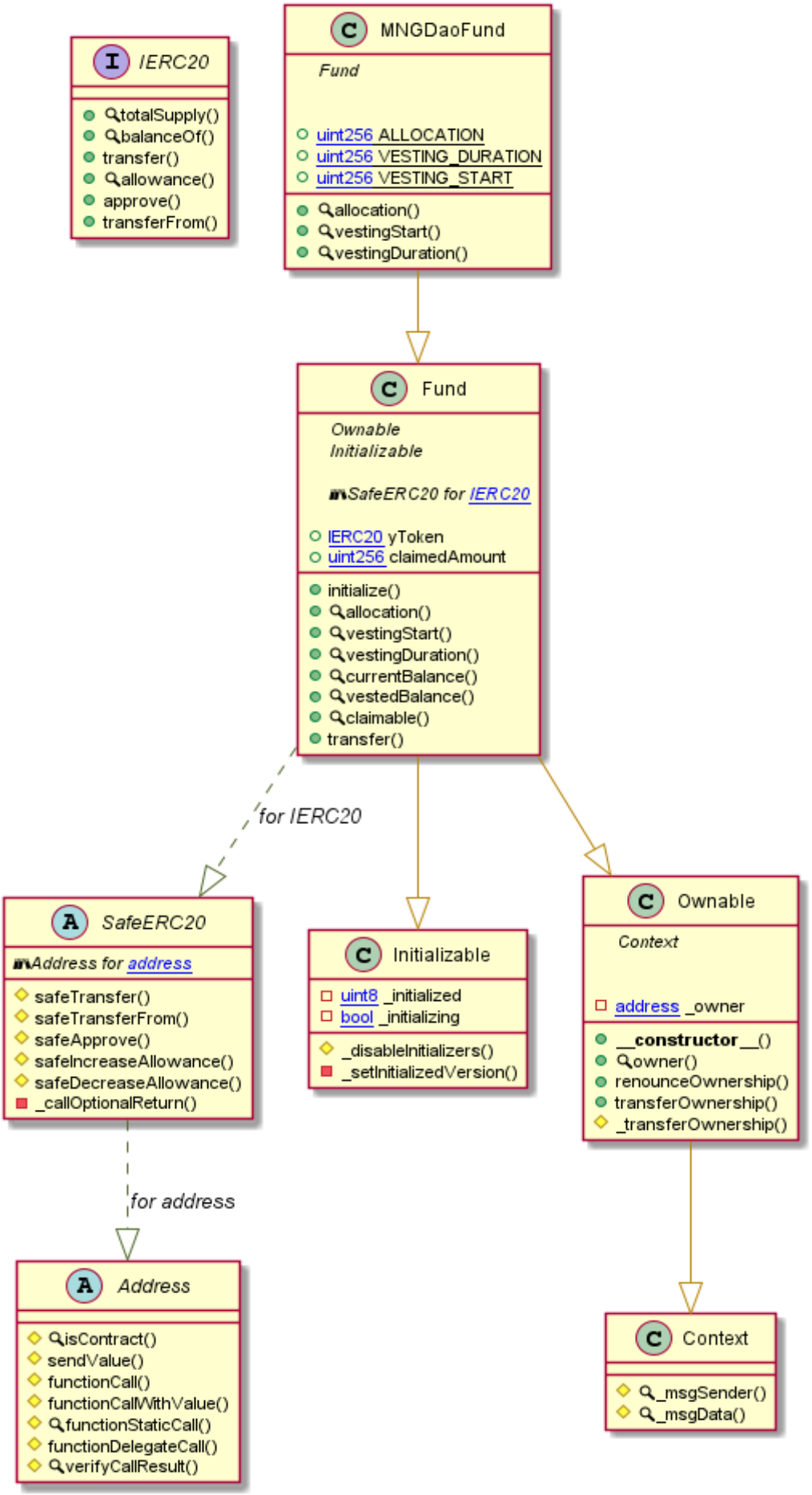
SaharaDaoStaking Diagram



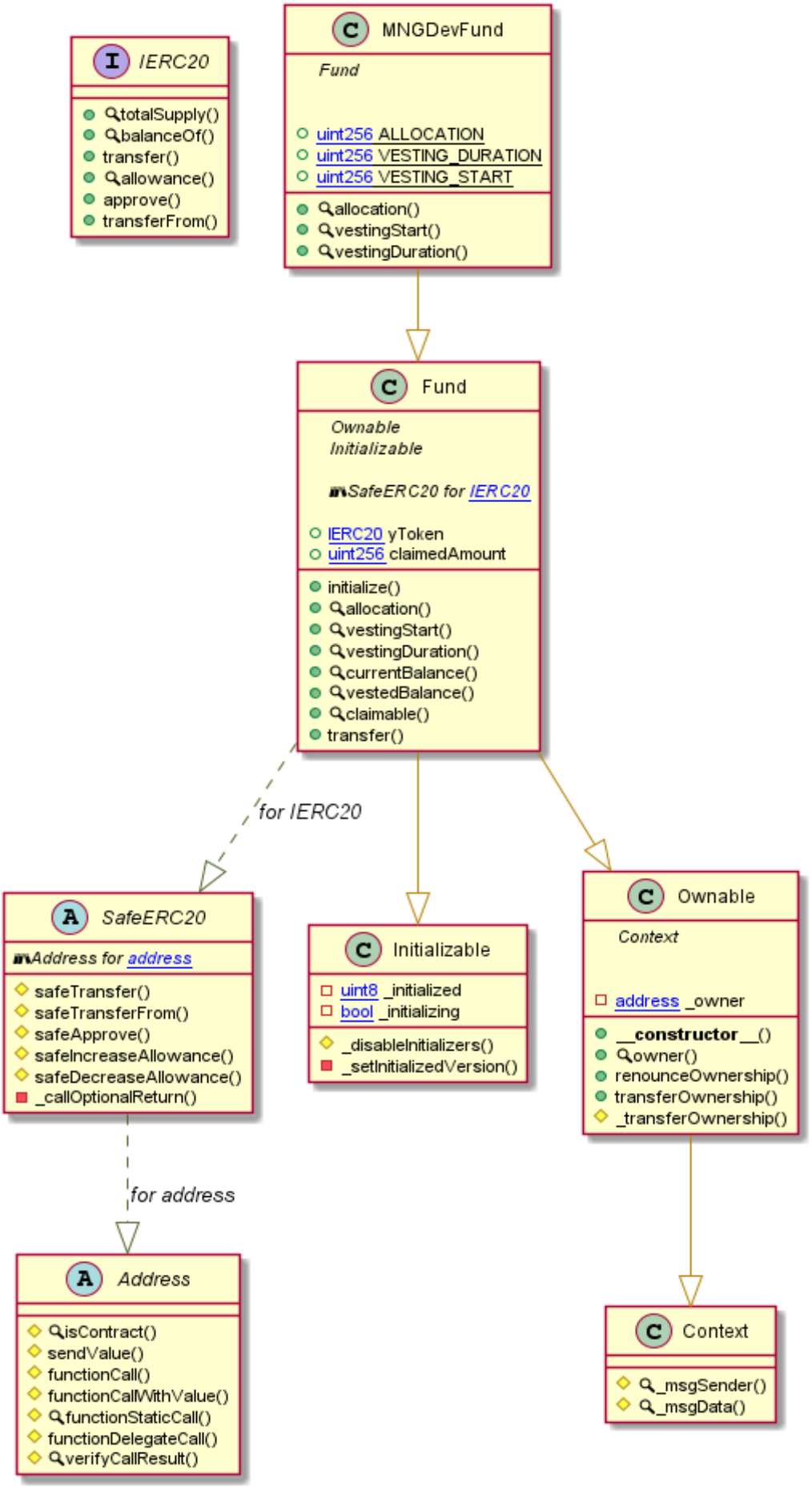
SaharaDaoZapMMSwap Diagram



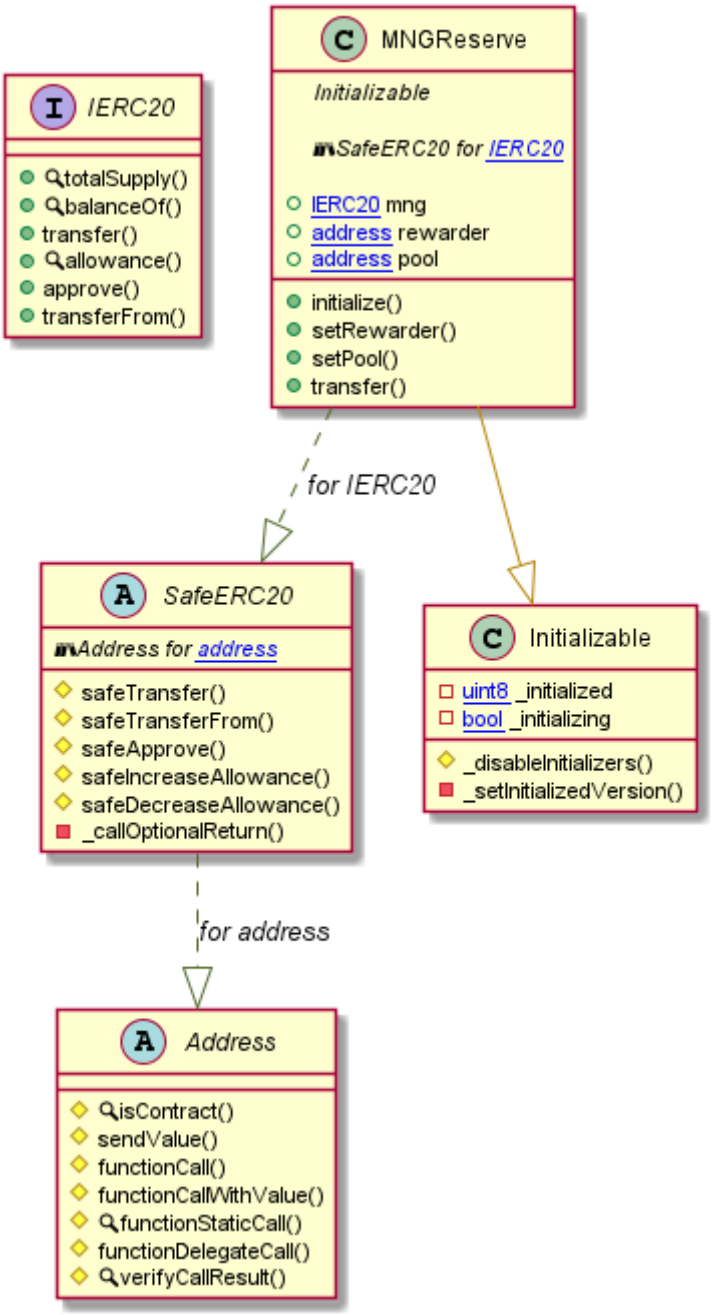
MNGDaoFund Diagram



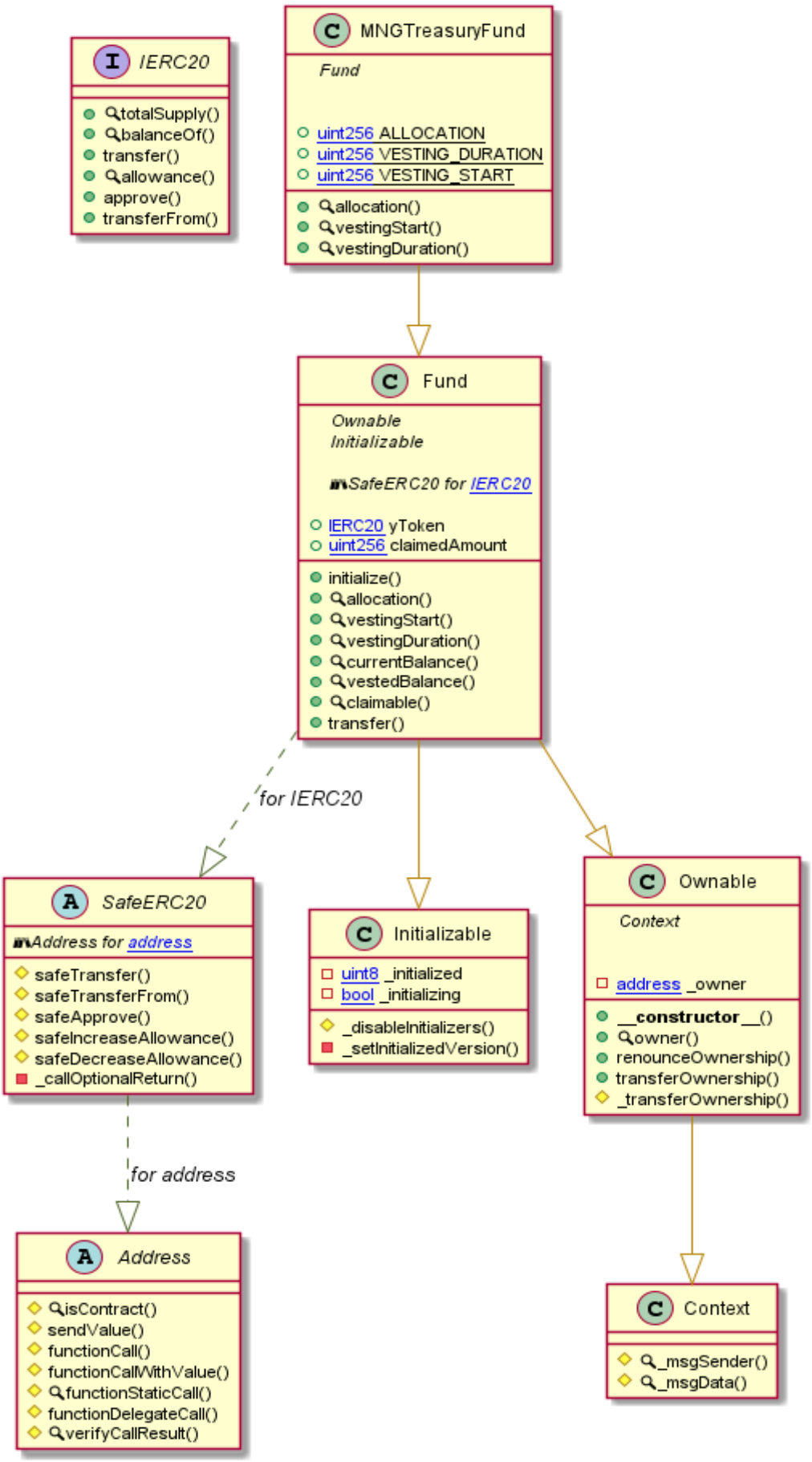
MNGDevFund Diagram



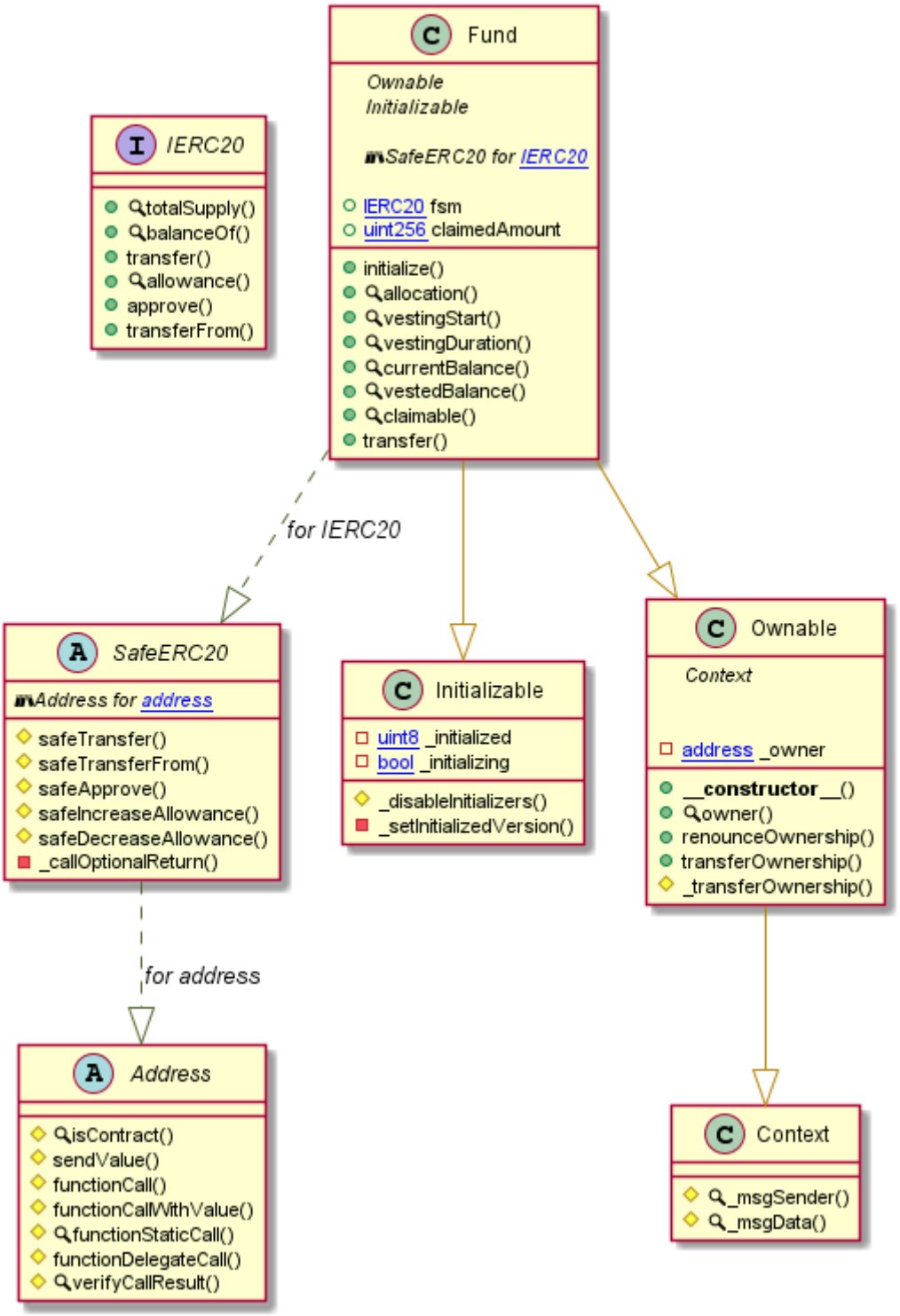
MNGReserve Diagram



MNGTreasuryFund Diagram



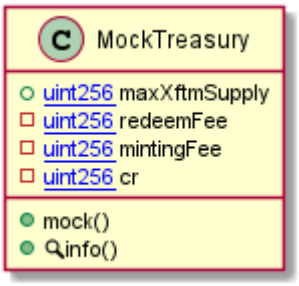
Fund Diagram



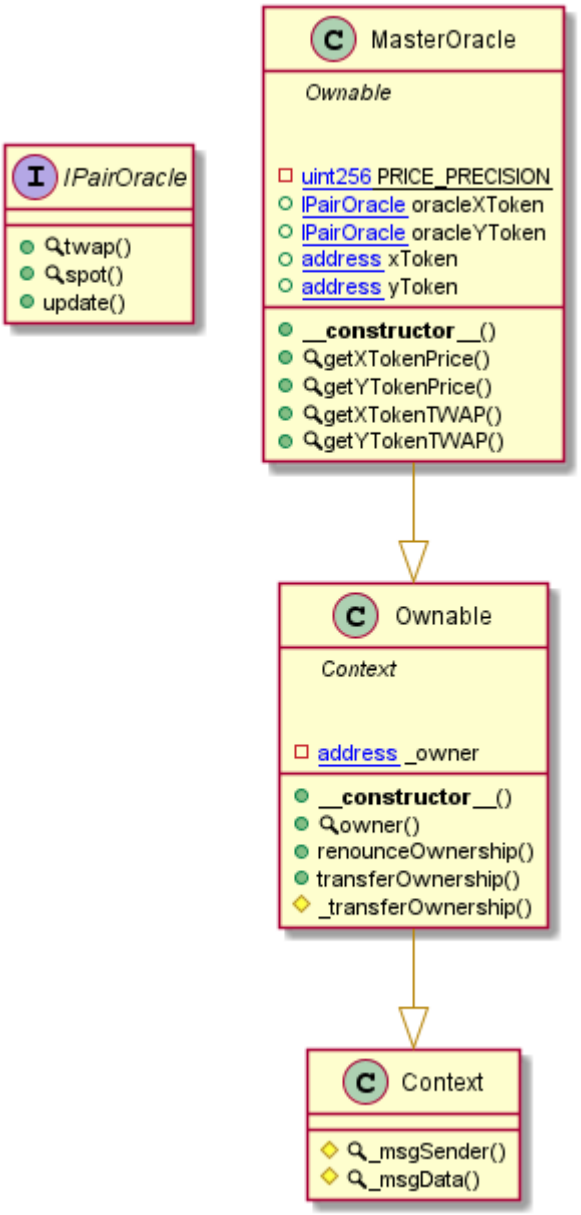
MockERC20 Diagram

Email: audit@EtherAuthority.io

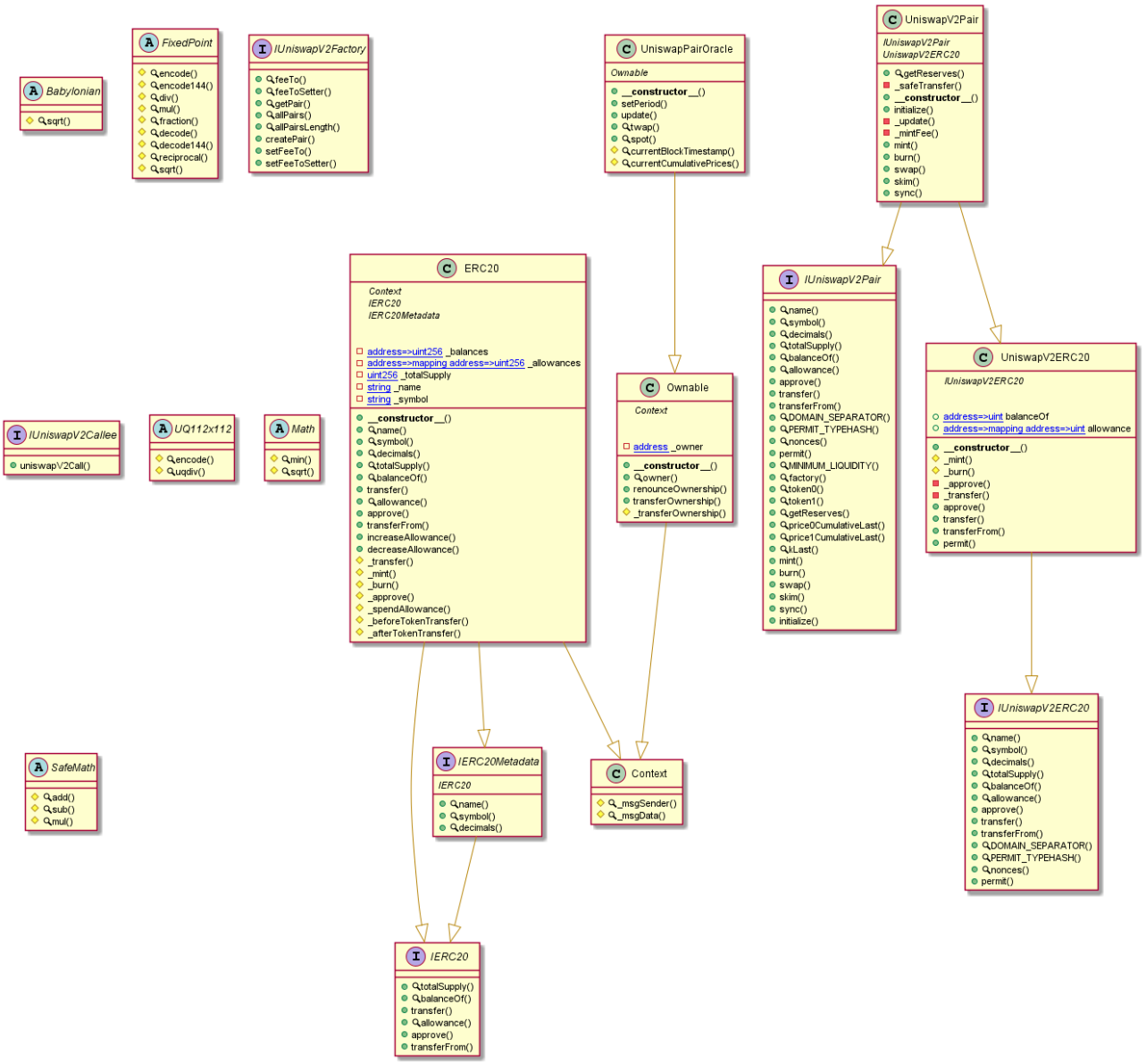
MockTreasury Diagram



MasterOracle Diagram



UniswapPairOracle Diagram

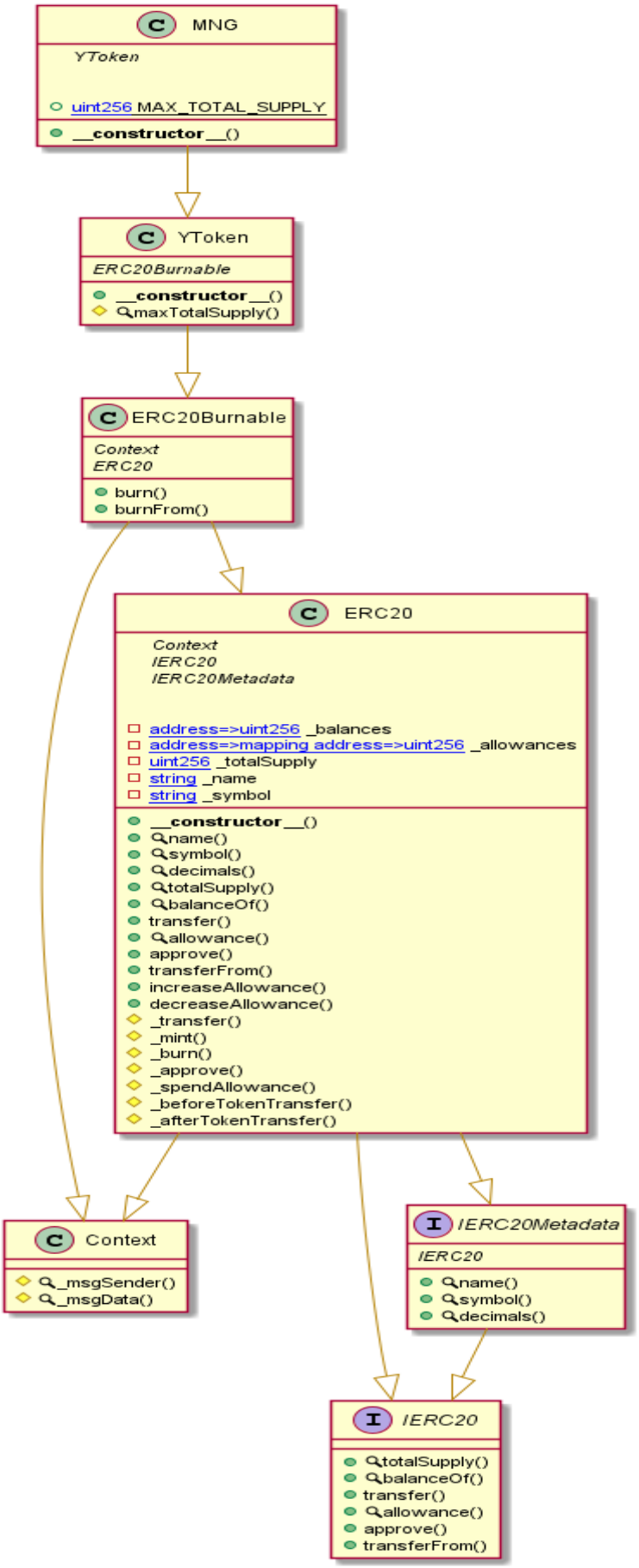


XToken Diagram

Email: audit@EtherAuthority.io

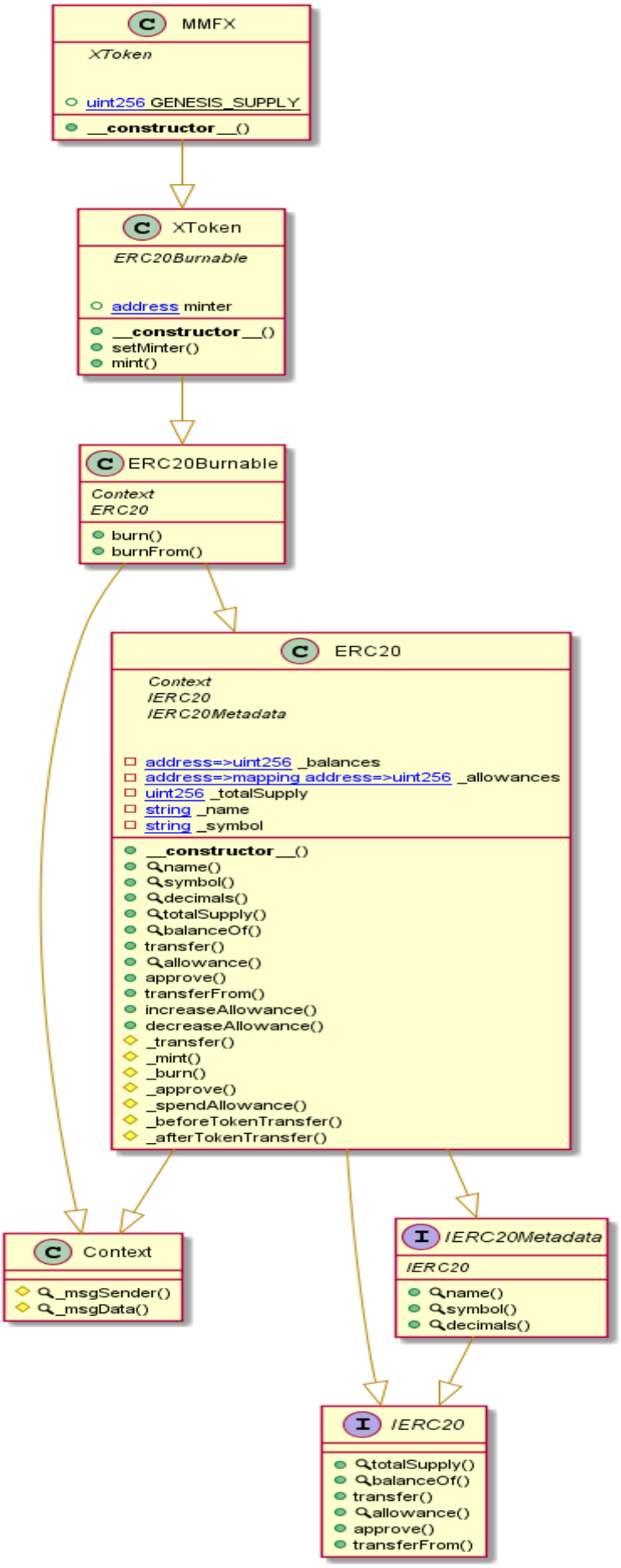
YToken Diagram

MNG Diagram

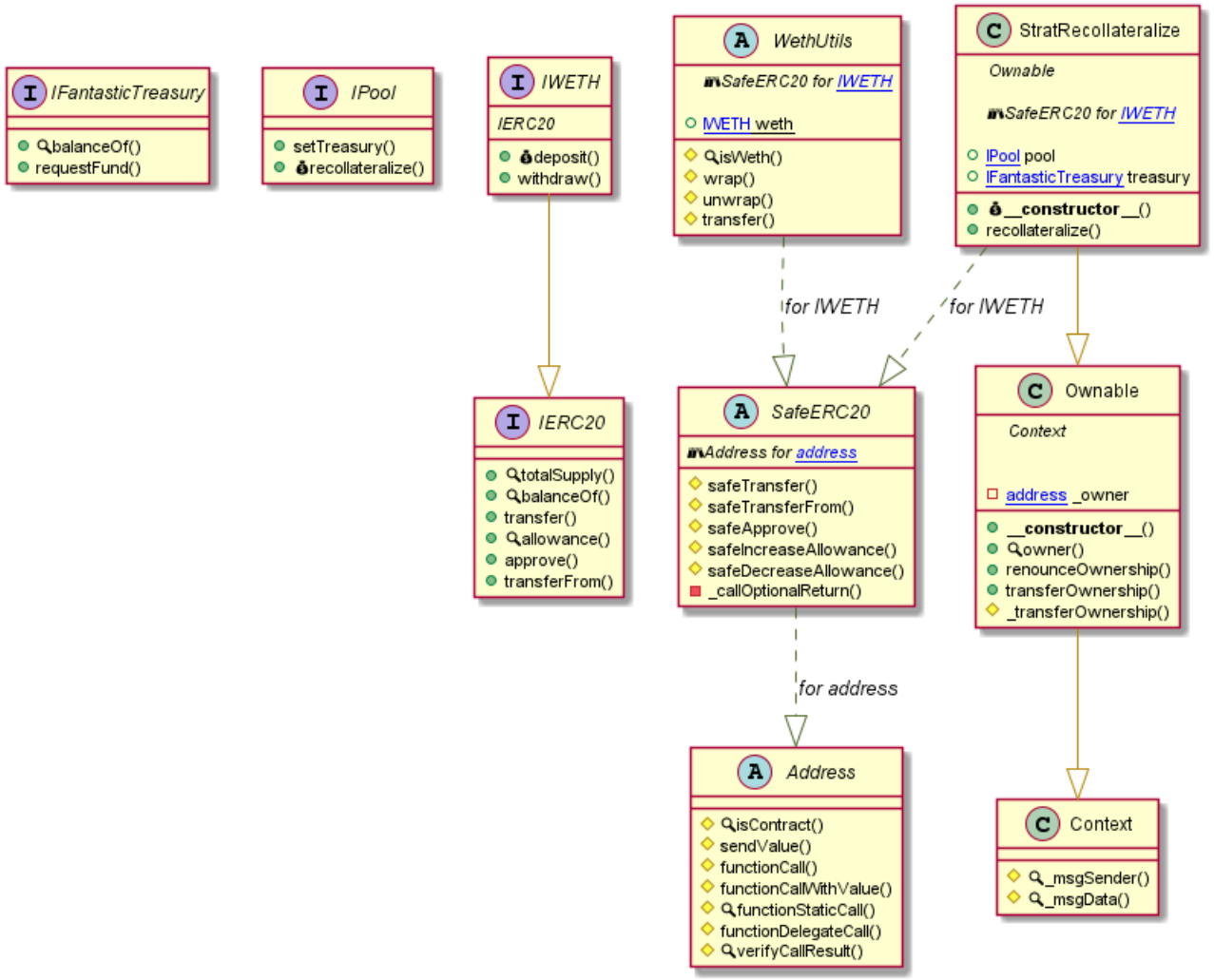


Email: audit@EtherAuthority.io

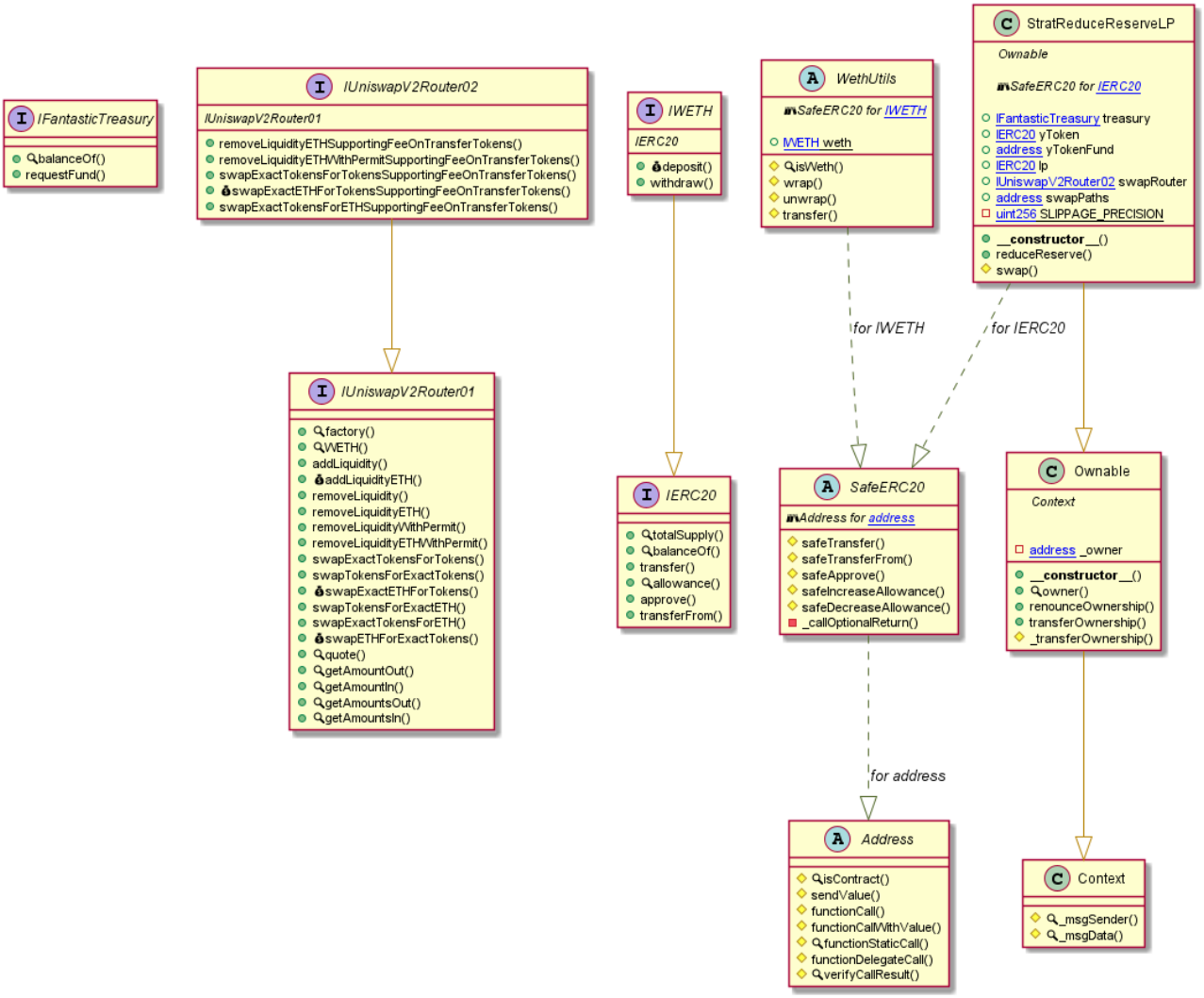
MMFX Diagram



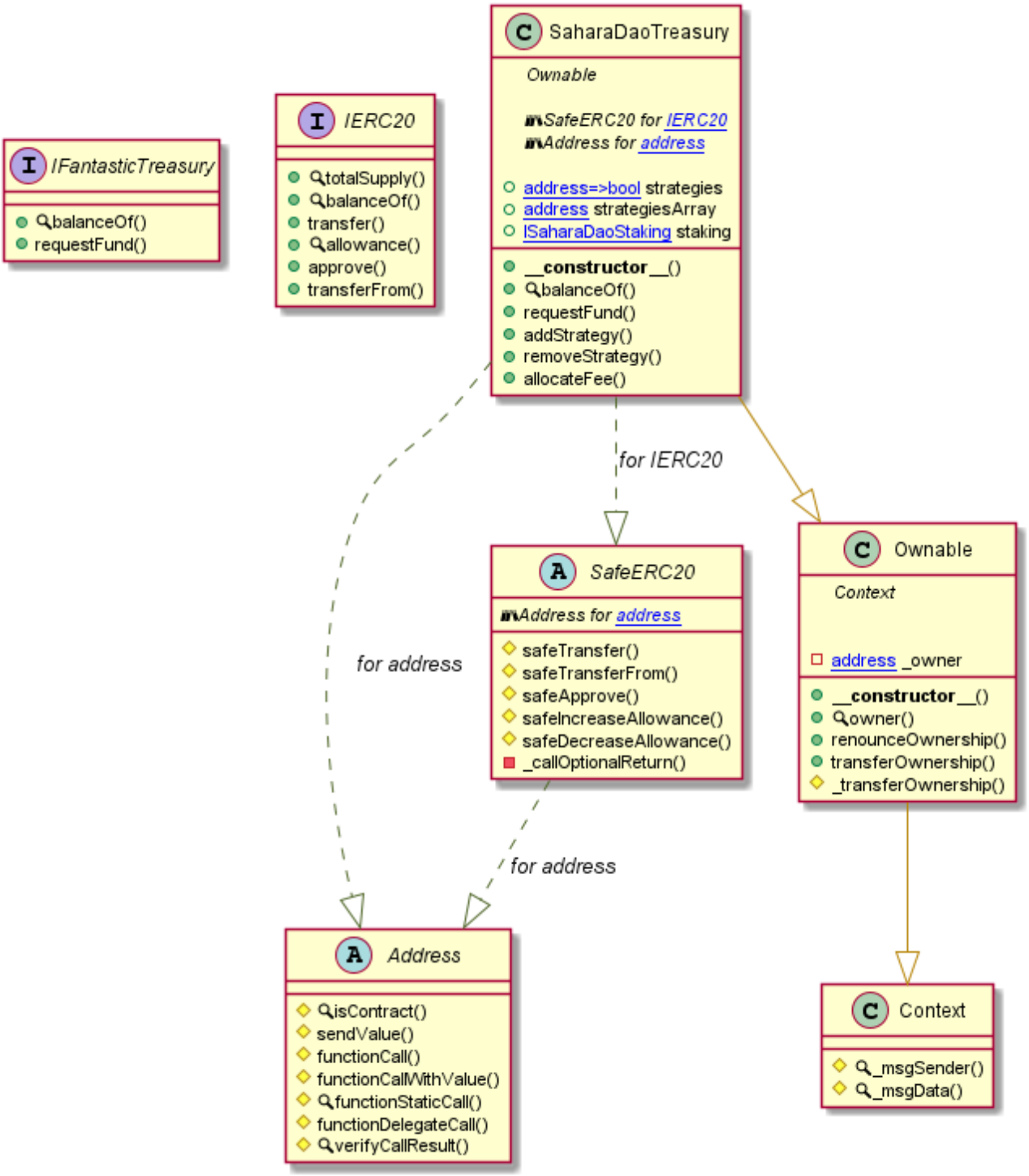
StratRecollateralize Diagram



StratReduceReserveLP Diagram



SaharaDaoTreasury Diagram



Slither Results Log

Slither log >> Pool.sol

```
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (Pool.sol#588-590)
symbol() should be declared external:
        - ERC20.symbol() (Pool.sol#596-598)
decimals() should be declared external:
        - ERC20.decimals() (Pool.sol#613-615)
totalSupply() should be declared external:
        - ERC20.totalSupply() (Pool.sol#620-622)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (Pool.sol#627-629)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (Pool.sol#639-643)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (Pool.sol#662-666)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (Pool.sol#684-693)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (Pool.sol#707-711)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (Pool.sol#727-736)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Pool.sol#944-946)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Pool.sol#952-955)
refreshCollateralRatio() should be declared external:
        - Pool.refreshCollateralRatio() (Pool.sol#1227-1253)
toggle(bool,bool) should be declared external:
        - Pool.toggle(bool,bool) (Pool.sol#1394-1398)
setCollateralRatioOptions(uint256,uint256,uint256,uint256) should be declared external:
        - Pool.setCollateralRatioOptions(uint256,uint256,uint256,uint256) (Pool.sol#1405-1416)
toggleCollateralRatio(bool) should be declared external:
        - Pool.toggleCollateralRatio(bool) (Pool.sol#1420-1425)
setFees(uint256,uint256) should be declared external:
        - Pool.setFees(uint256,uint256) (Pool.sol#1430-1436)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

INFO:Slither:Pool.sol analyzed (18 contracts with 75 detectors), 87 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SwapStrategyPOL.sol

```
INFO:Detectors:
Function IUniswapV2Router01.WETH() (SwapStrategyPOL.sol#12) is not in mixedCase
Constant WethUtils.weth (SwapStrategyPOL.sol#523) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter SwapStrategyPOL.execute(uint256,uint256)._wethIn (SwapStrategyPOL.sol#653) is not in mixedCase
Parameter SwapStrategyPOL.execute(uint256,uint256)._yTokenOut (SwapStrategyPOL.sol#653) is not in mixedCase
Parameter SwapStrategyPOL.swap(uint256,uint256)._wethToSwap (SwapStrategyPOL.sol#671) is not in mixedCase
Parameter SwapStrategyPOL.swap(uint256,uint256)._minYTokenOut (SwapStrategyPOL.sol#671) is not in mixedCase
Parameter SwapStrategyPOL.changeSlippage(uint256)._newSlippage (SwapStrategyPOL.sol#715) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Swap
StrategyPOL.sol#17) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,
uint256).amountBDesired (SwapStrategyPOL.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
SwapStrategyPOL.slitherConstructorVariables() (SwapStrategyPOL.sol#616-728) uses literals with too many digits:
        - swapSlippage = 200000 (SwapStrategyPOL.sol#625)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SwapStrategyPOL.sol#585-587)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SwapStrategyPOL.sol#593-596)
lpBalance() should be declared external:
        - SwapStrategyPOL.lpBalance() (SwapStrategyPOL.sol#644-646)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SwapStrategyPOL.sol analyzed (11 contracts with 75 detectors), 37 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SaharaDaoChef.sol

```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SaharaDaoChef.sol#442-444)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SaharaDaoChef.sol#450-453)
poolLength() should be declared external:
        - SaharaDaoChef.poolLength() (SaharaDaoChef.sol#502-504)
deposit(uint256,uint256,address) should be declared external:
        - SaharaDaoChef.deposit(uint256,uint256,address) (SaharaDaoChef.sol#553-574)
withdraw(uint256,uint256,address) should be declared external:
        - SaharaDaoChef.withdraw(uint256,uint256,address) (SaharaDaoChef.sol#580-601)
withdrawAndHarvest(uint256,uint256,address) should be declared external:
        - SaharaDaoChef.withdrawAndHarvest(uint256,uint256,address) (SaharaDaoChef.sol#632-660)
emergencyWithdraw(uint256,address) should be declared external:
        - SaharaDaoChef.emergencyWithdraw(uint256,address) (SaharaDaoChef.sol#665-679)
add(uint256,IERC20,IRewarder) should be declared external:
        - SaharaDaoChef.add(uint256,IERC20,IRewarder) (SaharaDaoChef.sol#704-718)
set(uint256,uint256,IRewarder,bool) should be declared external:
        - SaharaDaoChef.set(uint256,uint256,IRewarder,bool) (SaharaDaoChef.sol#725-738)
setRewardPerSecond(uint256) should be declared external:
        - SaharaDaoChef.setRewardPerSecond(uint256) (SaharaDaoChef.sol#742-747)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SaharaDaoChef.sol analyzed (8 contracts with 75 detectors), 51 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SaharaDaoStaking.sol

```
INFO:Detectors:
SaharaDaoStaking.userLocks (SaharaDaoStaking.sol#831) is never used in SaharaDaoStaking (SaharaDaoStaking.sol#776-1241)
SaharaDaoStaking.userEarnings (SaharaDaoStaking.sol#832) is never used in SaharaDaoStaking (SaharaDaoStaking.sol#776-1241)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
SaharaDaoStaking.lockedSupply (SaharaDaoStaking.sol#827) should be constant
SaharaDaoStaking.totalSupply (SaharaDaoStaking.sol#826) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SaharaDaoStaking.sol#708-710)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SaharaDaoStaking.sol#716-719)
addReward(address,address) should be declared external:
        - SaharaDaoStaking.addReward(address,address) (SaharaDaoStaking.sol#858-866)
withdrawableBalance(address) should be declared external:
        - SaharaDaoStaking.withdrawableBalance(address) (SaharaDaoStaking.sol#991-1009)
withdraw(uint256) should be declared external:
        - SaharaDaoStaking.withdraw(uint256) (SaharaDaoStaking.sol#1062-1105)
getReward() should be declared external:
        - SaharaDaoStaking.getReward() (SaharaDaoStaking.sol#1108-1124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SaharaDaoStaking.sol analyzed (13 contracts with 75 detectors), 73 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MNGDaoFund.sol

```
INFO:Detectors:
Pragma version0.8.4 (MNGDaoFund.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MNGDaoFund.sol#131-136):
        - (success) = recipient.call{value: amount}() (MNGDaoFund.sol#134)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (MNGDaoFund.sol#199-210):
        - (success,returndata) = target.call{value: value}(data) (MNGDaoFund.sol#208)
Low level call in Address.functionStaticCall(address,bytes,string) (MNGDaoFund.sol#228-237):
        - (success,returndata) = target.staticcall(data) (MNGDaoFund.sol#235)
Low level call in Address.functionDelegateCall(address,bytes,string) (MNGDaoFund.sol#255-264):
        - (success,returndata) = target.delegatecall(data) (MNGDaoFund.sol#262)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Fund.initialize(address)._yToken (MNGDaoFund.sol#545) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MNGDaoFund.sol#513-515)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MNGDaoFund.sol#521-524)
currentBalance() should be declared external:
        - Fund.currentBalance() (MNGDaoFund.sol#558-560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MNGDaoFund.sol analyzed (8 contracts with 75 detectors), 27 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MNGDevFund.sol

```
INFO:Detectors:
Pragma version0.8.4 (MNGDevFund.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MNGDevFund.sol#131-136):
        - (success) = recipient.call{value: amount}() (MNGDevFund.sol#134)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (MNGDevFund.sol#199-210):
        - (success,returndata) = target.call{value: value}(data) (MNGDevFund.sol#208)
Low level call in Address.functionStaticCall(address,bytes,string) (MNGDevFund.sol#228-237):
        - (success,returndata) = target.staticcall(data) (MNGDevFund.sol#235)
Low level call in Address.functionDelegateCall(address,bytes,string) (MNGDevFund.sol#255-264):
        - (success,returndata) = target.delegatecall(data) (MNGDevFund.sol#262)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Fund.initialize(address)._yToken (MNGDevFund.sol#545) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MNGDevFund.sol#513-515)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MNGDevFund.sol#521-524)
currentBalance() should be declared external:
        - Fund.currentBalance() (MNGDevFund.sol#558-560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MNGDevFund.sol analyzed (8 contracts with 75 detectors), 27 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MNGReserve.sol

```
INFO:Detectors:
MNGReserve.setRewarder(address)._rewarder (MNGReserve.sol#488) lacks a zero-check on :
                - rewarder = _rewarder (MNGReserve.sol#490)
MNGReserve.setPool(address)._pool (MNGReserve.sol#494) lacks a zero-check on :
                - pool = _pool (MNGReserve.sol#496)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (MNGReserve.sol#272-292) uses assembly
        - INLINE ASM (MNGReserve.sol#284-287)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Pragma version0.8.4 (MNGReserve.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MNGReserve.sol#131-136):
        - (success) = recipient.call{value: amount}() (MNGReserve.sol#134)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (MNGReserve.sol#199-210):
        - (success,returndata) = target.call{value: value}(data) (MNGReserve.sol#208)
Low level call in Address.functionStaticCall(address,bytes,string) (MNGReserve.sol#228-237):
        - (success,returndata) = target.staticcall(data) (MNGReserve.sol#235)
Low level call in Address.functionDelegateCall(address,bytes,string) (MNGReserve.sol#255-264):
        - (success,returndata) = target.delegatecall(data) (MNGReserve.sol#262)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter MNGReserve.initialize(address)._mng (MNGReserve.sol#481) is not in mixedCase
Parameter MNGReserve.setRewarder(address)._rewarder (MNGReserve.sol#488) is not in mixedCase
Parameter MNGReserve.setPool(address)._pool (MNGReserve.sol#494) is not in mixedCase
Parameter MNGReserve.transfer(address,uint256)._to (MNGReserve.sol#500) is not in mixedCase
Parameter MNGReserve.transfer(address,uint256)._amount (MNGReserve.sol#500) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:MNGReserve.sol analyzed (5 contracts with 75 detectors), 27 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MNGTreasuryFund.sol

```
INFO:Detectors:
Pragma version0.8.4 (MNGTreasuryFund.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.
7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MNGTreasuryFund.sol#131-136):
        - (success) = recipient.call{value: amount}() (MNGTreasuryFund.sol#134)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (MNGTreasuryFund.sol#199-210):
        - (success,returndata) = target.call{value: value}(data) (MNGTreasuryFund.sol#208)
Low level call in Address.functionStaticCall(address,bytes,string) (MNGTreasuryFund.sol#228-237):
        - (success,returndata) = target.staticcall(data) (MNGTreasuryFund.sol#235)
Low level call in Address.functionDelegateCall(address,bytes,string) (MNGTreasuryFund.sol#255-264):
        - (success,returndata) = target.delegatecall(data) (MNGTreasuryFund.sol#262)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Fund.initialize(address)._yToken (MNGTreasuryFund.sol#545) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MNGTreasuryFund.sol#513-515)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MNGTreasuryFund.sol#521-524)
currentBalance() should be declared external:
        - Fund.currentBalance() (MNGTreasuryFund.sol#558-560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MNGTreasuryFund.sol analyzed (8 contracts with 75 detectors), 27 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Fund.sol

```
INFO:Detectors:
Pragma version0.8.4 (Fund.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Fund.sol#131-136):
        - (success) = recipient.call{value: amount}() (Fund.sol#134)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Fund.sol#199-210):
        - (success,returndata) = target.call{value: value}(data) (Fund.sol#208)
Low level call in Address.functionStaticCall(address,bytes,string) (Fund.sol#228-237):
        - (success,returndata) = target.staticcall(data) (Fund.sol#235)
Low level call in Address.functionDelegateCall(address,bytes,string) (Fund.sol#255-264):
        - (success,returndata) = target.delegatecall(data) (Fund.sol#262)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Fund.initialize(address)._fsm (Fund.sol#545) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Fund (Fund.sol#538-588) does not implement functions:
        - Fund.allocation() (Fund.sol#552)
        - Fund.vestingDuration() (Fund.sol#556)
        - Fund.vestingStart() (Fund.sol#554)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Fund.sol#513-515)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Fund.sol#521-524)
currentBalance() should be declared external:
        - Fund.currentBalance() (Fund.sol#558-560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Fund.sol analyzed (7 contracts with 75 detectors), 28 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MockERC20.sol

```
INFO:Detectors:
MockERC20.constructor(string,string,uint8)._name (MockERC20.sol#460) shadows:
        - ERC20._name (MockERC20.sol#114) (state variable)
MockERC20.constructor(string,string,uint8)._symbol (MockERC20.sol#461) shadows:
        - ERC20._symbol (MockERC20.sol#115) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
INFO:Detectors:
Context._msgData() (MockERC20.sol#102-104) is never used and should be removed
ERC20._burn(address,uint256) (MockERC20.sol#352-367) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.4 (MockERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter MockERC20.mint(uint256)._amount (MockERC20.sol#467) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (MockERC20.sol#134-136)
symbol() should be declared external:
        - ERC20.symbol() (MockERC20.sol#142-144)
decimals() should be declared external:
        - ERC20.decimals() (MockERC20.sol#159-161)
        - MockERC20.decimals() (MockERC20.sol#471-473)
totalSupply() should be declared external:
        - ERC20.totalSupply() (MockERC20.sol#166-168)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (MockERC20.sol#173-175)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (MockERC20.sol#185-189)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (MockERC20.sol#208-212)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (MockERC20.sol#230-239)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (MockERC20.sol#253-257)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (MockERC20.sol#273-282)
mint(uint256) should be declared external:
        - MockERC20.mint(uint256) (MockERC20.sol#467-469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Slither:MockERC20.sol analyzed (5 contracts with 75 detectors), 18 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MockTreasury.sol

```
INFO:Detectors:
Pragma version0.8.4 (MockTreasury.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter MockTreasury.mock(uint256,uint256,uint256,uint256)._maxXftmSupply (MockTreasury.sol#10) is not in mixedCase
Parameter MockTreasury.mock(uint256,uint256,uint256,uint256)._cr (MockTreasury.sol#11) is not in mixedCase
Parameter MockTreasury.mock(uint256,uint256,uint256,uint256)._mintingFee (MockTreasury.sol#12) is not in mixedCase
Parameter MockTreasury.mock(uint256,uint256,uint256,uint256)._redeemFee (MockTreasury.sol#13) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
mock(uint256,uint256,uint256,uint256) should be declared external:
        - MockTreasury.mock(uint256,uint256,uint256,uint256) (MockTreasury.sol#9-19)
info() should be declared external:
        - MockTreasury.info() (MockTreasury.sol#21-31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MockTreasury.sol analyzed (1 contracts with 75 detectors), 8 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MasterOracle.sol

```
INFO:Detectors:
Context._msgData() (MasterOracle.sol#19-21) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (MasterOracle.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable MasterOracle.constructor(address,address,address,address)._oracleXToken (MasterOracle.sol#93) is too similar to Maste
rOracle.constructor(address,address,address,address)._oracleYToken (MasterOracle.sol#94)
Variable MasterOracle.oracleXToken (MasterOracle.sol#84) is too similar to MasterOracle.oracleYToken (MasterOracle.sol#85)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MasterOracle.sol#57-59)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MasterOracle.sol#65-68)
getXTokenPrice() should be declared external:
        - MasterOracle.getXTokenPrice() (MasterOracle.sol#106-108)
getYTokenPrice() should be declared external:
        - MasterOracle.getYTokenPrice() (MasterOracle.sol#110-112)
getXTokenTWAP() should be declared external:
        - MasterOracle.getXTokenTWAP() (MasterOracle.sol#114-116)
getYTokenTWAP() should be declared external:
        - MasterOracle.getYTokenTWAP() (MasterOracle.sol#118-120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MasterOracle.sol analyzed (4 contracts with 75 detectors), 11 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> UniswapPairOracle.sol

```
INFO:Detectors:
UniswapV2Pair.initialize(address,address)._token0 (UniswapPairOracle.sol#858) lacks a zero-check on :
        - token0 = _token0 (UniswapPairOracle.sol#860)
UniswapV2Pair.initialize(address,address)._token1 (UniswapPairOracle.sol#858) lacks a zero-check on :
        - token1 = _token1 (UniswapPairOracle.sol#861)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (UniswapPairOracle.sol#356-358)
symbol() should be declared external:
        - ERC20.symbol() (UniswapPairOracle.sol#364-366)
decimals() should be declared external:
        - ERC20.decimals() (UniswapPairOracle.sol#381-383)
totalSupply() should be declared external:
        - ERC20.totalSupply() (UniswapPairOracle.sol#388-390)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (UniswapPairOracle.sol#395-397)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (UniswapPairOracle.sol#407-411)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (UniswapPairOracle.sol#430-434)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (UniswapPairOracle.sol#452-461)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (UniswapPairOracle.sol#475-479)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (UniswapPairOracle.sol#495-504)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (UniswapPairOracle.sol#714-716)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (UniswapPairOracle.sol#722-725)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:UniswapPairOracle.sol analyzed (17 contracts with 75 detectors), 66 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> XToken.sol

```
INFO:Detectors:
XToken.constructor(string,string)._name (XToken.sol#498) shadows:
        - ERC20._name (XToken.sol#114) (state variable)
XToken.constructor(string,string)._symbol (XToken.sol#498) shadows:
        - ERC20._symbol (XToken.sol#115) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
XToken.setMinter(address)._minter (XToken.sol#504) lacks a zero-check on :
                - minter = _minter (XToken.sol#506)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Context._msgData() (XToken.sol#102-104) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (XToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter XToken.setMinter(address)._minter (XToken.sol#504) is not in mixedCase
Parameter XToken.mint(address,uint256)._address (XToken.sol#512) is not in mixedCase
Parameter XToken.mint(address,uint256)._amount (XToken.sol#512) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (XToken.sol#134-136)
symbol() should be declared external:
        - ERC20.symbol() (XToken.sol#142-144)
decimals() should be declared external:
        - ERC20.decimals() (XToken.sol#159-161)
totalSupply() should be declared external:
        - ERC20.totalSupply() (XToken.sol#166-168)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (XToken.sol#173-175)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (XToken.sol#185-189)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (XToken.sol#208-212)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (XToken.sol#230-239)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (XToken.sol#253-257)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (XToken.sol#273-282)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (XToken.sol#463-465)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (XToken.sol#478-481)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:XToken.sol analyzed (6 contracts with 75 detectors), 21 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> YToken.sol

```
INFO:Detectors:
Context._msgData() (YToken.sol#102-104) is never used and should be removed
ERC20._mint(address,uint256) (YToken.sol#329-339) is never used and should be removed
YToken.maxTotalSupply() (YToken.sol#487) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.4 (YToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
YToken (YToken.sol#484-488) does not implement functions:
        - YToken.maxTotalSupply() (YToken.sol#487)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
```

```
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (YToken.sol#134-136)
symbol() should be declared external:
        - ERC20.symbol() (YToken.sol#142-144)
decimals() should be declared external:
        - ERC20.decimals() (YToken.sol#159-161)
totalSupply() should be declared external:
        - ERC20.totalSupply() (YToken.sol#166-168)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (YToken.sol#173-175)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (YToken.sol#185-189)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (YToken.sol#208-212)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (YToken.sol#230-239)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (YToken.sol#253-257)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (YToken.sol#273-282)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (YToken.sol#463-465)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (YToken.sol#478-481)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:YToken.sol analyzed (6 contracts with 75 detectors), 20 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MNG.sol

```
INFO:Detectors:
MNG (MNG.sol#491-508) does not implement functions:
        - YToken.maxTotalSupply() (MNG.sol#487)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (MNG.sol#134-136)
symbol() should be declared external:
        - ERC20.symbol() (MNG.sol#142-144)
decimals() should be declared external:
        - ERC20.decimals() (MNG.sol#159-161)
totalSupply() should be declared external:
        - ERC20.totalSupply() (MNG.sol#166-168)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (MNG.sol#173-175)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (MNG.sol#185-189)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (MNG.sol#208-212)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (MNG.sol#230-239)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (MNG.sol#253-257)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (MNG.sol#273-282)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (MNG.sol#463-465)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (MNG.sol#478-481)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MNG.sol analyzed (7 contracts with 75 detectors), 22 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MMFX.sol

```
INFO:Detectors:
Parameter XToken.setMinter(address)._minter (MMFX.sol#504) is not in mixedCase
Parameter XToken.mint(address,uint256)._address (MMFX.sol#512) is not in mixedCase
Parameter XToken.mint(address,uint256)._amount (MMFX.sol#512) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (MMFX.sol#134-136)
symbol() should be declared external:
        - ERC20.symbol() (MMFX.sol#142-144)
decimals() should be declared external:
        - ERC20.decimals() (MMFX.sol#159-161)
totalSupply() should be declared external:
        - ERC20.totalSupply() (MMFX.sol#166-168)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (MMFX.sol#173-175)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (MMFX.sol#185-189)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (MMFX.sol#208-212)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (MMFX.sol#230-239)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (MMFX.sol#253-257)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (MMFX.sol#273-282)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (MMFX.sol#463-465)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (MMFX.sol#478-481)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MMFX.sol analyzed (7 contracts with 75 detectors), 23 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> StratRecollateralize.sol

```
INFO:Detectors:
Pragma version0.8.4 (StratRecollateralize.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.
12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (StratRecollateralize.sol#141-146):
        - (success) = recipient.call{value: amount}() (StratRecollateralize.sol#144)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (StratRecollateralize.sol#209-220):
        - (success,returndata) = target.call{value: value}(data) (StratRecollateralize.sol#218)
Low level call in Address.functionStaticCall(address,bytes,string) (StratRecollateralize.sol#238-247):
        - (success,returndata) = target.staticcall(data) (StratRecollateralize.sol#245)
Low level call in Address.functionDelegateCall(address,bytes,string) (StratRecollateralize.sol#265-274):
        - (success,returndata) = target.delegatecall(data) (StratRecollateralize.sol#272)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Constant WethUtils.weth (StratRecollateralize.sol#395) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter StratRecollateralize.recollateralize(uint256)._amount (StratRecollateralize.sol#495) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (StratRecollateralize.sol#457-459)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (StratRecollateralize.sol#465-468)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StratRecollateralize.sol analyzed (10 contracts with 75 detectors), 28 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> StratReduceReserveLP.sol

```
INFO:Detectors:
Function IUniswapV2Router01.WETH() (StratReduceReserveLP.sol#13) is not in mixedCase
Constant WethUtils.weth (StratReduceReserveLP.sol#524) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter StratReduceReserveLP.reduceReserve(uint256,uint256)._amount (StratReduceReserveLP.sol#644) is not in mixedCase
Parameter StratReduceReserveLP.reduceReserve(uint256,uint256)._minYTokenAmount (StratReduceReserveLP.sol#644) is not in mixedC
ase
Parameter StratReduceReserveLP.swap(uint256,uint256)._wethToSwap (StratReduceReserveLP.sol#668) is not in mixedCase
Parameter StratReduceReserveLP.swap(uint256,uint256)._minYTokenOut (StratReduceReserveLP.sol#668) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Stra
tReduceReserveLP.sol#18) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,add
ress,uint256).amountBDesired (StratReduceReserveLP.sol#19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
StratReduceReserveLP.SLIPPAGE_PRECISION (StratReduceReserveLP.sol#622) is never used in StratReduceReserveLP (StratReduceReser
veLP.sol#611-676)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (StratReduceReserveLP.sol#587-589)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (StratReduceReserveLP.sol#595-598)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StratReduceReserveLP.sol analyzed (11 contracts with 75 detectors), 34 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SaharaDaoTreasury.sol

```
INFO:Detectors:
Pragma version0.8.4 (SaharaDaoTreasury.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/
0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (SaharaDaoTreasury.sol#136-141):
        - (success) = recipient.call{value: amount}() (SaharaDaoTreasury.sol#139)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (SaharaDaoTreasury.sol#204-215):
        - (success,returndata) = target.call{value: value}(data) (SaharaDaoTreasury.sol#213)
Low level call in Address.functionStaticCall(address,bytes,string) (SaharaDaoTreasury.sol#233-242):
        - (success,returndata) = target.staticcall(data) (SaharaDaoTreasury.sol#240)
Low level call in Address.functionDelegateCall(address,bytes,string) (SaharaDaoTreasury.sol#260-269):
        - (success,returndata) = target.delegatecall(data) (SaharaDaoTreasury.sol#267)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
SaharaDaoTreasury (SaharaDaoTreasury.sol#456-525) should inherit from IFantasticTreasury (SaharaDaoTreasury.sol#5-9)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
INFO:Detectors:
Parameter SaharaDaoTreasury.balanceOf(address)._token (SaharaDaoTreasury.sol#471) is not in mixedCase
Parameter SaharaDaoTreasury.requestFund(address,uint256)._token (SaharaDaoTreasury.sol#480) is not in mixedCase
Parameter SaharaDaoTreasury.requestFund(address,uint256)._amount (SaharaDaoTreasury.sol#480) is not in mixedCase
Parameter SaharaDaoTreasury.addStrategy(address)._strategy (SaharaDaoTreasury.sol#489) is not in mixedCase
Parameter SaharaDaoTreasury.removeStrategy(address)._strategy (SaharaDaoTreasury.sol#499) is not in mixedCase
Parameter SaharaDaoTreasury.allocateFee(address,uint256)._token (SaharaDaoTreasury.sol#516) is not in mixedCase
Parameter SaharaDaoTreasury.allocateFee(address,uint256)._amount (SaharaDaoTreasury.sol#516) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SaharaDaoTreasury.sol#426-428)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SaharaDaoTreasury.sol#434-437)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SaharaDaoTreasury.sol analyzed (7 contracts with 75 detectors), 30 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SaharaDaoZapMMSwap.sol

```
INFO:Detectors:
UniswapV2Pair.initialize(address,address)._token0 (SaharaDaoZapMMSwap.sol#916) lacks a zero-check on :
                - token0 = _token0 (SaharaDaoZapMMSwap.sol#918)
UniswapV2Pair.initialize(address,address)._token1 (SaharaDaoZapMMSwap.sol#916) lacks a zero-check on :
                - token1 = _token1 (SaharaDaoZapMMSwap.sol#919)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in UniswapV2Pair.burn(address) (SaharaDaoZapMMSwap.sol#984-1006):
        External calls:
        - _safeTransfer(_token0,to,amount0) (SaharaDaoZapMMSwap.sol#998)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (SaharaDaoZapMMSwap.sol#905)
        - _safeTransfer(_token1,to,amount1) (SaharaDaoZapMMSwap.sol#999)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (SaharaDaoZapMMSwap.sol#905)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (SaharaDaoZapMMSwap.sol#1003)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (SaharaDaoZapMMS
wap.sol#929)
        - _update(balance0,balance1,_reserve0,_reserve1) (SaharaDaoZapMMSwap.sol#1003)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (SaharaDaoZapMMS
wap.sol#930)
```

```
        Event emitted after the call(s):
        - Zapped(_zapId,_ethIn,_liquidity) (SaharaDaoZapMMSwap.sol#1493)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
UniswapV2ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (SaharaDaoZapMMSwap.sol#856-868) uses timestamp f
or comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,UniswapV2: EXPIRED) (SaharaDaoZapMMSwap.sol#857)
UniswapV2Pair._update(uint256,uint256,uint112,uint112) (SaharaDaoZapMMSwap.sol#923-936) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (SaharaDaoZapMMSwap.sol#927)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
UniswapV2ERC20.DOMAIN_SEPARATOR (SaharaDaoZapMMSwap.sol#807) should be constant
UniswapV2Pair.factory (SaharaDaoZapMMSwap.sol#878) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (SaharaDaoZapMMSwap.sol#414-416)
symbol() should be declared external:
        - ERC20.symbol() (SaharaDaoZapMMSwap.sol#422-424)
decimals() should be declared external:
        - ERC20.decimals() (SaharaDaoZapMMSwap.sol#439-441)
totalSupply() should be declared external:
        - ERC20.totalSupply() (SaharaDaoZapMMSwap.sol#446-448)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (SaharaDaoZapMMSwap.sol#453-455)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (SaharaDaoZapMMSwap.sol#465-469)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (SaharaDaoZapMMSwap.sol#488-492)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (SaharaDaoZapMMSwap.sol#510-519)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (SaharaDaoZapMMSwap.sol#533-537)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (SaharaDaoZapMMSwap.sol#553-562)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SaharaDaoZapMMSwap.sol#772-774)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SaharaDaoZapMMSwap.sol#780-783)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SaharaDaoZapMMSwap.sol analyzed (22 contracts with 75 detectors), 79 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

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: 1227:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1251:33:

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: 1227:4:

Miscellaneous

Constant/View/Pure functions:

Pool.transferToTreasury(uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1488:4:

Similar variable names:

Pool.collect() : Variables have very similar names "_sendXToken" and "_sendYToken". Note: Modifiers are currently not considered by this static analysis.
Pos: 1329:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1489:8:

SwapStrategyPOL.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SwapStrategyPOL.addLiquidity(uint256,uint256,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 684:4:

Gas & Economy

Gas costs:

Gas requirement of function SwapStrategyPOL.changeSlippage 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: 715:4:

Miscellaneous

Constant/View/Pure functions:

SwapStrategyPOL.cleanDust() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 710:4:

Similar variable names:

SwapStrategyPOL.addLiquidity(uint256,uint256,uint256) : Variables have very similar names "_amountA" and "_amountB". Note: Modifiers are currently not considered by this static analysis.
Pos: 706:54:

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: 716:8:

SaharaDaoChef.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SaharaDaoChef.emergencyWithdraw(uint256,address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 665:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 715:72:

Gas & Economy

Gas costs:

Gas requirement of function SaharaDaoChef.pendingReward 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: 510:4:

Miscellaneous

Constant/View/Pure functions:

IRewarder.onReward(uint256,address,address,uint256,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 384:4:

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: 752:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 643:53:

SaharaDaoStaking.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SaharaDaoStaking.getReward(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1108:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1149:44:

Gas & Economy

Gas costs:

Gas requirement of function SaharaDaoStaking.lockDuration 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: 813:4:

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: 1129:8:

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: 1217:8:

Miscellaneous

Constant/View/Pure functions:

SaharaDaoStaking.lockedBalances(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 963:4:

Similar variable names:

SaharaDaoStaking.lockedBalances(address) : Variables have very similar names "locks" and "locked". Note: Modifiers are currently not considered by this static analysis.
Pos: 976:16:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1196:8:

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: 1082:24:

MNGDaoFund.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: 572:31:

Gas & Economy

Gas costs:

Gas requirement of function MNGDaoFund.transfer is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 580:4:

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: 572:15:

MNGDevFund.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 199:4:

Gas & Economy

Gas costs:

Gas requirement of function MNGDevFund.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: 558:4:

Miscellaneous

Constant/View/Pure functions:

SafeERC20._callOptionalReturn(contract IERC20,bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 364:4:

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: 581:8:

MNGReserve.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeDecreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 345:4:

Gas & Economy

Gas costs:

Gas requirement of function MNGReserve.transfer is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 500:4:

Miscellaneous

Constant/View/Pure functions:

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

more

Pos: 500:4:

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: 502:8:

MNGTreasuryFund.sol

Security

Check-effects-interaction:

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

more

Pos: 199:4:

Gas & Economy

Gas costs:

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

Pos: 580:4:

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: 522:8:

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: 572:31:

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: 583:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 572:15:

MockERC20.sol

Gas & Economy

Gas costs:

Gas requirement of function MockERC20.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: 467:4:

Miscellaneous

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 450:4:

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: 409:12:

MockTreasury.sol

Gas & Economy

Gas costs:

Gas requirement of function MockTreasury.mock 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: 9:4:

Gas costs:

Gas requirement of function MockTreasury.info 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: 21:4:

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: 106:4:

Miscellaneous

Constant/View/Pure functions:

IPairOracle.update() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 10:4:

Similar variable names:

MasterOracle.(address,address,address,address) : Variables have very similar names "oracleXToken" and "oracleYToken". Note: Modifiers are currently not considered by this static analysis.
Pos: 102:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 99:8:

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: 1082:22:

Gas & Economy

Gas costs:

Gas requirement of function UniswapV2Pair.sync 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: 990:4:

Gas costs:

Gas requirement of function UniswapPairOracle.pair 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: 1003:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 237:4:

Miscellaneous

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 125:4:

Constant/View/Pure functions:

UniswapPairOracle.currentCumulativePrices(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1086:4:

Similar variable names:

UniswapV2Pair.swap(uint256,uint256,address,bytes) : Variables have very similar names "reserve1" and "_reserve0". Note: Modifiers are currently not considered by this static analysis.
Pos: 968:73:

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: 1069:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1074:21:

XToken.sol

Gas & Economy

Gas costs:

Gas requirement of function ERC20.decreaseAllowance 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: 273:4:

Miscellaneous

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 450:4:

Similar variable names:

ERC20Burnable.burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 480:23:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 505:8:

YToken.sol

Gas & Economy

Gas costs:

Gas requirement of function ERC20.decreaseAllowance 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: 273:4:

Miscellaneous

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 450:4:

Similar variable names:

ERC20Burnable.burnFrom(address,uint256) : Variables have very similar names "account" and "amount".
Pos: 480:23:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 409:12:

MNG.sol

Gas & Economy

Gas costs:

Gas requirement of function ERC20.decreaseAllowance 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: 273:4:

Miscellaneous

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 450:4:

Similar variable names:

ERC20._mint(address,uint256) : Variables have very similar names "account" and "amount".
Pos: 336:34:

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: 409:12:

MMFX.sol

Gas & Economy

Gas costs:

Gas requirement of function MMFX.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: 512:4:

Miscellaneous

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 450:4:

Similar variable names:

ERC20Burnable.burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 480:14:

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: 505:8:

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: 495:4:

Gas & Economy

Gas costs:

Gas requirement of function StratRecollateralize.pool 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: 485:4:

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: 497:8:

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: 653:107:

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: 644:4:

Miscellaneous

Constant/View/Pure functions:

WethUtils.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 538:4:

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: 646:8:

SaharaDaoTreasury.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeDecreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 350:4:

Gas & Economy

Gas costs:

Gas requirement of function SaharaDaoTreasury.balanceOf 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: 471:4:

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: 503:8:

Miscellaneous

Constant/View/Pure functions:

SafeERC20._callOptionalReturn(contract IERC20,bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 369:4:

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: 500:8:

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: 501:8:

SaharaDaoZapMMSwap.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in UniswapV2Pair._mintFee(uint112,uint112): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 939:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1538:12:

Gas & Economy

Gas costs:

Gas requirement of function SaharaDaoZapMMSwap.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: 1587:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 295:4:

Miscellaneous

Constant/View/Pure functions:

SaharaDaoZapMMSwap.approveToken(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1548:4:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1568:12:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1588:8:

Solhint Linter

Pool.sol

```
Pool.sol:508:18: Error: Parse error: missing ';' at '{'
Pool.sol:731:18: Error: Parse error: missing ';' at '{'
Pool.sol:764:18: Error: Parse error: missing ';' at '{'
Pool.sol:813:18: Error: Parse error: missing ';' at '{'
Pool.sol:864:22: Error: Parse error: missing ';' at '{'
```

SwapStrategyPOL.sol

```
SwapStrategyPOL.sol:488:18: Error: Parse error: missing ';' at '{'
```

SaharaDaoChef.sol

```
SaharaDaoChef.sol:356:18: Error: Parse error: missing ';' at '{'
```

SaharaDaoStaking.sol

```
SaharaDaoStaking.sol:57:18: Error: Parse error: missing ';' at '{'
SaharaDaoStaking.sol:70:18: Error: Parse error: missing ';' at '{'
SaharaDaoStaking.sol:82:18: Error: Parse error: missing ';' at '{'
SaharaDaoStaking.sol:99:18: Error: Parse error: missing ';' at '{'
SaharaDaoStaking.sol:111:18: Error: Parse error: missing ';' at '{'
SaharaDaoStaking.sol:207:18: Error: Parse error: missing ';' at '{'
SaharaDaoStaking.sol:230:18: Error: Parse error: missing ';' at '{'
SaharaDaoStaking.sol:256:18: Error: Parse error: missing ';' at '{'
SaharaDaoStaking.sol:607:18: Error: Parse error: missing ';' at '{'
```

SaharaDaoZapMMSwap.sol

```
SaharaDaoZapMMSwap.sol:557:18: Error: Parse error: missing ';' at '{'
SaharaDaoZapMMSwap.sol:590:18: Error: Parse error: missing ';' at '{'
SaharaDaoZapMMSwap.sol:639:18: Error: Parse error: missing ';' at '{'
SaharaDaoZapMMSwap.sol:690:22: Error: Parse error: missing ';' at '{'
SaharaDaoZapMMSwap.sol:1338:18: Error: Parse error: missing ';' at
'{'
```

MNGDaoFund.sol

```
MNGDaoFund.sol:350:18: Error: Parse error: missing ';' at '{'
```

MNGDevFund.sol

```
MNGDevFund.sol:350:18: Error: Parse error: missing ';' at '{'
```

MNGReserve.sol

```
MNGReserve.sol:350:18: Error: Parse error: missing ';' at '{'
```

MNGTreasuryFund.sol

```
MNGTreasuryFund.sol:350:18: Error: Parse error: missing ';' at '{'
```

Fund.sol

```
Fund.sol:350:18: Error: Parse error: missing ';' at '{'
```

MockERC20.sol

```
FantasticTreasury.sol:277:18: Error: Parse error: missing ';' at '{'
FantasticTreasury.sol:310:18: Error: Parse error: missing ';' at '{'
FantasticTreasury.sol:359:18: Error: Parse error: missing ';' at '{'
FantasticTreasury.sol:410:22: Error: Parse error: missing ';' at '{
```

MockTreasury.sol

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

MasterOracle.sol

```
MasterOracle.sol:3:1: Error: Compiler version 0.8.4 does not satisfy
the r semver requirement
MasterOracle.sol:31:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
MasterOracle.sol:90:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

UniswapPairOracle.sol

```
UniswapPairOracle.sol:499:18: Error: Parse error: missing ';' at '{'
UniswapPairOracle.sol:532:18: Error: Parse error: missing ';' at '{'
UniswapPairOracle.sol:581:18: Error: Parse error: missing ';' at '{'
UniswapPairOracle.sol:632:22: Error: Parse error: missing ';' at '{'
UniswapPairOracle.sol:1035:18: Error: Parse error: missing ';' at '{'
UniswapPairOracle.sol:1102:18: Error: Parse error: missing ';' at '{'
```

XToken.sol

```
XToken.sol:277:18: Error: Parse error: missing ';' at '{'
XToken.sol:310:18: Error: Parse error: missing ';' at '{'
XToken.sol:359:18: Error: Parse error: missing ';' at '{'
XToken.sol:410:22: Error: Parse error: missing ';' at '{'
```

YToken.sol

```
YToken.sol:277:18: Error: Parse error: missing ';' at '{'
YToken.sol:310:18: Error: Parse error: missing ';' at '{'
YToken.sol:359:18: Error: Parse error: missing ';' at '{'
YToken.sol:410:22: Error: Parse error: missing ';' at '{'
```

MNG.sol

```
MNG.sol:277:18: Error: Parse error: missing ';' at '{'
MNG.sol:310:18: Error: Parse error: missing ';' at '{'
MNG.sol:359:18: Error: Parse error: missing ';' at '{'
MNG.sol:410:22: Error: Parse error: missing ';' at '{'
```

MMFX.sol

```
MMFX.sol:277:18: Error: Parse error: missing ';' at '{'
MMFX.sol:310:18: Error: Parse error: missing ';' at '{'
MMFX.sol:359:18: Error: Parse error: missing ';' at '{'
MMFX.sol:410:22: Error: Parse error: missing ';' at '{'
```

StratRecollateralize.sol

```
StratRecollateralize.sol:360:18: Error: Parse error: missing ';' at
'{'
```

StratReduceReserveLP.sol

```
StratReduceReserveLP.sol:489:18: Error: Parse error: missing ';' at
'{'
```

SaharaDaoTreasury.sol

```
SaharaDaoTreasury.sol:355:18: Error: Parse error: missing ';' at '{'
```

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