Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: MadKing Finance Protocol
Platform: Cronos Blockchain
Language: Solidity
Date: May 26th, 2022

Table of contents

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Introduction

EtherAuthority was contracted by MadKing Finance to perform the Security audit of the MadKing Finance Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on May 26th, 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

- MadKing's unique NFT distribution model, and first of its kind in cronos: NFT-as-a-Subscription (NaaS).
- MadKing art design originally comes from the world famous "Game of Thrones". We are launching this series using NFT as a subscription (NaaS) service.
- MadKing DAO's mission is to utilize NFT as a subscription (NaaS) service technology to help artists and communities to launch their own NFT series and to help raise their own DAO treasury. The backbone of a DAO is its smart contract and we will further build more DAO technology in this space.
- MadKing NFTs holders are the core members of DAO, and it has higher voting power in directing the DAO directions and the use of the treasury.
- The MadKing Finance Protocol is a Defi Program which has functions like mint, swap, OpenTrade, burn, twap, spot, update, mock, info, transfer, set pool, claimable, zap, addLiquidity, cleanDust, etc.
- The MadKing Finance contract inherits the IERC721, ERC20, SafeERC20, Ownable, ReentrancyGuard, Address, IUniswapV2Router02, IUniswapV2Pair, IERC20, Math, SafeMath, Initializable, ERC20Burnable, TransparentUpgradeableProxy 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 MadKing Finance Protocol Smart Contracts
Platform	**Cronos / Solidity**
File 1	Pool.sol
File 1 MD5 Hash	96E877B7E3ED19870D5ED2502F73CAB7
File 2	SwapStrategyPOL.sol
File 2 MD5 Hash	04929BE63CE8E467FB26F4B5F2287BAF
File 3	Timelock.sol
File 3 MD5 Hash	94F559046B7CB4335EE0F49341A23DA0
File 4	MadkingDaoChef.sol
File 4 MD5 Hash	2EF23F582AAE0ECE54146205AAD892B9
Update File 4 MD5 Hash	4924B2D33F85875818FEFCA67CCEA58C
File 5	MadkingDaoStaking.sol
File 5 MD5 Hash	F6D07D088345B64A5B83D32AACBD3612
File 6	MadkingDaoZapMMSwap.sol
File 6 MD5 Hash	5A6D6CE8B9452FE643DDF71B5C1EA48D
File 7	NFTController.sol
File 7 MD5 Hash	0F8B98F08EB464E5F1098F448B95ECCC
File 8	Fund.sol
File 8 MD5 Hash	FDD809CCA8A37A7689B86AAAEF65BB8C
File 9	MDSDaoFund.sol
File 9 MD5 Hash	53F499A5A29F5C5F1D99342561B5E440
File 10	MDSDevFund.sol
File 10 MD5 Hash	4DA5643E68FAA5789A9F9E115F8BF49B
File 11	MDSReserve.sol

File 11 MD5 Hash	6B7F1BAC85E4E5ABA10F7D9F56A74EA4
File 12	MDSTreasuryFund.sol
File 12 MD5 Hash	6B7F1BAC85E4E5ABA10F7D9F56A74EA4
File 13	MasterOracle.sol
File 13 MD5 Hash	858E77FFF07996BAFD879F91F4AB5FA2
File 14	UniswapPairOracle.sol
File 14 MD5 Hash	C541DE95E99AC9AC297B7A5FCB1DC6CB
File 15	XToken.sol
File 15 MD5 Hash	395A1DA7EA3BE191DECE50ACA1B7DA66
File 16	YToken.sol
File 16 MD5 Hash	64ED8F5421F4C542F1C4F308548A31D2
File 17	MDK.sol
File 17 MD5 Hash	9B037041D588B4003E32367FCDC53713
File 18	MDS.sol
File 18 MD5 Hash	8B08B61F8DFF27710227EB99043C5EEE
File 19	MadkingDaoTreasury.sol
File 19 MD5 Hash	8B08B61F8DFF27710227EB99043C5EEE
File 20	StratRecollateralize.sol
File 20 MD5 Hash	CB0011910A0C00C6C7C7EA7A5F82850A
File 21	StratReduceReserveLP.sol
File 21 MD5 Hash	EBBB464C813D8CFF7EB559A6217B9AB1
Audit Date	May 26th,2022

Claimed Smart Contract Features

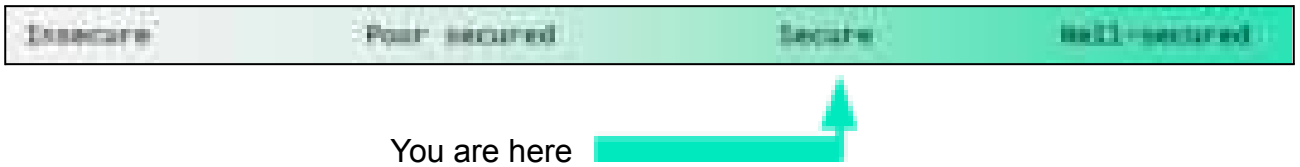
Claimed Feature Detail	Our Observation
File 1 Pool.sol • Refresh Cooldown: 1 hour • Ratio StepUp: 0.002% • Ratio StepDown: 0.002% • Price Target: 1 SVN • Price Band: 0.005 • Minimum Collateral Ratio: 9,00,000 • YToken Slippage: 20% • Redemption Fee: 0.67% • Redemption Fee Maximum: 0.9% • Minting Fee: 0.33% • Minting Fee Maximum: 0.5%	**YES, This is valid.**
File 2 SwapStrategyPOL.sol • Swap Slippage: 20%	**YES, This is valid.**
File 3 Timelock.sol • Grace Period: 14 Days • Minimum Delay: 12 Hours • Maximum Delay: 30 Days	**YES, This is valid.**
File 4 MadkingDaoChef.sol • Maximum Imumreward: 10 Token Per Second • NFT Boost Rate: 100	**YES, This is valid.**
File 5 MadkingDaoStaking.sol • Rewards Duration: 1 week • Lock Duration: 4 weeks • Team Reward Percent: 20%	**YES, This is valid.**
File 6 MadkingDaoZapMMSwap.sol • MadkingDaoZap is a ZapperFi's simplified version	**YES, This is valid.**

of zapper contract which will: 1. use ETH to swap to target token 2. make LP between ETH and target token 3. add into MadkingDaoChef farm	
File 7 Fund.sol • Fund has functions like: allocation, transfer, etc.	**YES, This is valid.**
File 8 MDSDaoFund.sol • Allocation: 10% • Vesting Duration: 3 Years	**YES, This is valid.**
File 9 MDSDevFund.sol • Allocation: 10% • Vesting Duration: 2 Years	**YES, This is valid.**
File 10 MDSReserve.sol • MDSReserve has functions like: initialize, etc.	**YES, This is valid.**
File 11 MDSTreasuryFund.sol • Allocation: 10% • Vesting Duration: 3 Years	**YES, This is valid.**
File 12 MasterOracle.sol • MasterOracle has functions like: getYTokenPrice, getYTokenTWAP, etc.	**YES, This is valid.**
File 13 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 14 XToken.sol • XToken has functions like: mint, etc.	**YES, This is valid.**

File 15 YToken.sol • YToken has functions like: burn, etc.	**YES, This is valid.**
File 16 MDK.sol • Genesis Supply: 100 MDK • Decimals: 18 • Total Supply: 100	**YES, This is valid.**
File 17 MDS.sol • Maximum Total Supply = 100 Million MDS	**YES, This is valid.**
File 18 MadkingDaoTreasury.sol • MadkingDaoTreasury has functions like: balanceOf, requestFund, etc.	**YES, This is valid.**
File 19 StratRecollateralize.sol • StratRecollateralize has functions like: recollateralize.	**YES, This is valid.**
File 20 StratReduceReserveLP.sol • StratReduceReserveLPhas functions like: reduceReserve, swap.	**YES, This is valid.**
File 21 NFTController.sol • NFTController has functions like: setBoostRate, setWhitelist, etc.	**YES, This is valid.**

Audit Summary

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



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

We found 0 critical, 0 high, 0 medium and 2 low and some very low level issues.

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

Technical Quick Stats

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

Code Quality

This audit scope has 2 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 MadKing Finance Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the MadKing Finance Protocol.

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

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

Documentation

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

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

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	write	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	Passed	No Issue
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	calculateSwapInAmount	internal	Passed	No Issue
10	swap	internal	Passed	No Issue
11	addLiquidity	internal	Passed	No Issue
12	cleanDust	external	access only Owner	No Issue
13	changeSlippage	external	access only Owner	No Issue

Timelock.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	setDelay	write	Passed	No Issue
3	acceptAdmin	write	Passed	No Issue
4	setPendingAdmin	write	Passed	No Issue
5	queueTransaction	write	Passed	No Issue
6	cancelTransaction	write	Passed	No Issue
7	executeTransaction	write	Passed	No Issue
8	getBlockTimestamp	internal	Passed	No Issue

MadkingDaoChef.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
22	getBoost	read	Passed	No Issue
23	getSlots	read	Passed	No Issue
24	getTokenIds	read	Passed	No Issue
25	depositNFT	write	Passed	No Issue
26	withdrawNFT	write	Passed	No Issue
27	setNftController	write	access only Owner	No Issue
28	setNftBoostRate	write	access only Owner	No Issue

MadkingDaoStaking.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	addReward	write	Function input parameters lack of check	Refer Audit Findings
9	approveRewardDistributor	external	access only Owner	No Issue
10	rewardPerToken	read	Passed	No Issue
11	_earned	internal	Passed	No Issue
12	lastTimeRewardApplicable	read	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	withdrawableBalance	read	Passed	No Issue
19	stake	external	Passed	No Issue
20	mint	external	Function input parameters lack of check, Division before multiplication	Refer Audit Findings
21	withdraw	write	Passed	No Issue
22	getReward	write	Passed	No Issue
23	emergencyWithdraw	external	Critical operation lacks event log	Refer Audit Findings
24	_notifyReward	internal	Passed	No Issue
25	notifyRewardAmount	external	Passed	No Issue
26	recoverERC20	external	access only Owner	No Issue

27	setTeamWalletAddress	external	Passed	No Issue
28	setTeamRewardPercent	external	Passed	No Issue
29	updateReward	modifier	Passed	No Issue

MadkingDaoZapMMSwap.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	zap	external	Passed	No Issue
9	swap	internal	Passed	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

NFTController.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	initializer	modifier	Passed	No Issue
8	reinitializer	modifier	Passed	No Issue
9	onlyInitializing	modifier	Passed	No Issue
10	_disableInitializers	internal	Passed	No Issue
11	_setInitializedVersion	write	Passed	No Issue
12	initialize	write	Passed	No Issue
13	getBoostRate	external	Passed	No Issue
14	setWhitelist	external	access only Owner	No Issue
15	setDefaultBoostRate	external	access only Owner	No Issue
16	setBoostRate	external	access only Owner	No Issue

NFTControllerProxy.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	ifAdmin	modifier	Passed	No Issue
3	admin	external	access if Admin	No Issue
4	implementation	external	access if Admin	No Issue
5	changeAdmin	external	access if Admin	No Issue
6	upgradeTo	external	access if Admin	No Issue
7	upgradeToAndCall	external	access if Admin	No Issue
8	_admin	internal	Passed	No Issue
9	_beforeFallback	internal	Passed	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	initializer	modifier	Passed	No Issue
8	reinitializer	modifier	Passed	No Issue
9	onlyInitializing	modifier	Passed	No Issue
10	_disableInitializers	internal	Passed	No Issue
11	_setInitializedVersion	write	Passed	No Issue
12	initialize	external	Passed	No Issue
13	allocation	read	Passed	No Issue
14	vestingStart	read	Passed	No Issue
15	vestingDuration	read	Passed	No Issue
16	currentBalance	read	Passed	No Issue
17	vestedBalance	read	Passed	No Issue
18	claimable	read	Passed	No Issue
19	transfer	external	access only Owner	No Issue

MDSDaoFund.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

MDSDevFund.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

MDSReserve.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

MDSTreasuryFund.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	write	Passed	No Issue
8	getYTokenPrice	write	Passed	No Issue
9	getXTokenTWAP	write	Passed	No Issue
10	getYTokenTWAP	write	Passed	No Issue

UniswapPairOracle.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	setPeriod	external	access only Owner	No Issue
8	update	external	Passed	No Issue
9	twap	external	Passed	No Issue
10	spot	external	Passed	No Issue
11	currentBlockTimestamp	internal	Passed	No Issue
12	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	Unlimited Minting	Refer Audit Findings

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

MDK.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	access only Minter	No Issue
5	OpenTrade	external	Passed	No Issue
6	includeToWhitelist	write	Passed	No Issue
7	excludeFromWhitlist	write	Passed	No Issue

MDS.sol

Functions

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

MadkingDaoTreasury.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

StratRecollateralize.sol

Functions

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

StratReduceReserveLP.sol

Functions

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

Severity Definitions

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

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Critical operation lacks event log:**- MadkingDaoStaking.sol**

Missing event log for: emergencyWithdraw

Resolution: Write an event log for listed events.

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

Variable validation is not performed in below functions:

- addReward =_rewardsToken
- mint = user

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

Very Low / Informational / Best practices:

(1) Unlimited Minting: **- XToken.sol**

Minter can mint unlimited tokens.

Resolution: We suggest putting a minting limit.

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

```
// Mint new tokens
// Minted tokens receive rewards normally but incur a 50% penalty when
// withdrawn before lockDuration has passed.
function mint(address user, uint256 amount) external updateReward(user) {
    require(minters[msg.sender], "MultiFeeDistribution::mint: Only minters allowed");
    totalSupply = totalSupply.add(amount);
    Balances storage bal = balances[user];
    bal.total = bal.total.add(amount);
    bal.earned = bal.earned.add(amount);
    uint256 unlockTime = block.timestamp.div(rewardsDuration).mul(rewardsDuration).add
    LockedBalance[] storage earnings = userEarnings[user];
```

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

Resolution: Consider ordering multiplication before division.

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 owner can set the minimum Collateral Ratio.
- reduceExcessCollateral: Pool owner 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.
- setTreasury: Pool owner can set the address of the Treasury.
- cleanDust: SwapStrategyPOL owner can clean dust.
- changeSlippage: SwapStrategyPOL owner can change slippage value.
- add: MadkingDaoChef owner can add a new LP to the pool.
- set: MadkingDaoChef owner can update the given pool's reward allocation point and `IRewarder` contract.
- setRewardPerSecond: MadkingDaoChef owner can set the reward per second to be distributed.
- setRewardMinter: MadkingDaoChef owner can set the address of rewardMinter.
- setNftController: MadkingDaoChef owner can set NFT controller address.
- setNftBoostRate: MadkingDaoChef owner can set NFT Boost Rate value.
- addReward: MadkingDaoStaking owner can add a new reward token to be distributed to stakers.
- approveRewardDistributor: MadkingDaoStaking owner can modify approval for an address to call notifyRewardAmount.
- recoverERC20:MadkingDaoStaking owner can be added to support recovering LP Rewards from other systems such as BAL to be distributed to holders.

- setTeamWalletAddress:MadkingDaoStaking owner can set the address of the team wallet.
- setTeamRewardPercent:MadkingDaoStaking owner can set percent of team reward.
- addZap: MadkingDaoZapMMSwap owner can add new zap configuration.
- removeZap: MadkingDaoZapMMSwap owner can deactivate a Zap configuration.
- setWhitelist: NFTController owner can set whitelist address,
- setDefaultBoostRate: NFTController owner can set default boost rate value.
- setBoostRate: NFTController owner can set boost rate value.
- transfer: Fund owners can transfer tokens.
- transfer: MDSReserve::transfer owner can only allow funds to withdraw.
- setPeriod: UniswapPairOracle owner can set maximum and minimum period.
- setMinter: XToken minter can set minter for XToken.
- mint: XToken minter can mint new XToken.
- OpenTrade: MDK owners can trade openly.
- includeToWhitelist: MDK owner can include address to whitelist.
- excludeFromWhitlist: MDK owner can exclude address to whitelist.
- OpenTrade: MDS owners can trade openly.
- includeToWhitelist: MDS owner can include address to whitelist.
- excludeFromWhitlist: MDS owner can exclude address to whitelist.
- addStrategy: MadkingDaoTreasury owner can add new strategy.
- removeStrategy: MadkingDaoTreasury owner can remove current strategy.
- allocateFee:MadkingDaoTreasury owner can allocate protocol's fee to stakers.
- recollateralize: StratRecollateralize owner can 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 have not observed any major issues in the smart contracts. **So, the smart contracts are ready for the mainnet deployment**.

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

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

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

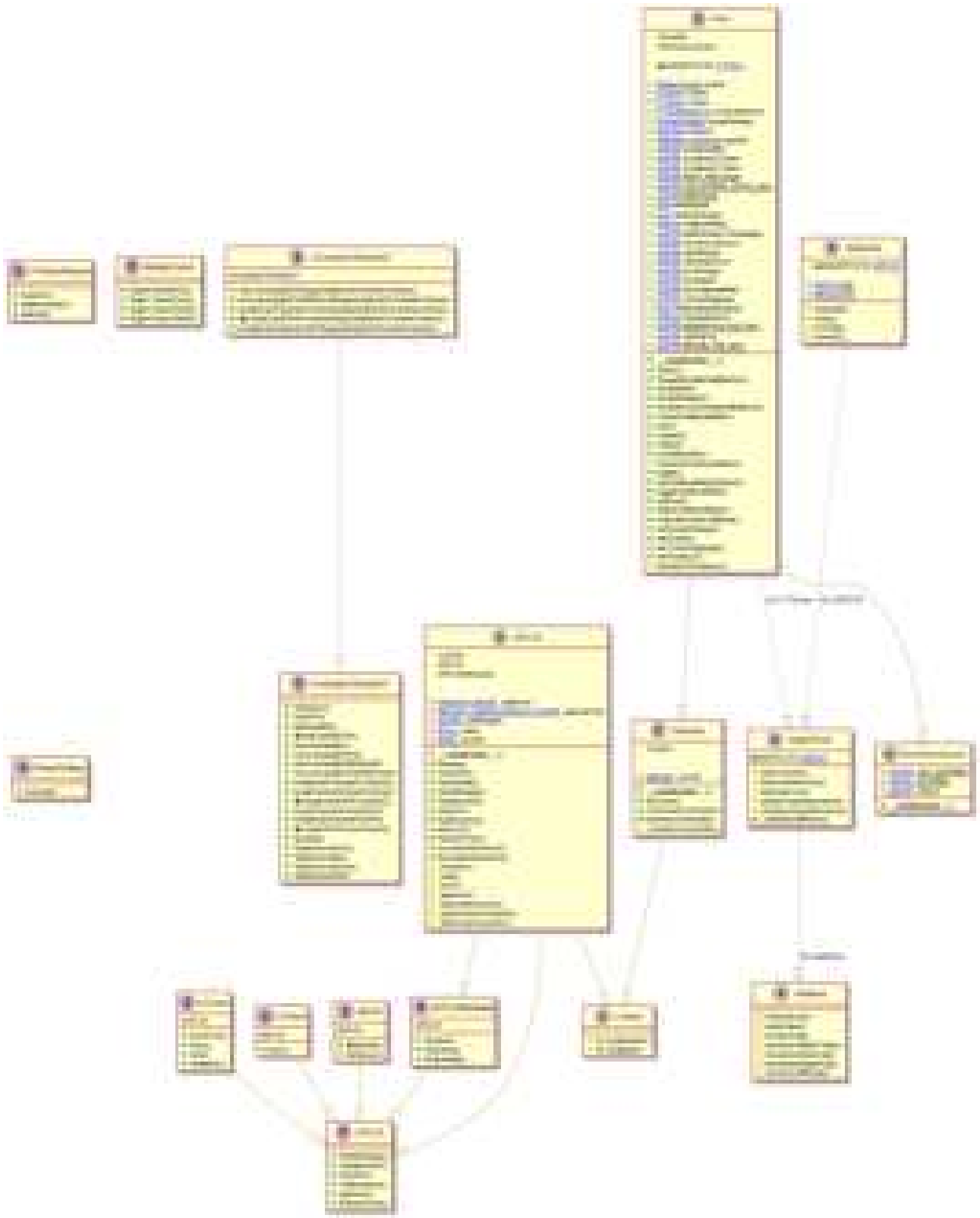
Technical Disclaimer

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

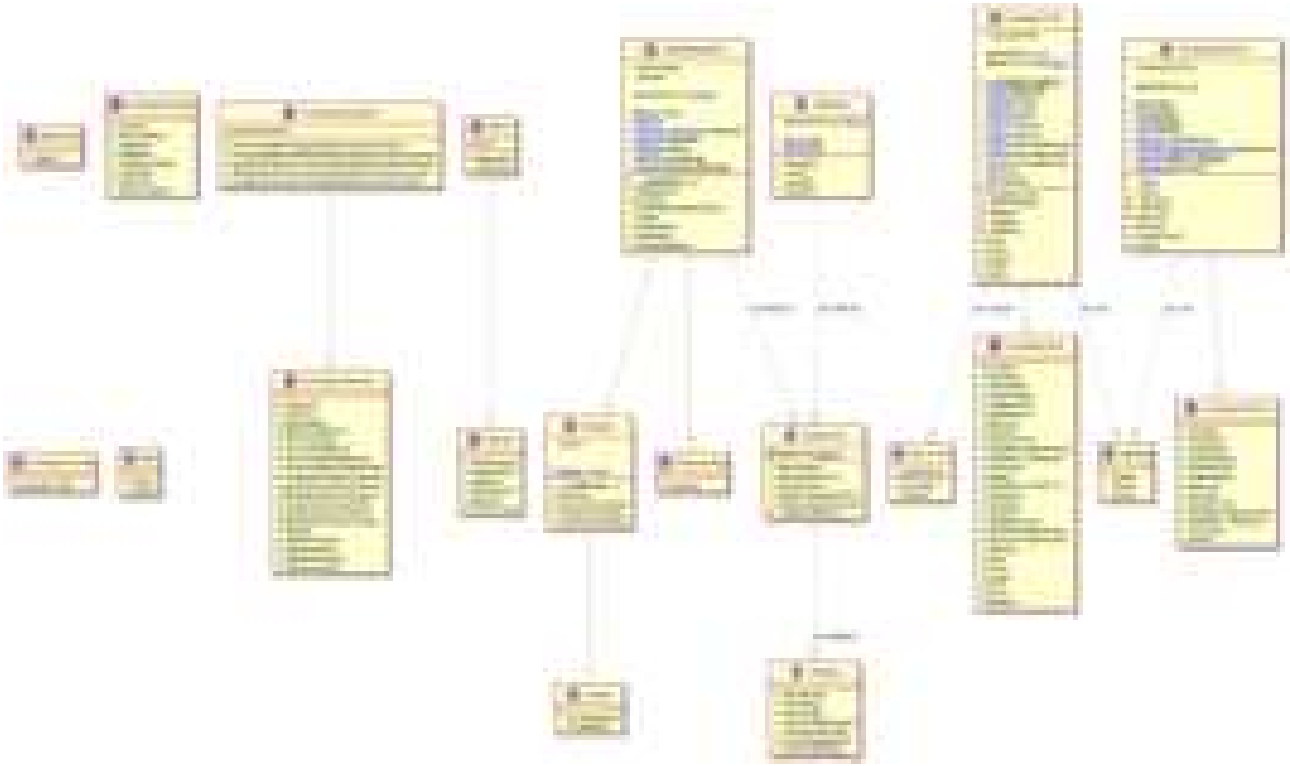
Appendix

Code Flow Diagram - MadKing Finance Protocol

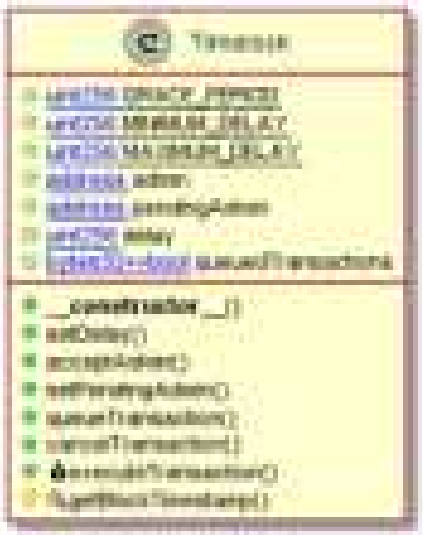
Pool Diagram



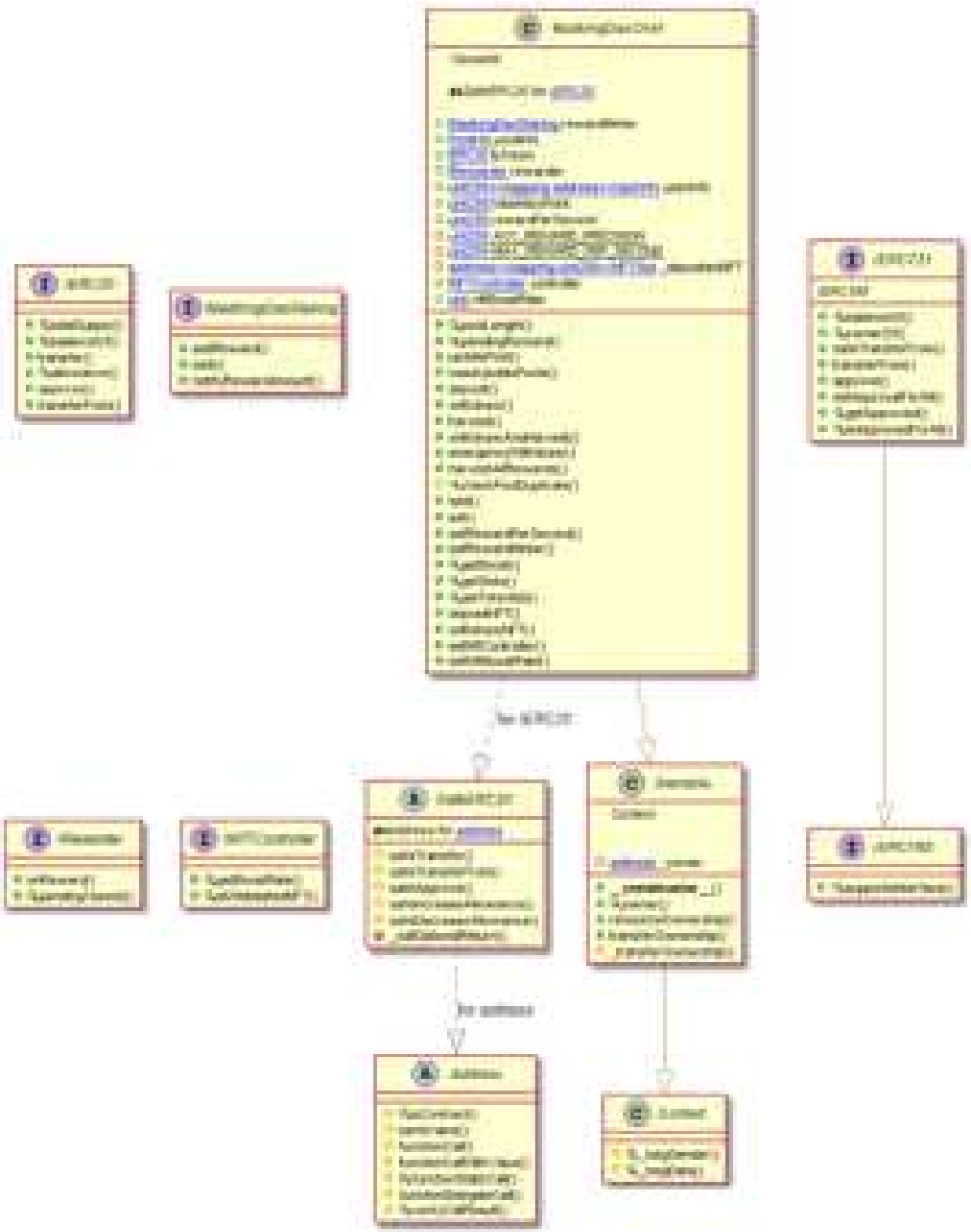
SwapStrategyPOL Diagram



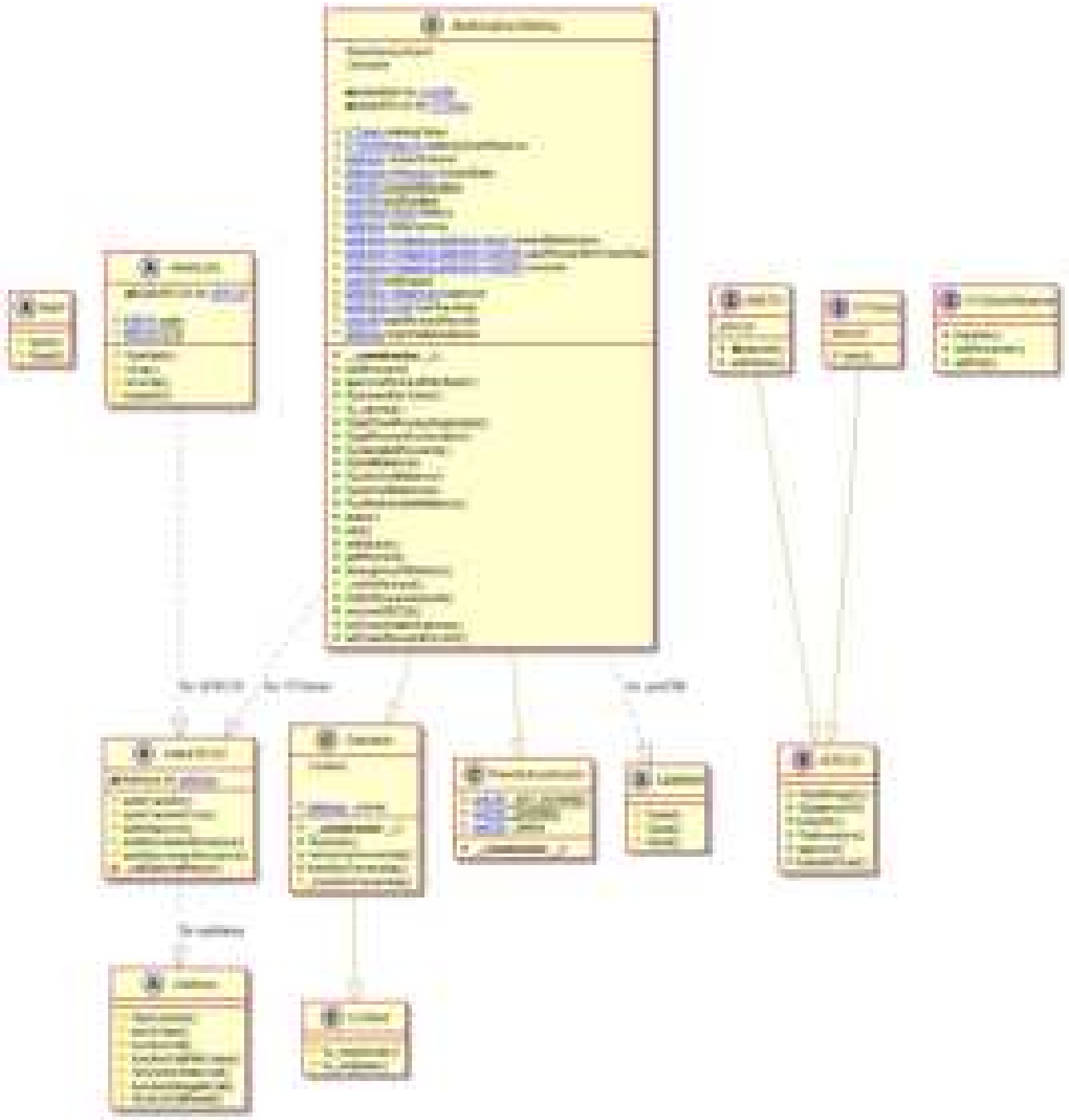
Timelock Diagram



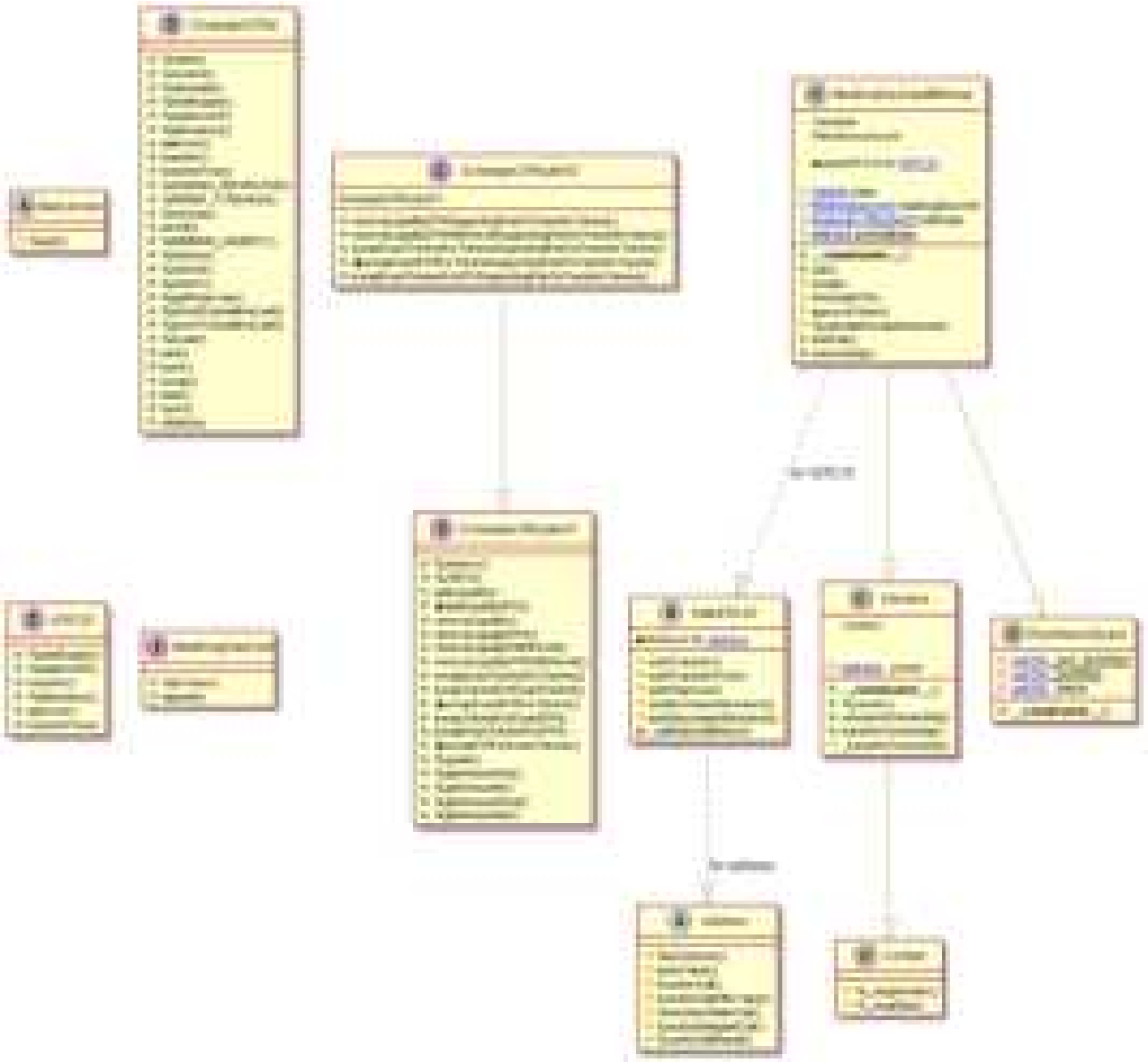
MadkingDaoChef Diagram



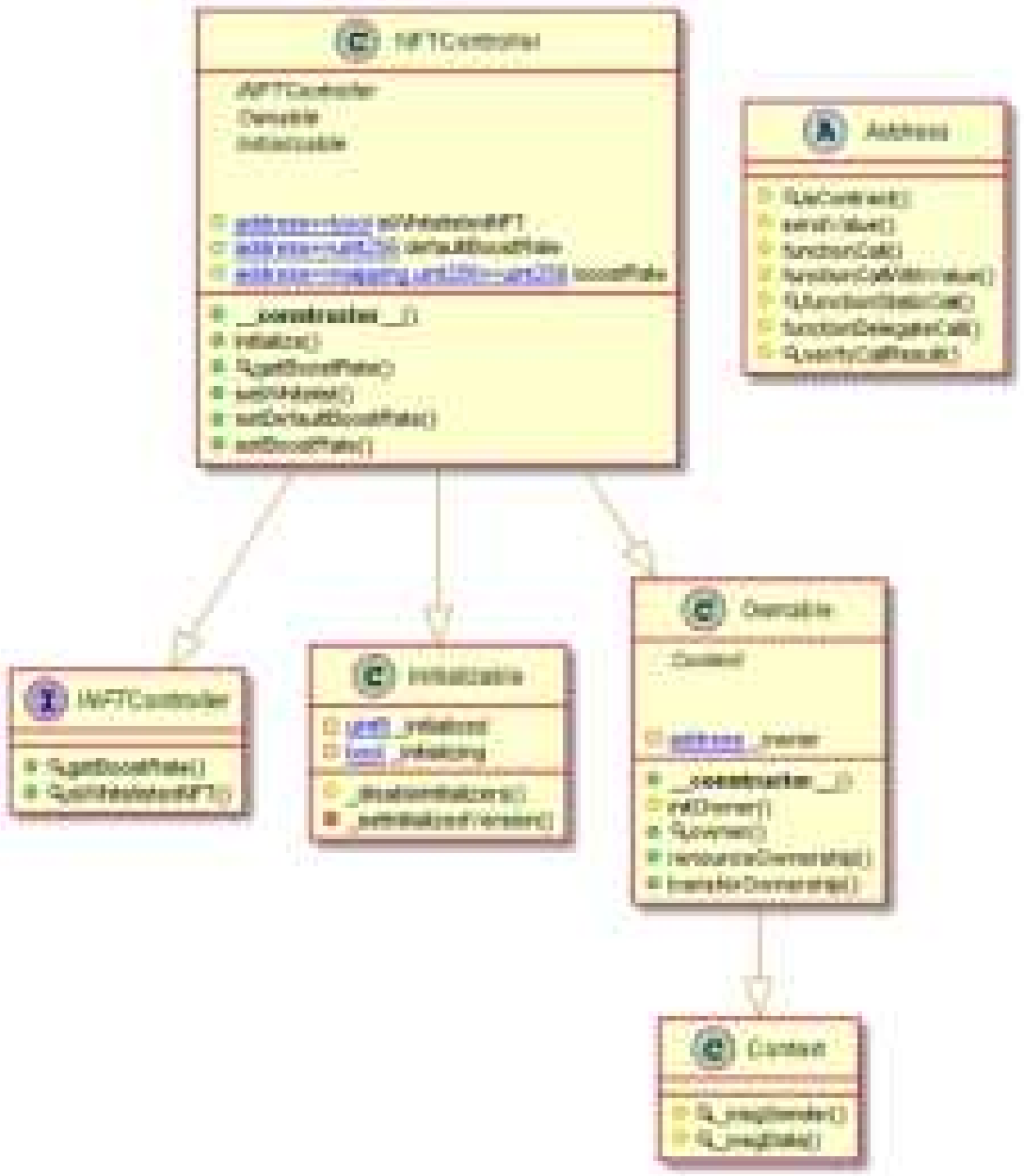
MadkingDaoStaking Diagram



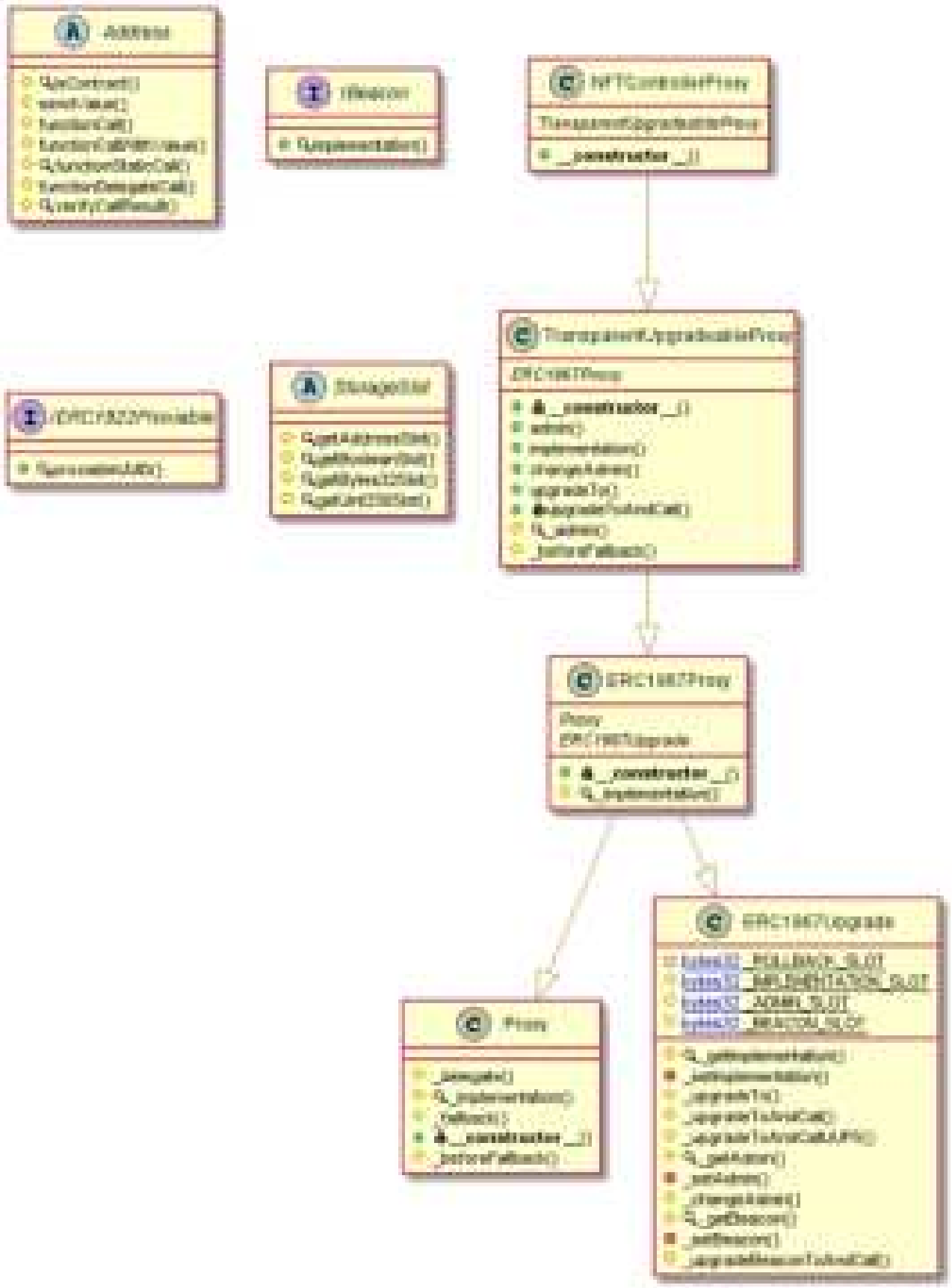
MadkingDaoZapMMSwap Diagram



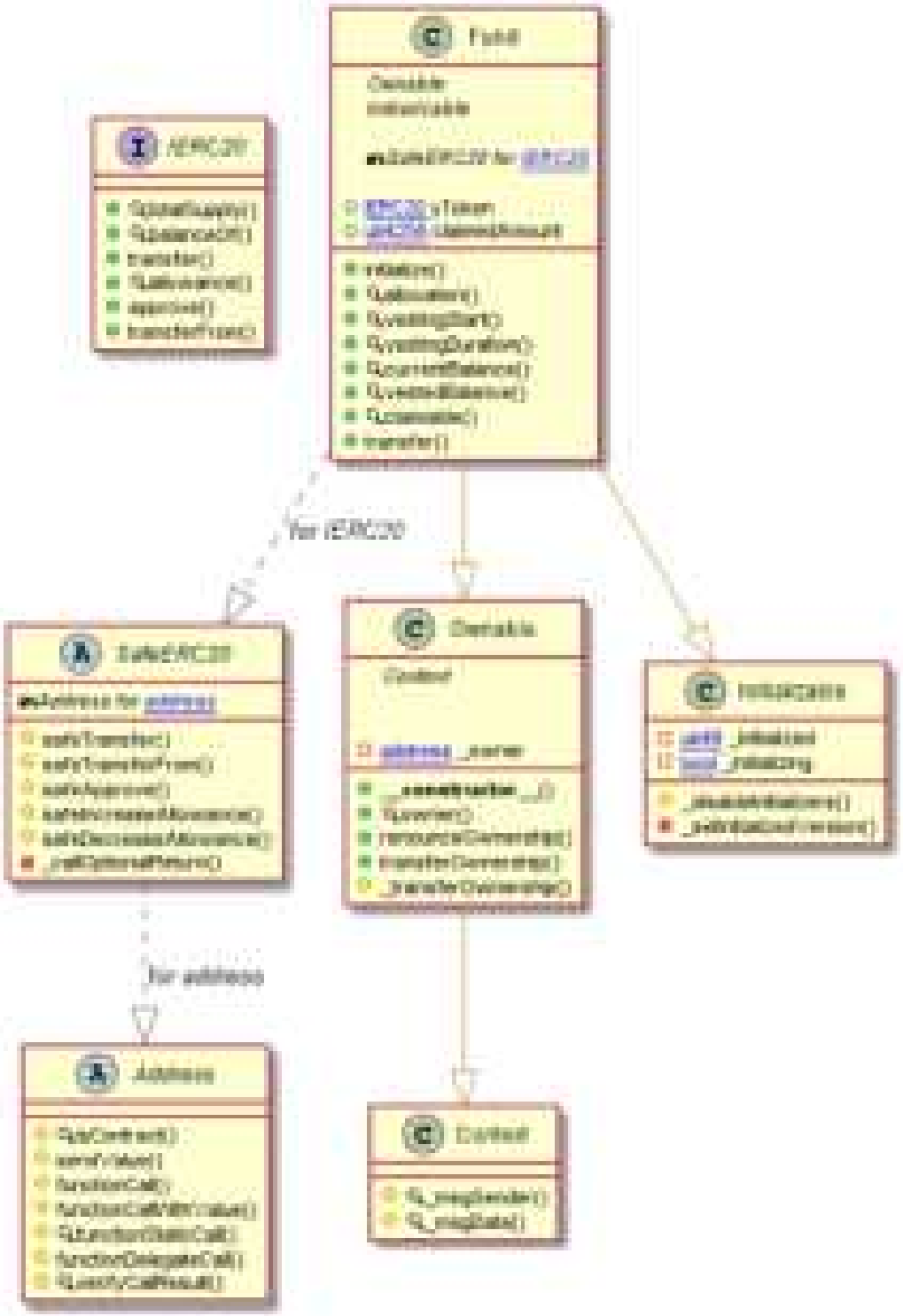
NFTController Diagram



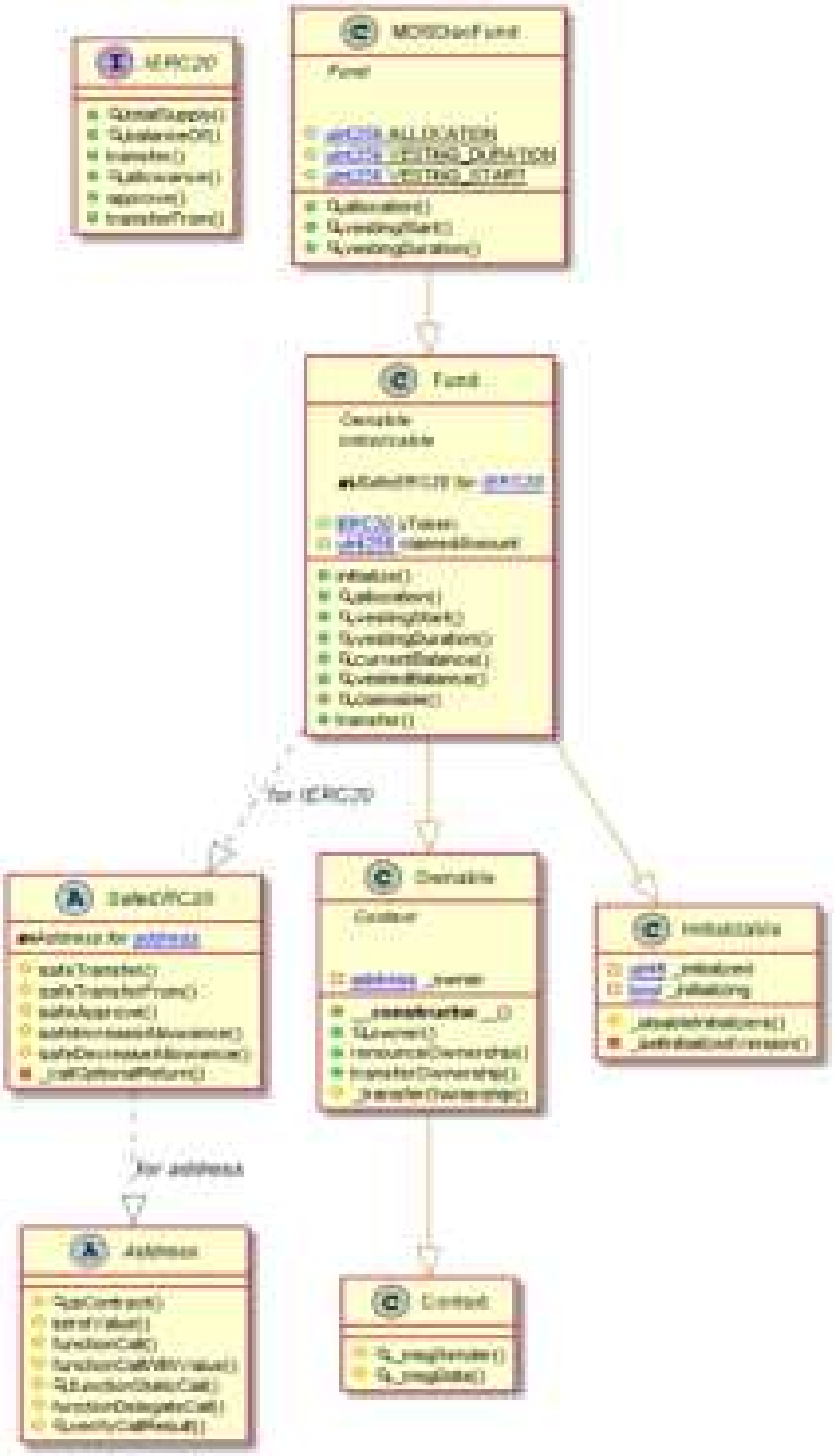
NFTControllerProxy Diagram



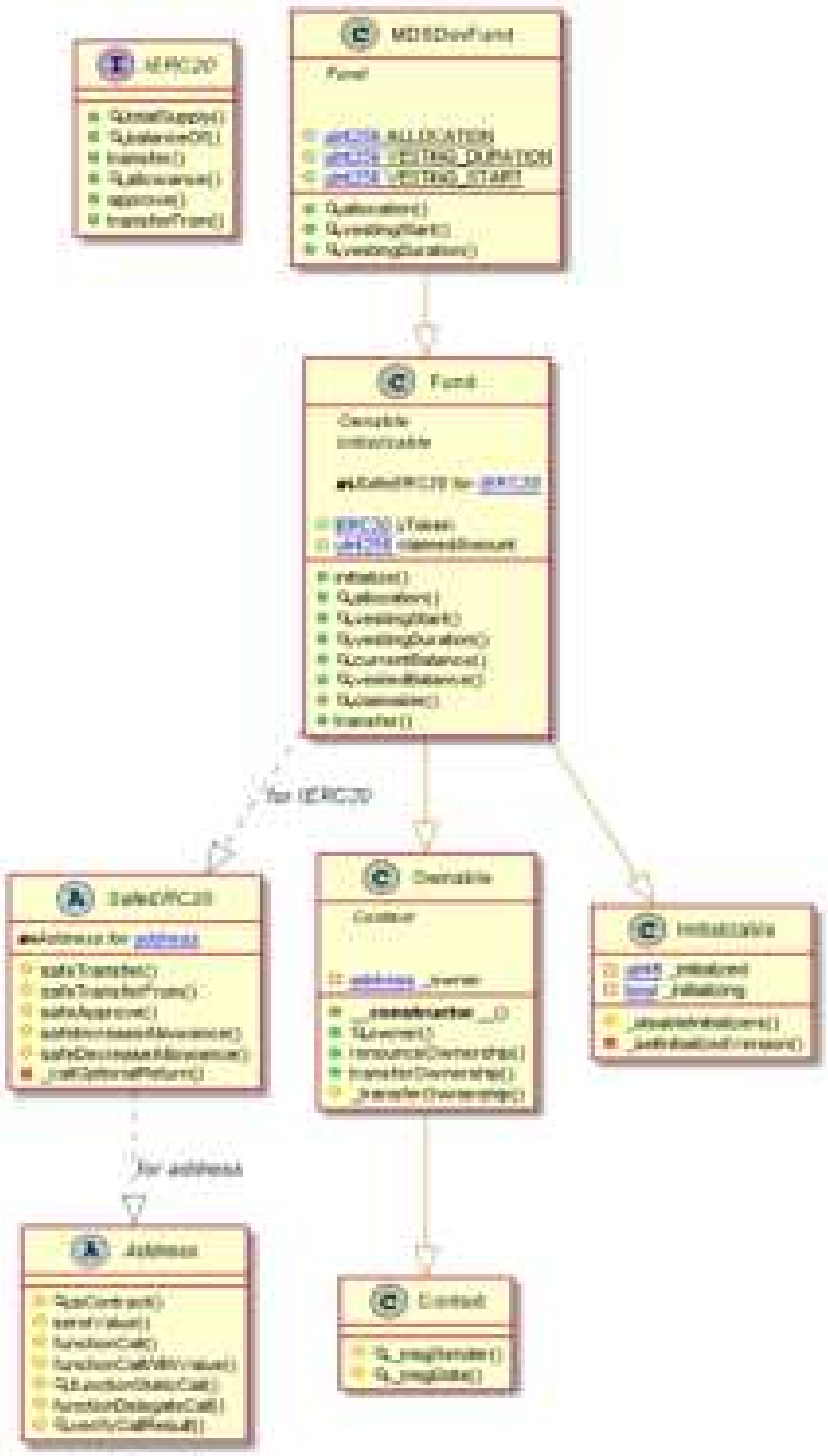
Fund Diagram



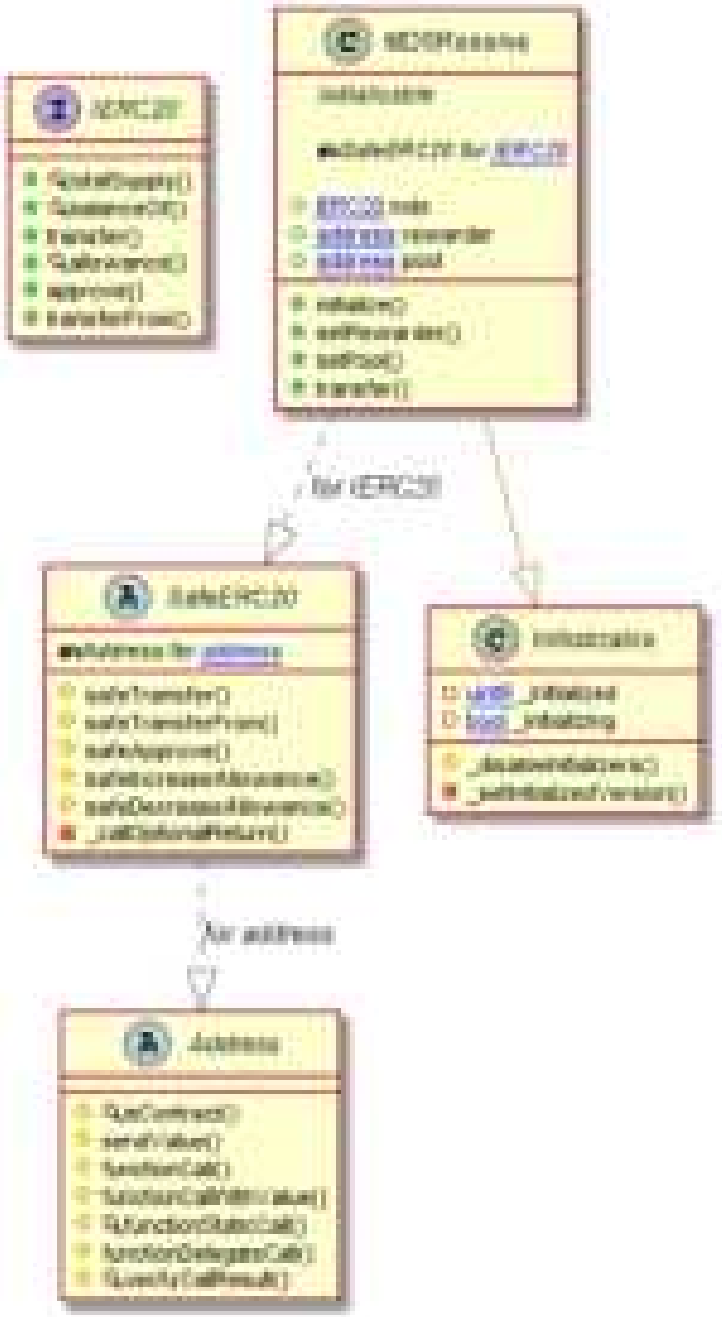
MDSDaoFund Diagram



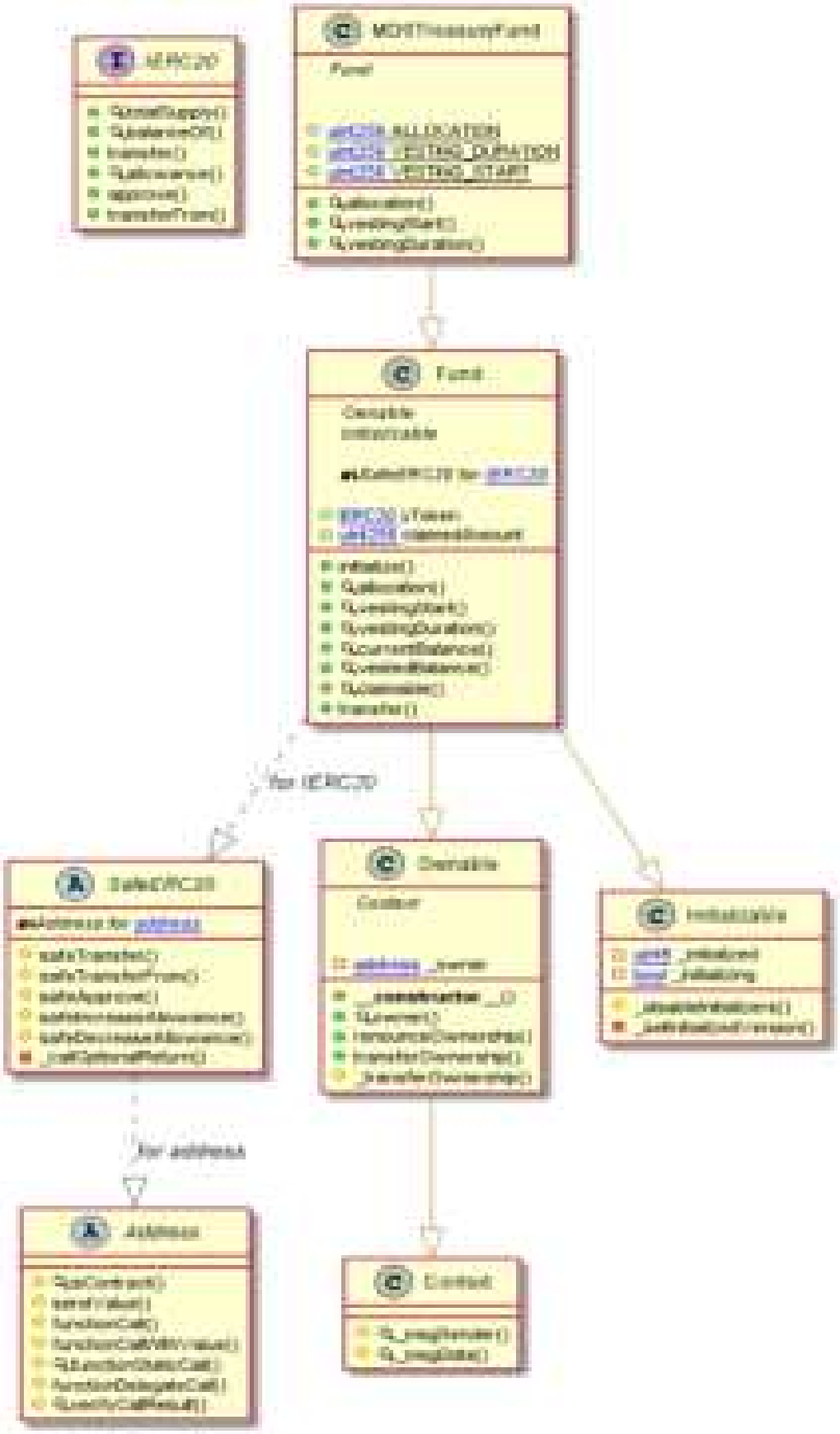
MDSDevFund Diagram



MDSReserve Diagram



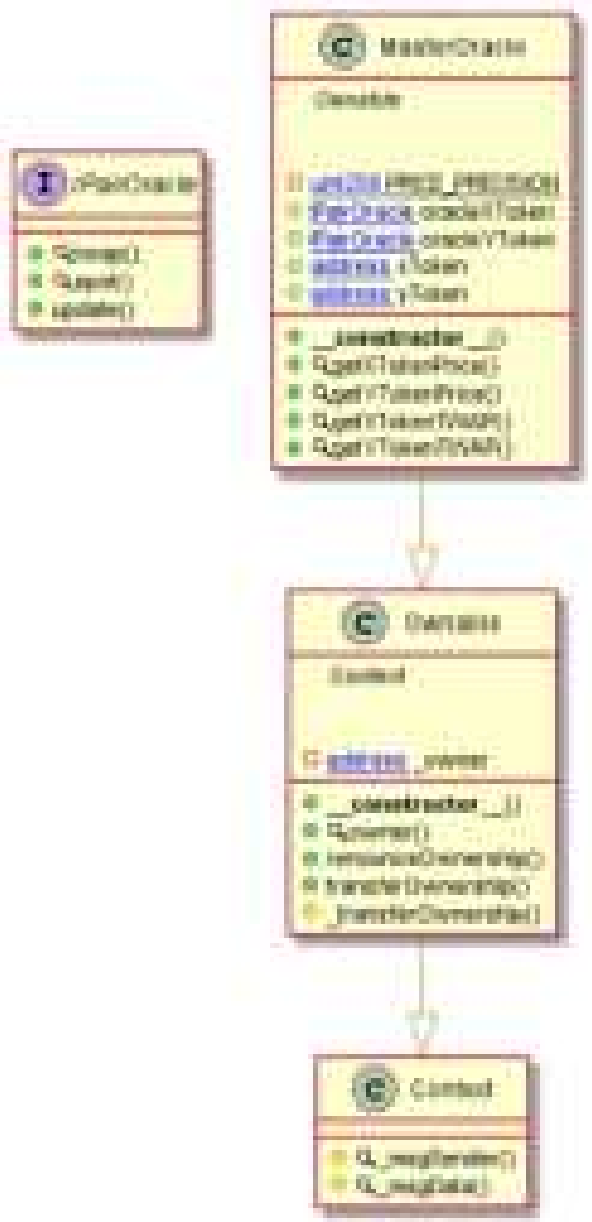
MDSTreasuryFund Diagram



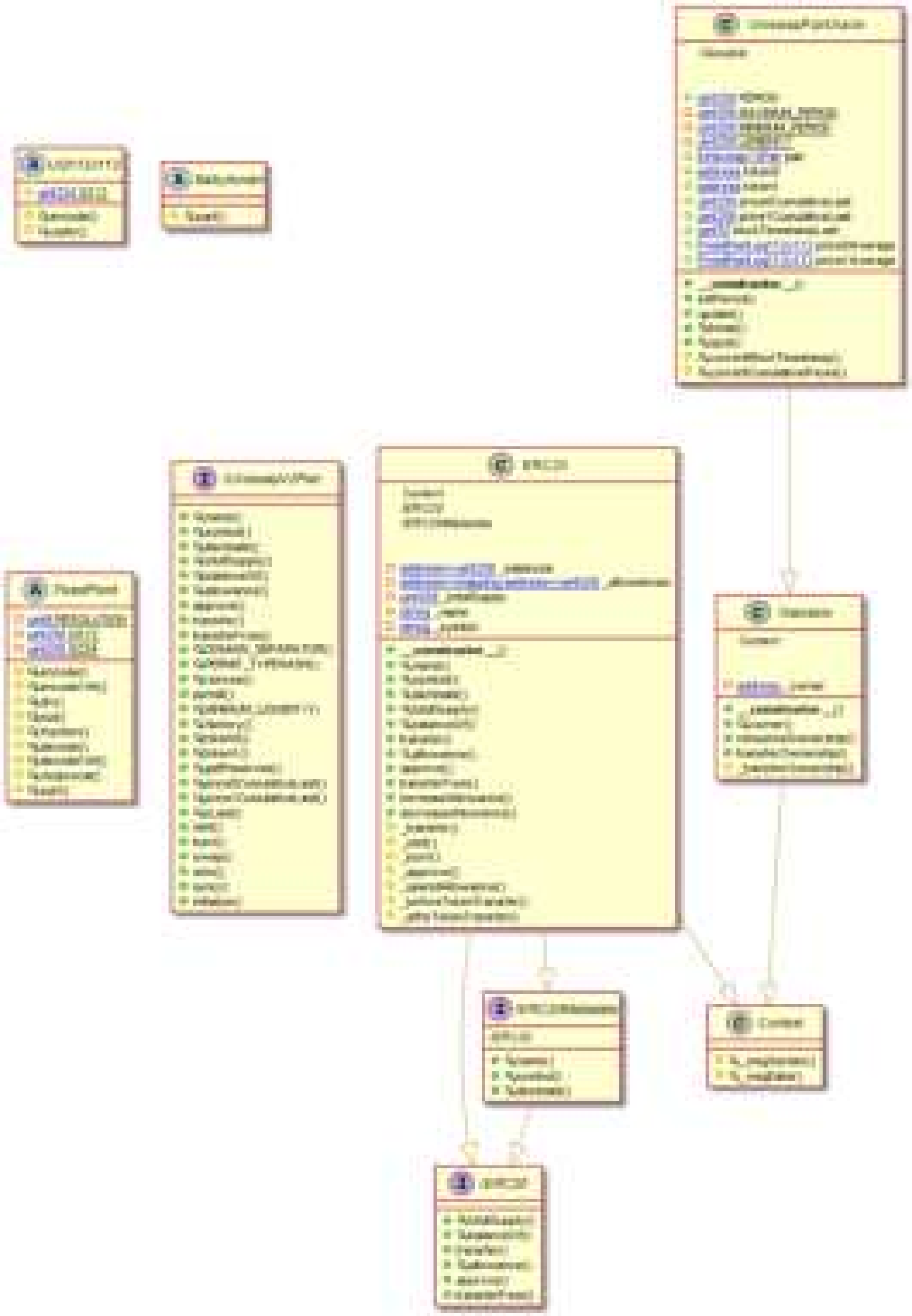
MockERC20 Diagram

MockTreasury Diagram

MasterOracle Diagram



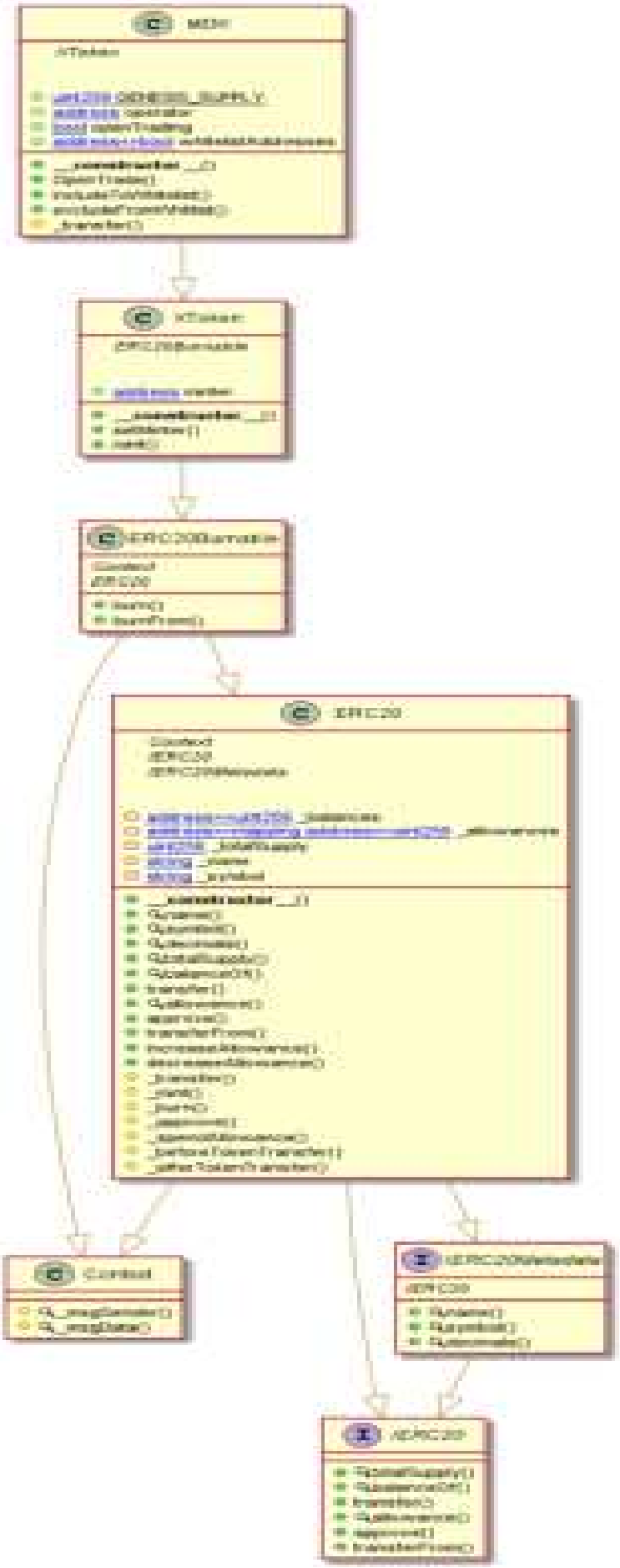
UniswapPairOracle Diagram



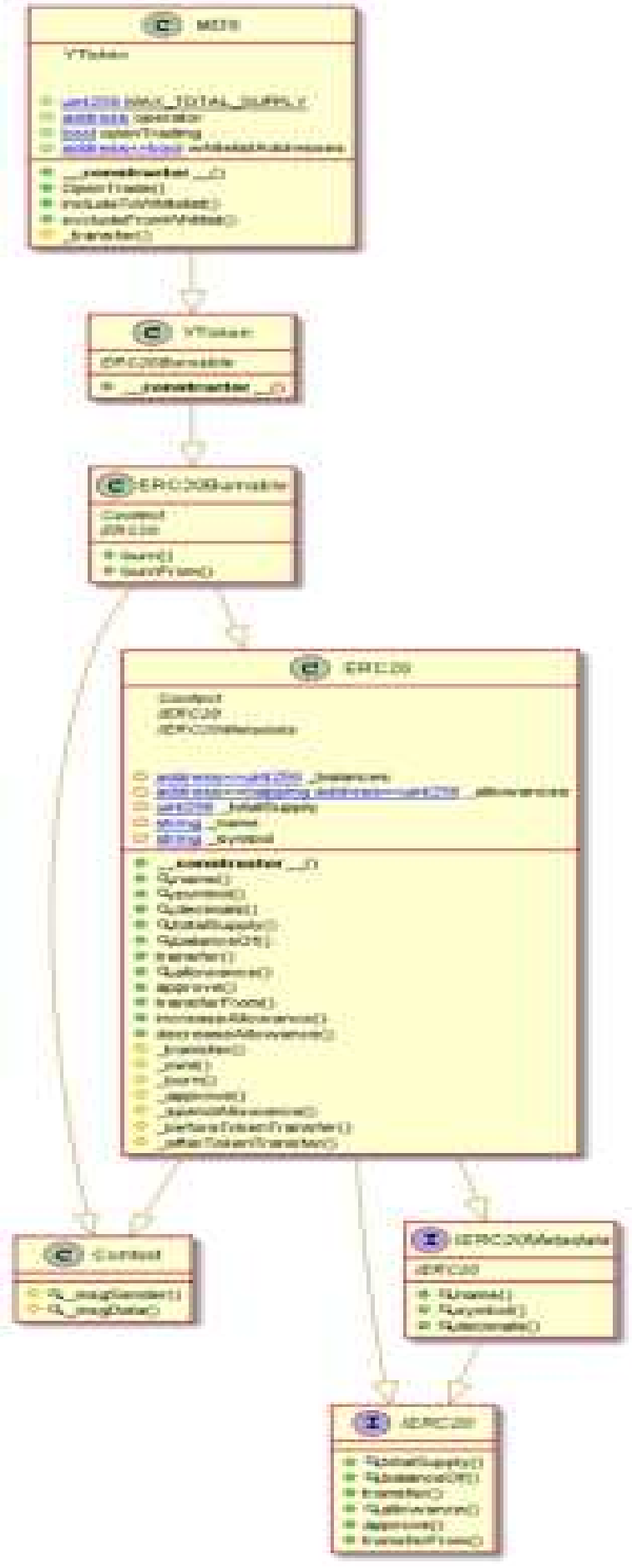
XToken Diagram

YToken Diagram

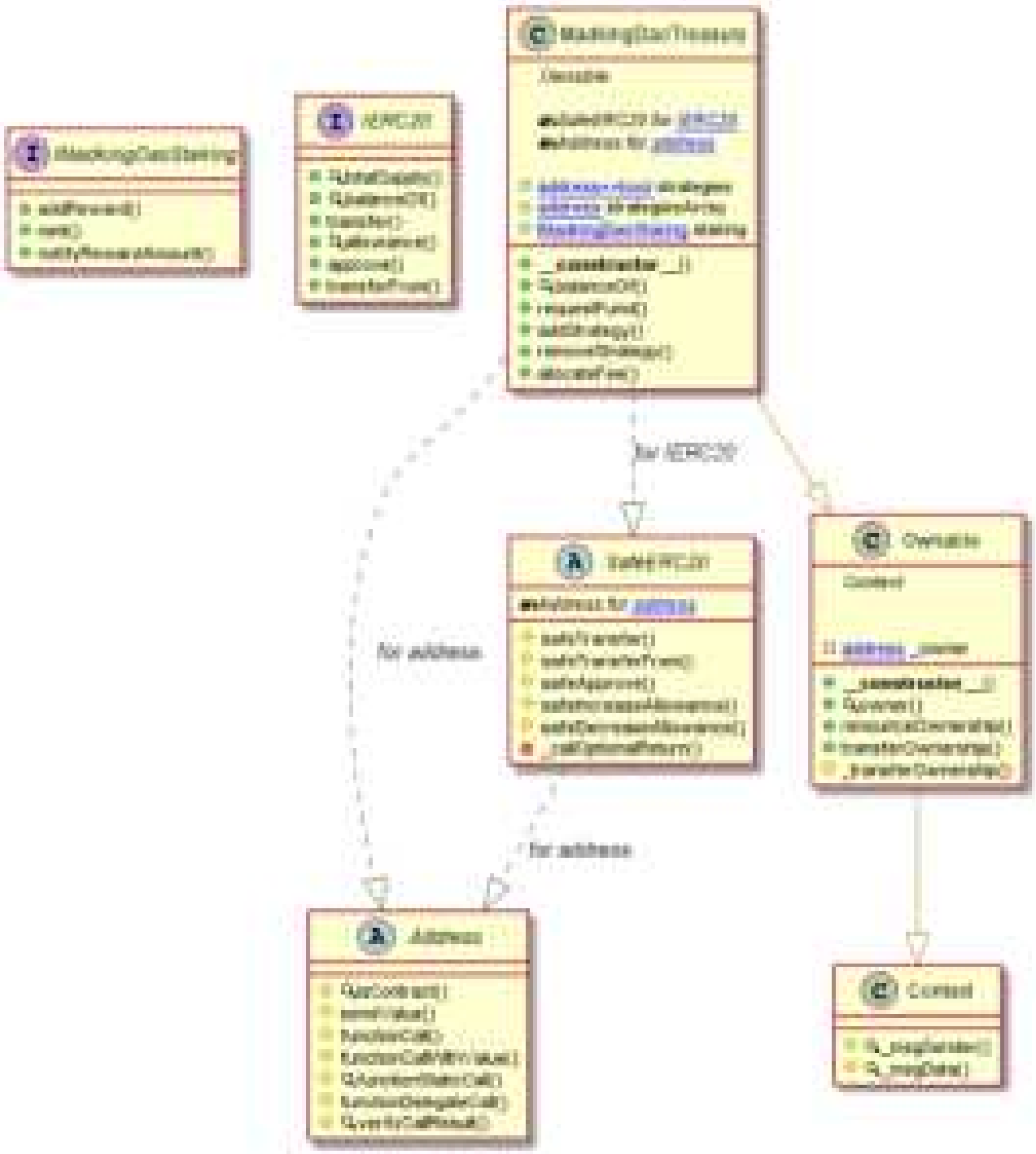
MDK Diagram



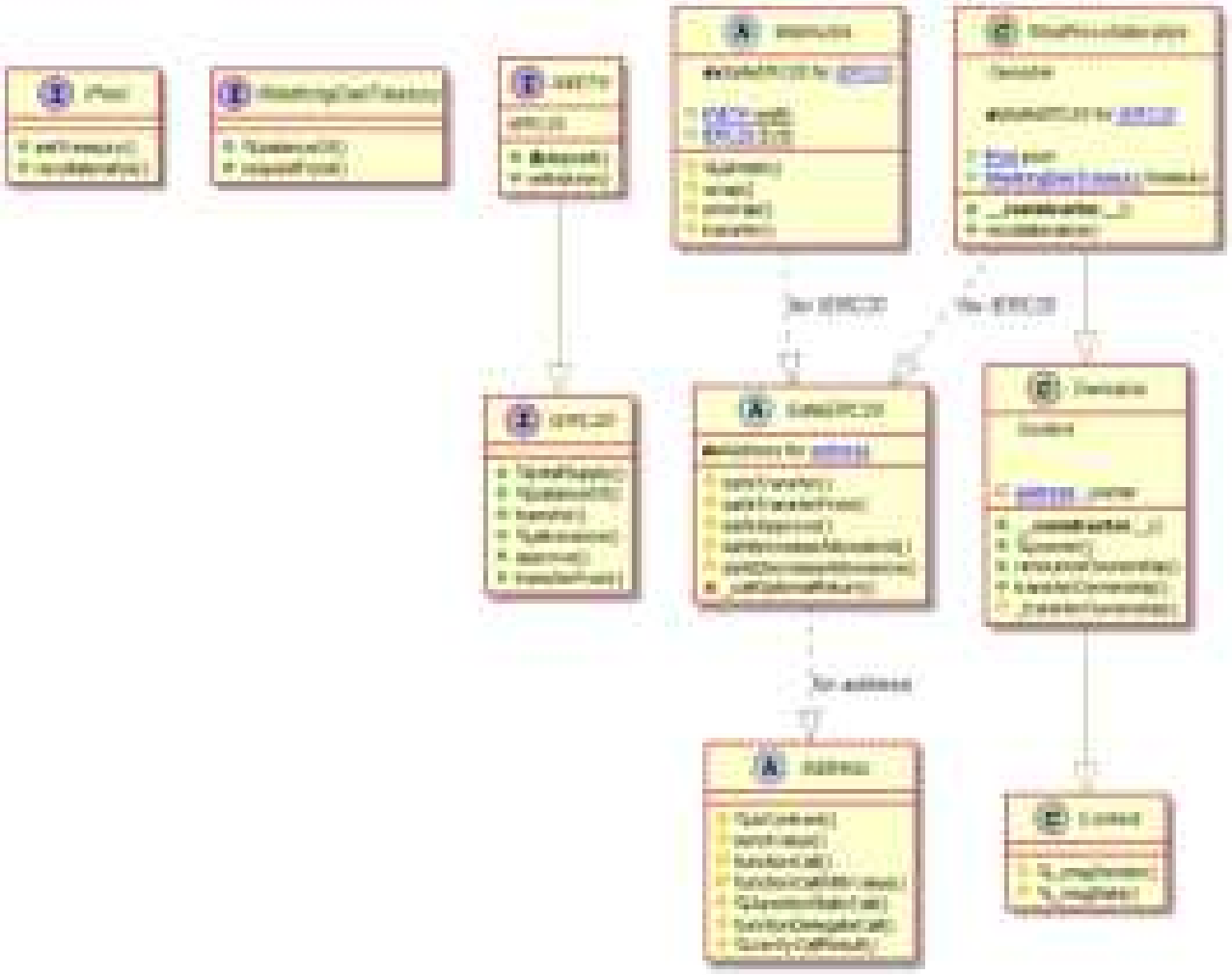
MDS Diagram



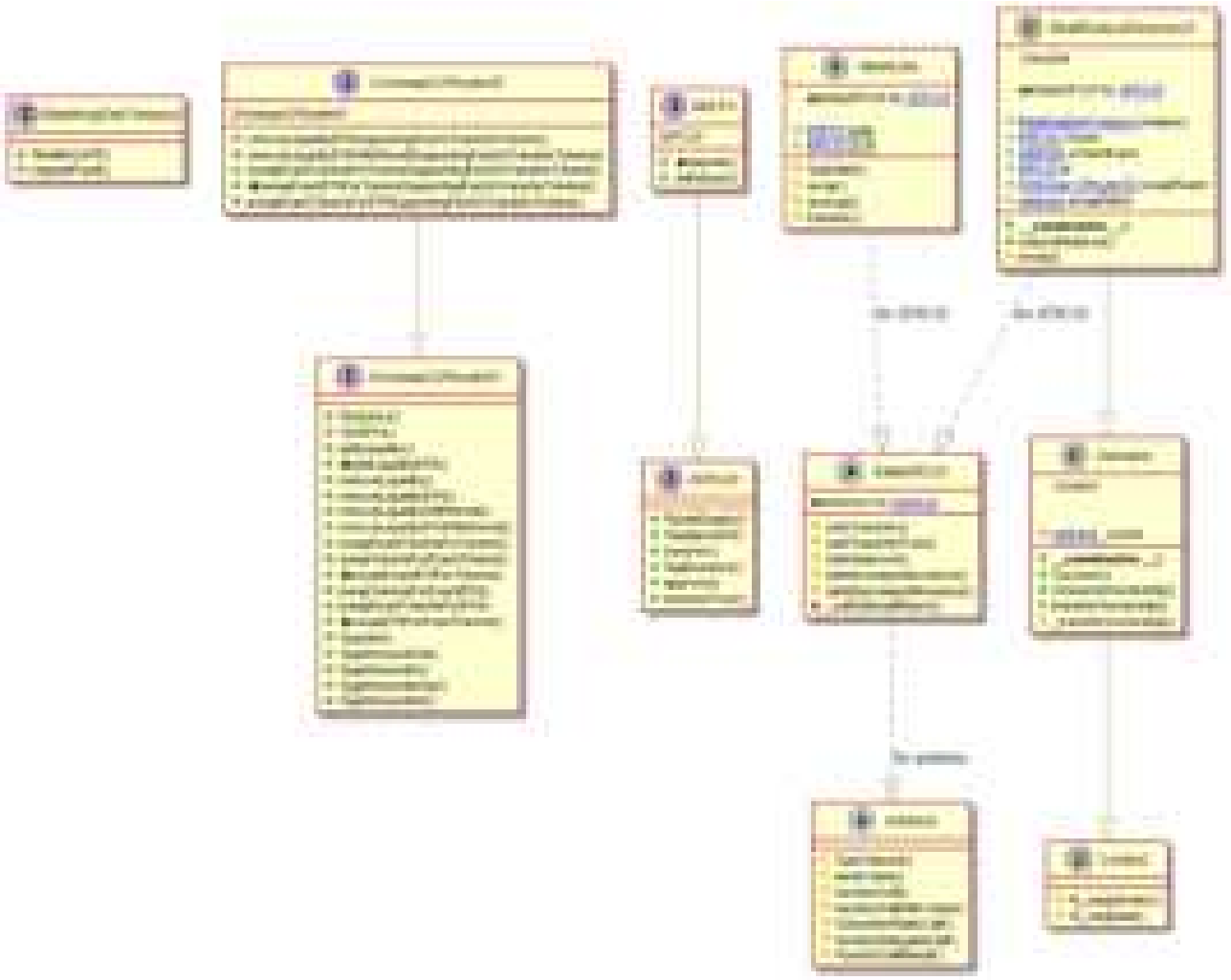
MadkingDaoTreasury Diagram



StratRecollateralize Diagram

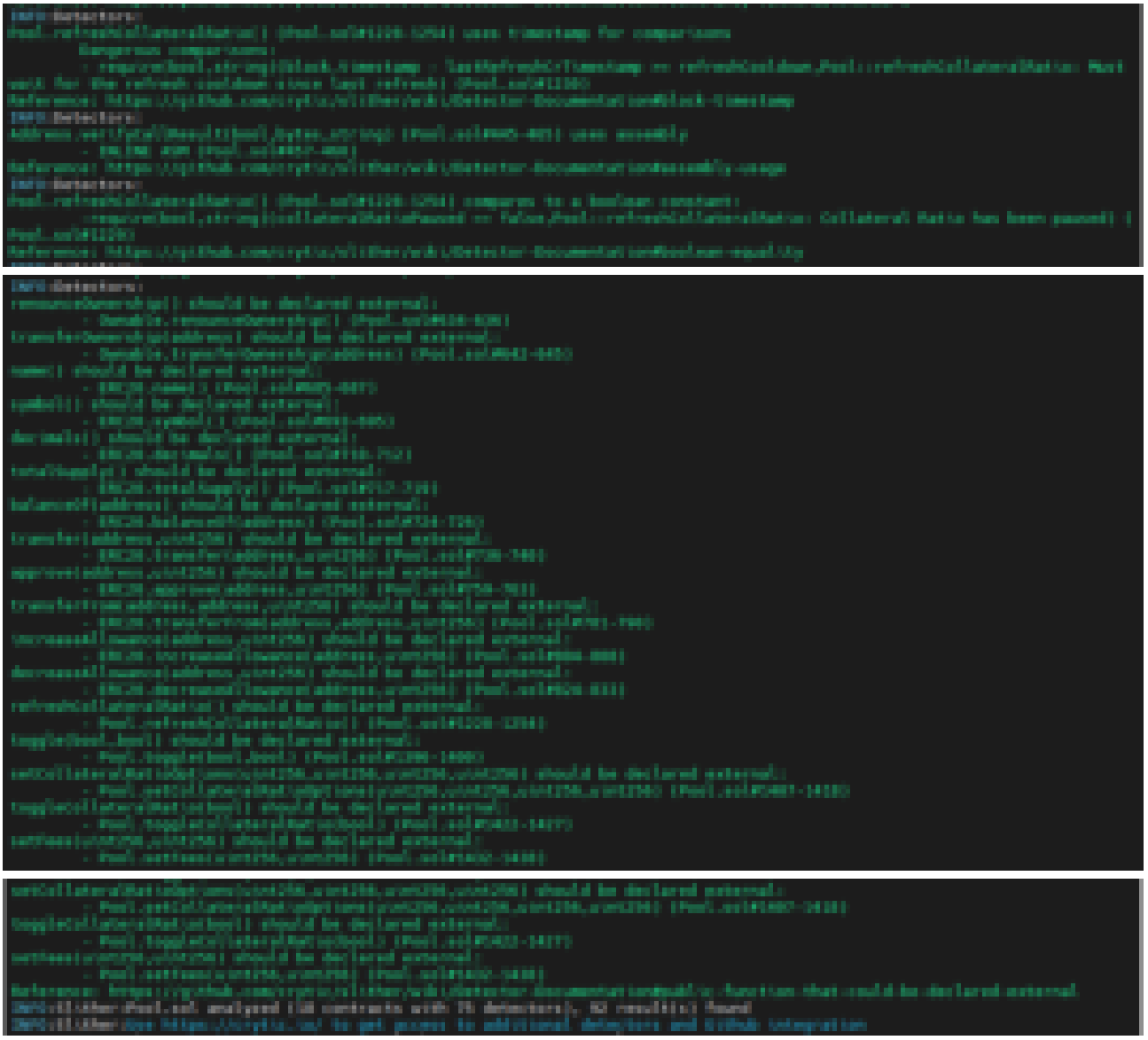


StratReduceReserveLP Diagram

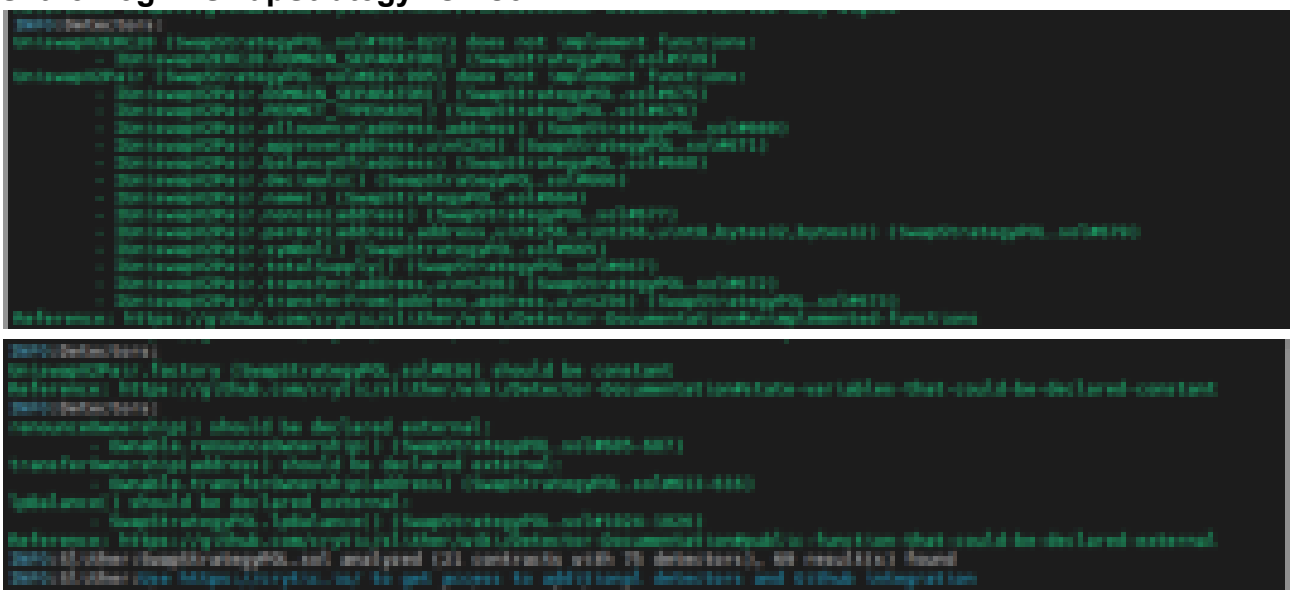


Slither Results Log

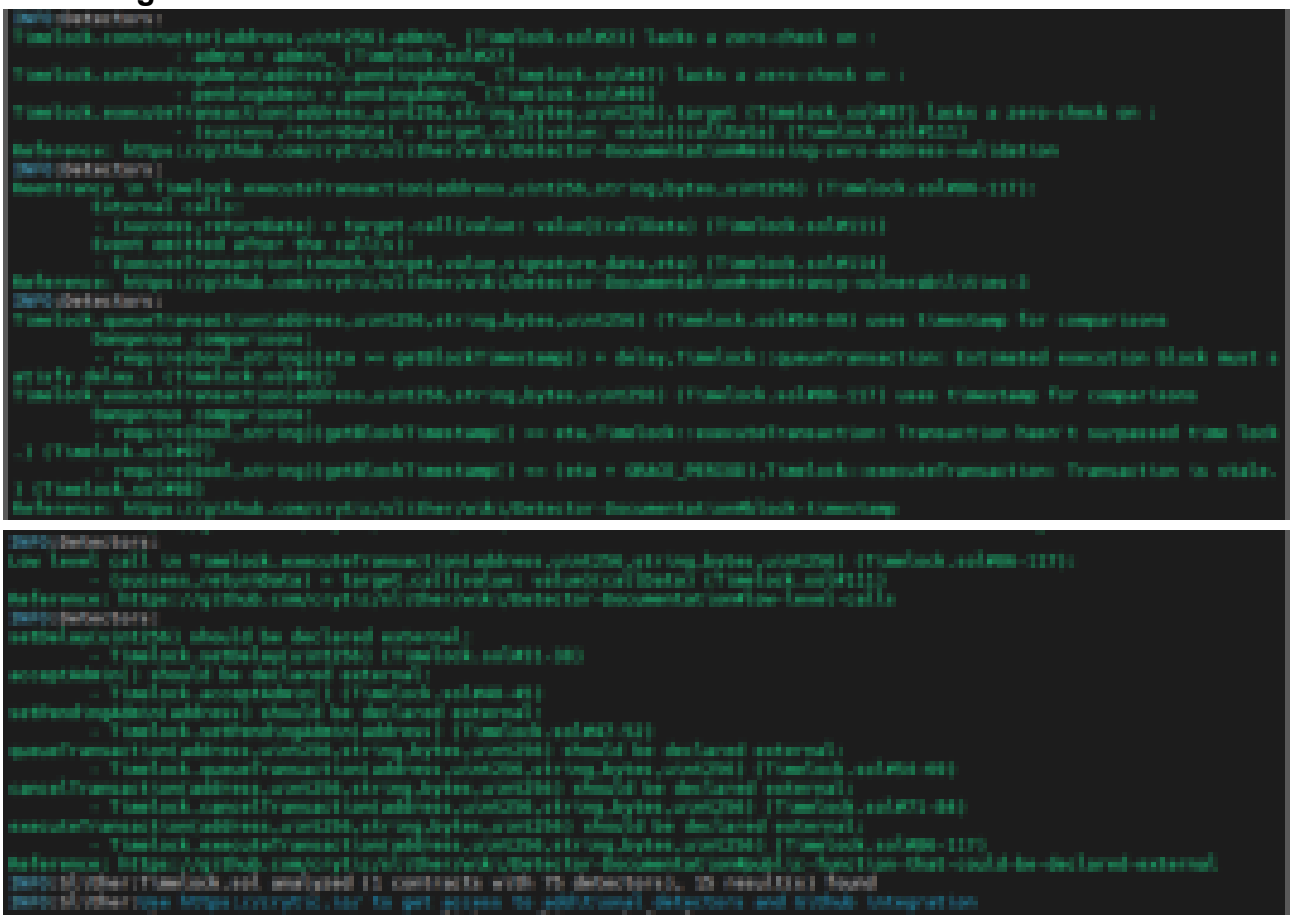
Slither log >>Pool.sol



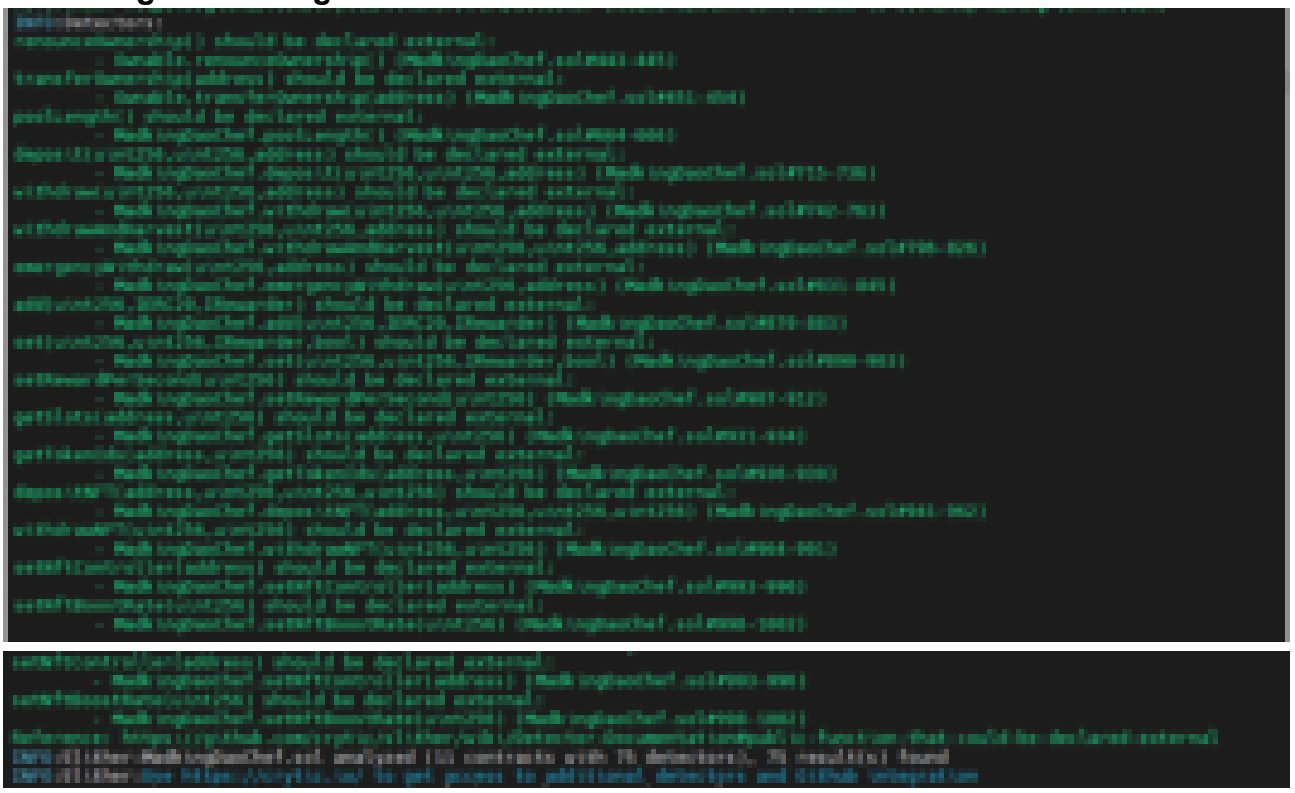
Slither log >>SwapStrategyPOL.sol



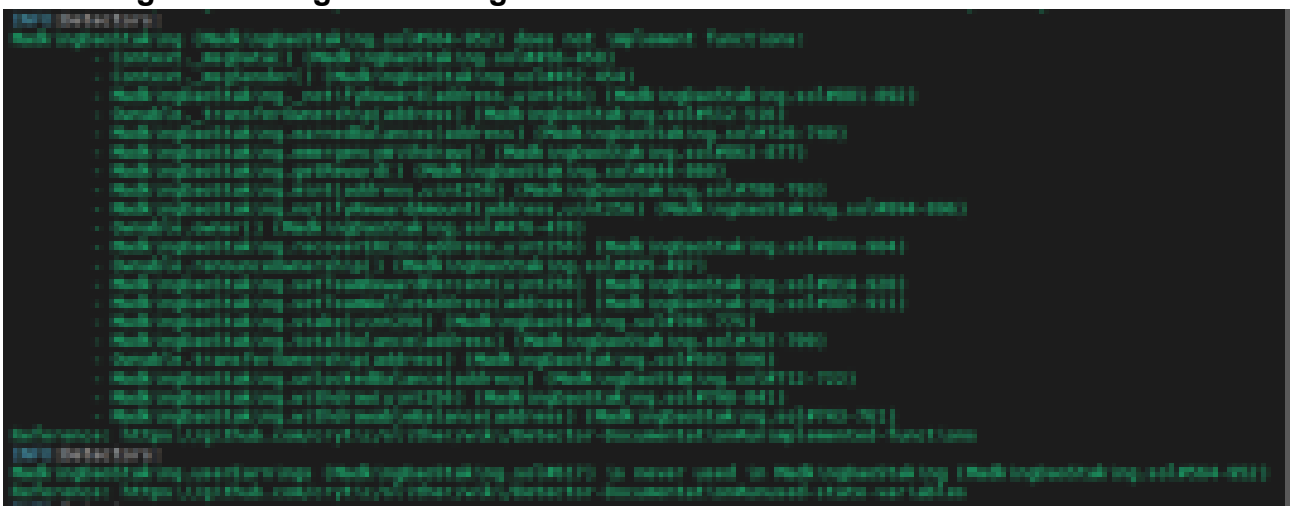
Slither log >>Timelock.sol

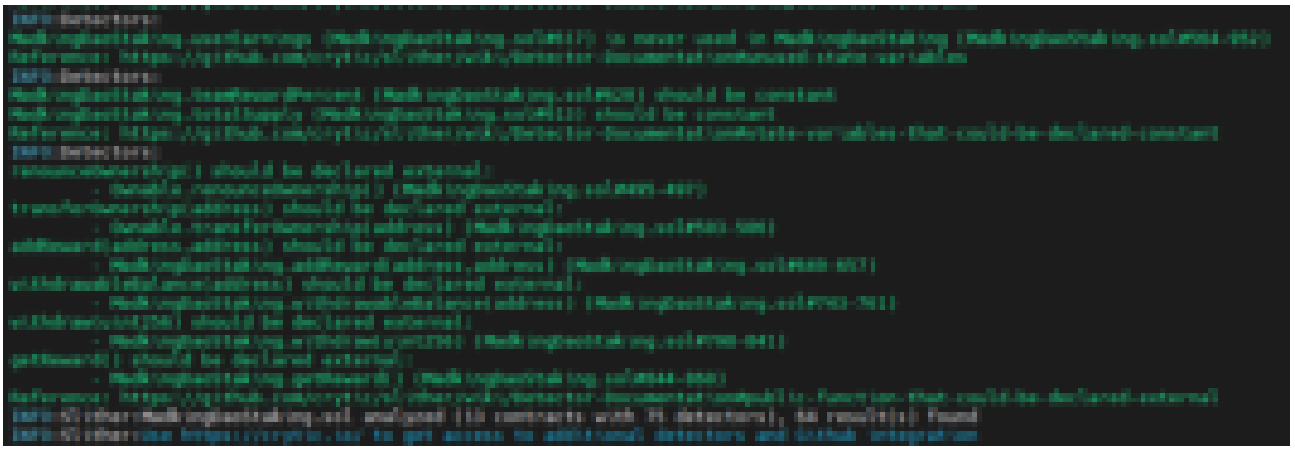


Slither log >>MadkingDaoChef.sol



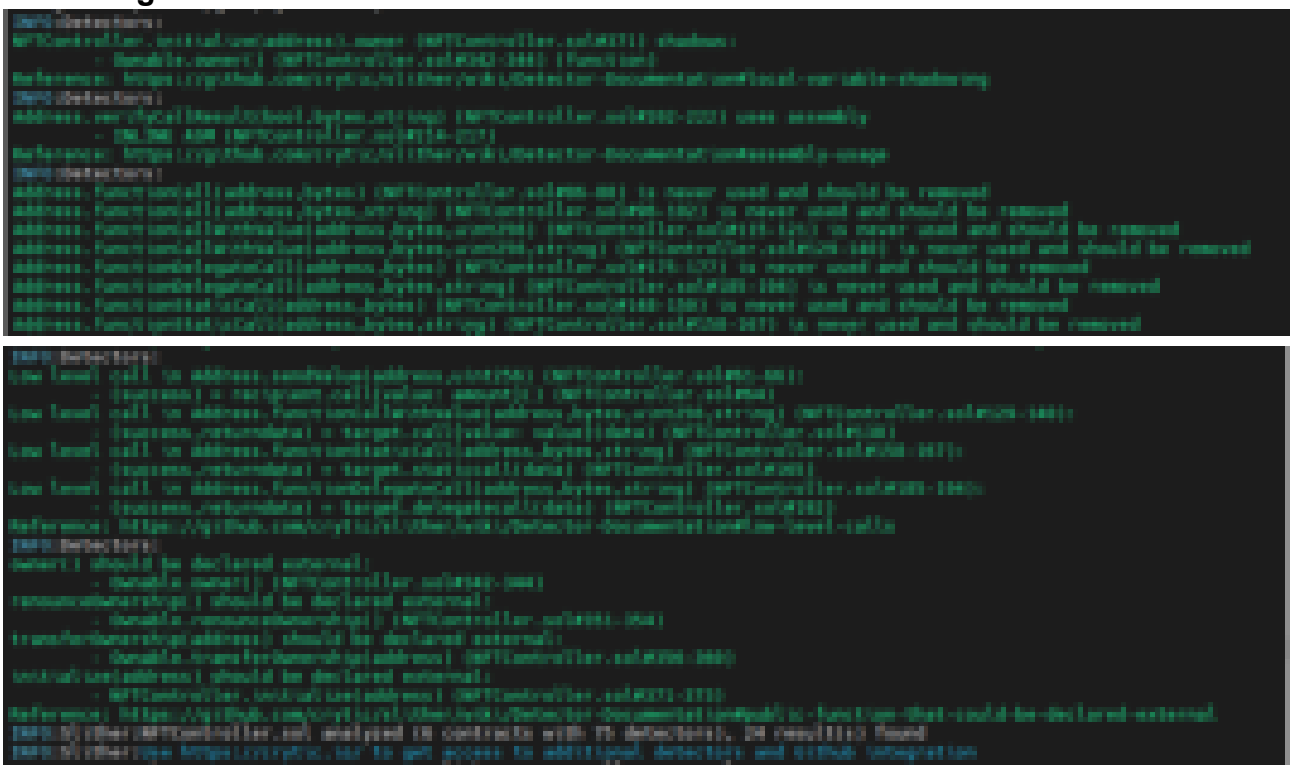
Slither log >>MadkingDaoStaking.sol



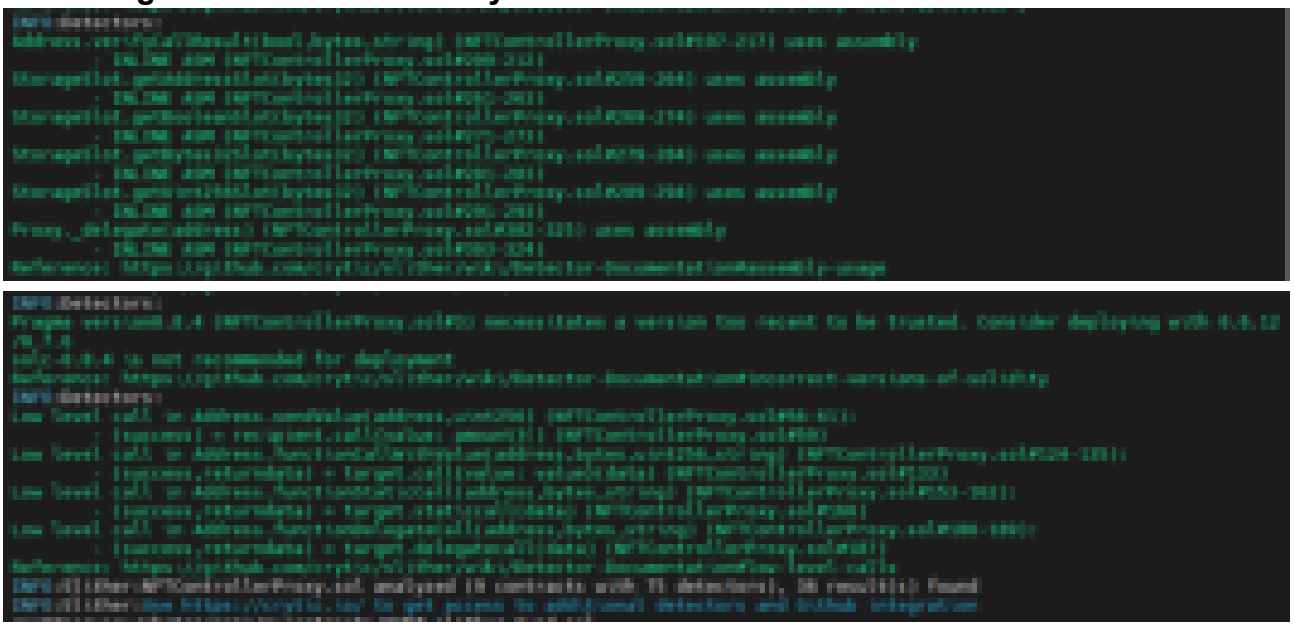


Slither log >>MadkingDaoZapMMSwap.sol

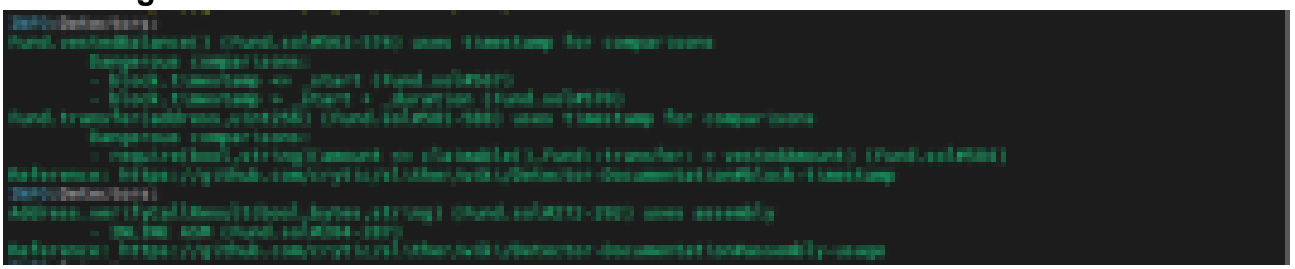
Slither log >>NFTController.sol

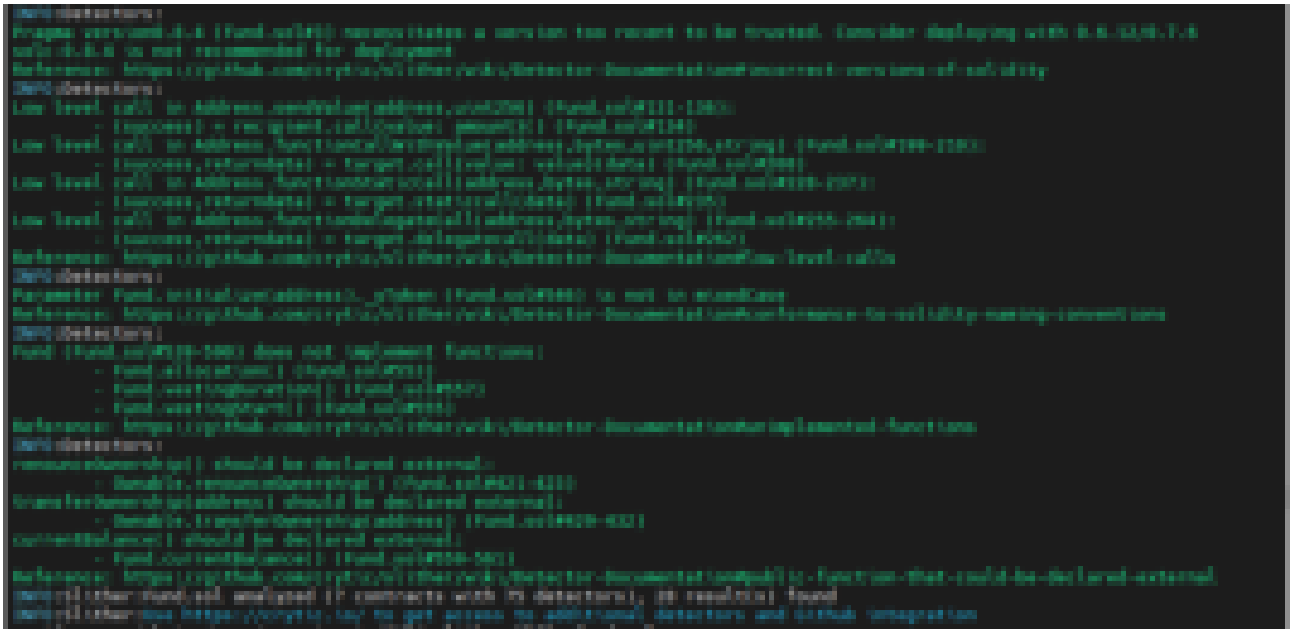


Slither log >>NFTControllerProxy.sol

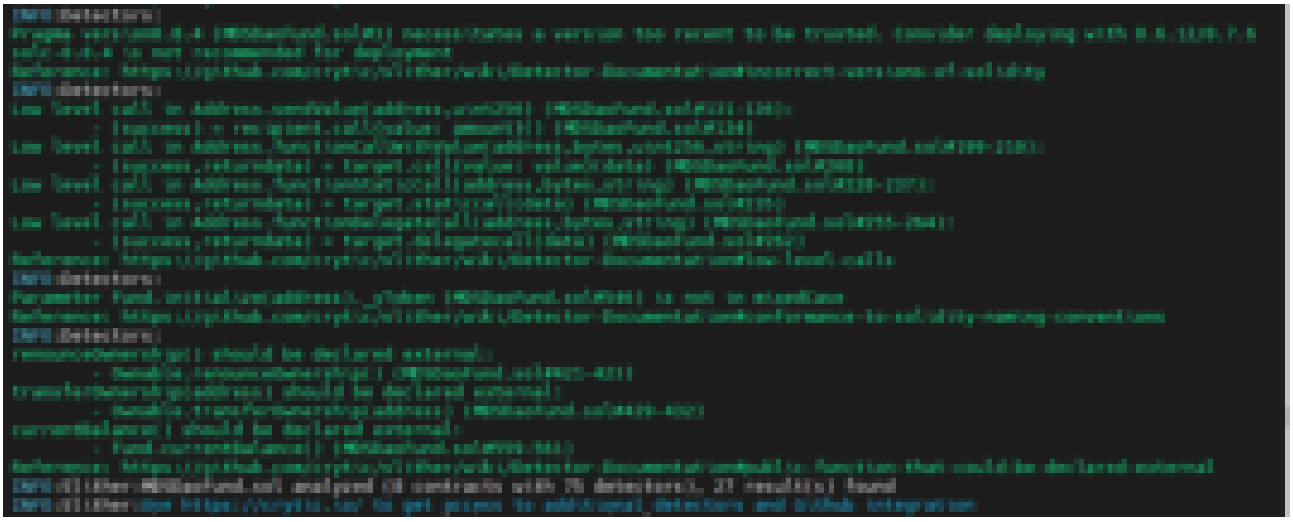


Slither log >>Fund.sol

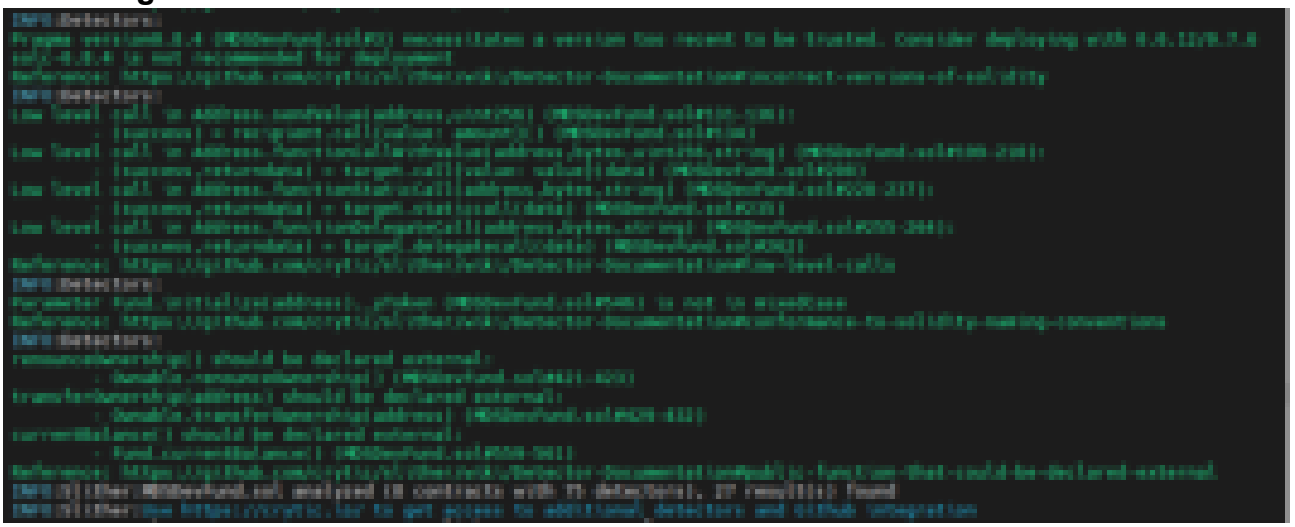




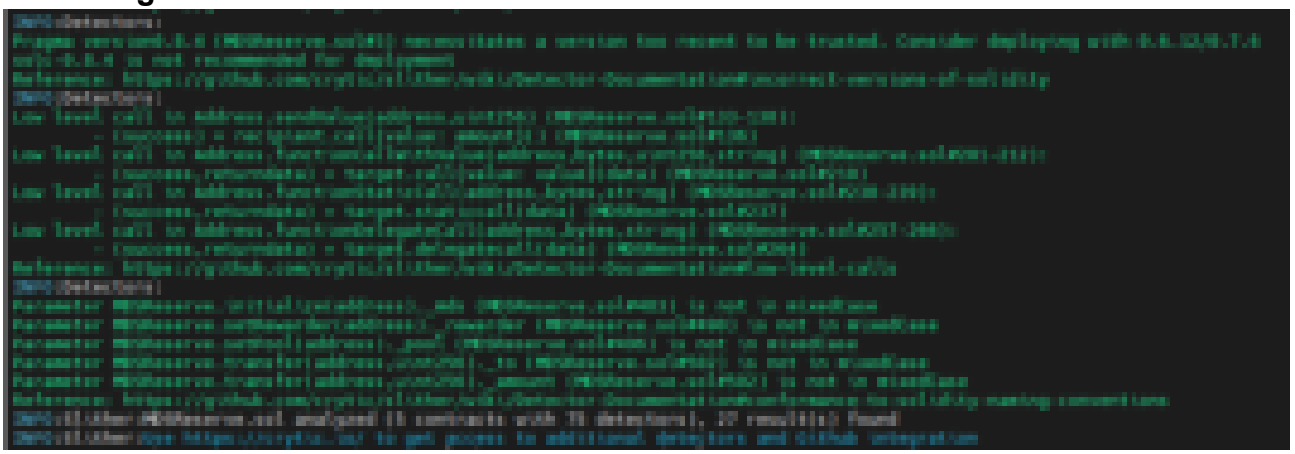
Slither log >>MDSDaoFund.sol



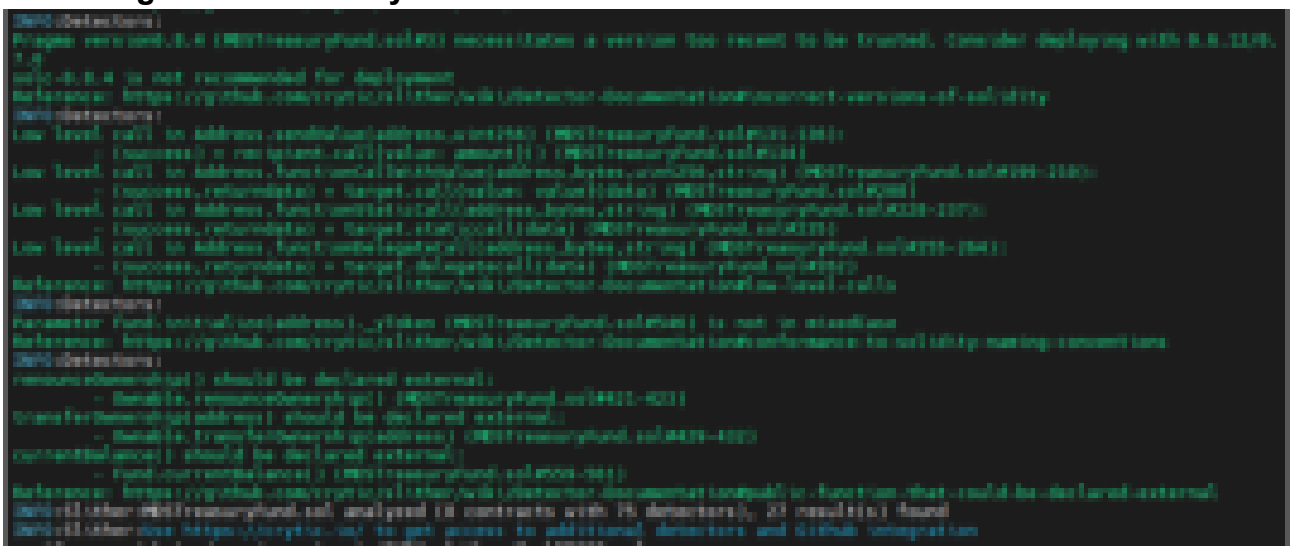
Slither log >>MDSDevFund.sol



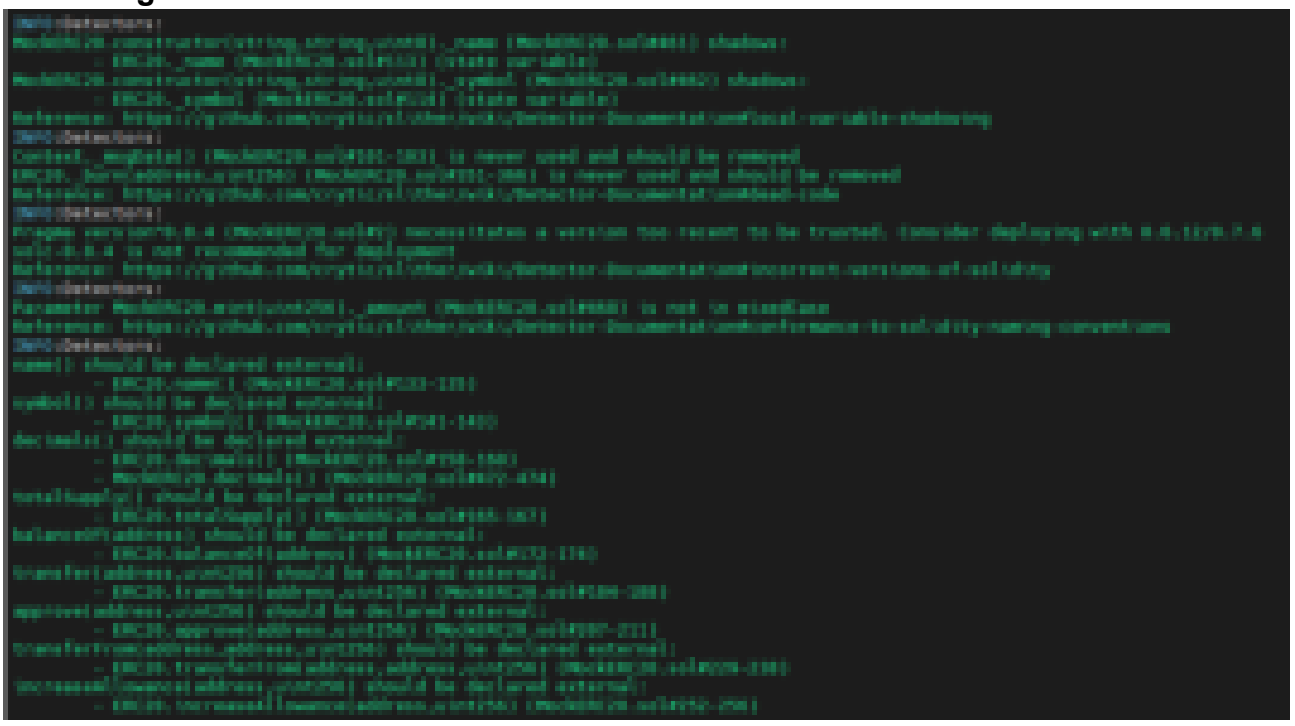
Slither log >>MDSReserve.sol



Slither log >>MDSTreasuryFund.sol

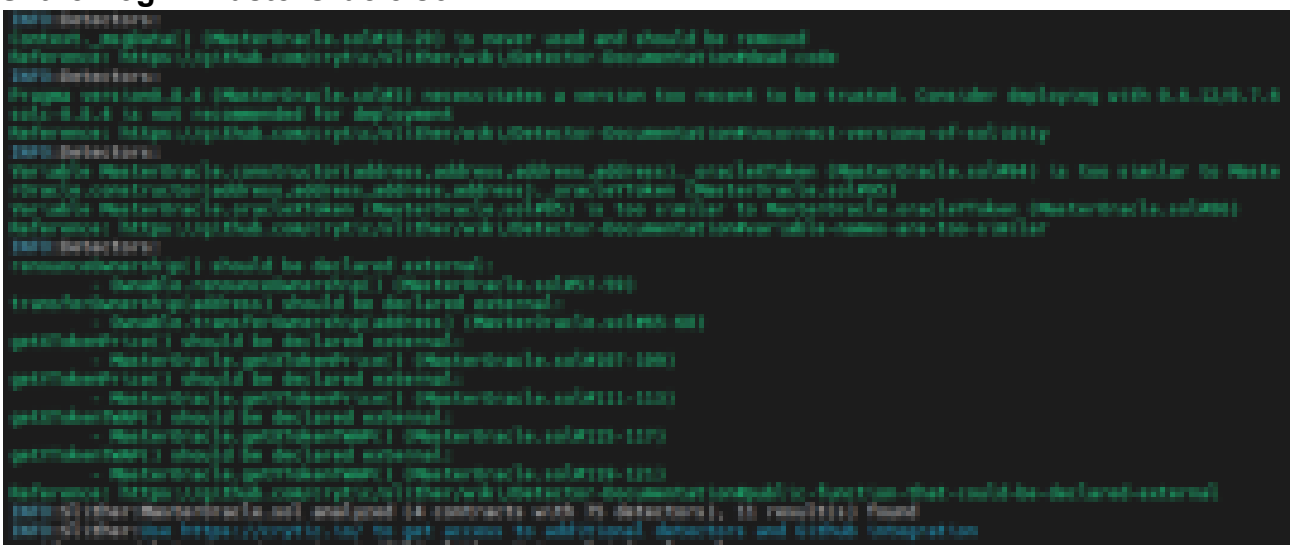


Slither log >>MockERC20.sol

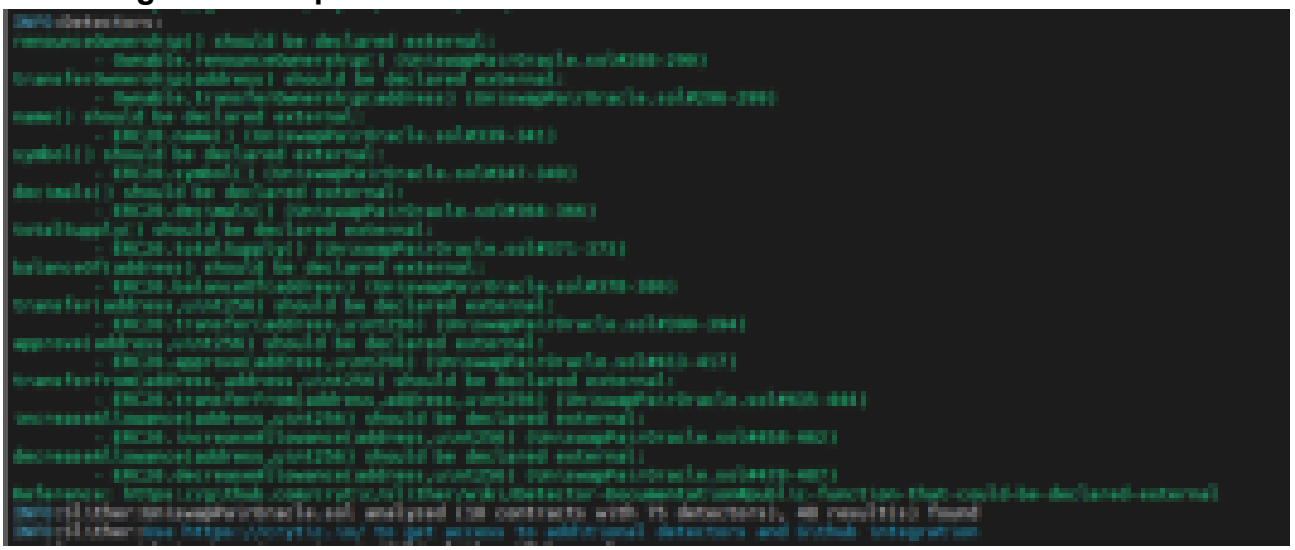


Slither log >>MockTreasury.sol

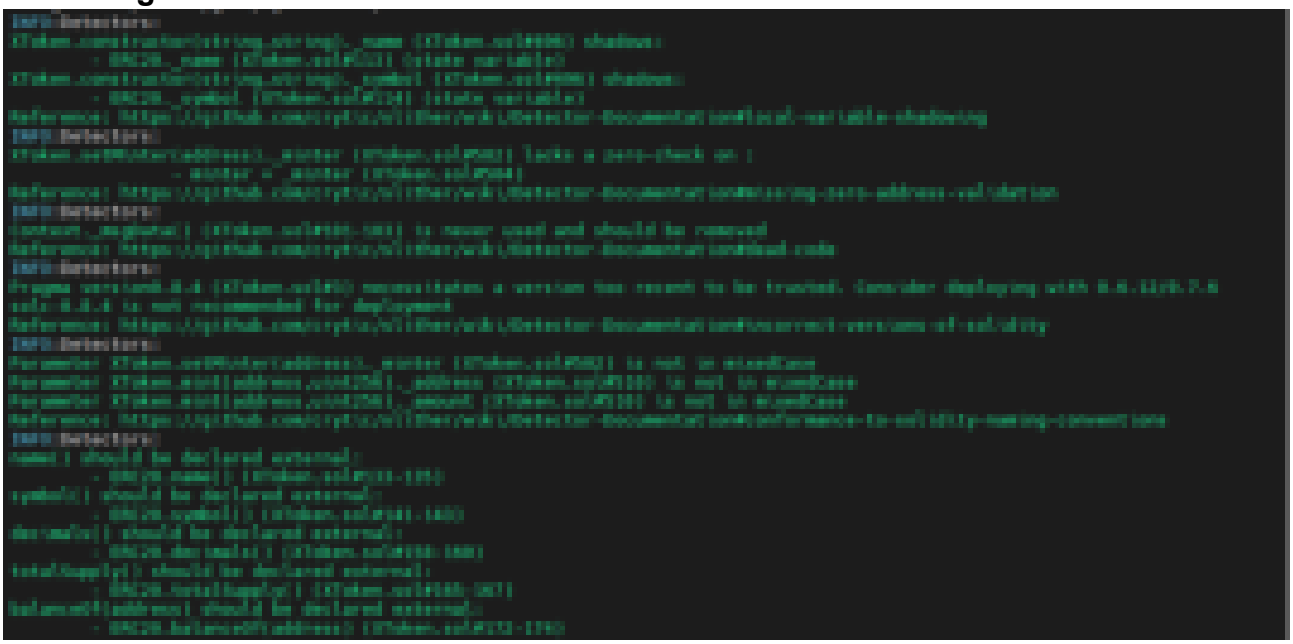
Slither log >>MasterOracle.sol



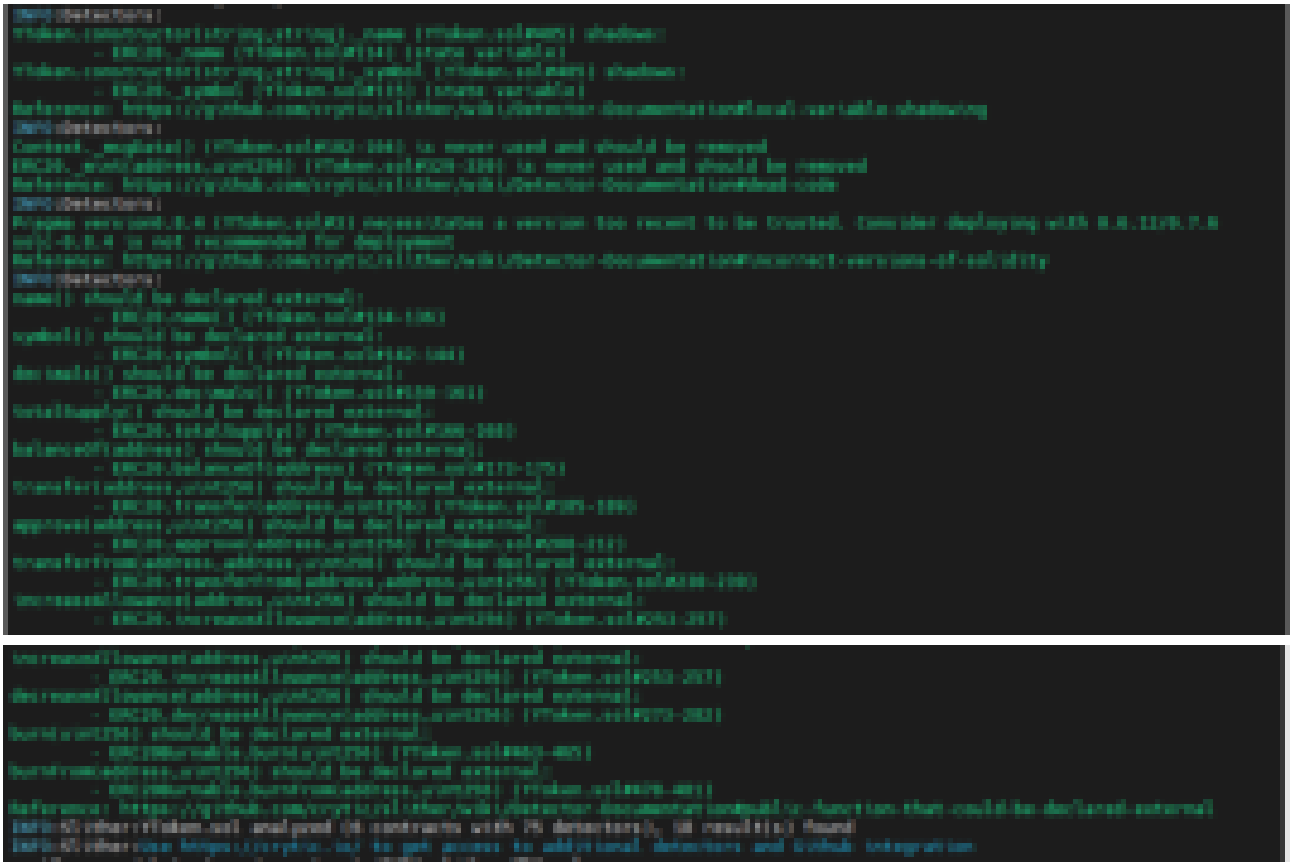
Slither log >>UniswapPairOracle.sol



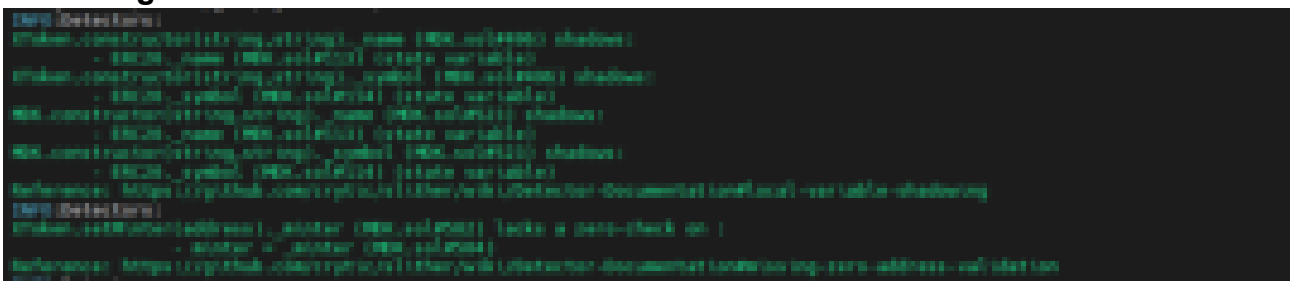
Slither log >>XToken.sol

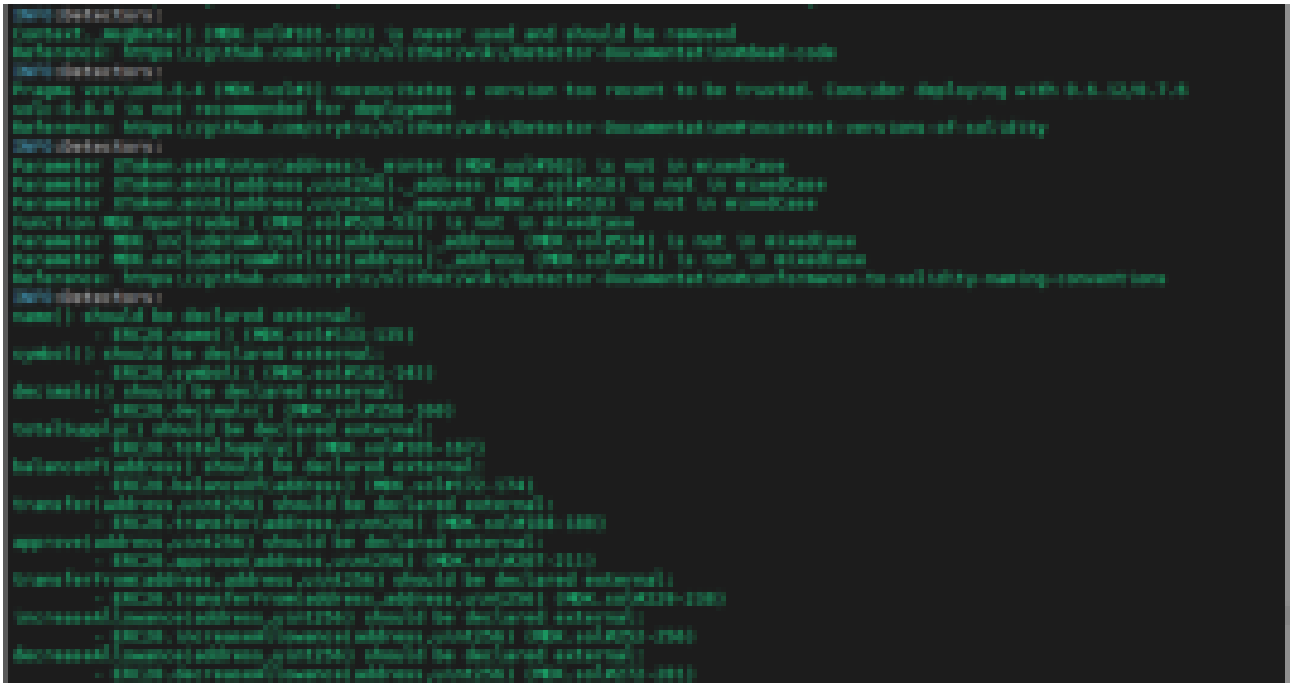


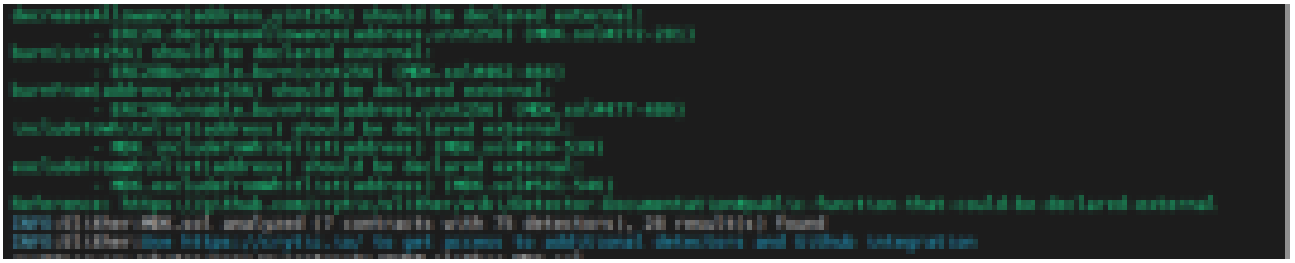
Slither log >>YToken.sol



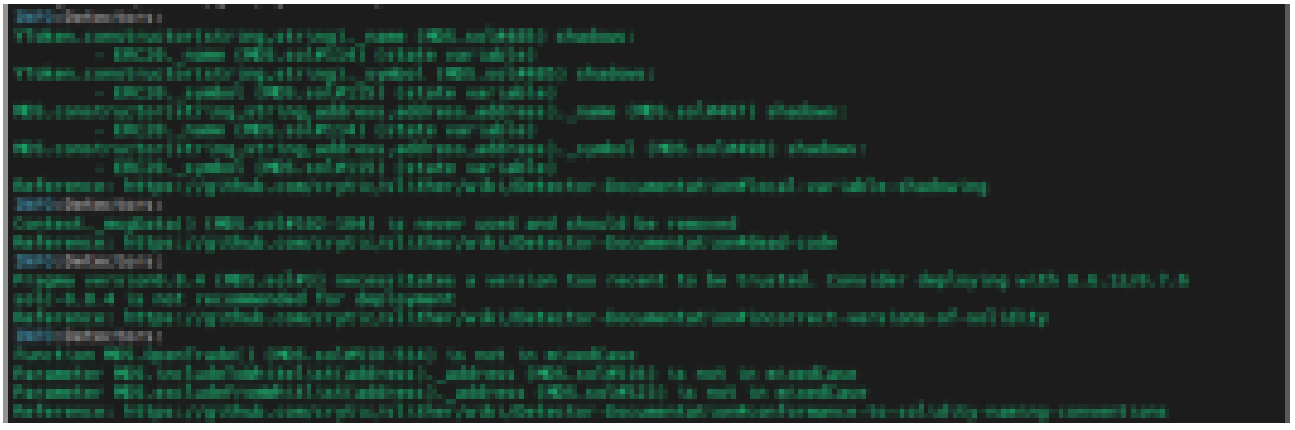
Slither log >>MDK.sol

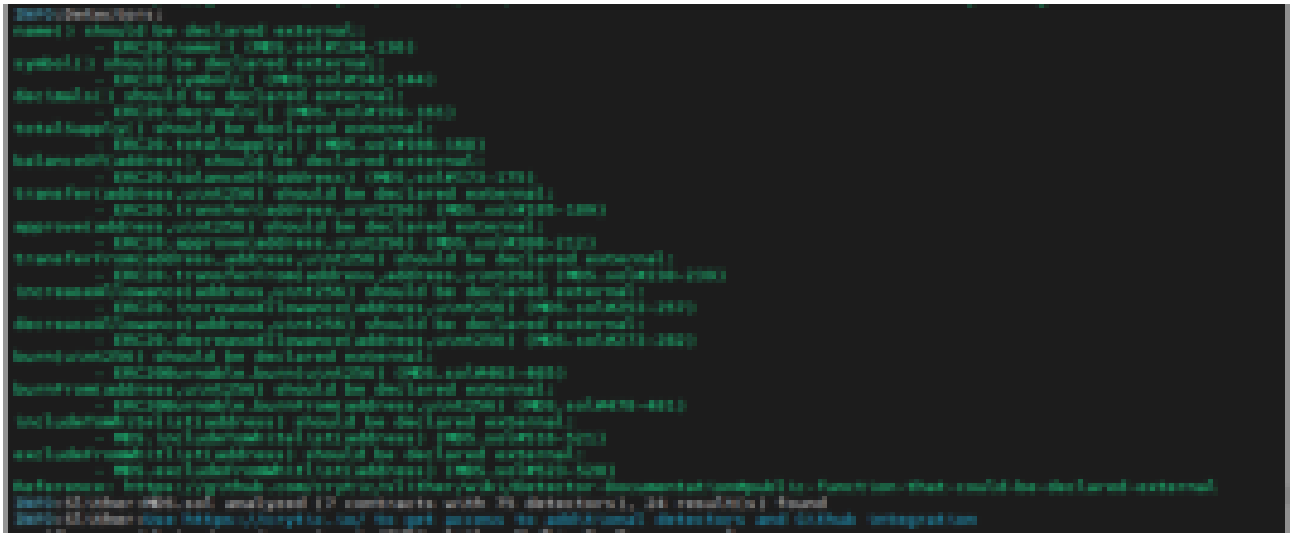




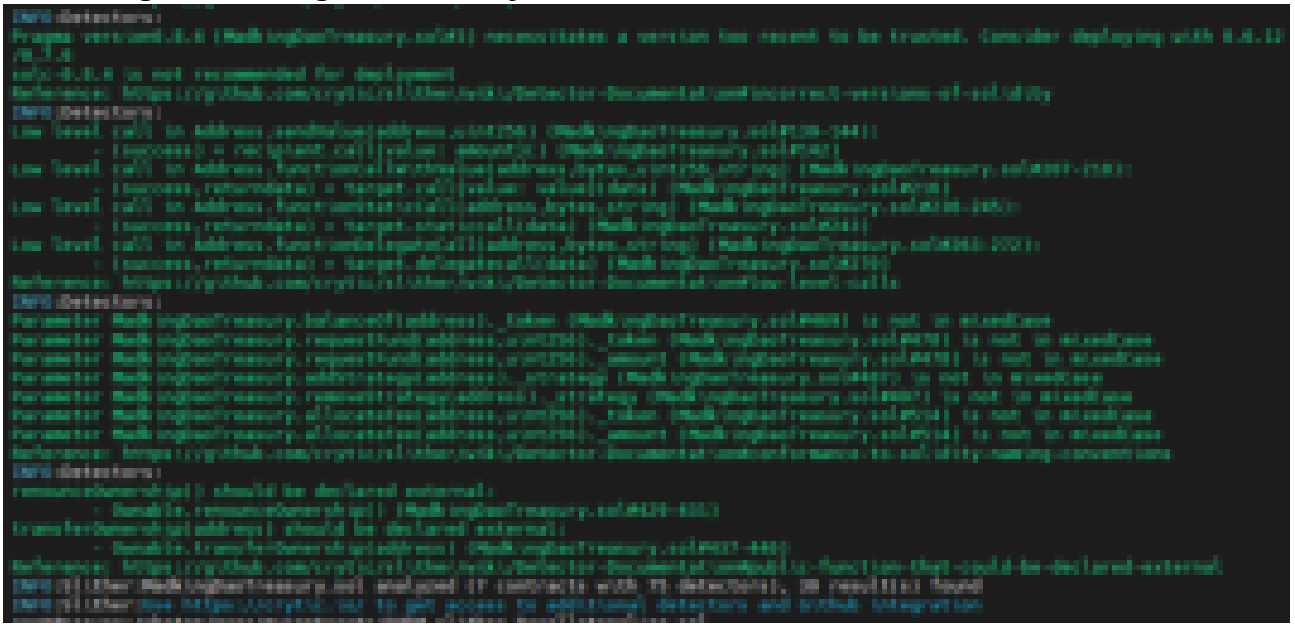


Slither log >>MDS.sol

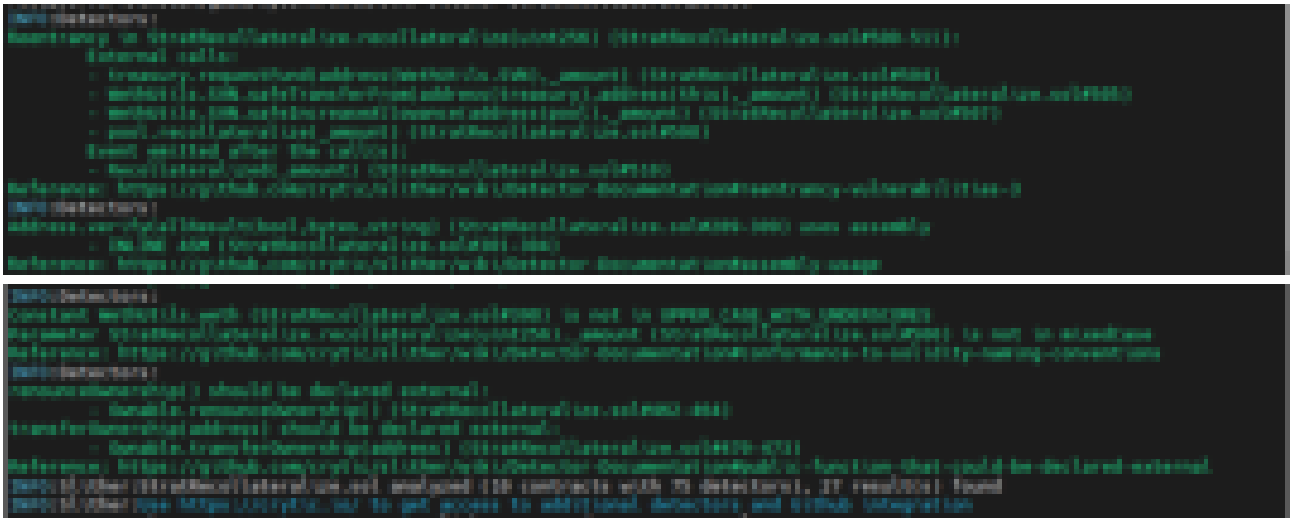




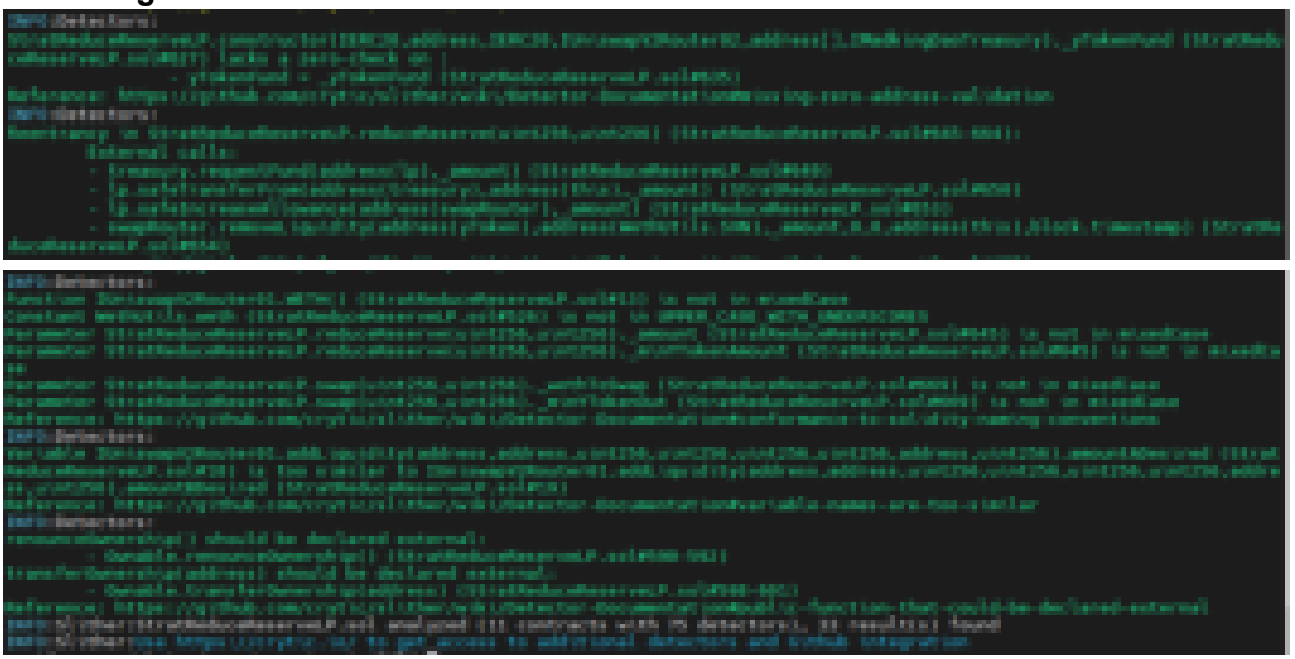
Slither log >>MadkingDaoTreasury.sol



Slither log >>StratRecollateralize.sol



Slither log >>StratReduceReserveLP.sol



Email: audit@EtherAuthority.io