Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: VeZard Exchange
Website: https://vezard.exchange
Platform: ZkSync Era Chain
Language: Solidity
Date: May 8th, 2023

Table of contents

Email: audit@EtherAuthority.io

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

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by VeZard Exchange to perform the Security audit of the VeZard Exchange 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 8th, 2023.

The purpose of this audit was to address the following:

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

Project Background

- veZard Exchange is built on the zkSync Era network using a governance model called the ve(3,3) system. Purpose of the ve(3,3) structure is to create an environment where users can actively choose to participate and establish a cycle of growth that reinforces itself over time.
- The veZard Exchange contract inherits IERC20, TransparentUpgradeableProxy, SafeERC20, ReentrancyGuard, Ownable, ERC20, OwnableUpgradeable, SafeMath, IERC721Metadata, IERC721Receiver 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 VeZard Exchange Smart Contracts
Platform	**ZkSync Era Chain / Solidity**
File 1	Bribes.sol
File 1 MD5 Hash	22930973B1EE7005CB48E9400E4CBC4D
File 2	FaucetToken.sol
File 2 MD5 Hash	3CFACF5B2D6E9CCB150911A49ADE67D7
File 3	GaugeV2.sol
File 3 MD5 Hash	7A3F2E2A748573CDFB8654A714DCCCF0
File 4	MinterUpgradeable.sol
File 4 MD5 Hash	B28E301E4724B23D7396651183B13C2C
File 5	Multicall.sol
File 5 MD5 Hash	B31A5401C236F10109672BC3D903C9DA
File 6	Pair.sol
File 6 MD5 Hash	11E9CF8F52D2324B3E1A964D55EFF83C
File 7	PairFees.sol
File 7 MD5 Hash	FBEB940CDE074480C2DCBA9D1BF404F1
File 8	RewardsDistributor.sol
File 8 MD5 Hash	708D98975EC0DB4DE3ED85C9803BA155
File 9	RouterV2.sol
File 9 MD5 Hash	DF5BF916C6DAA34A4D3FAEBFF7BB5AF5
File 10	SwapLibrary.sol
File 10 MD5 Hash	B6DBF1D160C62F3CA689D0E38E457075
File 11	VoterV2.sol
File 11 MD5 Hash	111C4D010010BA87915329BC488C63DA
File 12	VotingEscrow.sol

File 12 MD5 Hash	FC2AF6575DE1BB2344088C7C123383F4
File 13	Zard.sol
File 13 MD5 Hash	3596FD4176DBCC8F6FB40C53CB2BB6F4
File 14	BribeFactoryV2.sol
File 14 MD5 Hash	11AE5E800B94E9650FD617985B91BAC9
File 15	GaugeFactoryV2.sol
File 15 MD5 Hash	10EF53C0D003B7CD9B16E94E981EDD80
File 16	PairFactoryUpgradeable.sol
File 16 MD5 Hash	B60C422FA97157362B396A6F31A73BE2
File 17	AdminUpgradeabilityProxy.sol
File 17 MD5 Hash	9DB89ED56B653E26510B7013EFFE47B0
File 18	VeArtProxyUpgradeable.sol
File 18 MD5 Hash	5074D64AF05AB31C410E9431B02FFB65
Audit Date	May 8th, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 Bribes.sol • Rewards are released over 7 days **<u>Owner has control over following functions:</u>** • Recover the ERC20 token address with the amount. • Set the Voter address. • Set the Reward address. • Set the Minter address. • Add a reward token address. • Set a new owner address.	**YES, This is valid.**
File 2 GaugeV2.sol **<u>Owner has control over following functions:</u>** • Set the distribution address. • Set the Gauge rewarder address. • Set the extra rewarder pid.	**YES, This is valid.**
File 3 import.sol • Import contract can inherit the TransparentUpgradeableProxy contract.	**YES, This is valid.**
File 4 Multicall.sol • Multicall - Aggregate results from multiple read-only function calls.	**YES, This is valid.**
File 5 MinterUpgradeable.sol **<u>Other Specifications:</u>** • MinterUpgradeable is used to codify the minting rules as per ve(3,3), abstracted from the token to support any token that allows minting. • Maximum Team rate: 5%	**YES, This is valid.**

• Allows minting once per week. **Owner has control over following functions:** • Set a team address. • Accept the team. • Set a voter address. • Set a team rate. • Set an emission rate value. • Set a Rebase rate value. • Set a reward distributor address.	
File 6 Pair.sol **Other Specifications:** • Decimals: 18 • Minimum Liquidity: 1000 • Capture oracle reading every 30 minutes.	**YES, This is valid.**
File 7 PairFees.sol • Pair Fees contract is used as a 1:1 pair relationship to split out fees, this ensures that the curve does not need to be modified for LP shares. **Owner Specifications:** • claimFeesFor us allow the pair to transfer fees to users.	**YES, This is valid.**
File 8 RewardsDistributor.sol • Instant Rate: 20 **Owner has control over following functions:** • check the checkpoint token. • Set the Depositor. • A new owner address can be set by the current Owner. • Withdraw ERC20 tokens from the contract.	**YES, This is valid.**

• Set an Instant rate.	
File 9 RouterV2.sol **<u>Owner has control over following functions:</u>** • RouterV2 : Support for Fee-on-Transfer Tokens. • Only accept ETH via fallback from the WETH contract.	**YES, This is valid.**
File 10 SwapLibrary.sol • SwapLibrary is used to fetch pair addresses by token addresses, sort tokens.	**YES, This is valid.**
File 11 VoterV2.sol • Rewards are released over 7 days **<u>Owner has control over following functions:</u>** • Set a minter address. • Set a Governor address. • Set an emergency council address.	**YES, This is valid.**
File 12 VotingEscrow.sol • Name: veZard • Symbol: veZARD • Decimals: 18 • version: 1.0.0 **<u>Other Specifications:</u>** • Voting Escrow: veNFT implementation that escrows ERC-20 tokens in the form of an ERC-721 NFT. **<u>Owner has control over following functions:</u>** • Set a team address. • Set an art proxy address.	**YES, This is valid.**
File 13 Zard.sol • Name: Zard Token	**YES, This is valid.**

• Symbol: ZARD • Decimals: 18 **Owner has control over following functions:** • Set a minter address. • Owner can mint a token.	
File 14 BribeFactoryV2.sol **Owner has control over following functions:** • Voter owners can create a new Bribe. • Voter address can be set by Owner. • Owner can add a new reward address.	**YES, This is valid.**
File 15 GaugeFactoryV2.sol **Owner has control over following functions:** • Distribution address can be set by Owner.	**YES, This is valid.**
File 16 PairFactoryUpgradeable.sol • Maximum Fee: 0.25% • Stable Fee: 0.02% • Volatile Fee: 0.2% **Owner has control over following functions:** • Set a Pauser address. • Set a dibs address. • Set a fee.	**YES, This is valid.**
File 17 VeArtProxyUpgradeable.sol • VeArtProxyUpgradeable contract can inherit OwnableUpgradeable contract.	**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, 0 low and some very low level issues.

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

Overall Audit Result: PASSED

Code Quality

This audit scope has 18 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 VeZard Exchange 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 VeZard Exchange Protocol.

The VeZard Exchange 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 VeZard Exchange Protocol smart contract code in the form of a file. The hash of that code is mentioned above in the table.

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

Another source of information was its official website: https://vezard.exchange 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

Bribes.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	nonReentrant	modifier	Passed	No Issue
3	_nonReentrantBefore	write	Passed	No Issue
4	_nonReentrantAfter	write	Passed	No Issue
5	_reentrancyGuardEntered	internal	Passed	No Issue
6	onlyOwner	modifier	Passed	No Issue
7	getEpochStart	read	Passed	No Issue
8	getNextEpochStart	read	Passed	No Issue
9	addReward	write	Passed	No Issue
10	rewardsListLength	external	Passed	No Issue
11	totalSupply	external	Passed	No Issue
12	totalSupplyAt	external	Passed	No Issue
13	balanceOfAt	read	Passed	No Issue
14	balanceOf	read	Passed	No Issue
15	earned	read	Passed	No Issue
16	_earned	internal	Passed	No Issue
17	rewardPerToken	read	Passed	No Issue
18	_deposit	external	Passed	No Issue
19	_withdraw	write	Passed	No Issue
20	getReward	external	Passed	No Issue
21	getRewardForOwner	write	Passed	No Issue
22	notifyRewardAmount	external	Passed	No Issue
23	recoverERC20	external	Passed	No Issue
24	setVoter	external	access only Owner	No Issue
25	setMinter	external	access only Owner	No Issue
26	addRewardToken	external	access only Owner	No Issue
27	setOwner	external	access only Owner	No Issue

GaugeV2.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	nonReentrant	modifier	Passed	No Issue
3	_nonReentrantBefore	write	Passed	No Issue
4	_nonReentrantAfter	write	Passed	No Issue
5	_reentrancyGuardEntered	internal	Passed	No Issue
6	onlyOwner	modifier	Passed	No Issue
7	owner	read	Passed	No Issue

8	_checkOwner	internal	Passed	No Issue
9	renounceOwnership	write	access only Owner	No Issue
10	transferOwnership	write	access only Owner	No Issue
11	_transferOwnership	internal	Passed	No Issue
12	updateReward	modifier	Passed	No Issue
13	onlyDistribution	modifier	Passed	No Issue
14	setDistribution	external	access only Owner	No Issue
15	setGaugeRewarder	external	access only Owner	No Issue
16	setRewarderPid	external	access only Owner	No Issue
17	totalSupply	read	Passed	No Issue
18	balanceOf	external	Passed	No Issue
19	lastTimeRewardApplicable	read	Passed	No Issue
20	rewardPerToken	read	Passed	No Issue
21	earned	read	Passed	No Issue
22	rewardForDuration	external	Passed	No Issue
23	depositAll	external	Passed	No Issue
24	deposit	external	Passed	No Issue
25	_deposit	internal	Passed	No Issue
26	withdrawAll	external	Passed	No Issue
27	withdraw	external	Passed	No Issue
28	_withdraw	internal	Passed	No Issue
29	withdrawAllAndHarvest	external	Passed	No Issue
30	getReward	write	Passed	No Issue
31	_periodFinish	external	Passed	No Issue
32	notifyRewardAmount	external	access only Distribution	No Issue
33	claimFees	external	Passed	No Issue
34	_claimFees	internal	Passed	No Issue

import.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
10	_requireZeroValue	write	Passed	No Issue

MinterUpgradeable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	access only Initializing	No Issue
3	__Ownable_init_unchained	internal	access only Initializing	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	owner	read	Passed	No Issue
6	_checkOwner	internal	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue
10	initialize	write	initializer	No Issue
11	_initialize	external	Passed	No Issue
12	setTeam	external	Passed	No Issue
13	acceptTeam	external	Passed	No Issue
14	setVoter	external	Passed	No Issue
15	setTeamRate	external	Passed	No Issue
16	setEmission	external	Passed	No Issue
17	setRebase	external	Passed	No Issue
18	circulating_supply	read	Passed	No Issue
19	calculate_emission	read	Passed	No Issue
20	weekly_emission	read	Passed	No Issue
21	circulating_emission	read	Passed	No Issue
22	calculate_rebate	read	Passed	No Issue
23	update_period	external	Passed	No Issue
24	check	external	Passed	No Issue
25	period	external	Passed	No Issue
26	setRewardDistributor	external	Passed	No Issue

Pair.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	lock	modifier	Passed	No Issue
3	observationLength	external	Passed	No Issue
4	lastObservation	read	Passed	No Issue
5	metadata	external	Passed	No Issue
6	tokens	external	Passed	No Issue
7	isStable	external	Passed	No Issue
8	claimFees	external	Passed	No Issue
9	_update0	internal	Passed	No Issue
10	_update1	internal	Passed	No Issue

11	_updateFor	internal	Passed	No Issue
12	getReserves	read	Passed	No Issue
13	_update	internal	Passed	No Issue
14	currentCumulativePrices	read	Passed	No Issue
15	current	external	Passed	No Issue
16	quote	external	Passed	No Issue
17	prices	external	Passed	No Issue
18	sample	read	Passed	No Issue
19	mint	external	Passed	No Issue
20	burn	external	Passed	No Issue
21	swap	external	Passed	No Issue
22	skim	external	Passed	No Issue
23	sync	external	Passed	No Issue
24	_f	internal	Passed	No Issue
25	_d	internal	Passed	No Issue
26	_get_y	internal	Passed	No Issue
27	getAmountOut	external	Passed	No Issue
28	_getAmountOut	internal	Passed	No Issue
29	_k	internal	Passed	No Issue
30	_mint	internal	Passed	No Issue
31	_burn	internal	Passed	No Issue
32	approve	external	Passed	No Issue
33	transfer	external	Passed	No Issue
34	transferFrom	external	Passed	No Issue
35	_transferTokens	internal	Passed	No Issue
36	_safeTransfer	internal	Passed	No Issue
37	_safeApprove	internal	Passed	No Issue

PairFees.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_safeTransfer	internal	Passed	No Issue
3	claimFeesFor	external	Passed	No Issue

RewardsDistributor.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	timestamp	external	Passed	No Issue
3	_checkpoint_token	internal	Passed	No Issue
4	checkpoint_token	external	Passed	No Issue
5	_find_timestamp_epoch	internal	Passed	No Issue
6	_find_timestamp_user_epoch	internal	Passed	No Issue
7	ve_for_at	external	Passed	No Issue

8	_checkpoint_total_supply	internal	Passed	No Issue
9	checkpoint_total_supply	external	Passed	No Issue
10	_claim	internal	Passed	No Issue
11	_claimable	internal	Passed	No Issue
12	claimable	external	Passed	No Issue
13	claim	external	Passed	No Issue
14	claim_many	external	Passed	No Issue
15	setDepositor	external	Passed	No Issue
16	setOwner	external	Passed	No Issue
17	withdrawERC20	external	Passed	No Issue
18	setInstantRate	external	Passed	No Issue

Router.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	ensure	modifier	Passed	No Issue
3	receive	external	Passed	No Issue
4	sortTokens	write	Passed	No Issue
5	pairFor	read	Passed	No Issue
6	quoteLiquidity	internal	Passed	No Issue
7	getReserves	read	Passed	No Issue
8	getAmountOut	external	Passed	No Issue
9	getAmountsOut	read	Passed	No Issue
10	isPair	external	Passed	No Issue
11	quoteAddLiquidity	external	Passed	No Issue
12	_addLiquidity	internal	Passed	No Issue
13	quoteRemoveLiquidity	external	Passed	No Issue
14	addLiquidity	external	Passed	No Issue
15	addLiquidityETH	external	Passed	No Issue
16	removeLiquidity	write	Passed	No Issue
17	removeLiquidityETH	write	Passed	No Issue
18	removeLiquidityWithPermit	external	Passed	No Issue
19	removeLiquidityETHWithPermit	external	Passed	No Issue
20	_swap	internal	Passed	No Issue
21	swapExactTokensForTokensSimple	external	Passed	No Issue
22	swapExactTokensForTokens	external	Passed	No Issue
23	swapExactETHForTokens	external	Passed	No Issue
24	swapExactTokensForETH	external	Passed	No Issue
25	_safeTransferETH	internal	Passed	No Issue
26	_safeTransfer	internal	Passed	No Issue
27	_safeTransferFrom	internal	Passed	No Issue
28	removeLiquidityETHSupportingFeeOnTransferTokens	write	Passed	No Issue

29	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	external	Passed	No Issue
30	_swapSupportingFeeOnTransferTokens	internal	Passed	No Issue
31	swapExactTokensForTokensSupportingFeeOnTransferTokens	external	Passed	No Issue
32	swapExactETHForTokensSupportingFeeOnTransferTokens	external	Passed	No Issue
33	swapExactTokensForETHSupportingFeeOnTransferTokens	external	Passed	No Issue

SwapLibrary.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_f	internal	Passed	No Issue
3	_d	internal	Passed	No Issue
4	_get_y	internal	Passed	No Issue
5	getTradeDiff	external	Passed	No Issue
6	getTradeDiffSimple	external	Passed	No Issue
7	getTradeDiff2	external	Passed	No Issue
8	getTradeDiff3	external	Passed	No Issue
9	_calcSample	internal	Passed	No Issue
10	getTradeDiff	external	Passed	No Issue
11	getSample	external	Passed	No Issue
12	getMinimumValue	external	Passed	No Issue
13	getAmountOut	external	Passed	No Issue
14	_getAmountOut	internal	Passed	No Issue
15	_k	internal	Passed	No Issue
16	getNormalizedReserves	external	Passed	No Issue
17	pairFor	read	Passed	No Issue
18	sortTokens	write	Passed	No Issue

VeArtProxyUpgradeable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	access only Initializing	No Issue

3	__Ownable_init_unchained	internal	access only Initializing	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	owner	read	Passed	No Issue
6	_checkOwner	internal	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue
10	initialize	write	initializer	No Issue
11	toString	internal	Passed	No Issue
12	_tokenURI	external	Passed	No Issue

VoterV2.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	access only Initializing	No Issue
3	__Ownable_init_unchained	internal	access only Initializing	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	owner	read	Passed	No Issue
6	_checkOwner	internal	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue
10	__ReentrancyGuard_init	internal	access only Initializing	No Issue
11	__ReentrancyGuard_init_unchained	internal	access only Initializing	No Issue
12	nonReentrant	modifier	Passed	No Issue
13	_nonReentrantBefore	write	Passed	No Issue
14	_nonReentrantAfter	write	Passed	No Issue
15	_reentrancyGuardEntered	internal	Passed	No Issue
16	initialize	write	Anyone can initialize contract	Refer to audit findings
17	_initialize	external	Infinite loop	Refer to audit findings
18	setMinter	external	Passed	No Issue
19	setGovernor	write	Passed	No Issue
20	setEmergencyCouncil	write	Passed	No Issue
21	reset	external	Passed	No Issue
22	_reset	internal	Infinite loop	Refer to audit findings
23	poke	external	Infinite loop	Refer to audit findings
24	_vote	internal	Infinite loop	Refer to audit findings
25	vote	external	Passed	No Issue

26	whitelist	write	Passed	No Issue
27	_whitelist	internal	Passed	No Issue
28	createGauge	external	Passed	No Issue
29	killGauge	external	Passed	No Issue
30	reviveGauge	external	Passed	No Issue
31	length	external	Passed	No Issue
32	poolVoteLength	external	Passed	No Issue
33	notifyRewardAmount	external	Passed	No Issue
34	updateFor	external	Passed	No Issue
35	updateForRange	write	Infinite loop	Refer to audit findings
36	updateAll	external	Passed	No Issue
37	updateGauge	external	Passed	No Issue
38	_updateFor	internal	Passed	No Issue
39	claimBribes	external	Infinite loop	Refer to audit findings
40	claimFees	external	Infinite loop	Refer to audit findings
41	distributeFees	external	Infinite loop	Refer to audit findings
42	distribute	write	Passed	No Issue
43	distributeAll	external	Passed	No Issue
44	distribute	write	Passed	No Issue
45	distribute	write	Passed	No Issue
46	_safeTransferFrom	internal	Passed	Fixed
47	setBribeFactory	external	Passed	No Issue
48	setGaugeFactory	external	Passed	Fixed
49	setPairFactory	external	Passed	Fixed
50	killGaugeTotally	external	Passed	No Issue
51	whitelist	write	Passed	No Issue
52	initGauges	write	Anyone can initGauges, Infinite loop	Refer to audit findings
53	increaseGaugeApprovals	external	Passed	Fixed
54	setNewBribe	external	Passed	Fixed

VotingEscrow.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	nonreentrant	modifier	Passed	No Issue
3	setTeam	external	Passed	No Issue
4	setArtProxy	external	Passed	No Issue
5	tokenURI	external	Passed	No Issue
6	ownerOf	read	Passed	No Issue
7	_balance	internal	Passed	No Issue
8	balanceOf	external	Passed	No Issue

9	getApproved	external	Passed	No Issue
10	isApprovedForAll	external	Passed	No Issue
11	approve	write	Passed	No Issue
12	setApprovalForAll	external	Passed	No Issue
13	_clearApproval	internal	Passed	No Issue
14	_isApprovedOrOwner	internal	Passed	No Issue
15	isApprovedOrOwner	external	Passed	No Issue
16	_transferFrom	internal	Passed	No Issue
17	transferFrom	external	Passed	No Issue
18	safeTransferFrom	external	Passed	No Issue
19	_isContract	internal	Passed	No Issue
20	safeTransferFrom	write	Passed	No Issue
21	supportsInterface	external	Passed	No Issue
22	tokenOfOwnerByIndex	external	Passed	No Issue
23	_addTokenToOwnerList	internal	Passed	No Issue
24	_addTokenTo	internal	Passed	No Issue
25	_mint	internal	Passed	No Issue
26	_removeTokenFromOwnerList	internal	Passed	No Issue
27	_removeTokenFrom	internal	Passed	No Issue
28	_burn	internal	Passed	No Issue
29	get_last_user_slope	external	Passed	No Issue
30	user_point_history__ts	external	Passed	No Issue
31	locked__end	external	Passed	No Issue
32	_checkpoint	internal	Passed	No Issue
33	_deposit_for	internal	Passed	No Issue
34	block_number	external	Passed	No Issue
35	checkpoint	external	Passed	No Issue
36	deposit_for	external	Passed	No Issue
37	_create_lock	internal	Passed	No Issue
38	create_lock	external	Passed	No Issue
39	create_lock_for	external	Passed	No Issue
40	increase_amount	external	Passed	No Issue
41	increase_unlock_time	external	Passed	No Issue
42	withdraw	external	Passed	No Issue
43	_find_block_epoch	internal	Passed	No Issue
44	_balanceOfNFT	internal	Passed	No Issue
45	balanceOfNFT	external	Passed	No Issue
46	balanceOfNFTAt	external	Passed	No Issue
47	_balanceOfAtNFT	internal	Passed	No Issue
48	balanceOfAtNFT	external	Passed	No Issue
49	totalSupplyAt	external	Passed	No Issue
50	_supply_at	internal	Passed	No Issue
51	totalSupply	external	Passed	No Issue
52	totalSupplyAtT	read	Passed	No Issue
53	setVoter	external	Passed	No Issue
54	voting	external	Passed	No Issue
55	abstain	external	Passed	No Issue

56	attach	external	Passed	No Issue
57	detach	external	Passed	No Issue
58	merge	external	Passed	No Issue
59	split	external	Passed	No Issue
60	delegates	read	Passed	No Issue
61	getVotes	external	Passed	No Issue
62	getPastVotesIndex	read	Passed	No Issue
63	getPastVotes	read	Passed	No Issue
64	getPastTotalSupply	external	Passed	No Issue
65	_moveTokenDelegates	internal	Passed	No Issue
66	_findWhatCheckpointToWrite	internal	Passed	No Issue
67	_moveAllDelegates	internal	Passed	No Issue
68	_delegate	internal	Passed	No Issue
69	delegate	write	Passed	No Issue

Zard.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	setMinter	external	Passed	No Issue
3	approve	external	Passed	No Issue
4	_mint	internal	Passed	No Issue
5	_transfer	internal	Passed	No Issue
6	transfer	external	Passed	No Issue
7	transferFrom	external	Passed	No Issue
8	mint	external	Passed	No Issue

BribeFactoryV2.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	access only Initializing	No Issue
3	__Ownable_init_unchained	internal	access only Initializing	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	owner	read	Passed	No Issue
6	_checkOwner	internal	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue
10	initialize	write	Passed	No Issue
11	createBribe	external	Passed	No Issue

12	setVoter	external	Passed	No Issue
13	addReward	external	Passed	No Issue
14	addRewards	external	Passed	No Issue

GaugeFactoryV2.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	access only Initializing	No Issue
3	__Ownable_init_unchained	internal	access only Initializing	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	owner	read	Passed	No Issue
6	_checkOwner	internal	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue
10	initialize	write	Passed	No Issue
11	createGaugeV2	external	Passed	No Issue
12	setDistribution	external	access only Owner	No Issue

PairFactoryUpgradeable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	access only Initializing	No Issue
3	__Ownable_init_unchained	internal	access only Initializing	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	owner	read	Passed	No Issue
6	_checkOwner	internal	Passed	No Issue
7	renounceOwnership	write	access only Owner	No Issue
8	transferOwnership	write	access only Owner	No Issue
9	_transferOwnership	internal	Passed	No Issue
10	onlyManager	modifier	Passed	No Issue
11	initialize	write	Passed	No Issue
12	allPairsLength	external	Passed	No Issue
13	pairs	external	Passed	No Issue
14	setPause	external	Passed	No Issue
15	setFeeManager	external	access only Manager	No Issue
16	acceptFeeManager	external	Passed	No Issue

17	address _dibs	external	access only Manager	No Issue
18	setNftFeeHandler	external	access only Manager	No Issue
19	setFee	external	access only Manager	No Issue
20	getFee	read	Passed	No Issue
21	pairCodeHash	external	Passed	No Issue
22	getInitializable	external	Passed	No Issue
23	createPair	external	Passed	No Issue
24	setSecondFee	external	Passed	No Issue

Severity Definitions

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

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No Low severity vulnerabilities were found.

Very Low / Informational / Best practices:

(1) Anyone can initialize contract: **VoterV2.sol**
The initialize function is public and accessible to anyone. Operator is not set during contract deployment, So any user can become an operator

Resolution: We suggest always making sure that the contract should be initialized by the owner.

(2) Anyone can initGauges : **VoterV2.sol**
The initGauges is a public function, emergencyCouncil can execute this unlimited times. This might lead to losing vote data.

Resolution: We suggest to re-check the logic and usage limit for this function.

(3) Infinite loop: **VoterV2.sol**

In below functions ,for loops do not have upper length limit , which costs more gas:

- claimBribes
- claimFees
- distributeFees
- initGauges
- updateForRange
- poke
- _reset
- _initialize

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

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:

Ownable.sol

- renounceOwnership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transferOwnership: Current owner can transfer ownership of the contract to a new account.
- _checkOwner: Thrown when the sender is not the owner.

Bribes.sol

- addReward: Owner can add a new reward address.
- recoverERC20: Owner can recover the ERC20 token address with the amount
- setVoter: Voter address can be set by the Owner.
- setMinter: Minter address can be set by the Owner.
- addRewardToken: Reward token address can be added by the Owner.
- setOwner: A new owner address can be set by the Owner.

GaugeV2.sol

- setDistribution: Distribution address can be set by the Owner.
- setGaugeRewarder: Gauge rewarder address can be set by the Owner.
- setRewarderPid: Extra rewarder pid can be set by the Owner.

MinterUpgradeable.sol

- setTeam: Team address can be set by the Owner.
- acceptTeam: Owner can accept the team.
- setVoter: Voter address can be set by the Owner.
- setTeamRate: Team rate value can be set by the Owner.
- setEmission: Emission rate can be set by the Owner.
- setRebase: Rebase rate can be set by the Owner.
- setRewardDistributor: Reward Distributor address can be set by the Owner.

RewardsDistributor.sol

- setDepositor: The Depositor can be set by the Owner.
- setOwner: A new owner address can be set by the current Owner.
- withdrawERC20: Owner can withdraw ERC20 tokens from the contract.
- setInstantRate: Owner can set an instant rate.

VoterV2.sol

- _initialize: Minter owner or EmergencyCouncil owner can initialize token addresses.
- setMinter: EmergencyCouncil owner can set minter address.
- setGovernor: Owner can set a new governor address.
- setEmergencyCouncil: Owner can set a new emergencyCouncil address.
- whitelist: Owner can add token address in whitelist.
- killGauge: Owner can kill gauge address.
- reviveGauge: Owner can revive gauge address.
- setBribeFactory: Owner can set a bribe factory address.
- setGaugeFactory: Owner can set a gauge factory address.
- setPairFactory: Owner can set a pair factory address.
- killGaugeTotally: Owner can kill gauge addresses.
- whitelist: Owner can add token address in the whitelist.
- initGauges: Owner can initialize gauges addresses.

- increaseGaugeApprovals: Owners can increase gauge approval addresses.
- setNewBribe: Owners can set new bribe addresses.

VotingEscrow.sol

- setTeam: Team address can be set by the Owner.
- setArtProxy: Proxy address can be set by the Owner.
- setVoter: Voter address can be set by the team Owner.
- voting: Voting tokenId can be set by the Voter Owner.
- abstain: Abstain tokenId can be set by the Voter Owner.
- attach: Attach tokenId can be set by the Voter Owner.
- detach: Detach tokenId can be set by the Voter Owner.
- delegate: Delegate votes from owner to `delegatee`.

BribeFactoryV2.sol

- createBribe: Voter owners can create a new Bribe.
- setVoter: Voter address can be set by the Owner.
- addReward: Owner can add a new reward address.
- addRewards: Owner can add multiple new reward addresses.

GaugeFactoryV2.sol

- setDistribution: Distribution address can be set by Owner.

PairFactoryUpgradeable.sol

- setPause: Pauser address can be set by the Owner.
- setFeeManager: Manager Owner can set a Fee Manager address.
- acceptFeeManager: Manager Owner can accept fee manager.
- setDibs: Manager Owner can set dibs address.
- setNftFeeHandler: Fee Manager Owner can set Nft fee.
- setSecondFee: Fee Manager Owner can set a second fee.
- setFee: Manager Owner can set a fee.

VeArtProxyUpgradeable.sol

- _checkOwner: Thrown when the sender is not the owner.

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

Import.sol

- admin: Admin can return the current admin address.
- implementation: Admin can return the current implementation.
- changeAdmin: Admin can change the admin of the proxy.
- upgradeTo: Admin can upgrade the implementation of the proxy.
- upgradeToAndCall: Admin can upgrade the implementation of the proxy, and then call a function from the new implementation as specified data.

PairFees.sol

- claimFeesFor: Owner can allow the pair to transfer fees to users.

Zard.sol

- setMinter: Owner can set the minter address.
- mint: Owner can mint a token from the address.

Multicall.sol

- aggregate: Owner can aggregate results from multiple function calls.

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

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We had observed some informational severity issues in the smart contracts, but those are not critical ones. **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.

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

The 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.

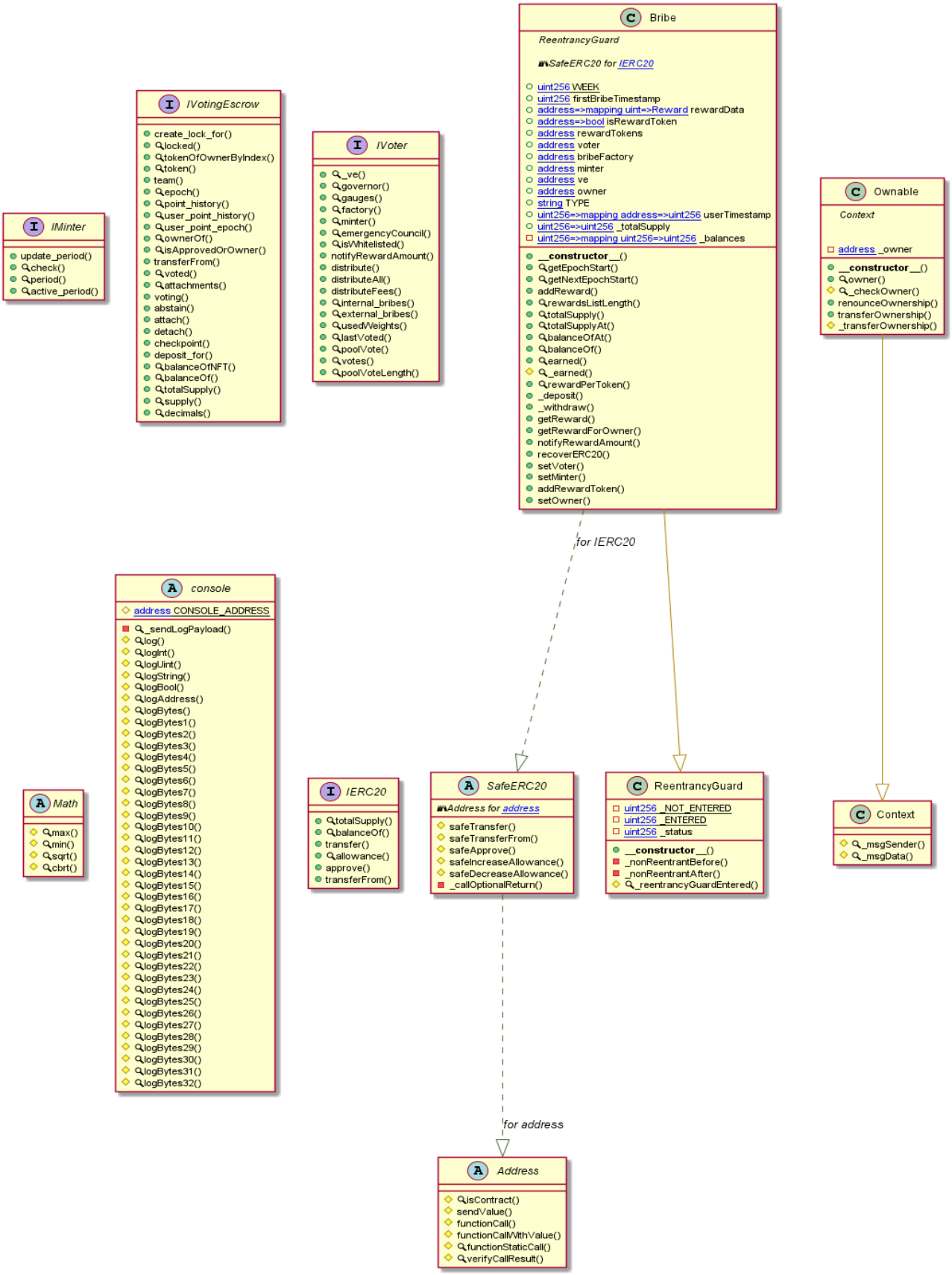
Technical Disclaimer

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.

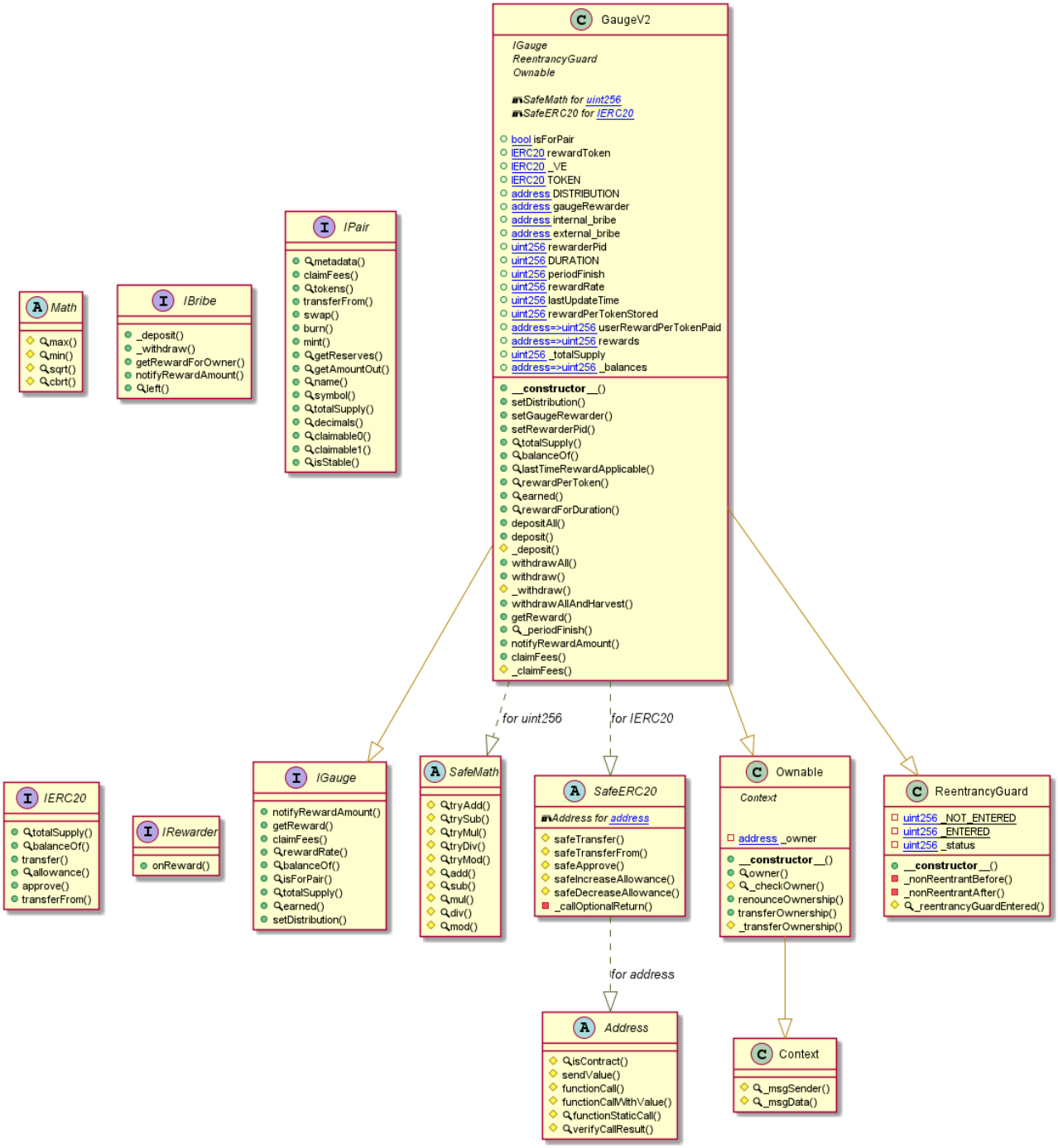
Appendix

Code Flow Diagram - VeZard Exchange

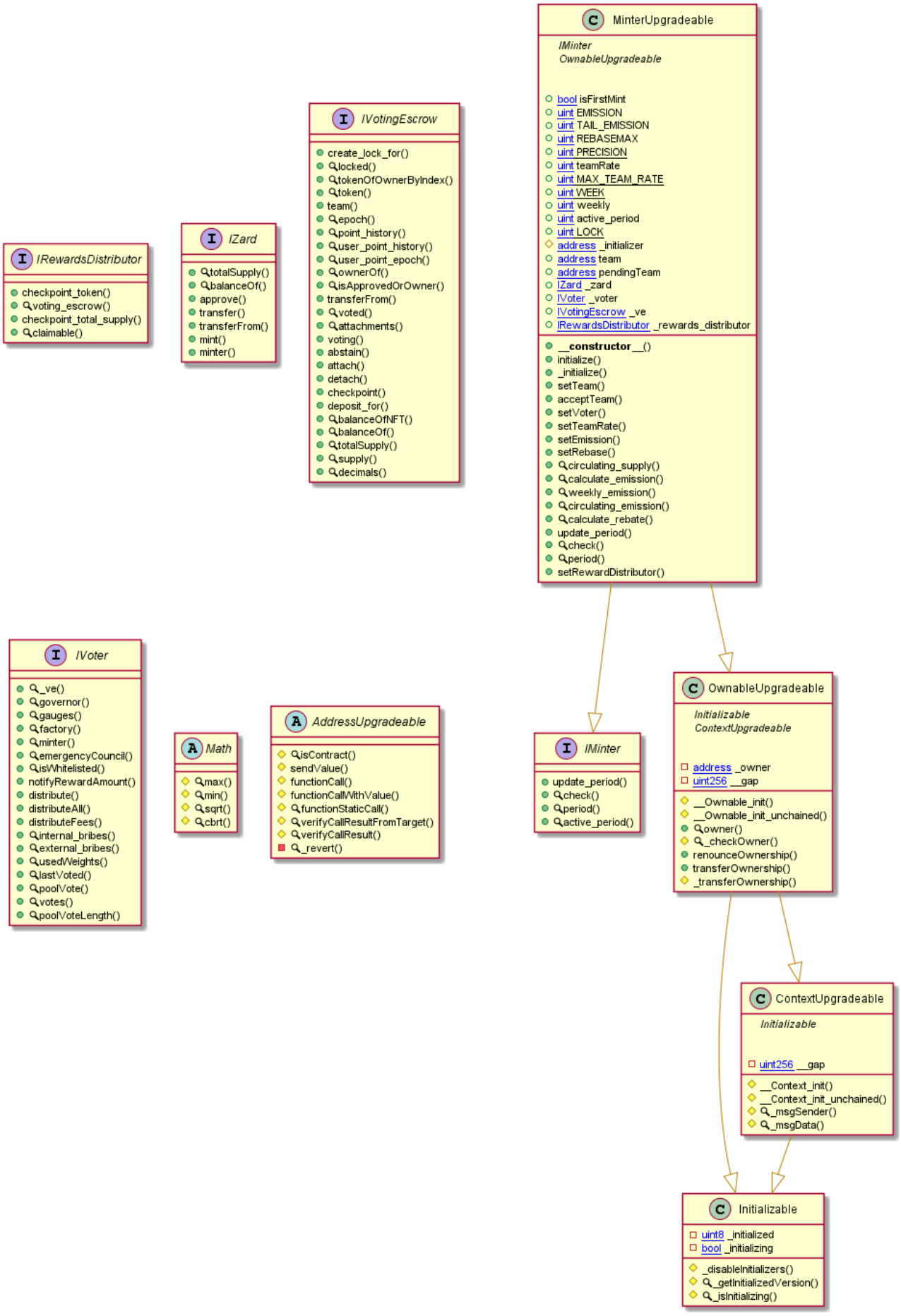
Bribes Diagram



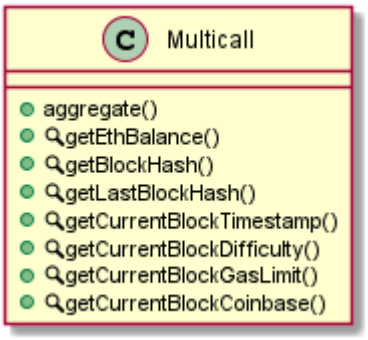
GaugeV2 Diagram



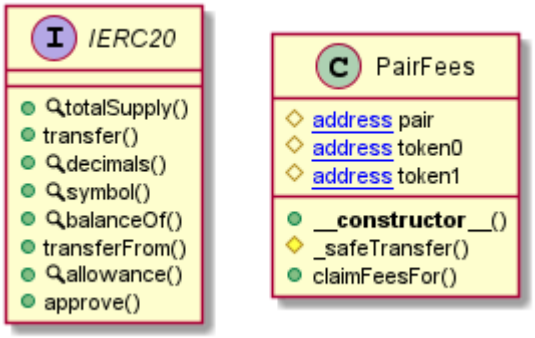
MinterUpgradeable Diagram



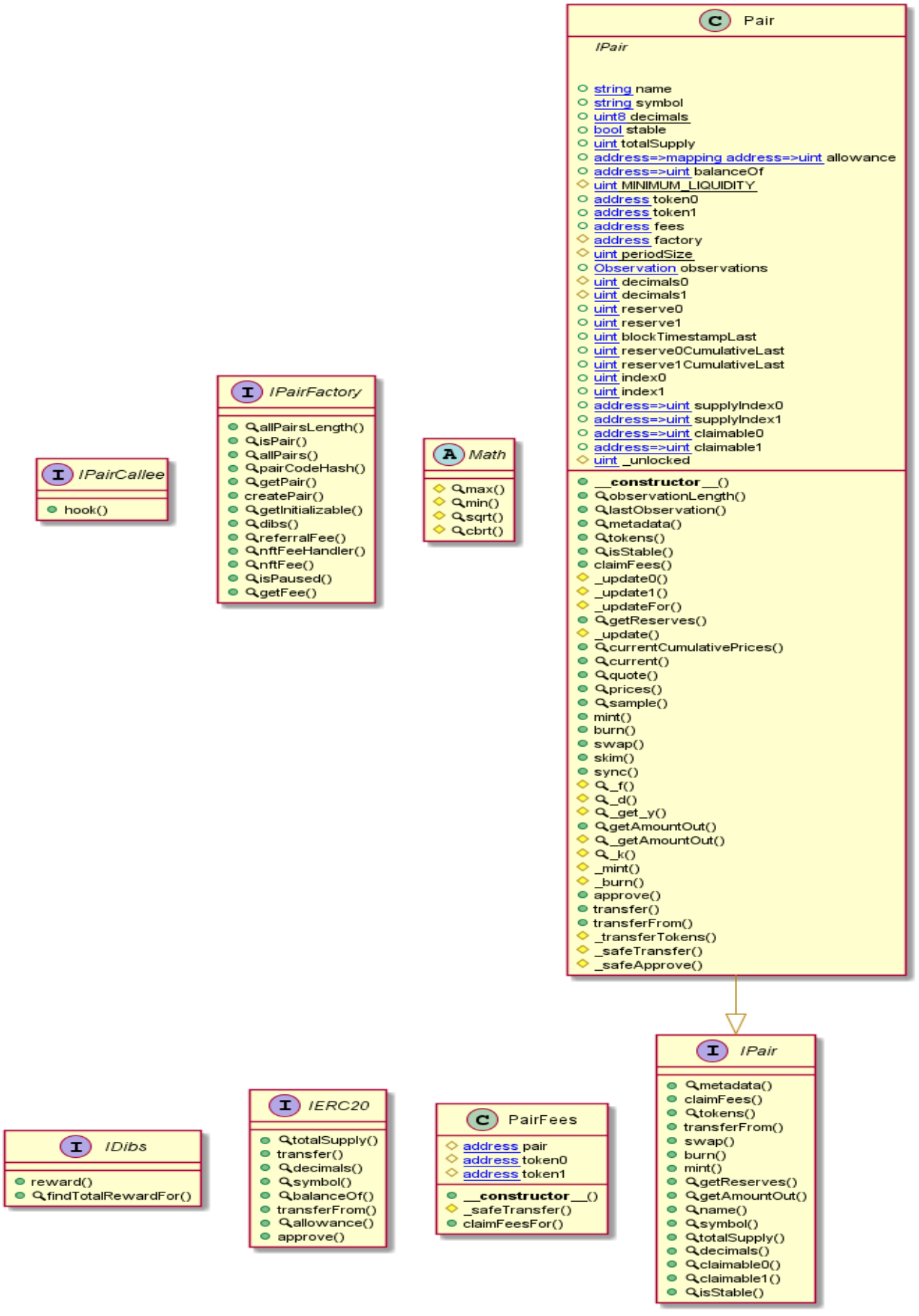
Multicall Diagram



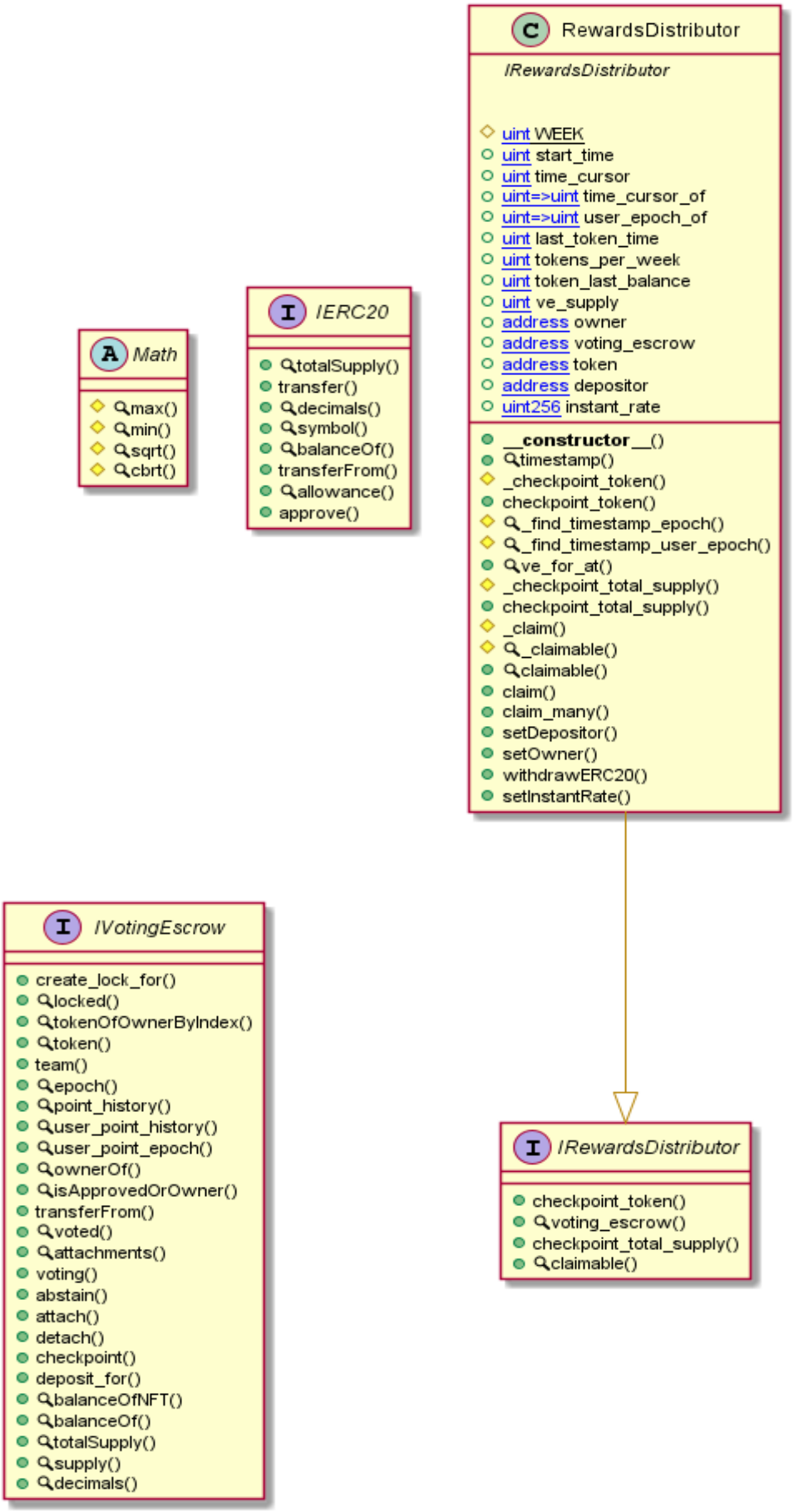
PairFees Diagram



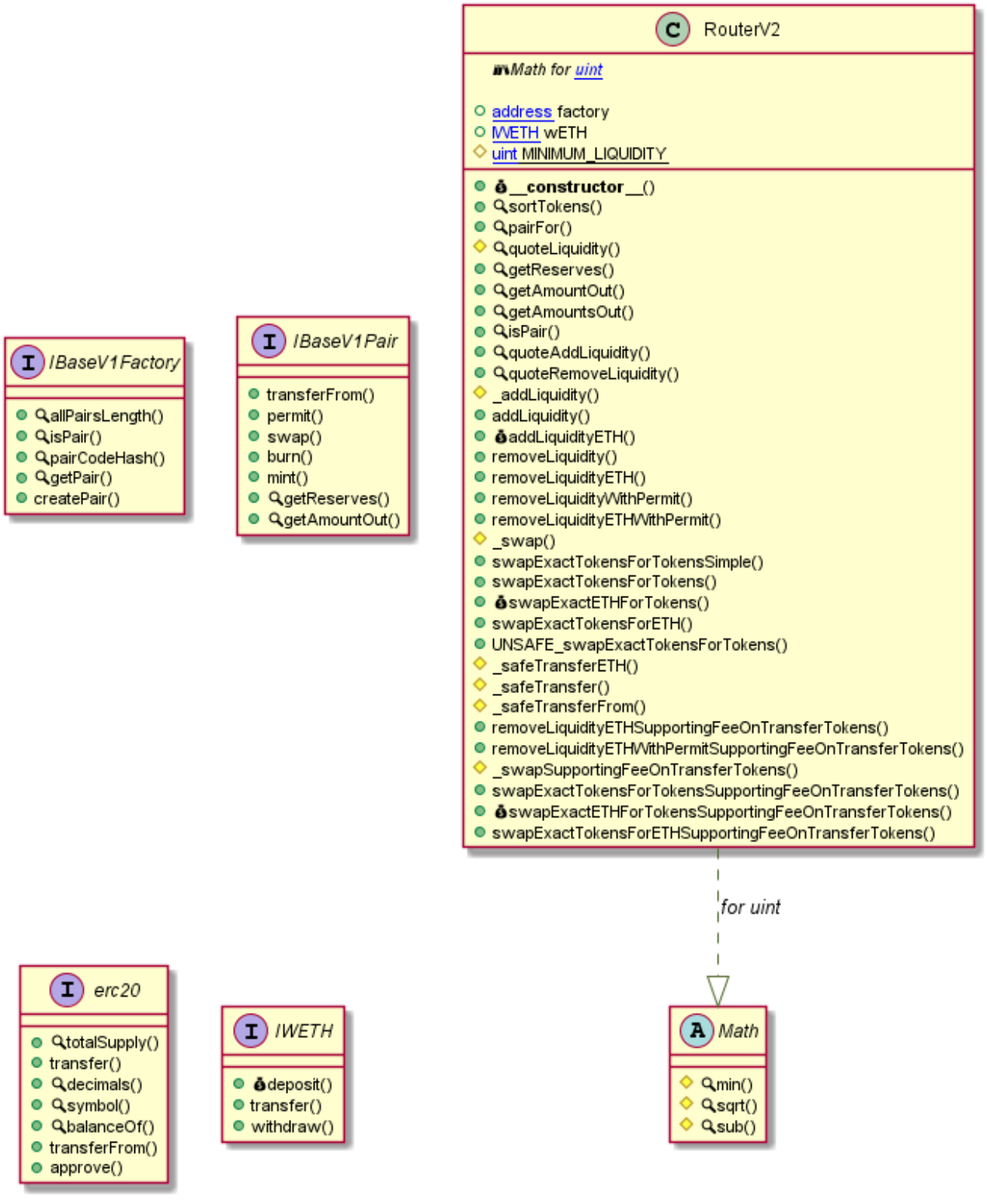
Pair Diagram



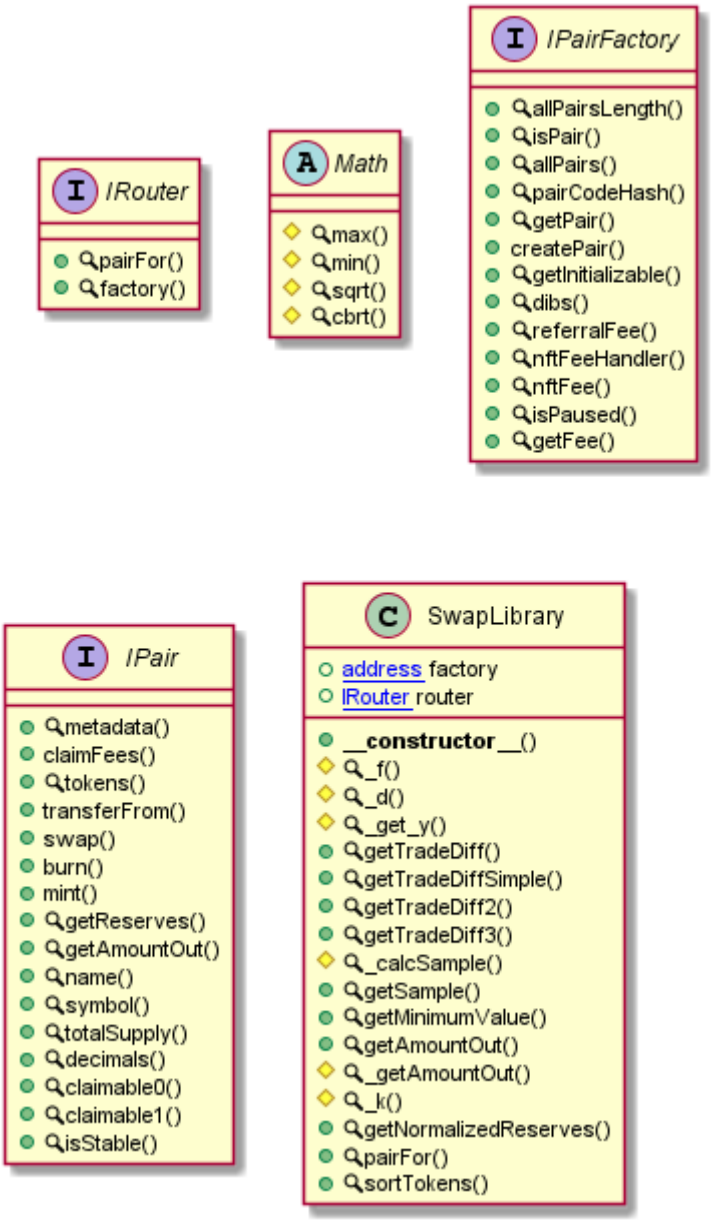
RewardsDistributor Diagram



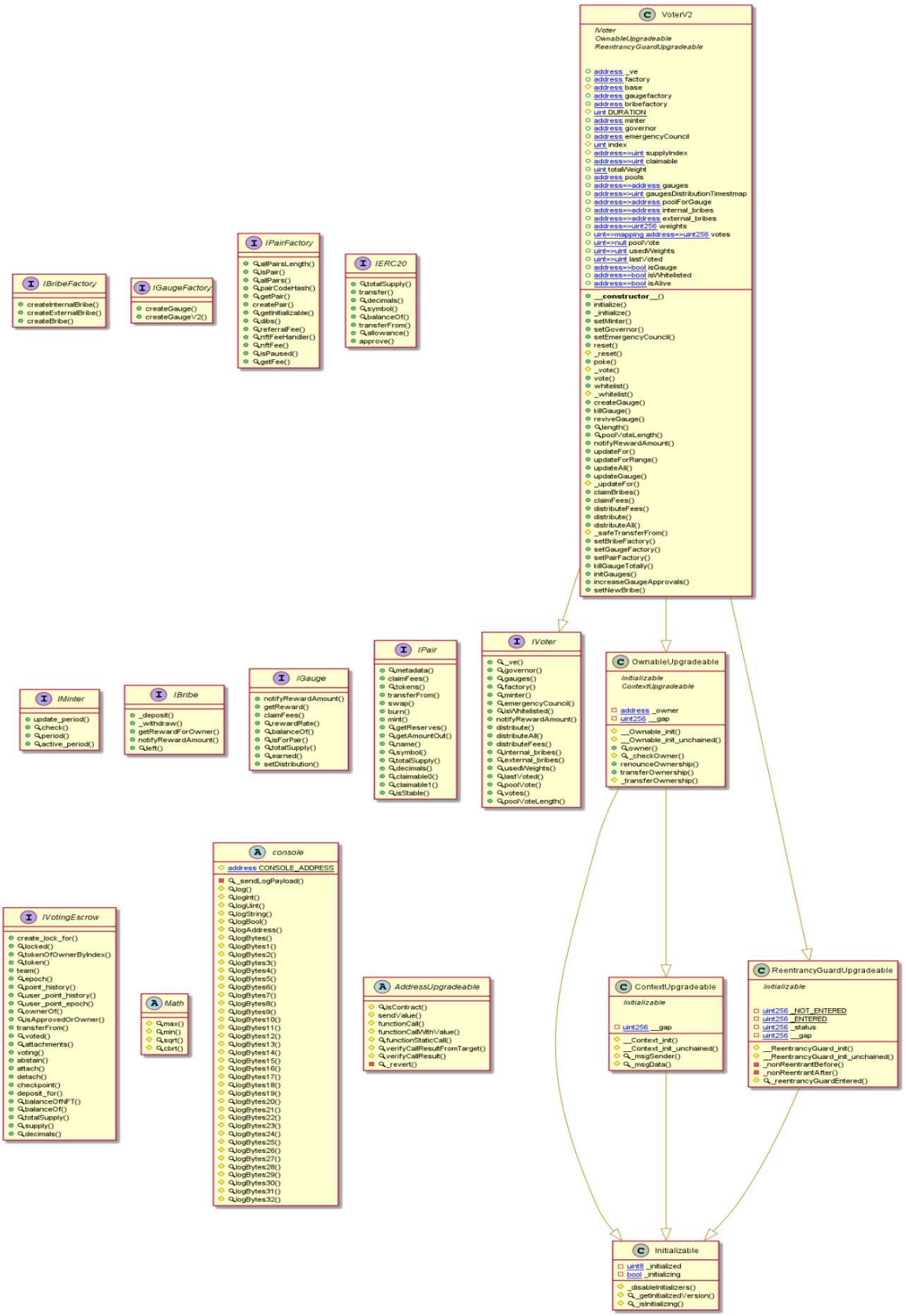
RouterV2 Diagram



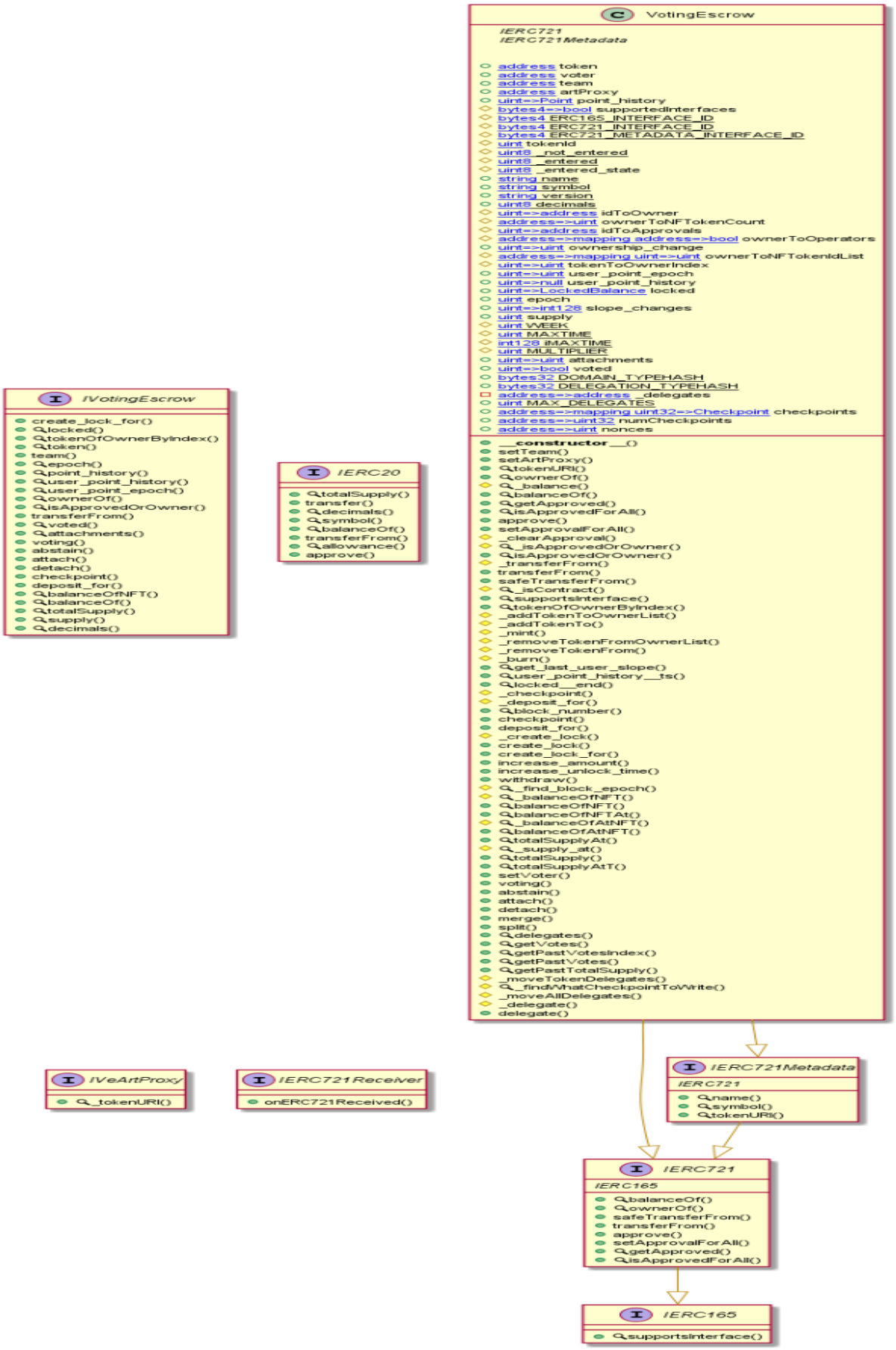
SwapLibrary Diagram



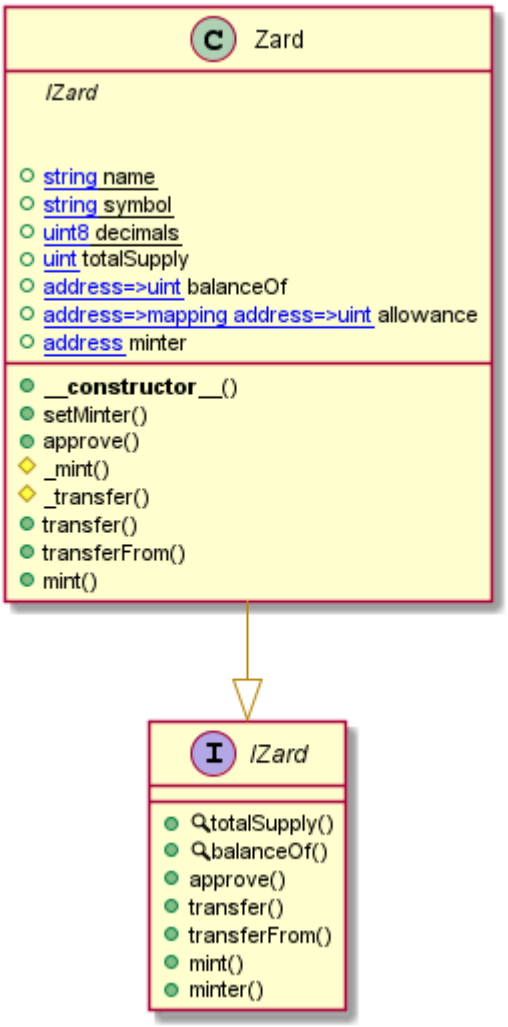
VoterV2 Diagram



VotingEscrow Diagram

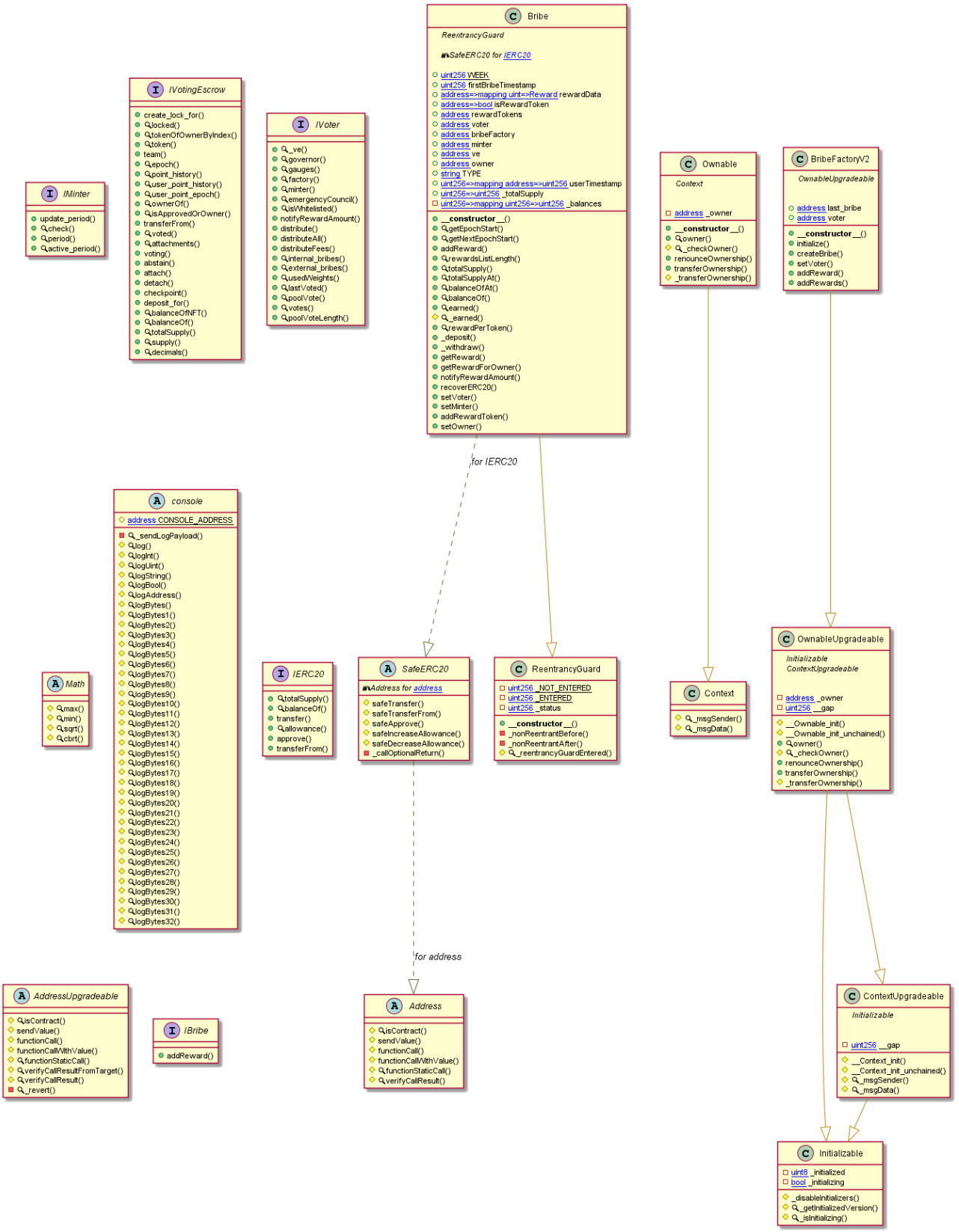


Zard Diagram

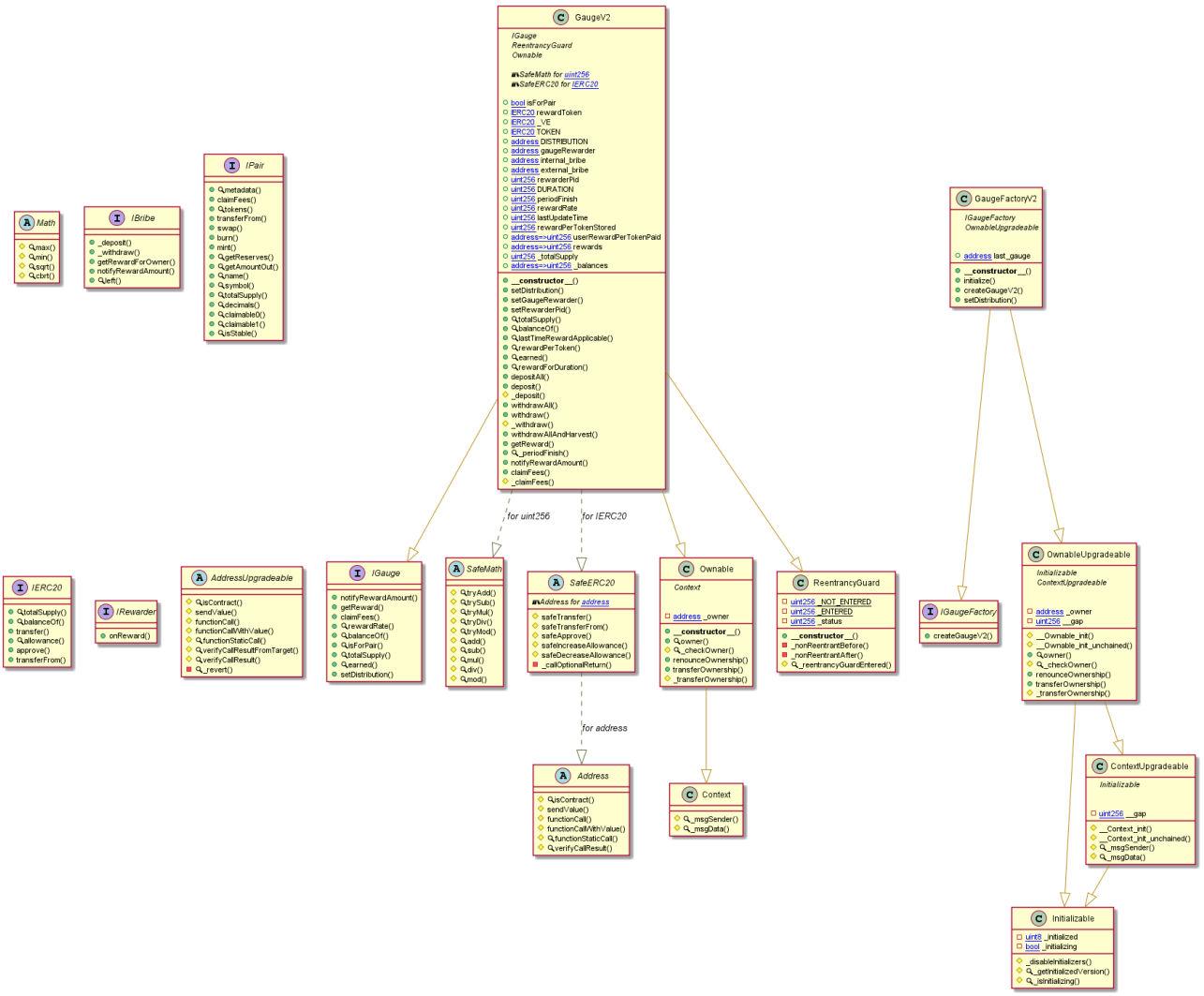


Email: audit@EtherAuthority.io

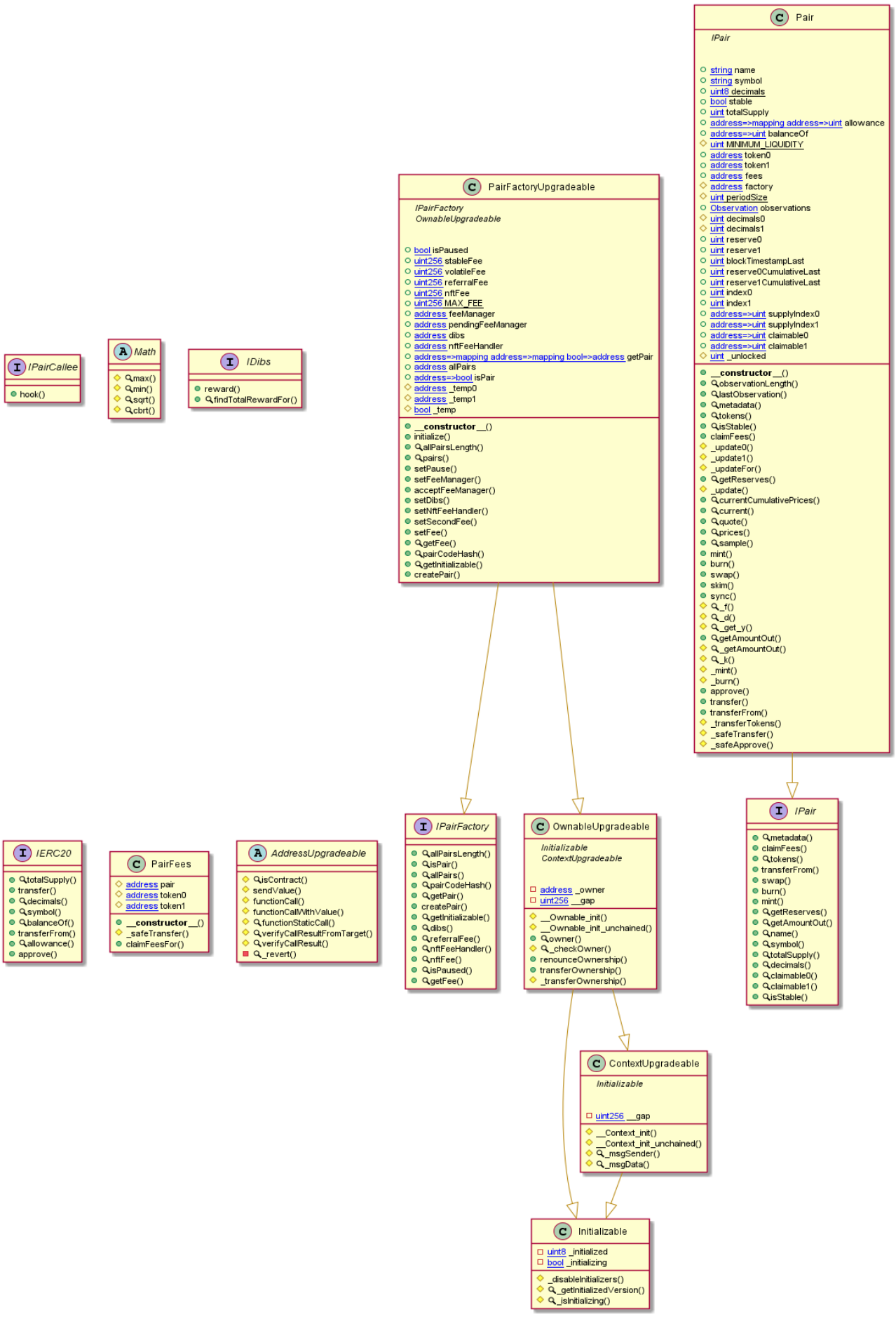
BribeFactoryV2 Diagram



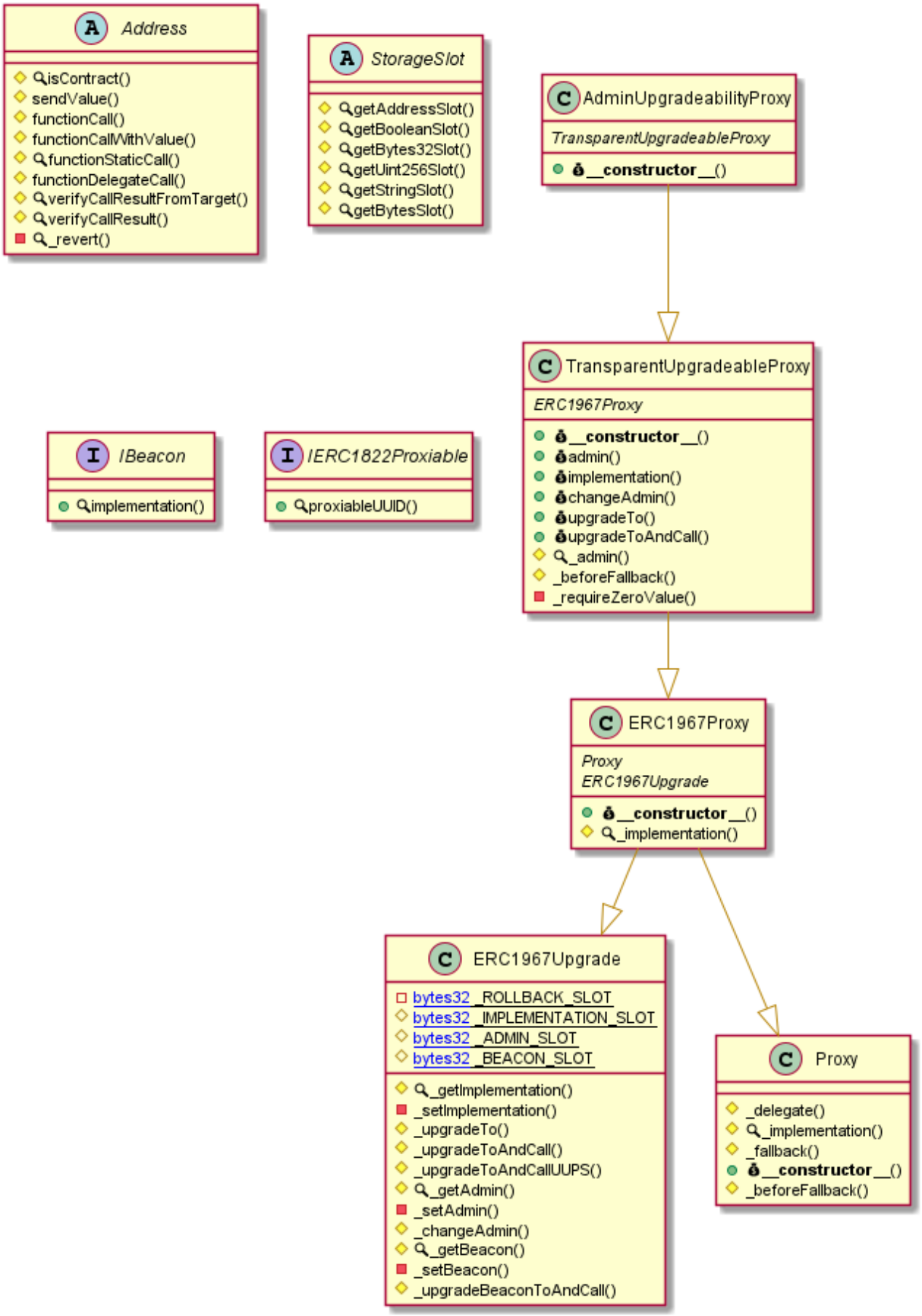
GaugeFactoryV2 Diagram



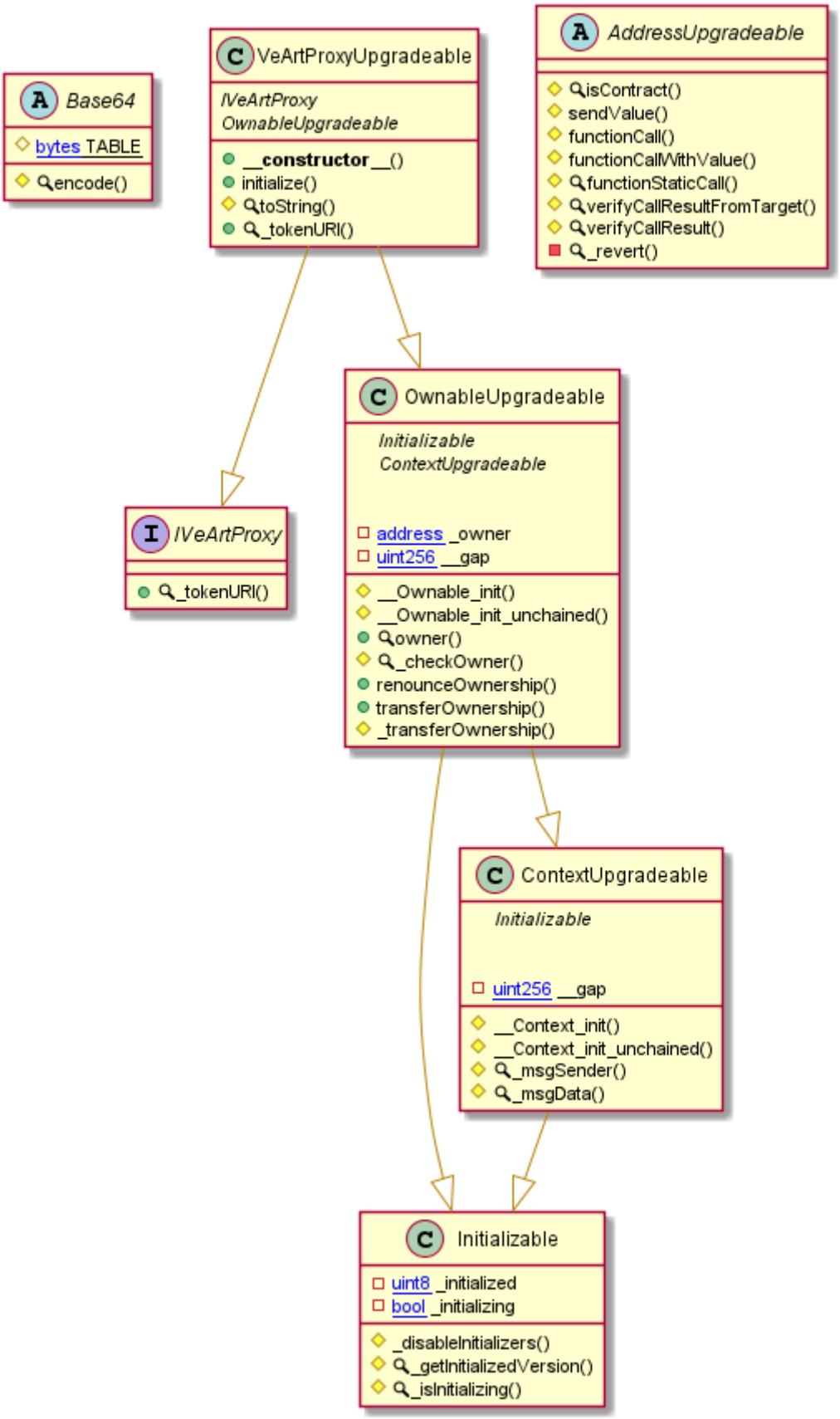
PairFactoryUpgradeable Diagram



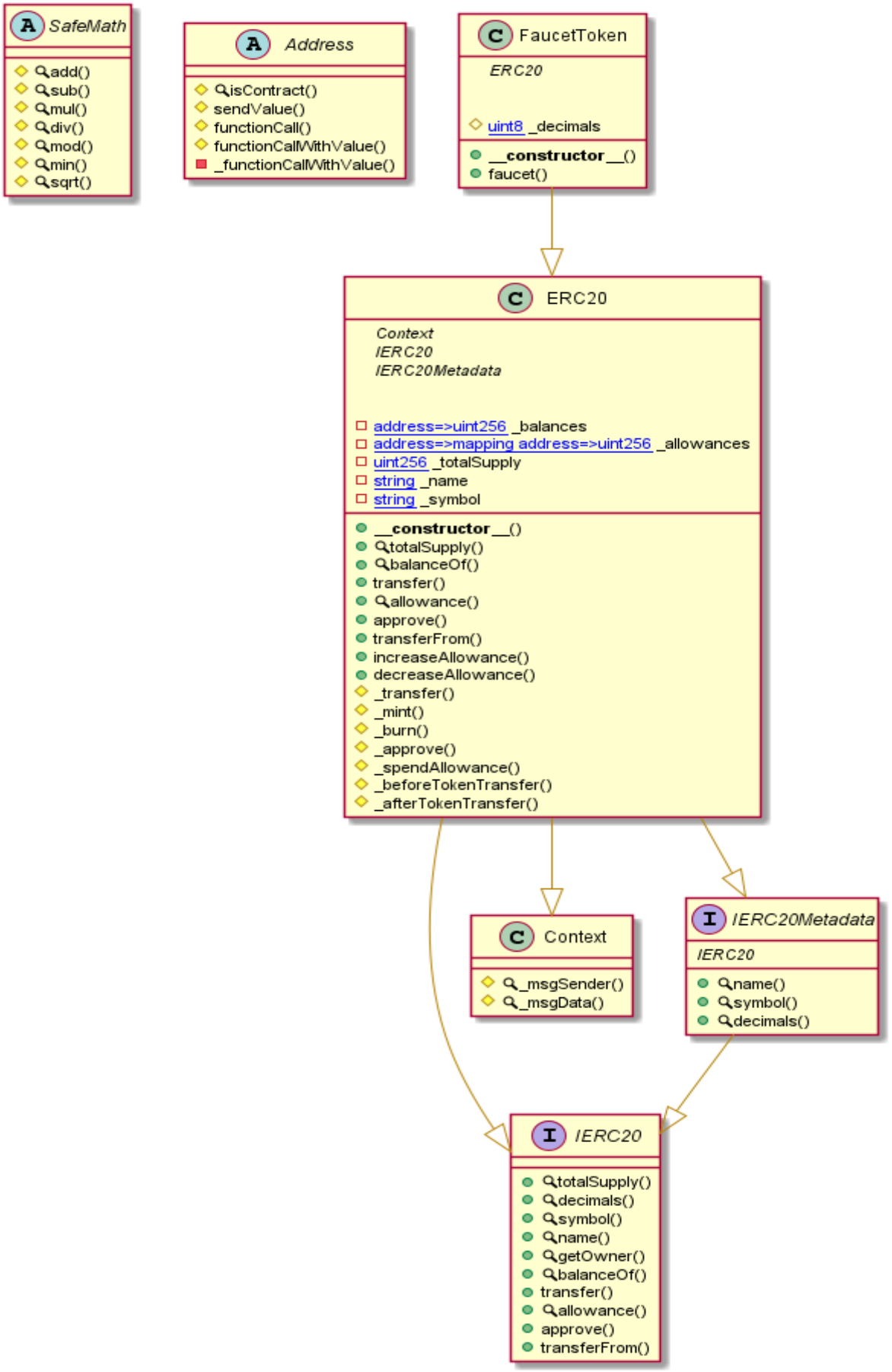
AdminUpgradeabilityProxy Diagram



VeArtProxyUpgradeable Diagram



FaucetToken Diagram



Email: audit@EtherAuthority.io

Slither Results Log

Slither log >> Bribes.sol

```
IVoter.votes(uint256,address).votes (Bribes.sol#81) shadows:
        - IVoter.votes(uint256,address) (Bribes.sol#81) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Bribe.setOwner(address) (Bribes.sol#2399-2402) should emit an event for:
        - owner = _owner (Bribes.sol#2401)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

Bribe.earned(uint256,address) (Bribes.sol#2249-2274) has external calls inside a loop: _endTimestamp = IMinter(minter).active_
period() (Bribes.sol#2252)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
Reentrancy in Bribe.notifyRewardAmount(address,uint256) (Bribes.sol#2356-2373):
        External calls:
        - IERC20(_rewardsToken).safeTransferFrom(msg.sender,address(this),reward) (Bribes.sol#2358)
        State variables written after the call(s):
        - firstBribeTimestamp = _startTimestamp (Bribes.sol#2363)
        - rewardData[_rewardsToken][_startTimestamp].rewardsPerEpoch = _lastReward + reward (Bribes.sol#2368)
        - rewardData[_rewardsToken][_startTimestamp].lastUpdateTime = block.timestamp (Bribes.sol#2369)
        - rewardData[_rewardsToken][_startTimestamp].periodFinish = _startTimestamp + WEEK (Bribes.sol#2370)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Pragma version^0.8.11 (Bribes.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Bribes.sol#1774-1779):
        - (success) = recipient.call{value: amount}() (Bribes.sol#1777)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Bribes.sol#1842-1853):
        - (success,returndata) = target.call{value: value}(data) (Bribes.sol#1851)
Low level call in Address.functionStaticCall(address,bytes,string) (Bribes.sol#1871-1880):
        - (success,returndata) = target.staticcall(data) (Bribes.sol#1878)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IMinter.update_period() (Bribes.sol#5) is not in mixedCase
Function IMinter.active_period() (Bribes.sol#8) is not in mixedCase
Function IVotingEscrow.create_lock_for(uint256,uint256,address) (Bribes.sol#26) is not in mixedCase
Parameter IVotingEscrow.create_lock_for(uint256,uint256,address)._lock_duration (Bribes.sol#26) is not in mixedCase
Function IVotingEscrow.point_history(uint256) (Bribes.sol#34) is not in mixedCase
Function IVotingEscrow.user_point_history(uint256,uint256) (Bribes.sol#35) is not in mixedCase
Function IVotingEscrow.user_point_epoch(uint256) (Bribes.sol#36) is not in mixedCase
Function IVotingEscrow.deposit_for(uint256,uint256) (Bribes.sol#50) is not in mixedCase
Function IVoter._ve() (Bribes.sol#63) is not in mixedCase
Function IVoter.internal_bribes(address) (Bribes.sol#75) is not in mixedCase
Function IVoter.external_bribes(address) (Bribes.sol#76) is not in mixedCase
Contract console (Bribes.sol#118-1646) is not in CapWords
Parameter Bribe.addReward(address)._rewardsToken (Bribes.sol#2215) is not in mixedCase
Parameter Bribe.totalSupplyAt(uint256)._timestamp (Bribes.sol#2233) is not in mixedCase
Parameter Bribe.balanceOfAt(uint256,uint256)._timestamp (Bribes.sol#2238) is not in mixedCase
Parameter Bribe.earned(uint256,address)._rewardToken (Bribes.sol#2249) is not in mixedCase
Parameter Bribe.rewardPerToken(address,uint256)._rewardsToken (Bribes.sol#2287) is not in mixedCase
Parameter Bribe.rewardPerToken(address,uint256)._timestmap (Bribes.sol#2287) is not in mixedCase
Function Bribe._deposit(uint256,uint256) (Bribes.sol#2297-2305) is not in mixedCase
Function Bribe._withdraw(uint256,uint256) (Bribes.sol#2307-2320) is not in mixedCase
Parameter Bribe.notifyRewardAmount(address,uint256)._rewardsToken (Bribes.sol#2356) is not in mixedCase
Parameter Bribe.setVoter(address)._Voter (Bribes.sol#2383) is not in mixedCase
Parameter Bribe.setMinter(address)._minter (Bribes.sol#2388) is not in mixedCase
```

```
Parameter Bribe.setVoter(address)._Voter (Bribes.sol#2383) is not in mixedCase
Parameter Bribe.setMinter(address)._minter (Bribes.sol#2388) is not in mixedCase
Parameter Bribe.addRewardToken(address)._token (Bribes.sol#2393) is not in mixedCase
Parameter Bribe.setOwner(address)._owner (Bribes.sol#2399) is not in mixedCase
Variable Bribe.TYPE (Bribes.sol#2178) is not in mixedCase
Variable Bribe._totalSupply (Bribes.sol#2184) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "k (Bribes.sol#2265)" inBribe (Bribes.sol#2155-2413)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements



Variable Bribe.getReward(uint256,address[])._rewardToken (Bribes.sol#2329) is too similar to Bribe.rewardTokens (Bribes.sol#21
71)
Variable Bribe._earned(uint256,address,uint256)._rewardToken (Bribes.sol#2276) is too similar to Bribe.rewardTokens (Bribes.so
l#2171)
Variable Bribe.earned(uint256,address)._rewardToken (Bribes.sol#2249) is too similar to Bribe.rewardTokens (Bribes.sol#2171)
Variable Bribe.getRewardForOwner(uint256,address[])._rewardToken (Bribes.sol#2346) is too similar to Bribe.rewardTokens (Bribe
s.sol#2171)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Bribe.TYPE (Bribes.sol#2178) should be immutable
Bribe.bribeFactory (Bribes.sol#2173) should be immutable
Bribe.ve (Bribes.sol#2175) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Bribes.sol analyzed (12 contracts with 84 detectors), 443 result(s) found
```

Slither log >> GaugeV2.sol

```
GaugeV2.setDistribution(address) (GaugeV2.sol#808-812) should emit an event for:
        - DISTRIBUTION = _distribution (GaugeV2.sol#811)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

GaugeV2.setRewarderPid(uint256) (GaugeV2.sol#822-826) should emit an event for:
        - rewarderPid = _pid (GaugeV2.sol#825)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
```

```
GaugeV2.constructor(address,address,address,address,address,address,bool)._distribution (GaugeV2.sol#793) lacks a zero-check o
n :
                - DISTRIBUTION = _distribution (GaugeV2.sol#797)
GaugeV2.constructor(address,address,address,address,address,address,bool)._internal_bribe (GaugeV2.sol#793) lacks a zero-check
 on :
                - internal_bribe = _internal_bribe (GaugeV2.sol#800)
GaugeV2.constructor(address,address,address,address,address,address,bool)._external_bribe (GaugeV2.sol#793) lacks a zero-check
 on :
                - external_bribe = _external_bribe (GaugeV2.sol#801)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
GaugeV2.getReward() (GaugeV2.sol#926-937) uses timestamp for comparisons
        Dangerous comparisons:
        - reward > 0 (GaugeV2.sol#928)
GaugeV2.notifyRewardAmount(address,uint256) (GaugeV2.sol#944-966) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= periodFinish (GaugeV2.sol#948)
        - require(bool,string)(rewardRate <= balance.div(DURATION),Provided reward too high) (GaugeV2.sol#961)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Pragma version0.8.13 (GaugeV2.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (GaugeV2.sol#368-373):
        - (success) = recipient.call{value: amount}() (GaugeV2.sol#371)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (GaugeV2.sol#417-428):
        - (success,returndata) = target.call{value: value}(data) (GaugeV2.sol#426)
Low level call in Address.functionStaticCall(address,bytes,string) (GaugeV2.sol#446-455):
        - (success,returndata) = target.staticcall(data) (GaugeV2.sol#453)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IBribe._deposit(uint256,uint256) (GaugeV2.sol#40) is not in mixedCase
Function IBribe._withdraw(uint256,uint256) (GaugeV2.sol#41) is not in mixedCase
Parameter GaugeV2.setDistribution(address)._distribution (GaugeV2.sol#808) is not in mixedCase
Parameter GaugeV2.setGaugeRewarder(address)._gaugeRewarder (GaugeV2.sol#815) is not in mixedCase
Parameter GaugeV2.setRewarderPid(uint256)._pid (GaugeV2.sol#822) is not in mixedCase
Function GaugeV2._periodFinish() (GaugeV2.sol#939-941) is not in mixedCase
Variable GaugeV2._VE (GaugeV2.sol#751) is not in mixedCase
Variable GaugeV2.TOKEN (GaugeV2.sol#752) is not in mixedCase
Variable GaugeV2.DISTRIBUTION (GaugeV2.sol#754) is not in mixedCase
Variable GaugeV2.internal_bribe (GaugeV2.sol#756) is not in mixedCase
Variable GaugeV2.external_bribe (GaugeV2.sol#757) is not in mixedCase
Variable GaugeV2.DURATION (GaugeV2.sol#760) is not in mixedCase
Variable GaugeV2._totalSupply (GaugeV2.sol#769) is not in mixedCase
Variable GaugeV2._balances (GaugeV2.sol#770) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
GaugeV2.DURATION (GaugeV2.sol#760) should be immutable
GaugeV2.TOKEN (GaugeV2.sol#752) should be immutable
GaugeV2._VE (GaugeV2.sol#751) should be immutable
GaugeV2.external_bribe (GaugeV2.sol#757) should be immutable
GaugeV2.internal_bribe (GaugeV2.sol#756) should be immutable
GaugeV2.isForPair (GaugeV2.sol#747) should be immutable
GaugeV2.rewardToken (GaugeV2.sol#750) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
GaugeV2.sol analyzed (13 contracts with 84 detectors), 66 result(s) found
```

Slither log >> MinterUpgradeable.sol

```
MinterUpgradeable.setTeamRate(uint256) (MinterUpgradeable.sol#649-653) should emit an event for:
        - teamRate = _teamRate (MinterUpgradeable.sol#652)
MinterUpgradeable.setEmission(uint256) (MinterUpgradeable.sol#655-659) should emit an event for:
        - EMISSION = _emission (MinterUpgradeable.sol#658)
MinterUpgradeable.setRebase(uint256) (MinterUpgradeable.sol#662-666) should emit an event for:
        - REBASEMAX = _rebase (MinterUpgradeable.sol#665)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

MinterUpgradeable.setTeam(address)._team (MinterUpgradeable.sol#633) lacks a zero-check on :
                - pendingTeam = _team (MinterUpgradeable.sol#635)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

MinterUpgradeable._initialize(address[],uint256[],uint256) (MinterUpgradeable.sol#615-631) has external calls inside a loop: _
ve.create_lock_for(amounts[i],LOCK,claimants[i]) (MinterUpgradeable.sol#625)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
Reentrancy in MinterUpgradeable.update_period() (MinterUpgradeable.sol#702-737):
        External calls:
        - _zard.mint(address(this),_required - _balanceOf) (MinterUpgradeable.sol#722)
        - require(bool)(_zard.transfer(team,_teamEmissions)) (MinterUpgradeable.sol#725)
        - require(bool)(_zard.transfer(address(_rewards_distributor),_rebase)) (MinterUpgradeable.sol#727)
        - _rewards_distributor.checkpoint_token() (MinterUpgradeable.sol#728)
        - _rewards_distributor.checkpoint_total_supply() (MinterUpgradeable.sol#729)
        - _zard.approve(address(_voter),_gauge) (MinterUpgradeable.sol#731)
        - _voter.notifyRewardAmount(_gauge) (MinterUpgradeable.sol#732)
        Event emitted after the call(s):
        - Mint(msg.sender,weekly,circulating_supply(),circulating_emission()) (MinterUpgradeable.sol#734)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

MinterUpgradeable.update_period() (MinterUpgradeable.sol#702-737) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= _period + WEEK && _initializer == address(0) (MinterUpgradeable.sol#704)
MinterUpgradeable.check() (MinterUpgradeable.sol#739-742) uses timestamp for comparisons
        Dangerous comparisons:
        - (block.timestamp >= _period + WEEK && _initializer == address(0)) (MinterUpgradeable.sol#741)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Pragma version0.8.13 (MinterUpgradeable.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Parameter MinterUpgradeable.initialize(address,address,address).__voter (MinterUpgradeable.sol#588) is not in mixedCase
Parameter MinterUpgradeable.initialize(address,address,address).__ve (MinterUpgradeable.sol#589) is not in mixedCase
Parameter MinterUpgradeable.initialize(address,address,address).__rewards_distributor (MinterUpgradeable.sol#590) is not in mi
xedCase
Function MinterUpgradeable._initialize(address[],uint256[],uint256) (MinterUpgradeable.sol#615-631) is not in mixedCase
Parameter MinterUpgradeable.setTeam(address)._team (MinterUpgradeable.sol#633) is not in mixedCase
Parameter MinterUpgradeable.setVoter(address).__voter (MinterUpgradeable.sol#643) is not in mixedCase
Parameter MinterUpgradeable.setTeamRate(uint256)._teamRate (MinterUpgradeable.sol#649) is not in mixedCase
Parameter MinterUpgradeable.setEmission(uint256)._emission (MinterUpgradeable.sol#655) is not in mixedCase
Parameter MinterUpgradeable.setRebase(uint256)._rebase (MinterUpgradeable.sol#662) is not in mixedCase
Function MinterUpgradeable.circulating_supply() (MinterUpgradeable.sol#669-671) is not in mixedCase
Function MinterUpgradeable.calculate_emission() (MinterUpgradeable.sol#674-676) is not in mixedCase
Function MinterUpgradeable.weekly_emission() (MinterUpgradeable.sol#679-681) is not in mixedCase
Function MinterUpgradeable.circulating_emission() (MinterUpgradeable.sol#684-686) is not in mixedCase
Function MinterUpgradeable.calculate_rebate(uint256) (MinterUpgradeable.sol#689-699) is not in mixedCase
Parameter MinterUpgradeable.calculate_rebate(uint256)._weeklyMint (MinterUpgradeable.sol#689) is not in mixedCase
Function MinterUpgradeable.update_period() (MinterUpgradeable.sol#702-737) is not in mixedCase
Parameter MinterUpgradeable.setRewardDistributor(address)._rewardDistro (MinterUpgradeable.sol#747) is not in mixedCase
Variable MinterUpgradeable.EMISSION (MinterUpgradeable.sol#562) is not in mixedCase
Variable MinterUpgradeable.TAIL_EMISSION (MinterUpgradeable.sol#563) is not in mixedCase
Variable MinterUpgradeable.REBASEMAX (MinterUpgradeable.sol#564) is not in mixedCase
Variable MinterUpgradeable.active_period (MinterUpgradeable.sol#571) is not in mixedCase
Variable MinterUpgradeable._initializer (MinterUpgradeable.sol#574) is not in mixedCase
Variable MinterUpgradeable._zard (MinterUpgradeable.sol#578) is not in mixedCase
Variable MinterUpgradeable._voter (MinterUpgradeable.sol#579) is not in mixedCase
Variable MinterUpgradeable._ve (MinterUpgradeable.sol#580) is not in mixedCase
Variable MinterUpgradeable._rewards_distributor (MinterUpgradeable.sol#581) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
MinterUpgradeable.sol analyzed (11 contracts with 84 detectors), 92 result(s) found
```

Slither log >> Multicall.sol

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

Pragma version>=0.5.0 (Multicall.sol#3) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.aggregate(Multicall.Call[]) (Multicall.sol#13-21):
        - (success,ret) = calls[i].target.call(calls[i].callData) (Multicall.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
Multicall.sol analyzed (1 contracts with 84 detectors), 4 result(s) found
```

Slither log >> Pair.sol

```
PairFees.constructor(address,address)._token0 (Pair.sol#114) lacks a zero-check on :
                - token0 = _token0 (Pair.sol#116)
PairFees.constructor(address,address)._token1 (Pair.sol#114) lacks a zero-check on :
                - token1 = _token1 (Pair.sol#117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Pair._update(uint256,uint256,uint256,uint256) (Pair.sol#367-384) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (Pair.sol#370)
        - timeElapsed > periodSize (Pair.sol#377)
Pair.currentCumulativePrices() (Pair.sol#387-400) uses timestamp for comparisons
        Dangerous comparisons:
        - _blockTimestampLast != blockTimestamp (Pair.sol#394)
Pair.current(address,uint256) (Pair.sol#403-414) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp == _observation.timestamp (Pair.sol#406)
Pair.quote(address,uint256,uint256) (Pair.sol#417-424) uses timestamp for comparisons
        Dangerous comparisons:
        - i < _prices.length (Pair.sol#420)
Pair.sample(address,uint256,uint256,uint256) (Pair.sol#431-451) uses timestamp for comparisons
        Dangerous comparisons:
        - i < length (Pair.sol#439)
Pair.swap(uint256,uint256,address,bytes) (Pair.sol#500-534) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_k(_balance0,_balance1) >= _k(_reserve0,_reserve1),K) (Pair.sol#529)
Pair._get_y(uint256,uint256,uint256) (Pair.sol#556-578) uses timestamp for comparisons
        Dangerous comparisons:
        - k < xy (Pair.sol#560)
        - y > y_prev (Pair.sol#567)
        - y - y_prev <= 1 (Pair.sol#568)
        - y_prev - y <= 1 (Pair.sol#572)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Pragma version0.8.13 (Pair.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Pair.decimals0 (Pair.sol#168) should be immutable
Pair.decimals1 (Pair.sol#169) should be immutable
Pair.factory (Pair.sol#154) should be immutable
Pair.fees (Pair.sol#153) should be immutable
Pair.name (Pair.sol#137) should be immutable
Pair.stable (Pair.sol#142) should be immutable
Pair.symbol (Pair.sol#138) should be immutable
Pair.token0 (Pair.sol#151) should be immutable
Pair.token1 (Pair.sol#152) should be immutable
PairFees.pair (Pair.sol#110) should be immutable
PairFees.token0 (Pair.sol#111) should be immutable
PairFees.token1 (Pair.sol#112) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Pair.sol analyzed (8 contracts with 84 detectors), 67 result(s) found
```

Slither log >> PairFees.sol

```
PairFees.constructor(address,address)._token0 (PairFees.sol#27) lacks a zero-check on :
                - token0 = _token0 (PairFees.sol#29)
PairFees.constructor(address,address)._token1 (PairFees.sol#27) lacks a zero-check on :
                - token1 = _token1 (PairFees.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Pragma version0.8.13 (PairFees.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in PairFees._safeTransfer(address,address,uint256) (PairFees.sol#33-37):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20.transfer.selector,to,value)) (PairFees.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

PairFees.pair (PairFees.sol#23) should be immutable
PairFees.token0 (PairFees.sol#24) should be immutable
PairFees.token1 (PairFees.sol#25) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
PairFees.sol analyzed (2 contracts with 84 detectors), 8 result(s) found
```

Slither log >> RewardsDistributor.sol

```
RewardsDistributor.setInstantRate(uint256) (RewardsDistributor.sol#461-465) should emit an event for:
        - instant_rate = _rate (RewardsDistributor.sol#464)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

RewardsDistributor.constructor(address)._token (RewardsDistributor.sol#155) lacks a zero-check on :
                - token = _token (RewardsDistributor.sol#156)
RewardsDistributor.constructor(address)._voting_escrow (RewardsDistributor.sol#150) lacks a zero-check on :
                - voting_escrow = _voting_escrow (RewardsDistributor.sol#157)
RewardsDistributor.setDepositor(address)._depositor (RewardsDistributor.sol#444) lacks a zero-check on :
                - depositor = _depositor (RewardsDistributor.sol#446)
RewardsDistributor.setOwner(address)._owner (RewardsDistributor.sol#449) lacks a zero-check on :
                - owner = _owner (RewardsDistributor.sol#451)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
RewardsDistributor._find_timestamp_epoch(address,uint256) (RewardsDistributor.sol#205-219) uses timestamp for comparisons
        Dangerous comparisons:
        - pt.ts <= _timestamp (RewardsDistributor.sol#212)
RewardsDistributor._find_timestamp_user_epoch(address,uint256,uint256,uint256) (RewardsDistributor.sol#221-235) uses timestamp
 for comparisons
        Dangerous comparisons:
        - pt.ts <= _timestamp (RewardsDistributor.sol#228)
RewardsDistributor._checkpoint_total_supply() (RewardsDistributor.sol#245-266) uses timestamp for comparisons
        Dangerous comparisons:
        - t > rounded_timestamp (RewardsDistributor.sol#252)
        - t > pt.ts (RewardsDistributor.sol#258)
RewardsDistributor._claim(uint256,address,uint256) (RewardsDistributor.sol#272-327) uses timestamp for comparisons
        Dangerous comparisons:
        - week_cursor == 0 (RewardsDistributor.sol#282)
        - week_cursor == 0 (RewardsDistributor.sol#292)
        - week_cursor >= last_token_time (RewardsDistributor.sol#293)
        - week_cursor < _start_time (RewardsDistributor.sol#294)
        - week_cursor >= _last_token_time (RewardsDistributor.sol#299)
        - week_cursor >= user_point.ts && user_epoch <= max_user_epoch (RewardsDistributor.sol#301)
        - balance_of == 0 && user_epoch > max_user_epoch (RewardsDistributor.sol#312)
        - balance_of != 0 (RewardsDistributor.sol#313)
RewardsDistributor._claimable(uint256,address,uint256) (RewardsDistributor.sol#329-378) uses timestamp for comparisons
        Dangerous comparisons:
        - week_cursor == 0 (RewardsDistributor.sol#339)
        - week_cursor == 0 (RewardsDistributor.sol#349)
        - week_cursor >= last_token_time (RewardsDistributor.sol#350)
        - week_cursor < _start_time (RewardsDistributor.sol#351)
        - week_cursor >= _last_token_time (RewardsDistributor.sol#356)
        - week_cursor >= user_point.ts && user_epoch <= max_user_epoch (RewardsDistributor.sol#358)
        - balance_of == 0 && user_epoch > max_user_epoch (RewardsDistributor.sol#369)
        - balance_of != 0 (RewardsDistributor.sol#370)
```

```
RewardsDistributor.claim(uint256) (RewardsDistributor.sol#385-407) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= time_cursor (RewardsDistributor.sol#386)
        - _locked.end < block.timestamp (RewardsDistributor.sol#393)
RewardsDistributor.claim_many(uint256[]) (RewardsDistributor.sol#409-442) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= time_cursor (RewardsDistributor.sol#410)
        - _locked.end < block.timestamp (RewardsDistributor.sol#423)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Function RewardsDistributor.ve_for_at(uint256,uint256) (RewardsDistributor.sol#237-243) is not in mixedCase
Parameter RewardsDistributor.ve_for_at(uint256,uint256)._tokenId (RewardsDistributor.sol#237) is not in mixedCase
Parameter RewardsDistributor.ve_for_at(uint256,uint256)._timestamp (RewardsDistributor.sol#237) is not in mixedCase
Function RewardsDistributor._checkpoint_total_supply() (RewardsDistributor.sol#245-266) is not in mixedCase
Function RewardsDistributor.checkpoint_total_supply() (RewardsDistributor.sol#268-270) is not in mixedCase
Parameter RewardsDistributor.claimable(uint256)._tokenId (RewardsDistributor.sol#380) is not in mixedCase
Parameter RewardsDistributor.claim(uint256)._tokenId (RewardsDistributor.sol#385) is not in mixedCase
Function RewardsDistributor.claim_many(uint256[]) (RewardsDistributor.sol#409-442) is not in mixedCase
Parameter RewardsDistributor.claim_many(uint256[])._tokenIds (RewardsDistributor.sol#409) is not in mixedCase
Parameter RewardsDistributor.setDepositor(address)._depositor (RewardsDistributor.sol#444) is not in mixedCase
Parameter RewardsDistributor.setOwner(address)._owner (RewardsDistributor.sol#449) is not in mixedCase
Parameter RewardsDistributor.withdrawERC20(address)._token (RewardsDistributor.sol#454) is not in mixedCase
Parameter RewardsDistributor.setInstantRate(uint256)._rate (RewardsDistributor.sol#461) is not in mixedCase
Variable RewardsDistributor.start_time (RewardsDistributor.sol#134) is not in mixedCase
Variable RewardsDistributor.time_cursor (RewardsDistributor.sol#135) is not in mixedCase
```

```
RewardsDistributor.start_time (RewardsDistributor.sol#134) should be immutable
RewardsDistributor.token (RewardsDistributor.sol#146) should be immutable
RewardsDistributor.voting_escrow (RewardsDistributor.sol#145) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
RewardsDistributor.sol analyzed (5 contracts with 84 detectors), 102 result(s) found
```

Slither log >> RouterV2.sol

```
RouterV2.constructor(address,address)._factory (RouterV2.sol#91) lacks a zero-check on :
                - factory = _factory (RouterV2.sol#92)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Pragma version0.8.13 (RouterV2.sol#7) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Contract erc20 (RouterV2.sol#28-36) is not in CapWords
Struct RouterV2.route (RouterV2.sol#72-76) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Variable RouterV2.quoteAddLiquidity(address,address,bool,uint256,uint256).amountADesired (RouterV2.sol#162) is too similar to
RouterV2._addLiquidity(address,address,bool,uint256,uint256,uint256,uint256).amountBDesired (RouterV2.sol#216)
Variable RouterV2.quoteAddLiquidity(address,address,bool,uint256,uint256).amountADesired (RouterV2.sol#162) is too similar to
RouterV2.addLiquidity(address,address,bool,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (RouterV2.sol#249)
Variable RouterV2.quoteAddLiquidity(address,address,bool,uint256,uint256).amountAOptimal (RouterV2.sol#183) is too similar to
RouterV2.quoteAddLiquidity(address,address,bool,uint256,uint256).amountBOptimal (RouterV2.sol#178)
Variable RouterV2._addLiquidity(address,address,bool,uint256,uint256,uint256,uint256).amountAOptimal (RouterV2.sol#236) is too
 similar to RouterV2._addLiquidity(address,address,bool,uint256,uint256,uint256,uint256).amountBOptimal (RouterV2.sol#231)
Variable RouterV2.quoteAddLiquidity(address,address,bool,uint256,uint256).amountAOptimal (RouterV2.sol#183) is too similar to
RouterV2._addLiquidity(address,address,bool,uint256,uint256,uint256,uint256).amountBOptimal (RouterV2.sol#231)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
RouterV2.sol analyzed (6 contracts with 84 detectors), 25 result(s) found
```

Slither log >> SwapLibrary.sol

```
SwapLibrary.constructor(address)._router (SwapLibrary.sol#90) lacks a zero-check on :
                - factory = IRouter(_router).factory() (SwapLibrary.sol#92)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Variable 'SwapLibrary._getAmountOut(uint256,address,uint256,uint256,address,uint256,uint256,bool).reserveB (SwapLibrary.sol#20
6)' in SwapLibrary._getAmountOut(uint256,address,uint256,uint256,address,uint256,uint256,bool) (SwapLibrary.sol#201-214) poten
tially used before declaration: (reserveA,reserveB) = (_reserve0,_reserve1) (SwapLibrary.sol#211)
Variable 'SwapLibrary._getAmountOut(uint256,address,uint256,uint256,address,uint256,uint256,bool).reserveA (SwapLibrary.sol#20
6)' in SwapLibrary._getAmountOut(uint256,address,uint256,uint256,address,uint256,uint256,bool) (SwapLibrary.sol#201-214) poten
tially used before declaration: (reserveA,reserveB) = (_reserve0,_reserve1) (SwapLibrary.sol#211)
Variable 'SwapLibrary._getAmountOut(uint256,address,uint256,uint256,address,uint256,uint256,bool).reserveB (SwapLibrary.sol#20
6)' in SwapLibrary._getAmountOut(uint256,address,uint256,uint256,address,uint256,uint256,bool) (SwapLibrary.sol#201-214) poten
tially used before declaration: (reserveA,reserveB) = (_reserve1,_reserve0) (SwapLibrary.sol#211)
Variable 'SwapLibrary._getAmountOut(uint256,address,uint256,uint256,address,uint256,uint256,bool).reserveA (SwapLibrary.sol#20
6)' in SwapLibrary._getAmountOut(uint256,address,uint256,uint256,address,uint256,uint256,bool) (SwapLibrary.sol#201-214) poten
tially used before declaration: (reserveA,reserveB) = (_reserve1,_reserve0) (SwapLibrary.sol#211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Math.cbrt(uint256) (SwapLibrary.sol#31-42) is never used and should be removed
Math.min(uint256,uint256) (SwapLibrary.sol#16-18) is never used and should be removed
Math.sqrt(uint256) (SwapLibrary.sol#19-30) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.13 (SwapLibrary.sol#3) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function SwapLibrary._get_y(uint256,uint256,uint256) (SwapLibrary.sol#103-125) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
SwapLibrary.sol analyzed (5 contracts with 84 detectors), 19 result(s) found
```

Slither log >> VoterV2.sol

```
VoterV2.initialize(address,address,address,address)._bribes (VoterV2.sol#2271) lacks a zero-check on :
                - bribefactory = _bribes (VoterV2.sol#2278)
VoterV2._initialize(address[],address)._minter (VoterV2.sol#2286) lacks a zero-check on :
                - minter = _minter (VoterV2.sol#2291)
VoterV2.setMinter(address)._minter (VoterV2.sol#2294) lacks a zero-check on :
                - minter = _minter (VoterV2.sol#2296)
VoterV2.setGovernor(address)._governor (VoterV2.sol#2299) lacks a zero-check on :
                - governor = _governor (VoterV2.sol#2301)
VoterV2.setEmergencyCouncil(address)._council (VoterV2.sol#2304) lacks a zero-check on :
                - emergencyCouncil = _council (VoterV2.sol#2306)
VoterV2.setBribeFactory(address)._bribeFactory (VoterV2.sol#2583) lacks a zero-check on :
                - bribefactory = _bribeFactory (VoterV2.sol#2585)
VoterV2.setGaugeFactory(address)._gaugeFactory (VoterV2.sol#2588) lacks a zero-check on :
                - gaugefactory = _gaugeFactory (VoterV2.sol#2590)
VoterV2.setPairFactory(address)._factory (VoterV2.sol#2593) lacks a zero-check on :
                - factory = _factory (VoterV2.sol#2595)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Variable VoterV2.createGauge(address)._external_bribe (VoterV2.sol#2432) is too similar to VoterV2.external_bribes (VoterV2.so
l#2246)
Variable VoterV2.createGauge(address)._internal_bribe (VoterV2.sol#2429) is too similar to VoterV2.internal_bribes (VoterV2.so
l#2245)
Variable VoterV2._vote(uint256,address[],uint256[])._usedWeight (VoterV2.sol#2362) is too similar to VoterV2.usedWeights (Vote
rV2.sol#2250)
Variable VoterV2.initGauges(address[],address[])._external_bribe (VoterV2.sol#2635) is too similar to VoterV2.external_bribes
(VoterV2.sol#2246)
Variable VoterV2.initGauges(address[],address[])._internal_bribe (VoterV2.sol#2633) is too similar to VoterV2.internal_bribes
(VoterV2.sol#2245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

VoterV2.DURATION (VoterV2.sol#2230) is never used in VoterV2 (VoterV2.sol#2223-2664)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
VoterV2.sol analyzed (18 contracts with 84 detectors), 535 result(s) found
```

Slither log >> VotingEscrow.sol

```
VotingEscrow.constructor(address,address).token_addr (VotingEscrow.sol#339) lacks a zero-check on :
                - token = token_addr (VotingEscrow.sol#340)
VotingEscrow.constructor(address,address).art_proxy (VotingEscrow.sol#339) lacks a zero-check on :
                - artProxy = art_proxy (VotingEscrow.sol#343)
VotingEscrow.setTeam(address)._team (VotingEscrow.sol#382) lacks a zero-check on :
                - team = _team (VotingEscrow.sol#384)
VotingEscrow.setArtProxy(address)._proxy (VotingEscrow.sol#387) lacks a zero-check on :
                - artProxy = _proxy (VotingEscrow.sol#389)
VotingEscrow.setVoter(address)._voter (VotingEscrow.sol#1288) lacks a zero-check on :
                - voter = _voter (VotingEscrow.sol#1290)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

VotingEscrow._deposit_for(uint256,uint256,uint256,VotingEscrow.LockedBalance,VotingEscrow.DepositType) (VotingEscrow.sol#949-9
82) has external calls inside a loop: assert(bool)(IERC20(token).transferFrom(from,address(this),_value)) (VotingEscrow.sol#97
7)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
VotingEscrow._checkpoint(uint256,VotingEscrow.LockedBalance,VotingEscrow.LockedBalance) (VotingEscrow.sol#806-941) uses timest
amp for comparisons
        Dangerous comparisons:
        - old_locked.end > block.timestamp && old_locked.amount > 0 (VotingEscrow.sol#820)
        - new_locked.end > block.timestamp && new_locked.amount > 0 (VotingEscrow.sol#824)
        - new_locked.end != 0 (VotingEscrow.sol#833)
        - new_locked.end == old_locked.end (VotingEscrow.sol#834)
        - block.timestamp > last_point.ts (VotingEscrow.sol#852)
        - t_i > block.timestamp (VotingEscrow.sol#866)
        - last_point.bias < 0 (VotingEscrow.sol#873)
        - last_point.slope < 0 (VotingEscrow.sol#877)
        - t_i == block.timestamp (VotingEscrow.sol#885)
        - last_point.slope < 0 (VotingEscrow.sol#902)
        - last_point.bias < 0 (VotingEscrow.sol#905)
        - old_locked.end > block.timestamp (VotingEscrow.sol#917)
        - new_locked.end == old_locked.end (VotingEscrow.sol#920)
        - new_locked.end > block.timestamp (VotingEscrow.sol#926)
        - new_locked.end > old_locked.end (VotingEscrow.sol#927)
VotingEscrow._deposit_for(uint256,uint256,uint256,VotingEscrow.LockedBalance,VotingEscrow.DepositType) (VotingEscrow.sol#949-9
82) uses timestamp for comparisons
        Dangerous comparisons:
        - unlock_time != 0 (VotingEscrow.sol#964)
VotingEscrow.deposit_for(uint256,uint256) (VotingEscrow.sol#998-1005) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_locked.end > block.timestamp,Cannot add to expired lock. Withdraw) (VotingEscrow.sol#1003)
VotingEscrow._create_lock(uint256,uint256,address) (VotingEscrow.sol#1011-1024) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(unlock_time > block.timestamp,Can only lock until time in the future) (VotingEscrow.sol#1015)
        - require(bool,string)(unlock_time <= block.timestamp + MAXTIME,Voting lock can be 2 years max) (VotingEscrow.sol#1016
)
```

```
Variable VotingEscrow._moveAllDelegates(address,address,address).tId_scope_1 (VotingEscrow.sol#1600) is too similar to VotingE
scrow._moveAllDelegates(address,address,address).tId_scope_3 (VotingEscrow.sol#1605)
Variable VotingEscrow._moveTokenDelegates(address,address,uint256).tId_scope_1 (VotingEscrow.sol#1529) is too similar to Votin
gEscrow._moveAllDelegates(address,address,address).tId_scope_3 (VotingEscrow.sol#1605)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
VotingEscrow.sol analyzed (8 contracts with 84 detectors), 151 result(s) found
```

Slither log >> Zard.sol

```
Zard.setMinter(address)._minter (Zard.sol#37) lacks a zero-check on :
                - minter = _minter (Zard.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Pragma version0.8.13 (Zard.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter Zard.setMinter(address)._minter (Zard.sol#37) is not in mixedCase
Parameter Zard.approve(address,uint256)._spender (Zard.sol#42) is not in mixedCase
Parameter Zard.approve(address,uint256)._value (Zard.sol#42) is not in mixedCase
Parameter Zard.transfer(address,uint256)._to (Zard.sol#66) is not in mixedCase
Parameter Zard.transfer(address,uint256)._value (Zard.sol#66) is not in mixedCase
Parameter Zard.transferFrom(address,address,uint256)._from (Zard.sol#70) is not in mixedCase
Parameter Zard.transferFrom(address,address,uint256)._to (Zard.sol#70) is not in mixedCase
Parameter Zard.transferFrom(address,address,uint256)._value (Zard.sol#70) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
Zard.sol analyzed (2 contracts with 84 detectors), 11 result(s) found
```

Slither log >> BribeFactoryV2.sol

```
Pragma version^0.8.11 (BribeFactoryV2.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (BribeFactoryV2.sol#1774-1779):
        - (success) = recipient.call{value: amount}() (BribeFactoryV2.sol#1777)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (BribeFactoryV2.sol#1842-1853):
        - (success,returndata) = target.call{value: value}(data) (BribeFactoryV2.sol#1851)
Low level call in Address.functionStaticCall(address,bytes,string) (BribeFactoryV2.sol#1871-1880):
        - (success,returndata) = target.staticcall(data) (BribeFactoryV2.sol#1878)
Low level call in AddressUpgradeable.sendValue(address,uint256) (BribeFactoryV2.sol#2470-2475):
        - (success) = recipient.call{value: amount}() (BribeFactoryV2.sol#2473)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (BribeFactoryV2.sol#2534-2543):
        - (success,returndata) = target.call{value: value}(data) (BribeFactoryV2.sol#2541)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (BribeFactoryV2.sol#2561-2568):
        - (success,returndata) = target.staticcall(data) (BribeFactoryV2.sol#2566)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Bribe (BribeFactoryV2.sol#2155-2413) should inherit from IBribe (BribeFactoryV2.sol#2834-2836)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
```

```
Redundant expression "k (BribeFactoryV2.sol#2265)" inBribe (BribeFactoryV2.sol#2155-2413)
Redundant expression "i (BribeFactoryV2.sol#2866)" inBribeFactoryV2 (BribeFactoryV2.sol#2838-2886)
Redundant expression "i (BribeFactoryV2.sol#2876)" inBribeFactoryV2 (BribeFactoryV2.sol#2838-2886)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements



Variable Bribe.getReward(uint256,address[])._rewardToken (BribeFactoryV2.sol#2329) is too similar to Bribe.rewardTokens (Bribe
FactoryV2.sol#2171)
Variable Bribe.getRewardForOwner(uint256,address[])._rewardToken (BribeFactoryV2.sol#2346) is too similar to Bribe.rewardToken
s (BribeFactoryV2.sol#2171)
Variable Bribe.earned(uint256,address)._rewardToken (BribeFactoryV2.sol#2249) is too similar to Bribe.rewardTokens (BribeFacto
ryV2.sol#2171)
Variable Bribe._earned(uint256,address,uint256)._rewardToken (BribeFactoryV2.sol#2276) is too similar to Bribe.rewardTokens (B
ribeFactoryV2.sol#2171)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Bribe.TYPE (BribeFactoryV2.sol#2178) should be immutable
Bribe.bribeFactory (BribeFactoryV2.sol#2173) should be immutable
Bribe.ve (BribeFactoryV2.sol#2175) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
BribeFactoryV2.sol analyzed (18 contracts with 84 detectors), 488 result(s) found
```

Slither log >> GaugeFactoryV2.sol

```
Pragma version0.8.13 (GaugeFactoryV2.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (GaugeFactoryV2.sol#368-373):
        - (success) = recipient.call{value: amount}() (GaugeFactoryV2.sol#371)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (GaugeFactoryV2.sol#417-428):
        - (success,returndata) = target.call{value: value}(data) (GaugeFactoryV2.sol#426)
Low level call in Address.functionStaticCall(address,bytes,string) (GaugeFactoryV2.sol#446-455):
        - (success,returndata) = target.staticcall(data) (GaugeFactoryV2.sol#453)
Low level call in AddressUpgradeable.sendValue(address,uint256) (GaugeFactoryV2.sol#1063-1068):
        - (success) = recipient.call{value: amount}() (GaugeFactoryV2.sol#1066)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (GaugeFactoryV2.sol#1127-1136):
        - (success,returndata) = target.call{value: value}(data) (GaugeFactoryV2.sol#1134)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (GaugeFactoryV2.sol#1154-1161):
        - (success,returndata) = target.staticcall(data) (GaugeFactoryV2.sol#1159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Parameter GaugeFactoryV2.createGaugeV2(address,address,address,address,address,address,bool)._ve (GaugeFactoryV2.sol#1436) is
not in mixedCase
Parameter GaugeFactoryV2.createGaugeV2(address,address,address,address,address,address,bool)._token (GaugeFactoryV2.sol#1436)
is not in mixedCase
Parameter GaugeFactoryV2.createGaugeV2(address,address,address,address,address,address,bool)._distribution (GaugeFactoryV2.sol
#1436) is not in mixedCase
Parameter GaugeFactoryV2.createGaugeV2(address,address,address,address,address,address,bool)._internal_bribe (GaugeFactoryV2.s
ol#1436) is not in mixedCase
Parameter GaugeFactoryV2.createGaugeV2(address,address,address,address,address,address,bool)._external_bribe (GaugeFactoryV2.s
ol#1436) is not in mixedCase
Parameter GaugeFactoryV2.createGaugeV2(address,address,address,address,address,address,bool)._isPair (GaugeFactoryV2.sol#1436)
 is not in mixedCase
Parameter GaugeFactoryV2.setDistribution(address,address)._gauge (GaugeFactoryV2.sol#1441) is not in mixedCase
Parameter GaugeFactoryV2.setDistribution(address,address)._newDistribution (GaugeFactoryV2.sol#1441) is not in mixedCase
Variable GaugeFactoryV2.last_gauge (GaugeFactoryV2.sol#1428) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

GaugeV2.DURATION (GaugeFactoryV2.sol#760) should be immutable
GaugeV2.TOKEN (GaugeFactoryV2.sol#752) should be immutable
GaugeV2._VE (GaugeFactoryV2.sol#751) should be immutable
GaugeV2.external_bribe (GaugeFactoryV2.sol#757) should be immutable
GaugeV2.internal_bribe (GaugeFactoryV2.sol#756) should be immutable
GaugeV2.isForPair (GaugeFactoryV2.sol#747) should be immutable
GaugeV2.rewardToken (GaugeFactoryV2.sol#750) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
GaugeFactoryV2.sol analyzed (19 contracts with 84 detectors), 110 result(s) found
```

Slither log >> PairFactoryUpgradeable.sol

```
PairFactoryUpgradeable.setFee(bool,uint256) (PairFactoryUpgradeable.sol#1184-1192) should emit an event for:
        - stableFee = _fee (PairFactoryUpgradeable.sol#1188)
        - volatileFee = _fee (PairFactoryUpgradeable.sol#1190)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
```

```
PairFactoryUpgradeable.pairCodeHash() (PairFactoryUpgradeable.sol#1198-1200) uses literals with too many digits:
        - keccak256(bytes)(type()(Pair).creationCode) (PairFactoryUpgradeable.sol#1199)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Pair.decimals0 (PairFactoryUpgradeable.sol#168) should be immutable
Pair.decimals1 (PairFactoryUpgradeable.sol#169) should be immutable
Pair.factory (PairFactoryUpgradeable.sol#154) should be immutable
Pair.fees (PairFactoryUpgradeable.sol#153) should be immutable
Pair.name (PairFactoryUpgradeable.sol#137) should be immutable
Pair.stable (PairFactoryUpgradeable.sol#142) should be immutable
Pair.symbol (PairFactoryUpgradeable.sol#138) should be immutable
Pair.token0 (PairFactoryUpgradeable.sol#151) should be immutable
Pair.token1 (PairFactoryUpgradeable.sol#152) should be immutable
PairFees.pair (PairFactoryUpgradeable.sol#110) should be immutable
PairFees.token0 (PairFactoryUpgradeable.sol#111) should be immutable
PairFees.token1 (PairFactoryUpgradeable.sol#112) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
PairFactoryUpgradeable.sol analyzed (13 contracts with 84 detectors), 110 result(s) found
```

Slither log >> VeArtProxyUpgradeable.sol

```
Base64.encode(bytes) (VeArtProxyUpgradeable.sol#9-59) uses assembly
        - INLINE ASM (VeArtProxyUpgradeable.sol#21-56)
AddressUpgradeable._revert(bytes,string) (VeArtProxyUpgradeable.sol#263-275) uses assembly
        - INLINE ASM (VeArtProxyUpgradeable.sol#268-271)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version0.8.13 (VeArtProxyUpgradeable.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (VeArtProxyUpgradeable.sol#121-126):
        - (success) = recipient.call{value: amount}() (VeArtProxyUpgradeable.sol#124)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (VeArtProxyUpgradeable.sol#185-194):
        - (success,returndata) = target.call{value: value}(data) (VeArtProxyUpgradeable.sol#192)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (VeArtProxyUpgradeable.sol#212-219):
        - (success,returndata) = target.staticcall(data) (VeArtProxyUpgradeable.sol#217)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Function VeArtProxyUpgradeable._tokenURI(uint256,uint256,uint256,uint256) (VeArtProxyUpgradeable.sol#517-526) is not in mixedC
ase
Parameter VeArtProxyUpgradeable._tokenURI(uint256,uint256,uint256,uint256)._tokenId (VeArtProxyUpgradeable.sol#517) is not in
mixedCase
Parameter VeArtProxyUpgradeable._tokenURI(uint256,uint256,uint256,uint256)._balanceOf (VeArtProxyUpgradeable.sol#517) is not i
n mixedCase
Parameter VeArtProxyUpgradeable._tokenURI(uint256,uint256,uint256,uint256)._locked_end (VeArtProxyUpgradeable.sol#517) is not
in mixedCase
Parameter VeArtProxyUpgradeable._tokenURI(uint256,uint256,uint256,uint256)._value (VeArtProxyUpgradeable.sol#517) is not in mi
xedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
VeArtProxyUpgradeable.sol analyzed (7 contracts with 84 detectors), 40 result(s) found
```

Slither log >> import.sol

```
AdminUpgradeabilityProxy.constructor(address,address,bytes).admin (import.sol#723) shadows:
        - TransparentUpgradeableProxy.admin() (import.sol#643-646) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Modifier TransparentUpgradeableProxy.ifAdmin() (import.sol#626-632) does not always execute _; or revertReference: https://git
hub.com/crytic/slither/wiki/Detector-Documentation#incorrect-modifier
```

```
Pragma version^0.8.0 (import.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (import.sol#59-64):
        - (success) = recipient.call{value: amount}() (import.sol#62)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (import.sol#123-132):
        - (success,returndata) = target.call{value: value}(data) (import.sol#130)
Low level call in Address.functionStaticCall(address,bytes,string) (import.sol#150-157):
        - (success,returndata) = target.staticcall(data) (import.sol#155)
Low level call in Address.functionDelegateCall(address,bytes,string) (import.sol#175-182):
        - (success,returndata) = target.delegatecall(data) (import.sol#180)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
import.sol analyzed (9 contracts with 84 detectors), 43 result(s) found
```

Slither log >> FaucetToken.sol

```
FaucetToken.constructor(string,string,uint256,uint8).name (FaucetToken.sol#513) shadows:
        - IERC20Metadata.name() (FaucetToken.sol#200) (function)
        - IERC20.name() (FaucetToken.sol#162) (function)
FaucetToken.constructor(string,string,uint256,uint8).symbol (FaucetToken.sol#513) shadows:
        - IERC20Metadata.symbol() (FaucetToken.sol#205) (function)
        - IERC20.symbol() (FaucetToken.sol#160) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
Pragma version0.8.13 (FaucetToken.sol#3) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (FaucetToken.sol#94-99):
        - (success) = recipient.call{value: amount}() (FaucetToken.sol#97)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (FaucetToken.sol#131-153):
        - (success,returndata) = target.call{value: weiValue}(data) (FaucetToken.sol#139)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable FaucetToken._decimals (FaucetToken.sol#511) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (FaucetToken.sol#191)" inContext (FaucetToken.sol#185-194)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

FaucetToken (FaucetToken.sol#510-521) does not implement functions:
        - IERC20Metadata.decimals() (FaucetToken.sol#210)
        - IERC20.getOwner() (FaucetToken.sol#164)
        - IERC20Metadata.name() (FaucetToken.sol#200)
        - IERC20Metadata.symbol() (FaucetToken.sol#205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

FaucetToken._decimals (FaucetToken.sol#511) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
FaucetToken.sol analyzed (7 contracts with 84 detectors), 31 result(s) found
```

Solidity Static Analysis

Bribes.sol

Security

Check-effects-interaction:

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

more

Pos: 57:4:

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: 232:68:

Gas & Economy

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: 191:8:

Miscellaneous

Constant/View/Pure functions:

IMinter.update_period() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 5:4:

Email: audit@EtherAuthority.io

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: 220:8:

GaugeV2.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: 248:23:

Gas & Economy

Gas costs:

Gas requirement of function GaugeV2.rewardToken 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: 34:4:

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: 229:8:

MinterUpgradeable.sol

Security

Check-effects-interaction:

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

more

Pos: 44:4:

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: 202:15:

Gas & Economy

Gas costs:

Gas requirement of function MinterUpgradeable._initialize 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: 72:4:

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: 81:12:

Miscellaneous

Guard conditions:

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

more

Pos: 182:12:

Multicall.sol

Security

Block timestamp:

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

more

Pos: 33:20:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

more

Pos: 17:47:

Gas & Economy

Gas costs:

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

Pos: 13:4:

For loop over dynamic array:

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

more

Pos: 16:8:

Miscellaneous

Guard conditions:

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

more

Pos: 18:12:

Pair.sol

Security

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

more

Pos: 21:44:

Gas & Economy

Gas costs:

Gas requirement of function Pair.quote 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: 296:4:

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: 299:8:

ERC

ERC20:

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

more

Pos: 7:4:

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: 27:8:

PairFees.sol

Security

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 21:44:

Gas & Economy

Gas costs:

Gas requirement of function PairFees.claimFeesFor is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 26:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 7:4:

Miscellaneous

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 6:4:

RewardsDistributor.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: 323:33:

Gas & Economy

Gas costs:

Gas requirement of function RewardsDistributor.checkpoint_token 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: 100:4:

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: 355:8:

RouterV2.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: 82:28:

Gas & Economy

Gas costs:

Gas requirement of function RouterV2.quoteRemoveLiquidity 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: 185:4:

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: 140:8:

Miscellaneous

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 228:16:

SwapLibrary.sol

Gas & Economy

Gas costs:

Gas requirement of function SwapLibrary.factory 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: 12:2:

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: 176:4:

VoterV2.sol

Security

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 377:8:

Gas & Economy

Gas costs:

Gas requirement of function VoterV2.initialize 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: 69:4:

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: 423:8:

Miscellaneous

Similar variable names:

VoterV2._vote(uint256,address[],uint256[]) : Variables have very similar names "usedWeights" and "_usedWeight". Note: Modifiers are currently not considered by this static analysis.
Pos: 160:8:

VotingEscrow.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: 1298:26:

Gas & Economy

Gas costs:

Gas requirement of function VotingEscrow.totalSupply 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: 1022:4:

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: 1353:16:

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: 498:8:

Zard.sol

Gas & Economy

Gas costs:

Gas requirement of function Zard.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: 68:4:

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: 69:8:

BribeFactoryV2.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 128:4:

Gas & Economy

Gas costs:

Gas requirement of function BribeFactoryV2.addRewards 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: 46:4:

Miscellaneous

Constant/View/Pure functions:

AddressUpgradeable.functionStaticCall(address,bytes) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 145:4:

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: 263:8:

GaugeFactoryV2.sol

Security

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 356:12:

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: 124:24:

Gas & Economy

Gas costs:

Gas requirement of function GaugeFactoryV2.createGaugeV2 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: 18:4:

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: 180:16:

PairFactoryUpgradeable.sol

Security

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 554:44:

Gas & Economy

Gas costs:

Gas requirement of function PairFees.claimFeesFor is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 26:4:

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: 299:8:

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: 346:33:

AdminUpgradeabilityProxy.sol

Security

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 185:50:

Gas & Economy

Gas costs:

Fallback function of contract AdminUpgradeabilityProxy requires too much gas (infinite). If the fallback function requires more than 2300 gas, the contract cannot receive Ether.
Pos: 75:4:

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: 42:8:

VeArtProxyUpgradeable.sol

Security

Check-effects-interaction:

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

more

Pos: 128:4:

Gas & Economy

Gas costs:

Gas requirement of function VeArtProxyUpgradeable._tokenURI 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: 41:4:

Miscellaneous

Constant/View/Pure functions:

AddressUpgradeable.functionStaticCall(address,bytes) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 145:4:

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: 61:8:

FaucetToken.sol

Gas & Economy

Gas costs:

Gas requirement of function FaucetToken.faucet 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: 19:4:

Miscellaneous

Constant/View/Pure functions:

ERC20._beforeTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not.
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: 260:8:

Solhint Linter

Bribes.sol

```
Bribes.sol:2:1: Error: Compiler version ^0.8.13 does not satisfy the
r semver requirement
Bribes.sol:5:8: Error: Use double quotes for string literals
Bribes.sol:15:8: Error: Use double quotes for string literals
Bribes.sol:41:19: Error: Variable name must be in mixedCase
Bribes.sol:57:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
Bribes.sol:232:69: Error: Avoid to make time-based decisions in your
business logic
Bribes.sol:246:23: Error: Variable name must be in mixedCase
```

GaugeV2.sol

```
GaugeV2.sol:2:1: Error: Compiler version 0.8.13 does not satisfy the
r semver requirement
GaugeV2.sol:13:8: Error: Use double quotes for string literals
GaugeV2.sol:27:1: Error: Contract has 18 states declarations but
allowed no more than 15
GaugeV2.sol:44:20: Error: Variable name must be in mixedCase
GaugeV2.sol:77:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
GaugeV2.sol:77:88: Error: Variable name must be in mixedCase
GaugeV2.sol:77:113: Error: Variable name must be in mixedCase
GaugeV2.sol:235:50: Error: Avoid to make time-based decisions in your
business logic
GaugeV2.sol:247:26: Error: Avoid to make time-based decisions in your
business logic
GaugeV2.sol:248:24: Error: Avoid to make time-based decisions in your
business logic
```

MinterUpgradeable.sol

```
MinterUpgradeable.sol:2:1: Error: Compiler version 0.8.13 does not
satisfy the r semver requirement
MinterUpgradeable.sol:146:5: Error: Function name must be in
mixedCase
MinterUpgradeable.sol:159:5: Error: Function name must be in
mixedCase
MinterUpgradeable.sol:198:17: Error: Avoid to make time-based
decisions in your business logic
MinterUpgradeable.sol:202:16: Error: Avoid to make time-based
decisions in your business logic
```

Multicall.sol

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

Pair.sol

```
Pair.sol:325:22: Error: Parse error: missing ';' at '{'
```

PairFees.sol

```
PairFees.sol:2:1: Error: Compiler version 0.8.13 does not satisfy the
r semver requirement
PairFees.sol:4:8: Error: Use double quotes for string literals
PairFees.sol:13:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
PairFees.sol:21:45: Error: Avoid using low level calls.
```

RewardsDistributor.sol

```
RewardsDistributor.sol:2:1: Error: Compiler version 0.8.13 does not
satisfy the r semver requirement
RewardsDistributor.sol:4:8: Error: Use double quotes for string
literals
RewardsDistributor.sol:5:8: Error: Use double quotes for string
literals
RewardsDistributor.sol:6:8: Error: Use double quotes for string
literalsmixedCase
RewardsDistributor.sol:247:9: Error: Variable name must be in
mixedCase
RewardsDistributor.sol:253:9: Error: Variable name must be in
mixedCase
RewardsDistributor.sol:268:17: Error: Variable name must be in
mixedCase
RewardsDistributor.sol:327:21: Error: Variable name must be in
mixedCase
RewardsDistributor.sol:338:13: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
```

RouterV2.sol

```
RouterV2.sol:2:1: Error: Compiler version 0.8.13 does not satisfy the
r semver requirement
RouterV2.sol:82:46: Error: Use double quotes for string literals
RouterV2.sol:86:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
RouterV2.sol:137:37: Error: Use double quotes for string literals
RouterV2.sol:228:55: Error: Use double quotes for string literals
RouterV2.sol:233:55: Error: Use double quotes for string literals
```

SwapLibrary.sol

```
SwapLibrary.sol:3:1: Error: Compiler version 0.8.13 does not satisfy
the r semver requirement
SwapLibrary.sol:15:3: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
SwapLibrary.sol:28:3: Error: Function name must be in mixedCase
SwapLibrary.sol:30:7: Error: Variable name must be in mixedCase
SwapLibrary.sol:176:31: Error: Use double quotes for string literals
SwapLibrary.sol:178:35: Error: Use double quotes for string literals
```

VoterV2.sol

```
VoterV2.sol:2:1: Error: Compiler version 0.8.13 does not satisfy the
r semver requirementError: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
VoterV2.sol:67:19: Error: Code contains empty blocks
VoterV2.sol:69:105: Error: Visibility modifier must be first in list
of modifiers
VoterV2.sol:109:31: Error: Avoid to make time-based decisions in your
business logic
VoterV2.sol:196:31: Error: Avoid to make time-based decisions in your
business logic
VoterV2.sol:227:9: Error: Variable name must be in mixedCase
VoterV2.sol:230:9: Error: Variable name must be in mixedCase
VoterV2.sol:377:9: Error: Avoid using low level calls.
VoterV2.sol:431:13: Error: Variable name must be in mixedCase
VoterV2.sol:433:13: Error: Variable name must be in mixedCase
```

VotingEscrow.sol

```
VotingEscrow.sol:2:1: Error: Compiler version 0.8.13 does not satisfy
the r semver requirement
VotingEscrow.sol:17:1: Error: Contract has 26 states declarations but
allowed no more than 15
VotingEscrow.sol:1118:9: Error: Variable name must be in mixedCase
VotingEscrow.sol:1119:31: Error: Avoid to make time-based decisions
in your business logic
VotingEscrow.sol:1120:59: Error: Use double quotes for string
```

```
literals
VotingEscrow.sol:1298:27: Error: Avoid to make time-based decisions
in your business logic
```

Zard.sol

```
Zard.sol:40:18: Error: Parse error: missing ';' at '{'
Zard.sol:49:18: Error: Parse error: missing ';' at '{'
```

BribeFactoryV2.sol

```
BribeFactoryV2.sol:2:1: Error: Compiler version ^0.8.11 does not
satisfy the r semver requirement
BribeFactoryV2.sol:13:20: Error: Variable name must be in mixedCase
BribeFactoryV2.sol:16:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
BribeFactoryV2.sol:16:19: Error: Code contains empty blocks
BribeFactoryV2.sol:17:54: Error: Visibility modifier must be first in
list of modifiers
BribeFactoryV2.sol:23:63: Error: Use double quotes for string
literals
BribeFactoryV2.sol:31:23: Error: Variable name must be in mixedCase
BribeFactoryV2.sol:47:40: Error: Use double quotes for string
literals
```

GaugeFactoryV2.sol

```
GaugeFactoryV2.sol:2:1: Error: Compiler version 0.8.13 does not
satisfy the r semver requirement
GaugeFactoryV2.sol:5:8: Error: Use double quotes for string literals
GaugeFactoryV2.sol:10:20: Error: Variable name must be in mixedCase
GaugeFactoryV2.sol:12:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
GaugeFactoryV2.sol:12:19: Error: Code contains empty blocks
GaugeFactoryV2.sol:14:40: Error: Visibility modifier must be first in
list of modifiers
GaugeFactoryV2.sol:18:99: Error: Variable name must be in mixedCase
GaugeFactoryV2.sol:18:124: Error: Variable name must be in mixedCase
```

PairFactoryUpgradeable.sol

```
PairFactoryUpgradeable.sol:2:1: Error: Compiler version 0.8.13 does
not satisfy the r semver requirementPairFactoryUpgradeable.sol:5:8:
Error: Use double quotes for string literals
PairFactoryUpgradeable.sol:39:5: Error: Explicitly mark visibility in
```

```
function (Set ignoreConstructors to true if using solidity >=0.7.0)
PairFactoryUpgradeable.sol:39:19: Error: Code contains empty blocks
PairFactoryUpgradeable.sol:40:40: Error: Visibility modifier must be
first in list of modifiers
PairFactoryUpgradeable.sol:112:35: Error: Use double quotes for
string literals
PairFactoryUpgradeable.sol:114:39: Error: Use double quotes for
string literals
PairFactoryUpgradeable.sol:115:64: Error: Use double quotes for
string literals
```

AdminUpgradeabilityProxy.sol

```
import.sol:2:1: Error: Compiler version ^0.8.0 does not satisfy the r
semver requirement
import.sol:8:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
import.sol:8:122: Error: Code contains empty blocks
```

VeArtProxyUpgradeable.sol

```
VeArtProxyUpgradeable.sol:2:1: Error: Compiler version 0.8.13 does
not satisfy the r semver requirement
VeArtProxyUpgradeable.sol:12:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
VeArtProxyUpgradeable.sol:12:19: Error: Code contains empty blocks
VeArtProxyUpgradeable.sol:14:39: Error: Visibility modifier must be
first in list of modifiers
VeArtProxyUpgradeable.sol:48:275: Error: Use double quotes for string
literals
VeArtProxyUpgradeable.sol:49:42: Error: Use double quotes for string
literals
```

FaucetToken.sol

```
FaucetToken.sol:3:1: Error: Compiler version 0.8.13 does not satisfy
the r semver requirement
FaucetToken.sol:8:5: Error: Explicitly mark visibility of state
FaucetToken.sol:10:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

Software analysis result:

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

Email: audit@EtherAuthority.io