Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Spoon Exchange
Website:	https://spoon.exchange
Platform:	Core Chain
Language:	Solidity
Date:	March 15th, 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 the Spoon Exchange team to perform the Security audit of the Spoon 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 March 15th, 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

- Spoon is a decentralized exchange platform derived from Andre Cronje's initial concept of a perpetual decentralized exchange.
- Spoon is built on Core Chain that offers low-cost token exchanges and reduced swap fees, using a governance model called the ve(3,3) system and with many improvements.

Audit scope

Name	**Code Review and Security Analysis Report for Spoon Exchange Smart Contracts**
Platform	**Core Chain / Solidity**
File 1	Bribes.sol
File 1 MD5 Hash	169EAAA52FA9568549466FEE15D56DAF
File 2	GaugeV2.sol
File 2 MD5 Hash	7A3F2E2A748573CDFB8654A714DCCCF0
File 3	import.sol
File 3 MD5 Hash	9DB89ED56B653E26510B7013EFFE47B0
File 4	MinterUpgradeable.sol
File 4 MD5 Hash	CC72DA59047D4EDFC18F63280583D9AD

File 5	Pair.sol
File 5 MD5 Hash	A0A52C205D83C869CB6E2F37177C8FCA
File 6	PairFees.sol
File 6 MD5 Hash	6DC3657D376FA99476F3DA48D1537310
File 7	RewardsDistributor.sol
File 7 MD5 Hash	9DE86D1D49D818493DE6850893712ED2
File 8	Router.sol
File 8 MD5 Hash	BA37485099CDE5B7CB4156D4E917C468
File 9	RouterV2.sol
File 9 MD5 Hash	AF5AFC6498B484220F717769E3505EAB
File 10	Spoon.sol
File 10 MD5 Hash	BDA0FDF1411C41BFAA5E138F8994503E
File 11	VeArtProxyUpgradeable.sol
File 11 MD5 Hash	FB4D2E453E58F8266B3295BDB1DE09F5
File 12	VoterV2_1.sol
File 12 MD5 Hash	48DB1D3D8035DA9F46A657243830FAD0
File 13	VotingEscrow.sol
File 13 MD5 Hash	E5CC7D7617C2AC7C80FEA6F69E6D66A3
File 14	BribeFactoryV2.sol
File 14 MD5 Hash	11AE5E800B94E9650FD617985B91BAC9
File 15	GaugeFactoryV2.sol
File 15 MD5 Hash	10ef53c0d003b7cd9b16e94e981edd80
File 16	PairFactory.sol
File 16 MD5 Hash	988D0EE6054E78F7D8C37B85207BA4E2
File 17	PairFactoryUpgradeable.sol
File 17 MD5 Hash	5119B35B5C61F5EFF047D36D72F91B4C
Audit Date	March 15th,2023

Claimed Smart Contract Features

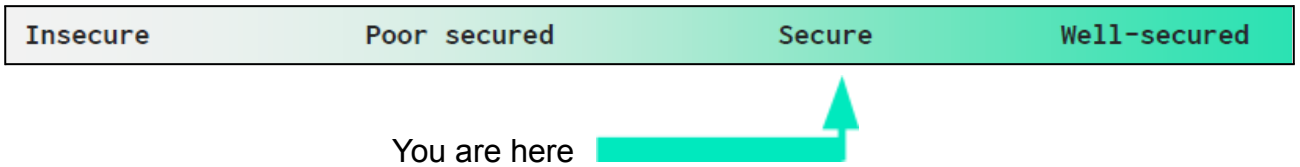
Claimed Feature Detail	Our Observation
File 1 Bribes.sol • Rewards are released over 7 days **Ownership Control:** • Owner can recover the ERC20 token address with the amount. • Voter address can be set by the Owner. • Reward address can be set by the Owner. • Minter address can be set by the Owner. • Reward token addresses can be added by the Owner. • A new owner address can be set by the current Owner.	**YES, This is valid.**
File 2 GaugeV2.sol **Ownership Control:** • Distribution address can be set by the Owner. • Gauge rewarder address can be set by the Owner. • Extra rewarder pid can be set by the Owner.	**YES, This is valid.**
File 3 import.sol • Import contract can inherit the TransparentUpgradeableProxy contract.	**YES, This is valid.**
File 4 MinterUpgradeable.sol • 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% • Allows minting once per week. **Ownership Control:** • Emission rate can be set by the Owner.	**YES, This is valid.**

• Team rate value can be set by the Owner.	
File 5 Pair.sol • Pools that are either stable or volatile, as the base pair. • Decimals: 18	**YES, This is valid.**
File 6 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. **Other Specifications:** • claimFeesFor us allow the pair to transfer fees to users.	**YES, This is valid.**
File 7 RewardsDistributor.sol • The Depositor can be set by the Owner. • A new owner address can be set by the current Owner. • Owner can withdraw ERC20 tokens from the contract.	**YES, This is valid.**
File 8 Router.sol • Minimum Liquidity: 1000	**YES, This is valid.**
File 9 RouterV2.sol • RouterV2 : Support for Fee-on-Transfer Tokens. • Only accept ETH via fallback from the WETH contract.	**YES, This is valid.**
File 10 Spoon.sol • Name: Spoon Token • Symbol: POON • Decimals: 18	**YES, This is valid.**
File 11 VeArtProxyUpgradeable.sol • VeArtProxyUpgradeable contract can inherit OwnableUpgradeable contract.	**YES, This is valid.**
File 12 VoterV2_1.sol	**YES, This is valid.**

• Rewards are released over 7 days	
File 13 VotingEscrow.sol • Name: veSpoon • Symbol: vePOON • Decimals: 18 • version: 1.0.0	**YES, This is valid.**
File 14 BribeFactoryV2.sol **Ownership Control:** • 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 **Ownership Control:** • Distribution address can be set by Owner.	**YES, This is valid.**
File 16 PairFactory.sol • Maximum Referral Fee: 12% • Maximum Fee: 0.25% • Stable Fee: 0.04% • Volatile Fee: 0.18%	**YES, This is valid.**
File 17 PairFactoryUpgradeable • Maximum Referral Fee: 12% • Maximum Fee: 0.25% • Stable Fee: 0.04% • Volatile Fee: 0.18%	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. 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, 1 medium and 2 low and some very low level issues.

Medium severity issue has been resolved in the revised code.

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

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	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	Moderated
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 17 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the Spoon Exchange 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 Spoon Exchange.

The Spoon 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 Spoon Exchange 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://spoon.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	Owner drain all tokens	Refer to audit findings
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	permit	external	Passed	No Issue
34	transfer	external	Passed	No Issue
35	transferFrom	external	Passed	No Issue
36	_transferTokens	internal	Passed	No Issue
37	_safeTransfer	internal	Passed	No Issue
38	_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

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

RouterV2.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	read	Passed	No Issue
9	getAmountsOut	read	Passed	No Issue
10	isPair	external	Passed	No Issue
11	quoteAddLiquidity	external	Passed	No Issue
12	quoteRemoveLiquidity	external	Passed	No Issue
13	_addLiquidity	internal	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	internal	Passed	No Issue
19	removeLiquidityETHWithPermit	internal	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	UNSAFE_swapExactTokensForTokens	external	Passed	No Issue
26	_safeTransferETH	internal	Passed	No Issue
27	_safeTransfer	internal	Passed	No Issue
28	_safeTransferFrom	internal	Passed	No Issue
29	removeLiquidityETHSupportingFeeOnTransferTokens	write	Passed	No Issue
30	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	external	Passed	No Issue
31	_swapSupportingFeeOnTransferTokens	internal	Passed	No Issue
32	swapExactTokensForTokensSupportingFeeOnTransferTokens	external	Passed	No Issue
33	swapExactETHForTokensSupportingFeeOnTransferTokens	external	Passed	No Issue

34	swapExactTokensForETHSupportingFeeOnTransferTokens	external	Passed	No Issue

Spoon.sol

Functions

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

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
70	delegateBySig	write	Passed	No Issue

VoterV2_1.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	Missing Error Message, Infinite loop	Refer to audit findings
18	setMinter	external	Missing Error Message	Refer to audit findings

19	setGovernor	write	Missing Error Message	Refer to audit findings
20	setEmergencyCouncil	write	Missing Error Message	Refer to audit findings
21	reset	external	Missing Error Message	Refer to audit findings
22	_reset	internal	Infinite loop	Refer to audit findings
23	poke	external	Missing Error Message, Infinite loop	Refer to audit findings
24	_vote	internal	Infinite loop	Refer to audit findings
25	vote	external	Missing Error Message	Refer to audit findings
26	whitelist	write	Missing Error Message	Refer to audit findings
27	_whitelist	internal	Missing Error Message	Refer to audit findings
28	createGauge	external	Ambiguous Error Message	Refer to audit findings
29	killGauge	external	Passed	No Issue
30	reviveGauge	external	Passed	No Issue
31	attachTokenToGauge	external	Missing Error Message	Refer to audit findings
32	emitDeposit	external	Missing Error Message, Unused functions	Refer to audit findings
33	detachTokenFromGauge	external	Missing Error Message	Refer to audit findings
34	emitWithdraw	external	Missing Error Message, Unused functions	Refer to audit findings
35	length	external	Passed	No Issue
36	poolVoteLength	external	Passed	No Issue
37	notifyRewardAmount	external	Passed	No Issue
38	updateFor	external	Passed	No Issue
39	updateForRange	write	Infinite loop	Refer to audit findings
40	updateAll	external	Passed	No Issue
41	updateGauge	external	Passed	No Issue
42	_updateFor	internal	Passed	No Issue
43	claimRewards	external	Removed	-
44	claimBribes	external	Missing Error Message, Infinite loop	Refer to audit findings
45	claimFees	external	Infinite loop	Refer to audit findings
46	distributeFees	external	Infinite loop	Refer to audit findings
47	distribute	write	Passed	No Issue

48	distributeAll	external	Passed	No Issue
49	distribute	write	Passed	No Issue
50	distribute	write	Passed	No Issue
51	_safeTransferFrom	internal	Missing Error Message	Refer to audit findings
52	setBribeFactory	external	Passed	No Issue
53	setGaugeFactory	external	Missing Error Message	Refer to audit findings
54	setPairFactory	external	Missing Error Message	Refer to audit findings
55	killGaugeTotally	external	Passed	No Issue
56	whitelist	write	Passed	No Issue
57	initGauges	write	Missing Error Message, Anyone can initGauges, Infinite loop	Refer to audit findings
58	increaseGaugeApprovals	external	Missing Error Message	Refer to audit findings
59	setNewBribe	external	Missing Error Message	Refer to audit findings

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	_removeTokenFromOwnerLi sti	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	write	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
70	delegateBySig	write	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

PairFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	allPairsLength	external	Passed	No Issue

3	pairs	external	Passed	No Issue
4	setPauser	external	Passed	No Issue
5	acceptPauser	external	Passed	No Issue
6	setPause	external	Passed	No Issue
7	setFeeManager	external	Passed	No Issue
8	acceptFeeManager	external	Passed	No Issue
9	setDibs	external	Passed	No Issue
10	setReferralFee	external	Passed	No Issue
11	setFee	external	Passed	No Issue
12	getFee	read	Passed	No Issue
13	pairCodeHash	external	Passed	No Issue
14	getInitializable	external	Passed	No Issue
15	createPair	external	Passed	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	setReferralFee	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

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

(1) ClaimRewards function is not working: **VoterV2_1.sol**

```
function claimRewards(address[] memory _gauges, address[][] memory _tokens) external {
    for (uint i = 0; i < _gauges.length; i++) {
        IGauge(_gauges[i]).getReward(msg.sender, _tokens[i]);
    }
}
```

The claimRewards function calls Gauge's getReward function with arguments. But Gauge's getReward function doesn't not have any parameters.

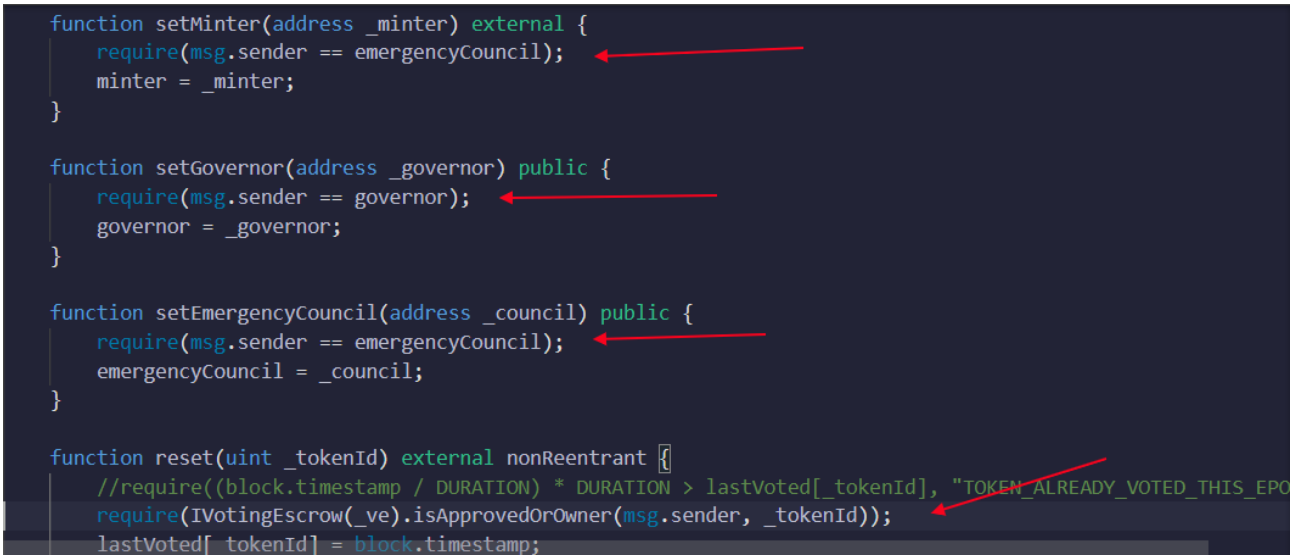
Resolution: We suggest removing parameters from claimRewards's getReward.
Status: **Fixed. This function has been removed in the revised code.**

Low

(1) Owner can drain all tokens: **Bribes.sol, RewardsDistributor.sol**
The owner can drain all tokens. This would create trust issues in the users.

Resolution: If this is a desired feature, then please disregard this issue.

(2) Missing Error Message: **VoterV2_1.sol**



```
function setMinter(address _minter) external {
    require(msg.sender == emergencyCouncil);
    minter = _minter;
}

function setGovernor(address _governor) public {
    require(msg.sender == governor);
    governor = _governor;
}

function setEmergencyCouncil(address _council) public {
    require(msg.sender == emergencyCouncil);
    emergencyCouncil = _council;
}

function reset(uint _tokenId) external nonReentrant {
    //require((block.timestamp / DURATION) * DURATION > lastVoted[_tokenId], "TOKEN_ALREADY_VOTED_THIS_EPO
    require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
    lastVoted[_tokenId] = block.timestamp;
```

A require is without error messages in these functions:

- reset
- setMinter
- _initialize
- setEmergencyCouncil
- setGovernor
- poke
- vote
- whitelist
- _whitelist
- attachTokenToGauge
- emitDeposit
- detachTokenFromGauge
- emitWithdraw
- claimBribes
- _safeTransferFrom
- setGaugeFactory
- setPairFactory
- initGauges
- increaseGaugeApprovals
- setNewBribe

Resolution: We advise writing appropriate error messages.

Very Low / Informational / Best practices:

(1) Anyone can initialize contract: **VoterV2_1.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 to always make sure that contract should be initialized by owner

(2) Anyone can initGauges : **VoterV2_1.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_1.sol**

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

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

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

(4) Unused functions, variables:

Unused variables: **GaugeV2.sol**

_VE , external_bribe are public variables which are not used anywhere in the contract.

Unused functions: VoterV2_1.sol

```
function emitDeposit(uint tokenId, address account, uint amount) external override {
    require(isGauge[msg.sender]);
    require(isAlive[msg.sender]);
    emit Deposit(account, msg.sender, tokenId, amount);
}

function detachTokenFromGauge(uint tokenId, address account) external override {
    require(isGauge[msg.sender]);
    if (tokenId > 0) IVotingEscrow(_ve).detach(tokenId);
    emit Detach(account, msg.sender, tokenId);
}

function emitWithdraw(uint tokenId, address account, uint amount) external override {
    require(isGauge[msg.sender]);
    emit Withdraw(account, msg.sender, tokenId, amount);
}
```

The emitDeposit , emitWithdraw functions only require and emit statements. No code logic is written.

Resolution: We suggest removing these unused functions and variables.

(5) Ambiguous Error Message: VoterV2_1.sol

```
function createGauge(address _pool) external returns (address) {
    require(gauges[_pool] == address(0x0), "exists");
    address[] memory allowedRewards = new address[](3);
    address[] memory internalRewards = new address[](2);
    bool isPair = IPairFactory(factory).isPair(_pool);
    address tokenA;
    address tokenB;

    if (isPair) {
        (tokenA, tokenB) = IPair(_pool).tokens();
        allowedRewards[0] = tokenA;
        allowedRewards[1] = tokenB;
        internalRewards[0] = tokenA;
        internalRewards[1] = tokenB;

        if (base != tokenA && base != tokenB) {
          allowedRewards[2] = base;
        }
    }

    if (msg.sender != governor) { // gov can create for any pool, even non-Spoon pairs
        require(isPair, "!_pool");
        require(isWhitelisted[tokenA] && isWhitelisted[tokenB], "!whitelisted");
    }
```

The mentioned error message does not explain exactly the error of the operation.

Resolution: As error messages are intended to notify users about failing conditions, they should provide enough information so that appropriate corrections can be made to interact with the system.

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:

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.
- _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.

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.
- _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.

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.

Spoon.sol

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

VoterV2_1.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 whitelist.
- initGauges: Owner can initialize gauges addresses.
- increaseGaugeApprovals: Owners can increase gauge approval addresses.
- setNewBribe: Owners can set new bribe addresses.
- _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.

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.

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.
- _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.

GaugeFactoryV2.sol

- setDistribution: Distribution address can be set by Owner.
- _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.

PairFactory.sol

- setPauser: Pauser address can be set by Owner.
- acceptPauser: Owner can accept Pauser address.

- setPause:Owner can set pause state.
- setFeeManager: Manager Owner can set a Fee Manager address.
- acceptFeeManager: Manager Owner can accept fee manager.
- setDibs: Manager Owner can set dibs address.
- setReferralFee: Manager Owner can set referral fee.
- setFee: Manager Owner can set a fee.

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.
- setReferralFee: Manager Owner can set referral fee.
- setFee: Manager Owner can set a fee.
- _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.

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.

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 have observed 1 medium severity issue, 2 low severity issues and some informational severity issues in the token smart contract. Medium severity issue has been resolved in the revised code and the rest are not critical issues. So, **it's good to go to production**.

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

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

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

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

Email: audit@EtherAuthority.io

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.

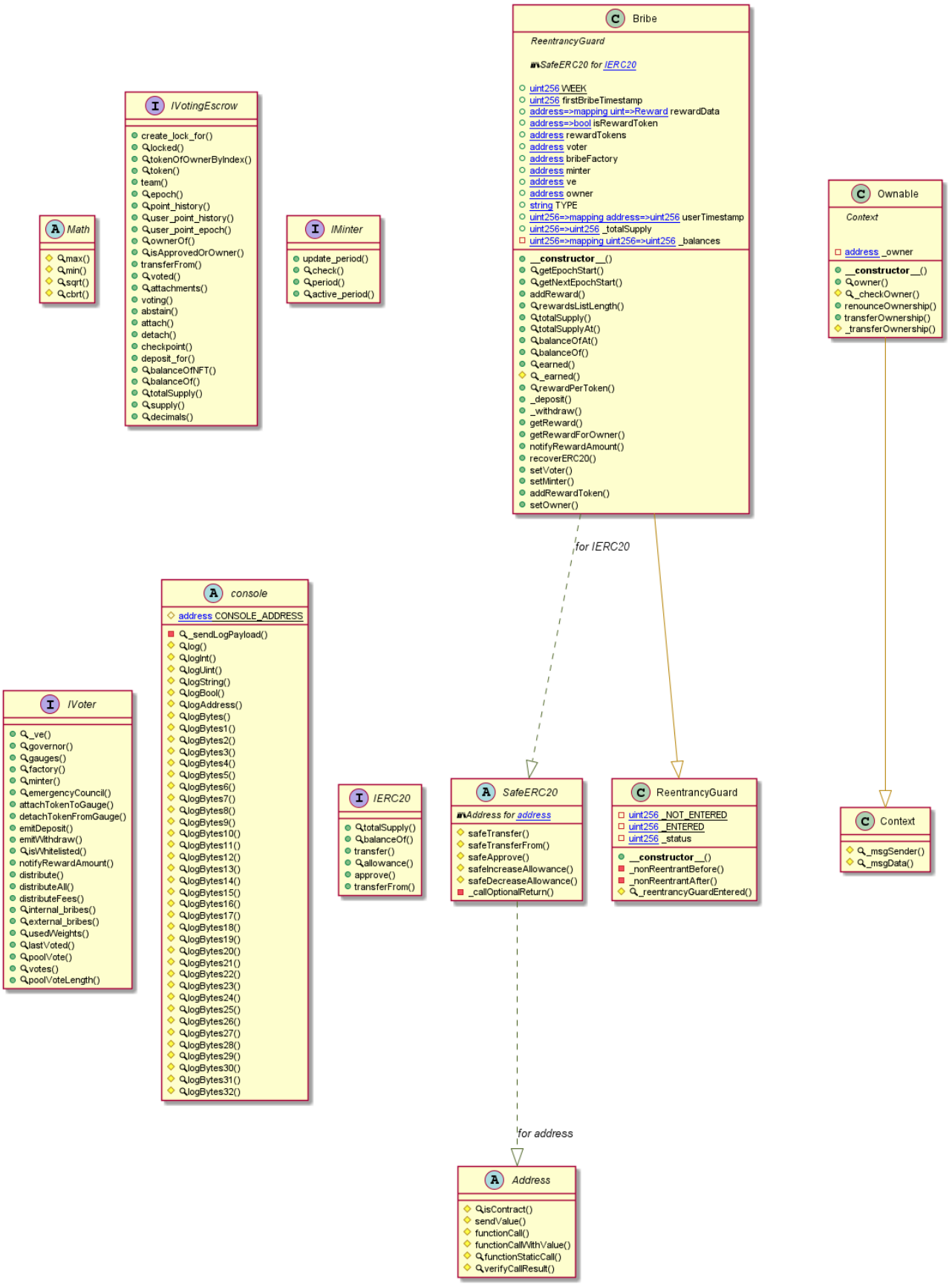
Technical Disclaimer

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.

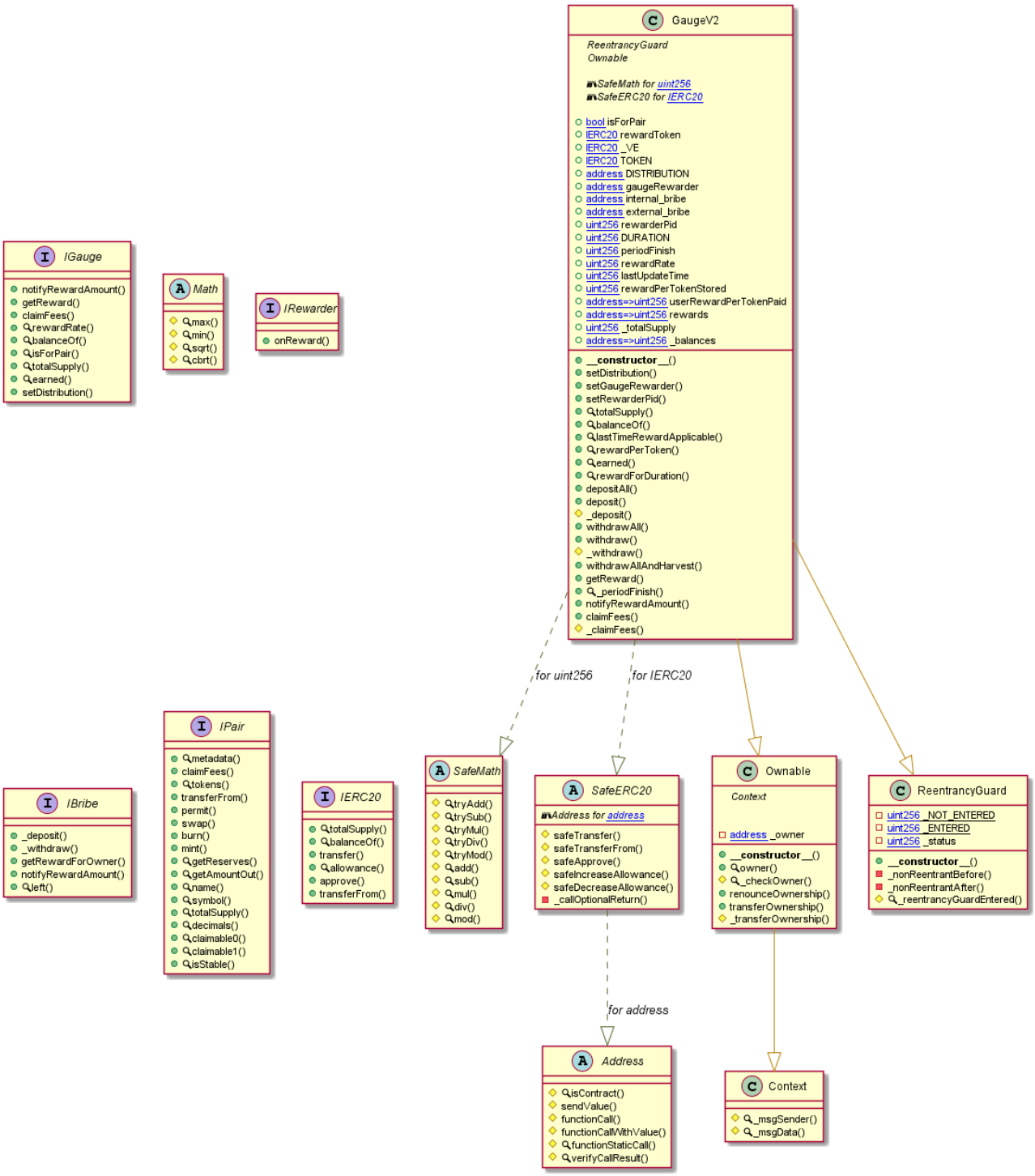
Appendix

Code Flow Diagram - Spoon Exchange

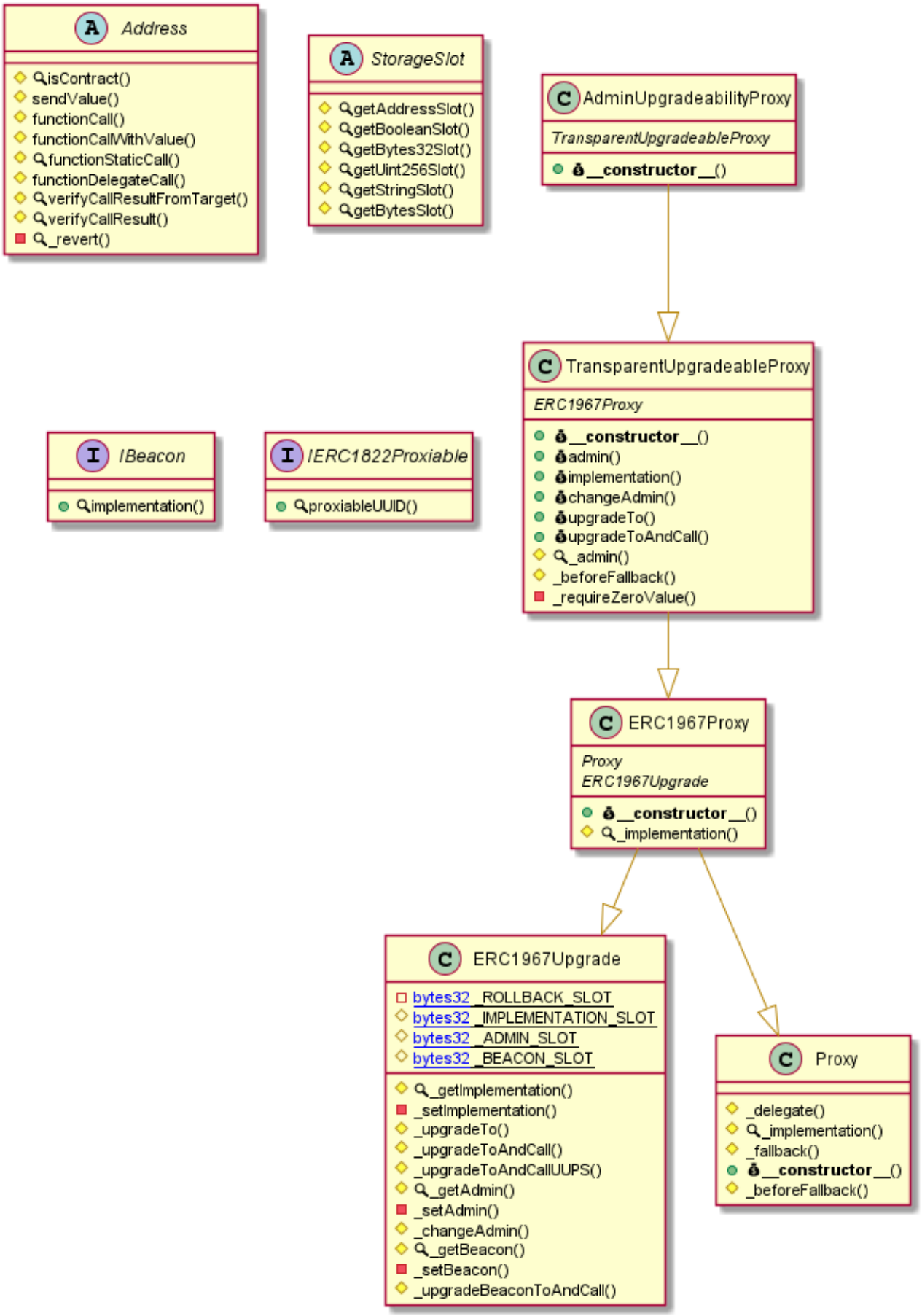
Bribes Diagram



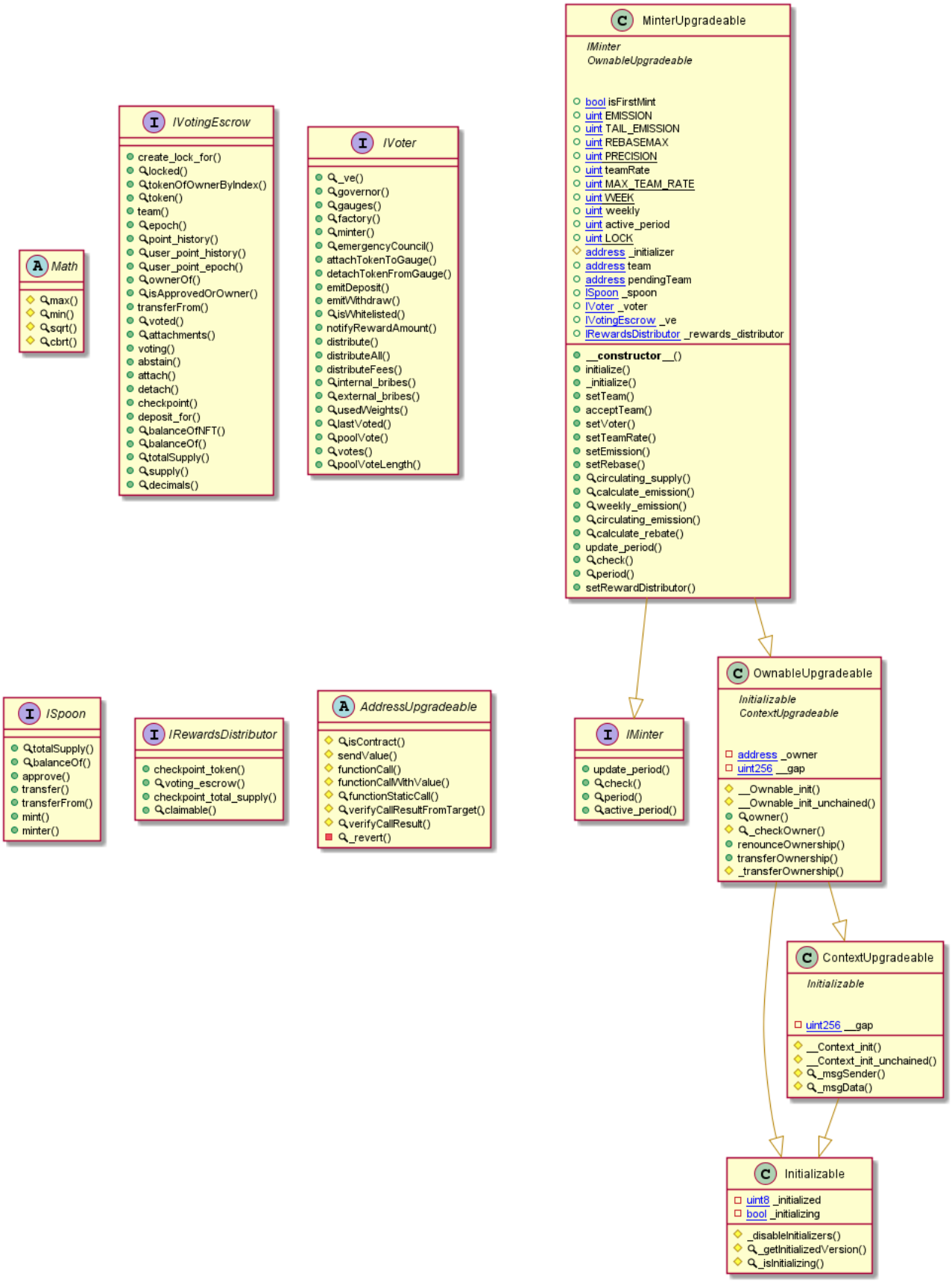
GaugeV2 Diagram



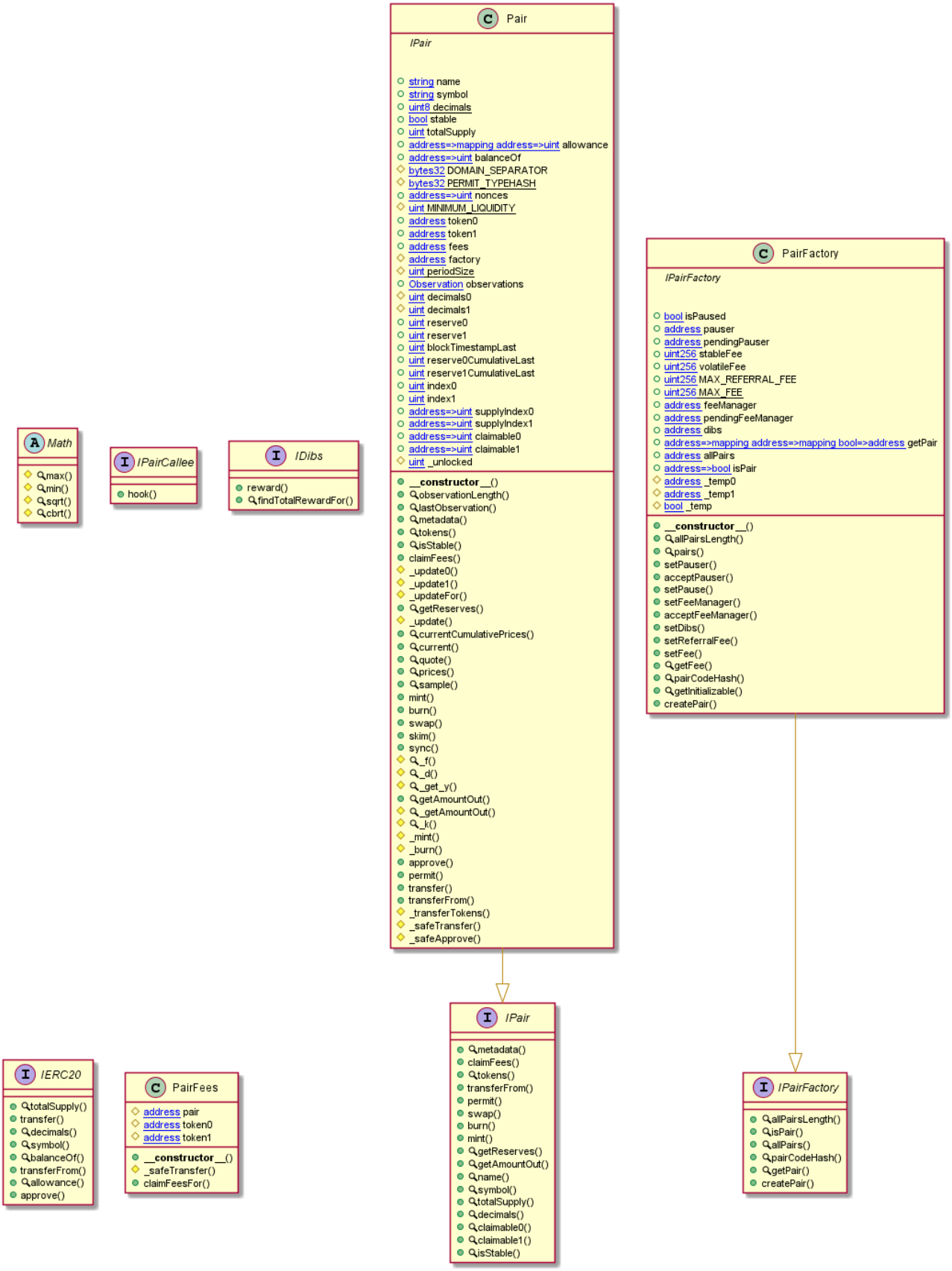
import Diagram



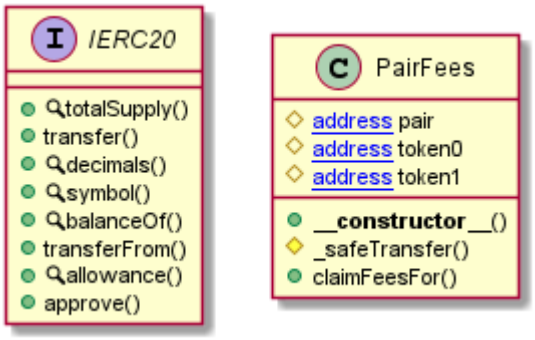
MinterUpgradeable Diagram



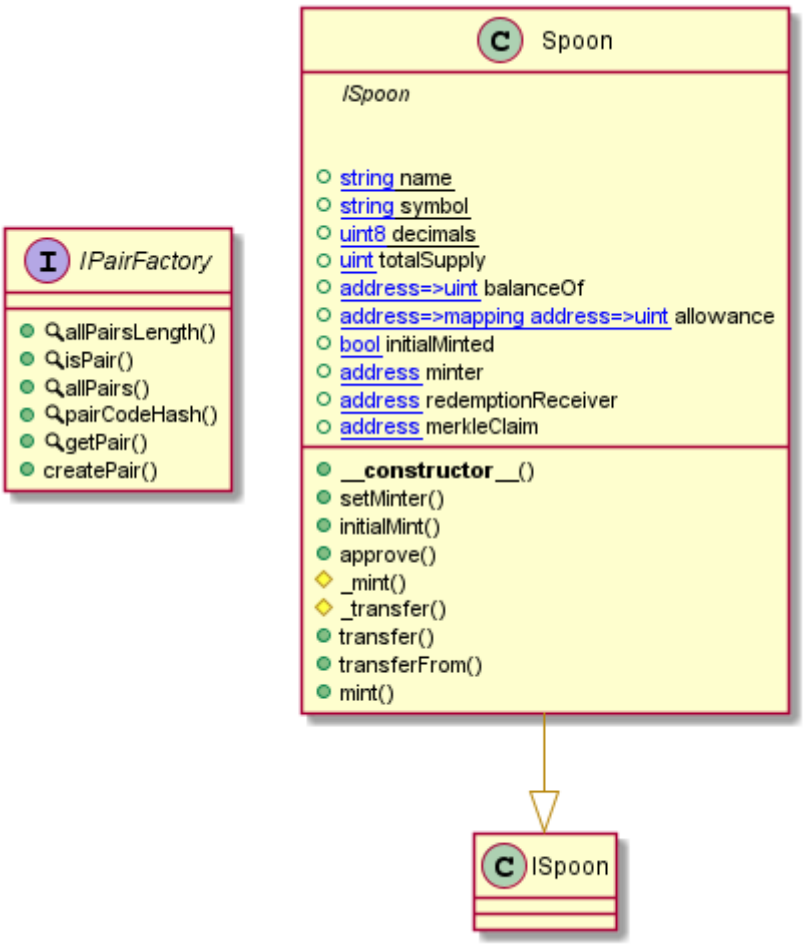
Pair Diagram



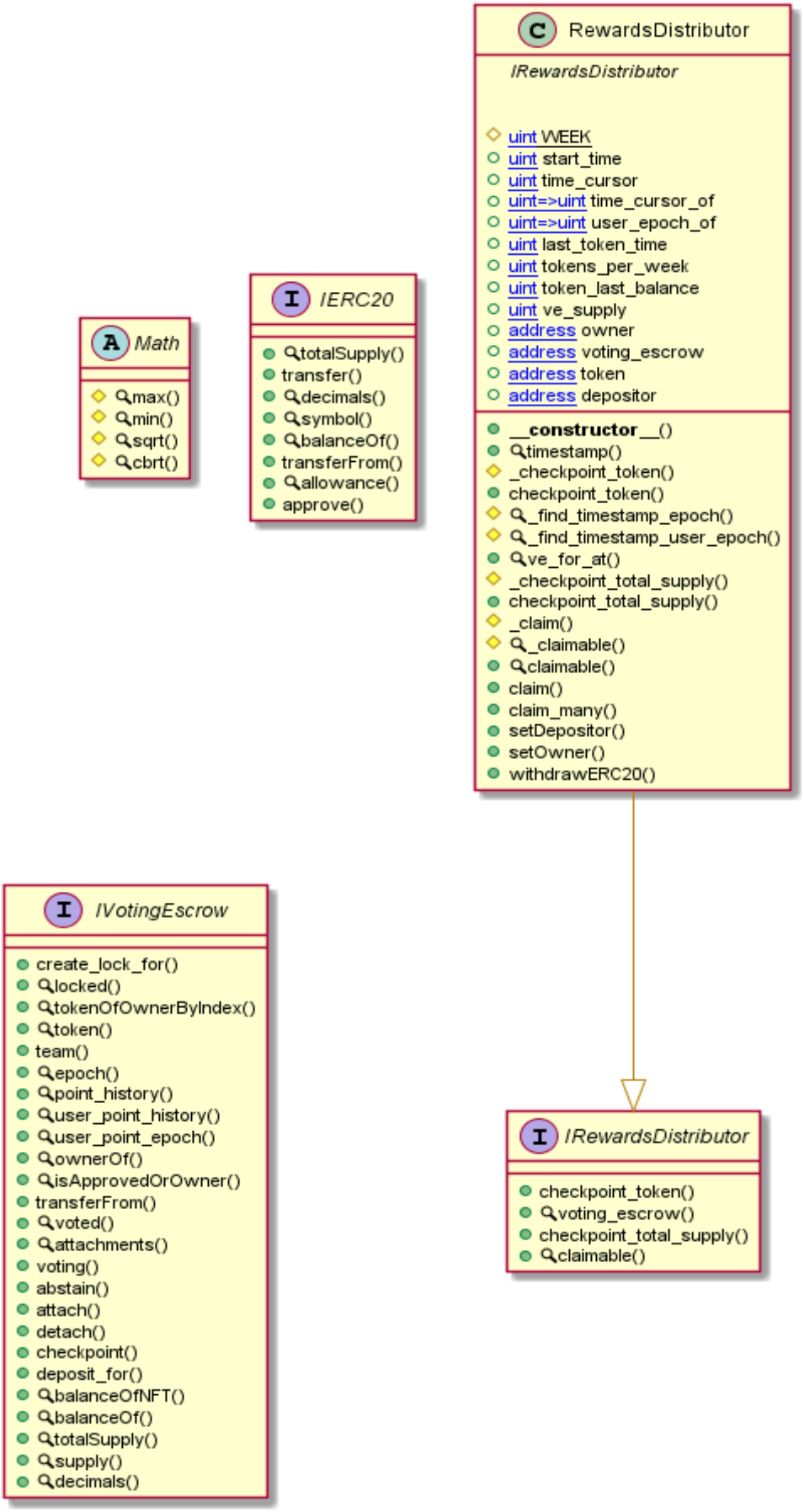
PairFees Diagram



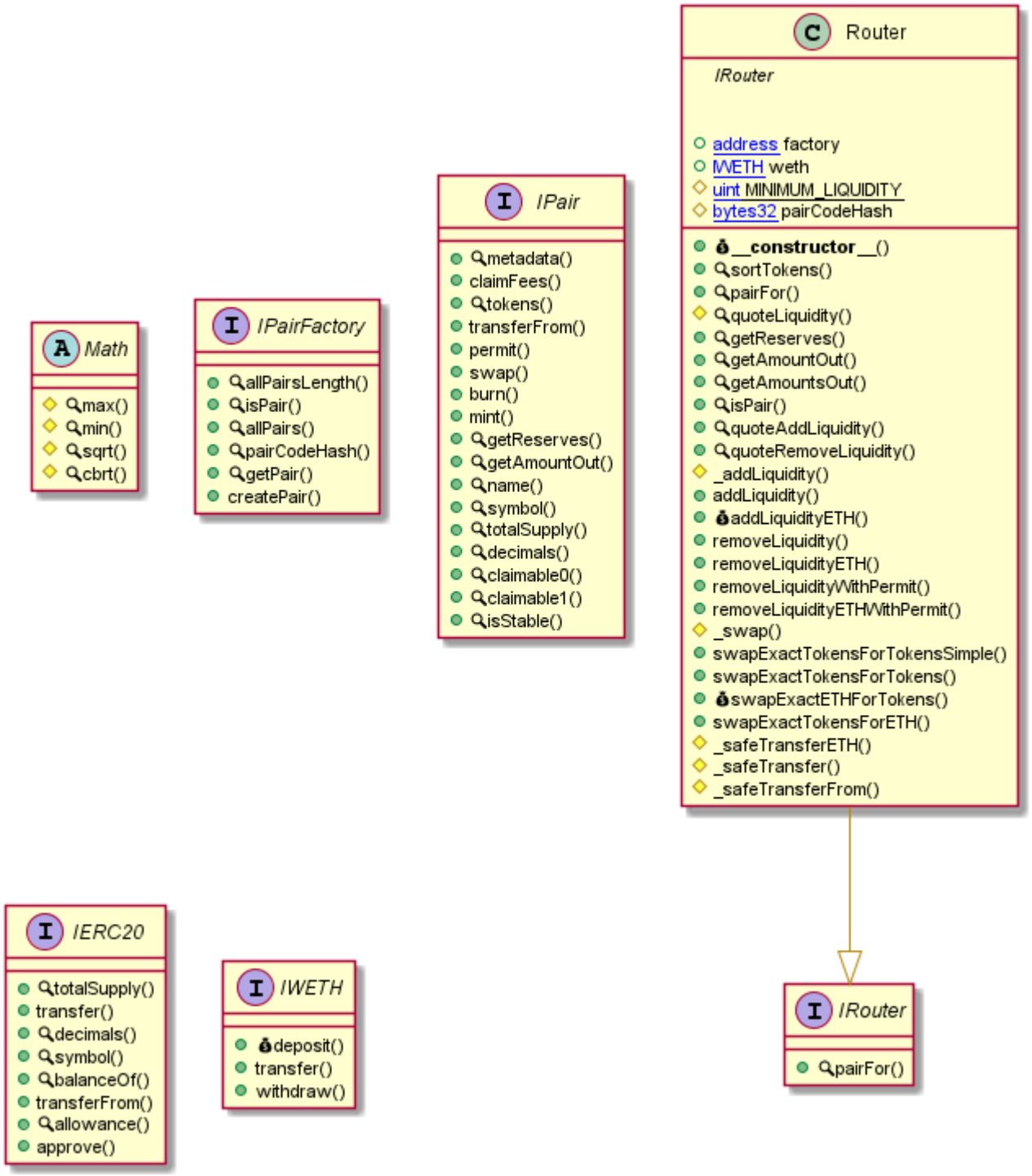
Spoon Diagram



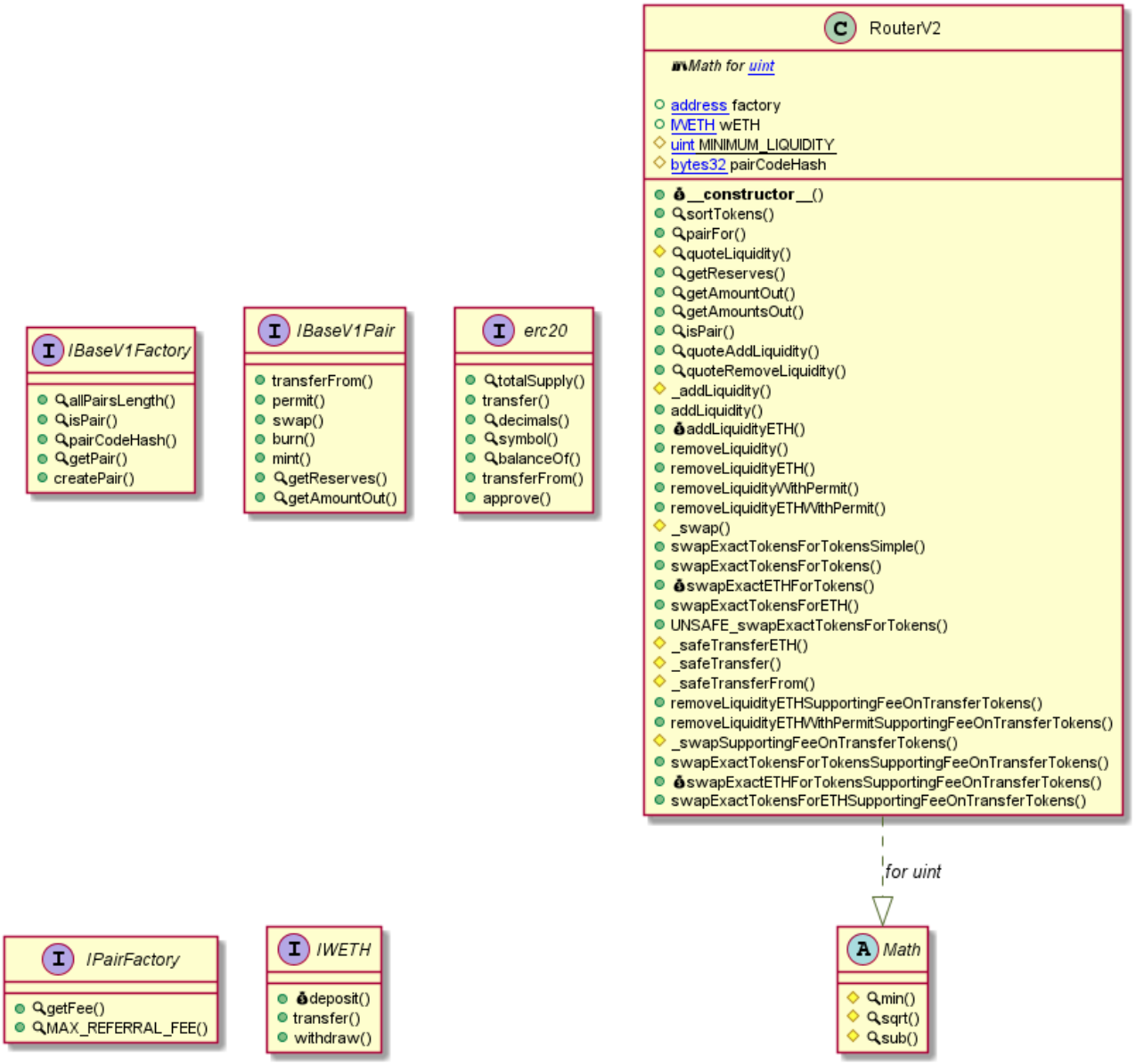
RewardsDistributor Diagram



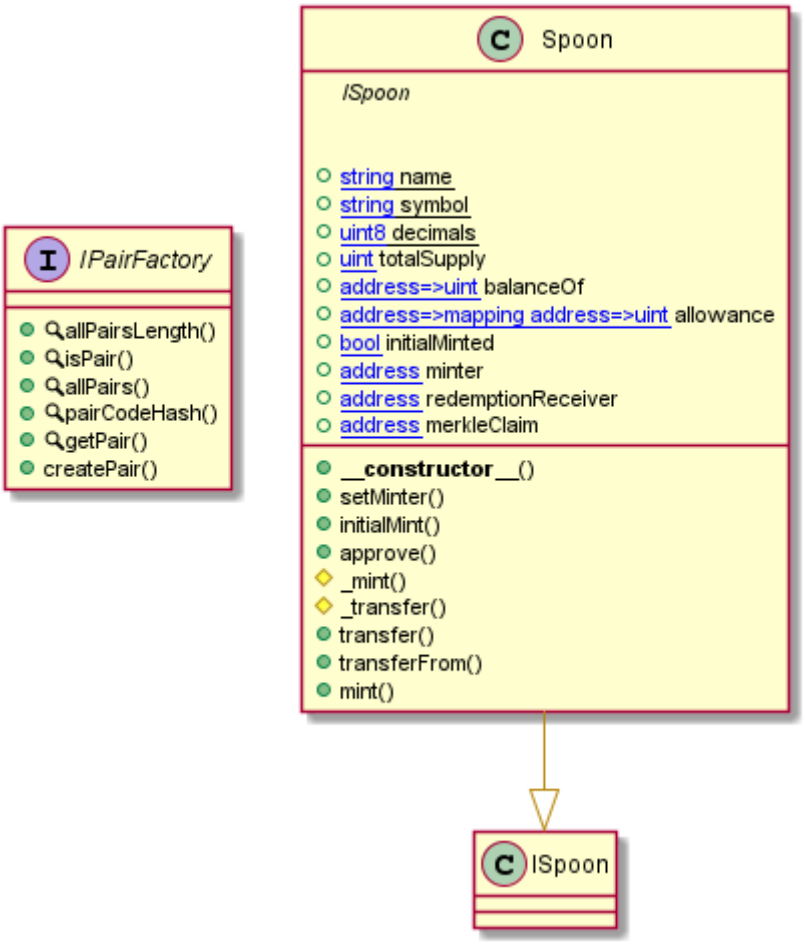
Router Diagram



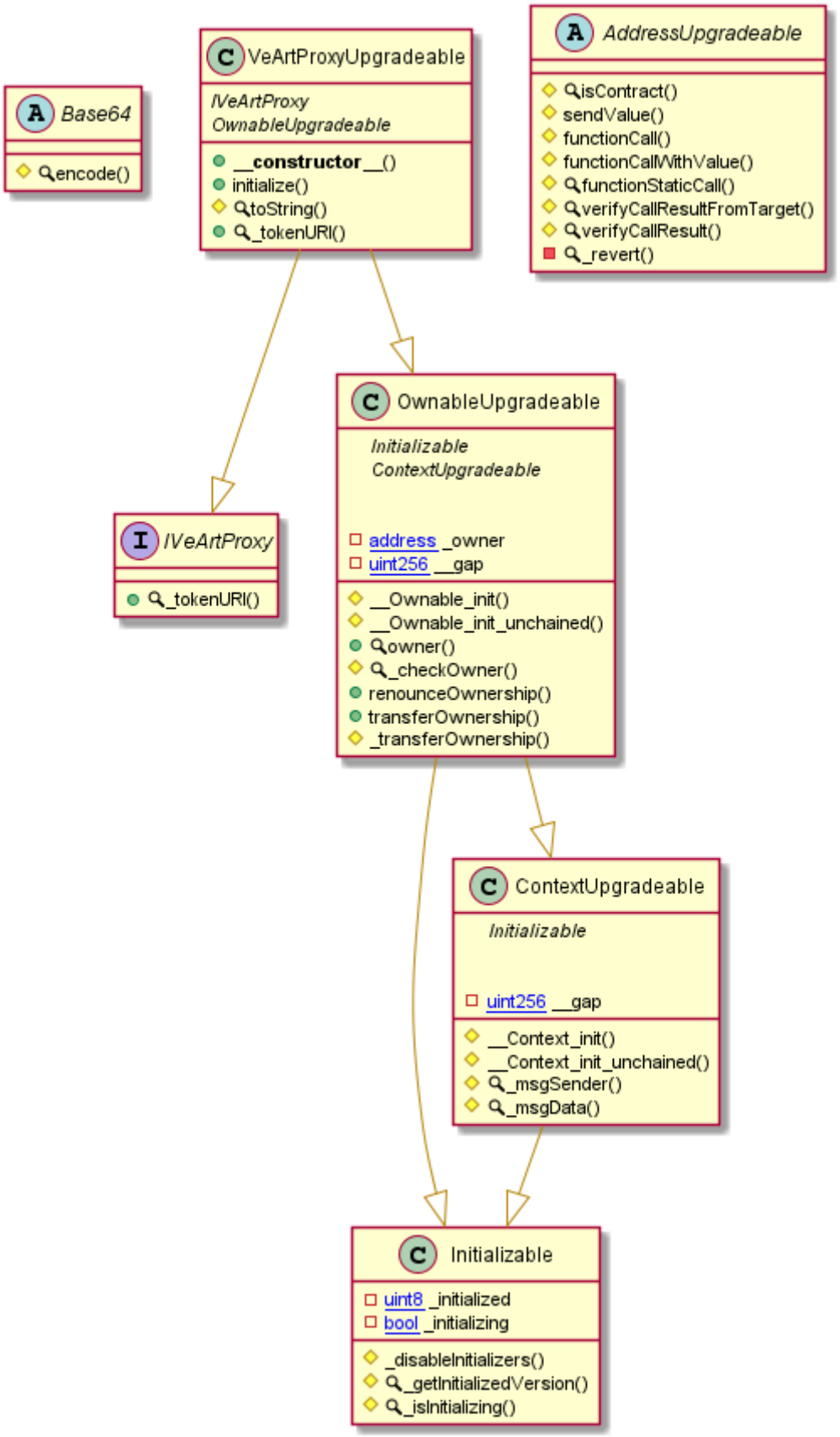
RouterV2 Diagram



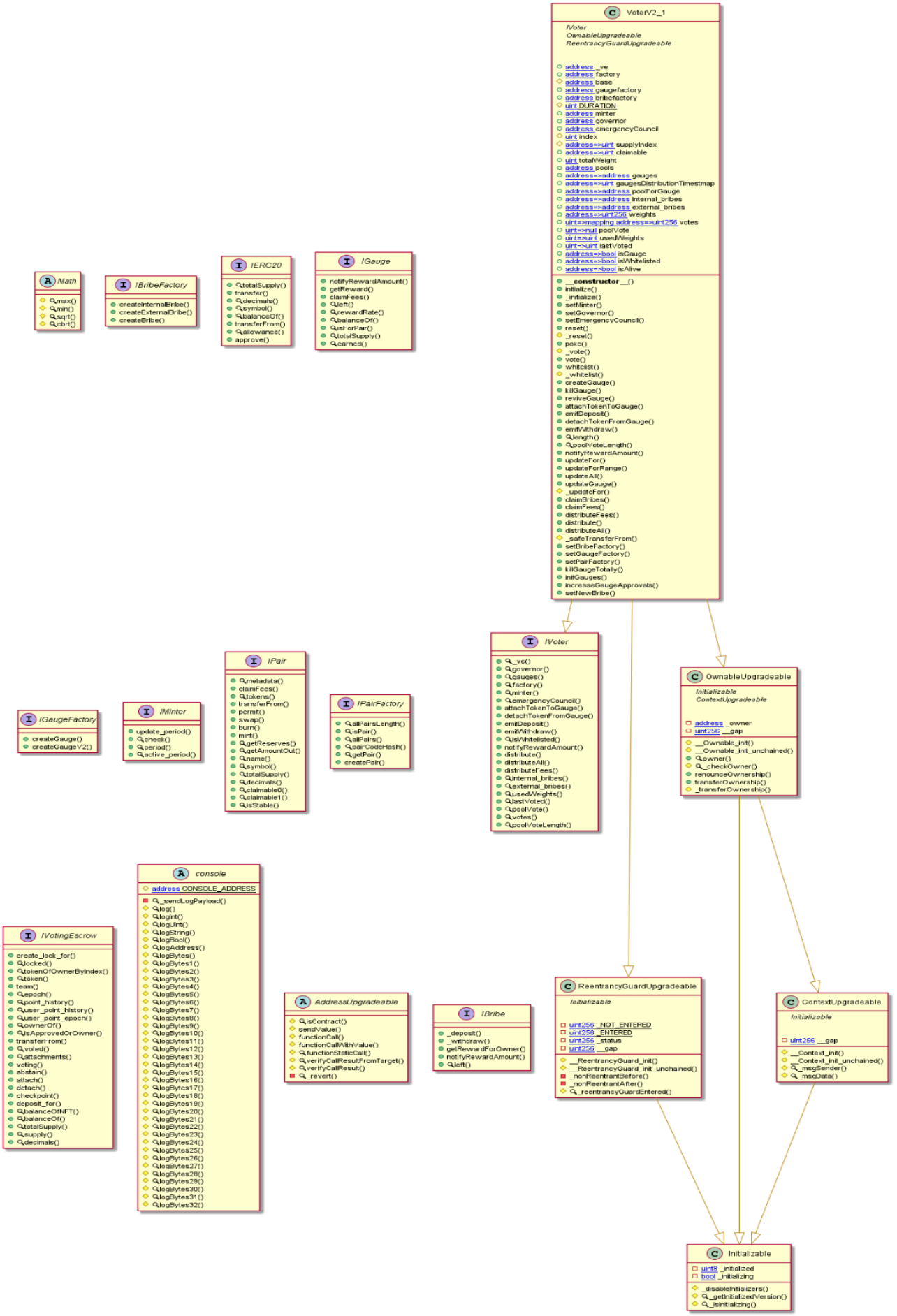
Spoon Diagram



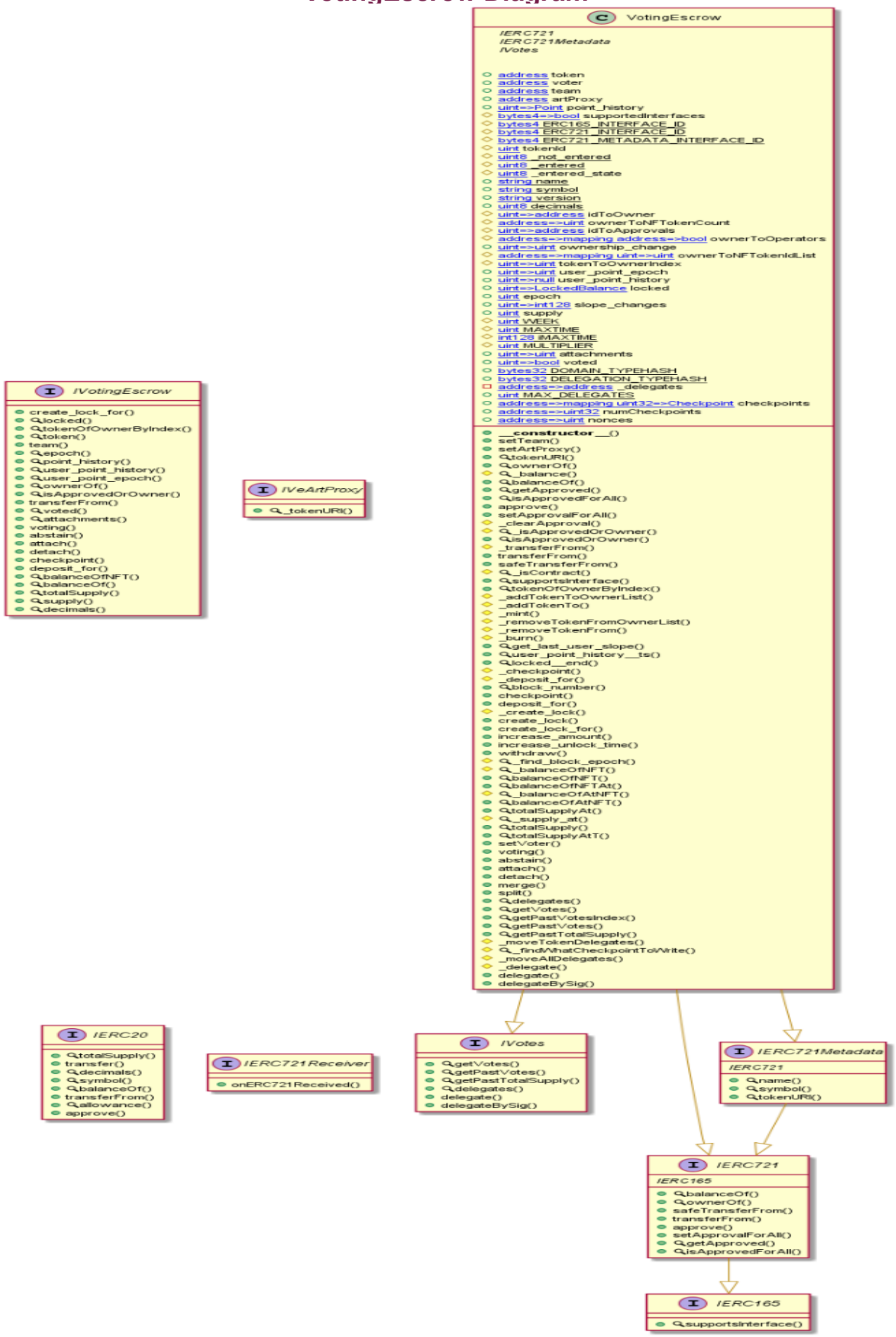
VeArtProxyUpgradeable Diagram



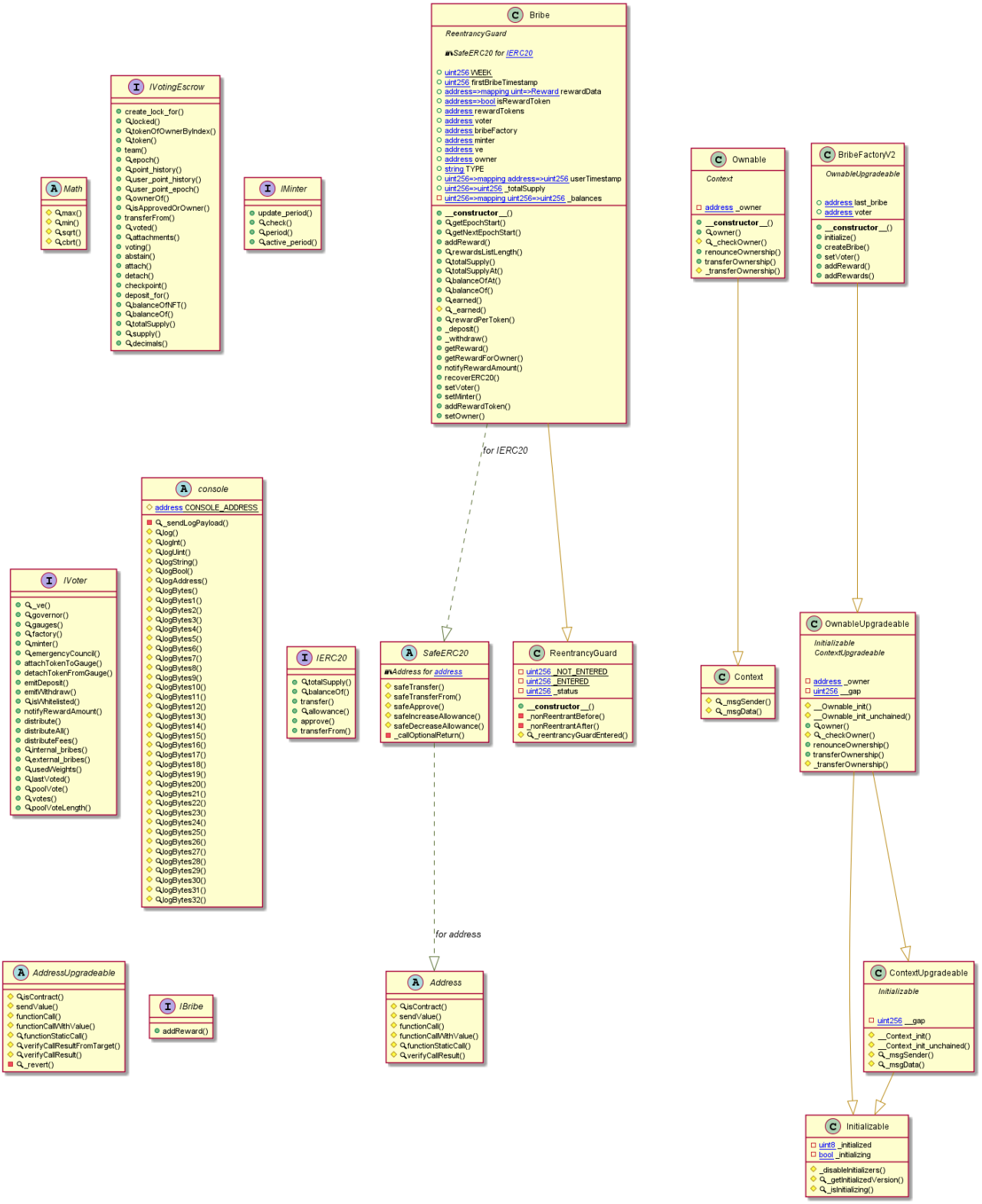
VoterV2_1 Diagram



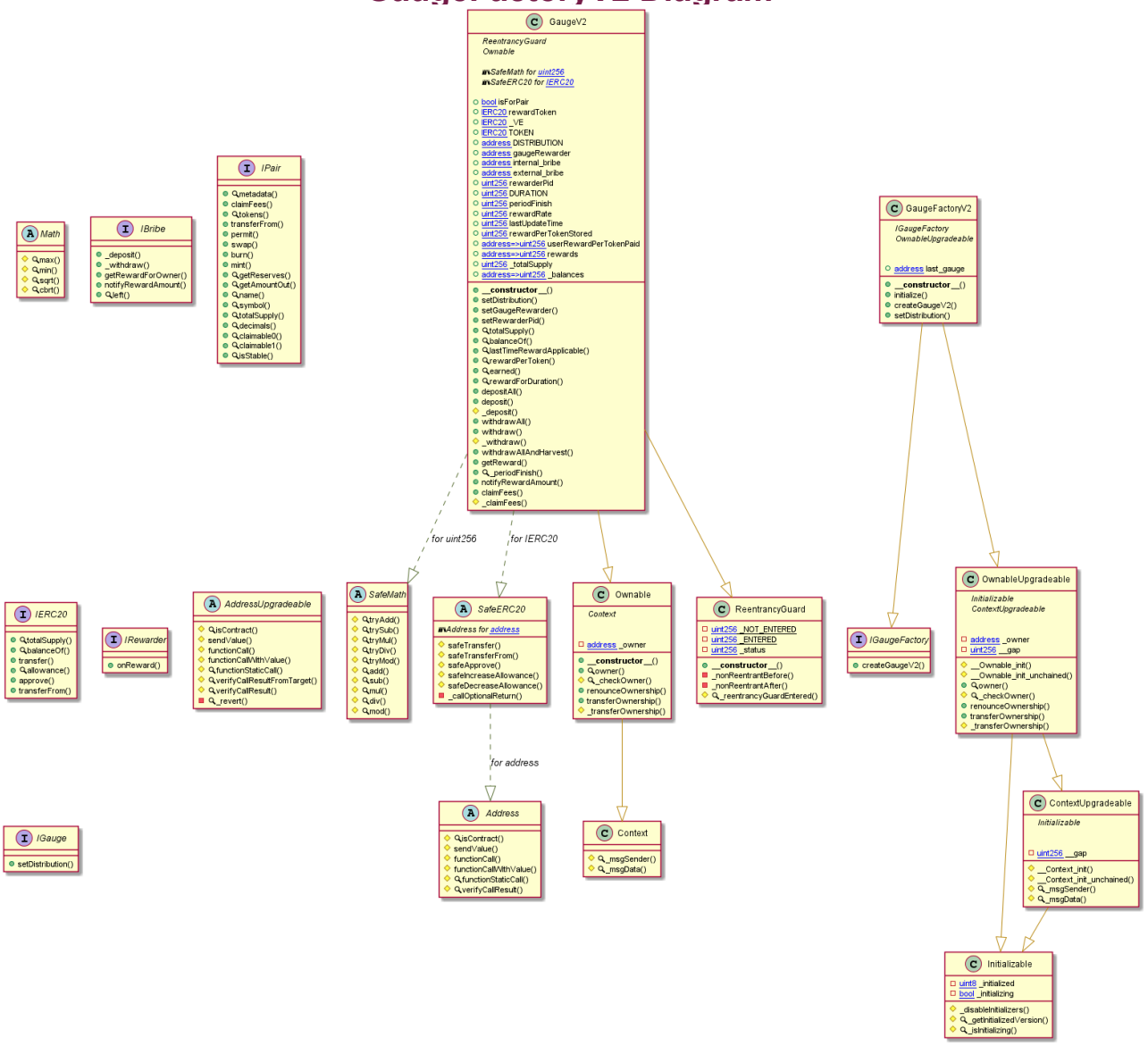
VotingEscrow Diagram



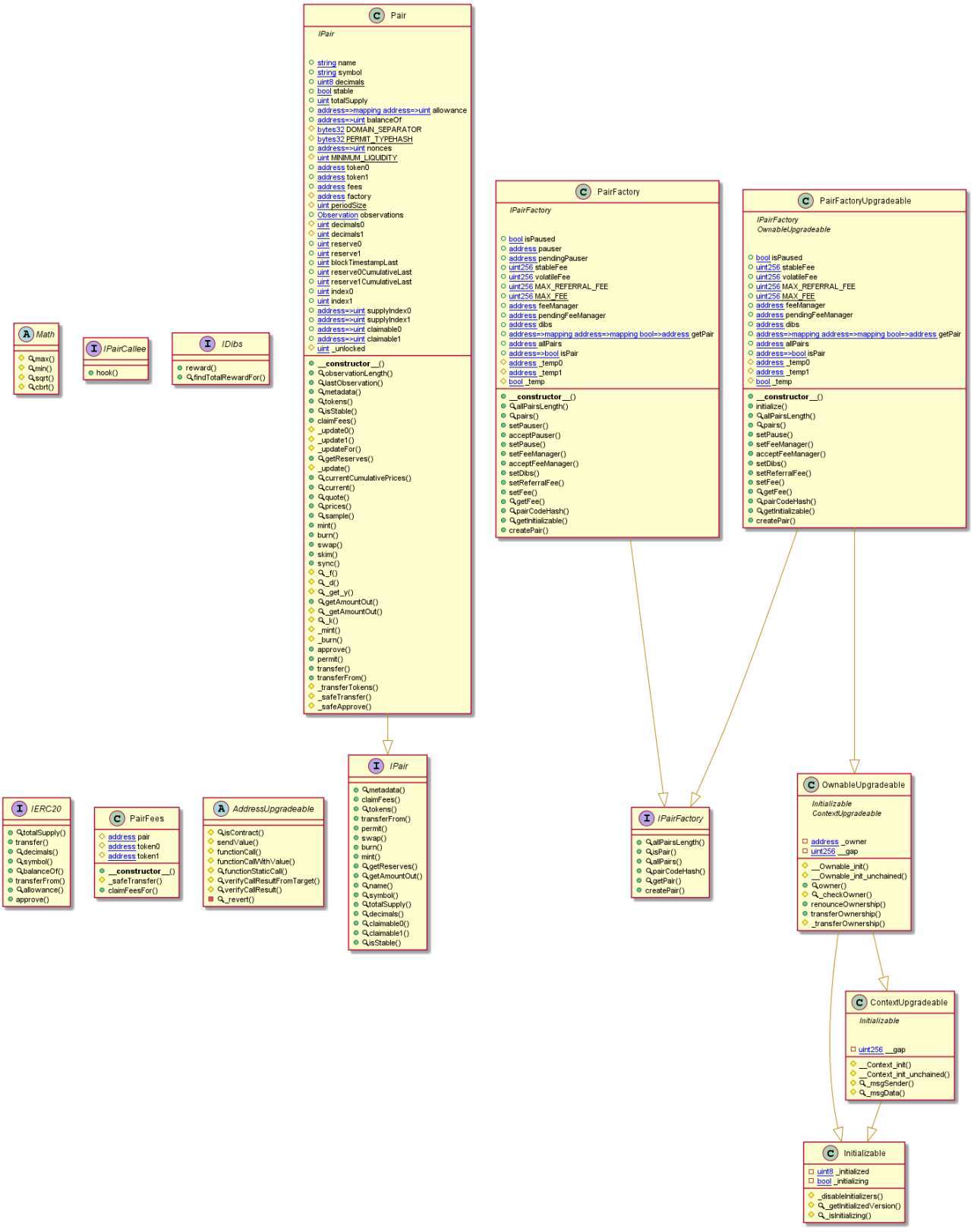
BribeFactoryV2 Diagram



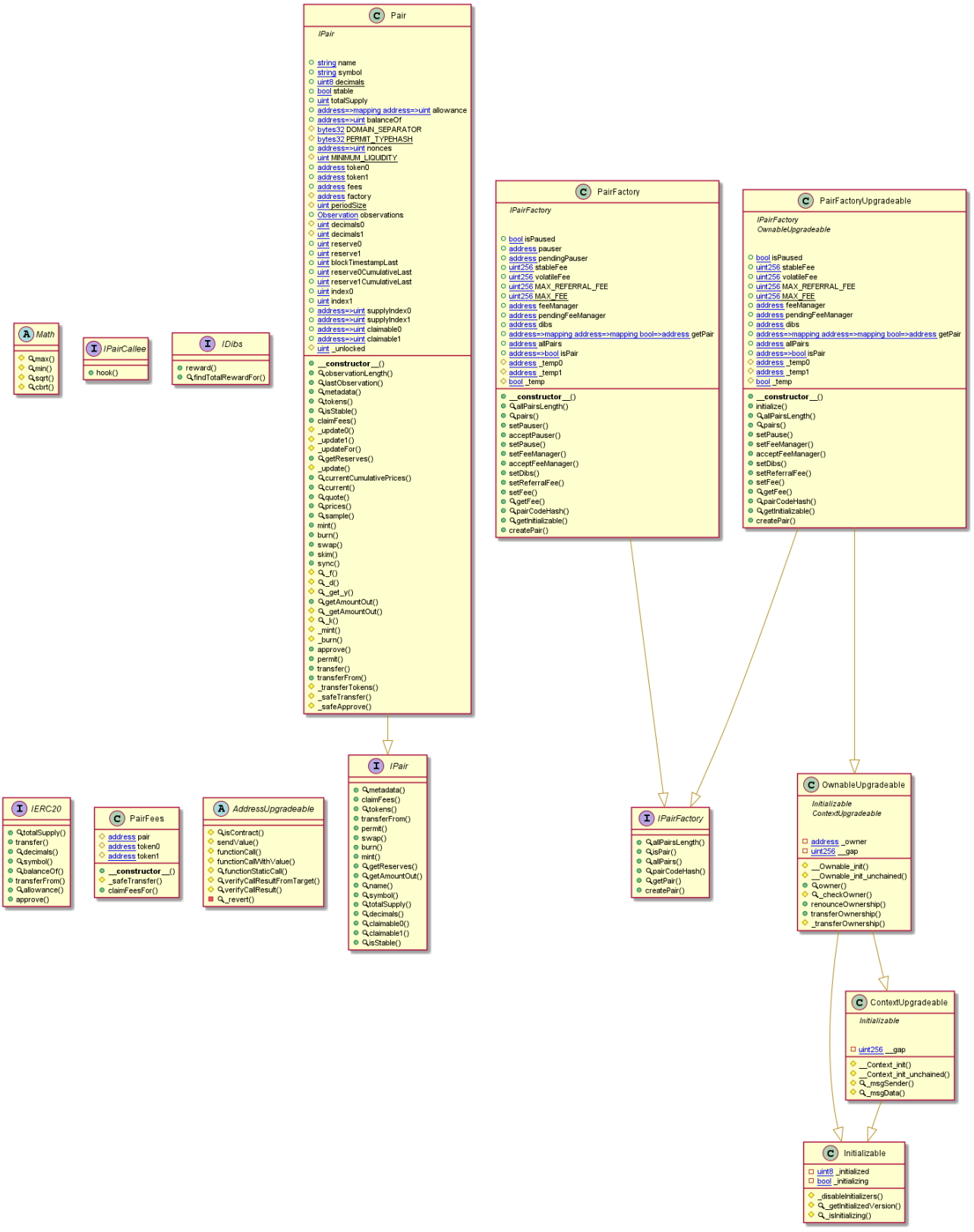
GaugeFactoryV2 Diagram



PairFactory Diagram



PairFactoryUpgradeable Diagram



Slither Results Log

Slither log >> Bribes.sol

```
IVoter.votes(uint256,address).votes (Bribes.sol#117) shadows:
        - IVoter.votes(uint256,address) (Bribes.sol#117) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Bribe.setOwner(address) (Bribes.sol#2400-2403) should emit an event for:
        - owner = _owner (Bribes.sol#2402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

Bribe.earned(uint256,address) (Bribes.sol#2250-2275) has external calls inside a loop: _endTimestamp = IMinter(minter).active_
period() (Bribes.sol#2253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in Bribe.notifyRewardAmount(address,uint256) (Bribes.sol#2357-2374):
        External calls:
        - IERC20(_rewardsToken).safeTransferFrom(msg.sender,address(this),reward) (Bribes.sol#2359)
        State variables written after the call(s):
        - firstBribeTimestamp = _startTimestamp (Bribes.sol#2364)
        - rewardData[_rewardsToken][_startTimestamp].rewardsPerEpoch = _lastReward + reward (Bribes.sol#2369)
        - rewardData[_rewardsToken][_startTimestamp].lastUpdateTime = block.timestamp (Bribes.sol#2370)
        - rewardData[_rewardsToken][_startTimestamp].periodFinish = _startTimestamp + WEEK (Bribes.sol#2371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Redundant expression "k (Bribes.sol#2265)" inBribe (Bribes.sol#2156-2414)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable Bribe.getRewardForOwner(uint256,address[])._rewardToken (Bribes.sol#2347) is too similar to Bribe.rewardTokens (Bribe
s.sol#2172)
Variable Bribe.getReward(uint256,address[])._rewardToken (Bribes.sol#2330) is too similar to Bribe.rewardTokens (Bribes.sol#21
72)
Variable Bribe._earned(uint256,address,uint256)._rewardToken (Bribes.sol#2277) is too similar to Bribe.rewardTokens (Bribes.so
l#2172)
Variable Bribe.earned(uint256,address)._rewardToken (Bribes.sol#2250) is too similar to Bribe.rewardTokens (Bribes.sol#2172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Bribe.TYPE (Bribes.sol#2179) should be immutable
Bribe.bribeFactory (Bribes.sol#2174) should be immutable
Bribe.ve (Bribes.sol#2176) 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#803-807) should emit an event for:
        - DISTRIBUTION = _distribution (GaugeV2.sol#806)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

GaugeV2.setRewarderPid(uint256) (GaugeV2.sol#817-821) should emit an event for:
        - rewarderPid = _pid (GaugeV2.sol#820)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

GaugeV2.constructor(address,address,address,address,address,address,bool)._distribution (GaugeV2.sol#788) lacks a zero-check o
n :
                - DISTRIBUTION = _distribution (GaugeV2.sol#792)
GaugeV2.constructor(address,address,address,address,address,address,bool)._internal_bribe (GaugeV2.sol#788) lacks a zero-check
 on :
                - internal_bribe = _internal_bribe (GaugeV2.sol#795)
GaugeV2.constructor(address,address,address,address,address,address,bool)._external_bribe (GaugeV2.sol#788) lacks a zero-check
 on :
                - external_bribe = _external_bribe (GaugeV2.sol#796)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Function IBribe._deposit(uint256,uint256) (GaugeV2.sol#39) is not in mixedCase
Function IBribe._withdraw(uint256,uint256) (GaugeV2.sol#40) is not in mixedCase
Parameter GaugeV2.setDistribution(address)._distribution (GaugeV2.sol#803) is not in mixedCase
Parameter GaugeV2.setGaugeRewarder(address)._gaugeRewarder (GaugeV2.sol#810) is not in mixedCase
Parameter GaugeV2.setRewarderPid(uint256)._pid (GaugeV2.sol#817) is not in mixedCase
Function GaugeV2._periodFinish() (GaugeV2.sol#934-936) is not in mixedCase
Variable GaugeV2._VE (GaugeV2.sol#746) is not in mixedCase
Variable GaugeV2.TOKEN (GaugeV2.sol#747) is not in mixedCase
Variable GaugeV2.DISTRIBUTION (GaugeV2.sol#749) is not in mixedCase
Variable GaugeV2.internal_bribe (GaugeV2.sol#751) is not in mixedCase
Variable GaugeV2.external_bribe (GaugeV2.sol#752) is not in mixedCase
Variable GaugeV2.DURATION (GaugeV2.sol#755) is not in mixedCase
Variable GaugeV2._totalSupply (GaugeV2.sol#764) is not in mixedCase
Variable GaugeV2._balances (GaugeV2.sol#765) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

GaugeV2.DURATION (GaugeV2.sol#755) should be immutable
GaugeV2.TOKEN (GaugeV2.sol#747) should be immutable
GaugeV2._VE (GaugeV2.sol#746) should be immutable
GaugeV2.external_bribe (GaugeV2.sol#752) should be immutable
GaugeV2.internal_bribe (GaugeV2.sol#751) should be immutable
GaugeV2.isForPair (GaugeV2.sol#742) should be immutable
GaugeV2.rewardToken (GaugeV2.sol#745) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
GaugeV2.sol analyzed (12 contracts with 84 detectors), 66 result(s) found
```

Slither log >> import.sol

```
Pragma version^0.8.4 (import.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (import.sol#61-66):
        - (success) = recipient.call{value: amount}() (import.sol#64)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (import.sol#125-134):
        - (success,returndata) = target.call{value: value}(data) (import.sol#132)
Low level call in Address.functionStaticCall(address,bytes,string) (import.sol#152-159):
        - (success,returndata) = target.staticcall(data) (import.sol#157)
Low level call in Address.functionDelegateCall(address,bytes,string) (import.sol#177-184):
        - (success,returndata) = target.delegatecall(data) (import.sol#182)
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 >> MinterUpgradeable.sol

```
Pragma version0.8.4 (MinterUpgradeable.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (MinterUpgradeable.sol#191-196):
        - (success) = recipient.call{value: amount}() (MinterUpgradeable.sol#194)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (MinterUpgradeable.sol#255-264):
        - (success,returndata) = target.call{value: value}(data) (MinterUpgradeable.sol#262)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (MinterUpgradeable.sol#282-289):
        - (success,returndata) = target.staticcall(data) (MinterUpgradeable.sol#287)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Variable MinterUpgradeable.EMISSION (MinterUpgradeable.sol#560) is not in mixedCase
Variable MinterUpgradeable.TAIL_EMISSION (MinterUpgradeable.sol#561) is not in mixedCase
Variable MinterUpgradeable.REBASEMAX (MinterUpgradeable.sol#562) is not in mixedCase
Variable MinterUpgradeable.active_period (MinterUpgradeable.sol#569) is not in mixedCase
Variable MinterUpgradeable._initializer (MinterUpgradeable.sol#572) is not in mixedCase
Variable MinterUpgradeable._spoon (MinterUpgradeable.sol#576) is not in mixedCase
Variable MinterUpgradeable._voter (MinterUpgradeable.sol#577) is not in mixedCase
Variable MinterUpgradeable._ve (MinterUpgradeable.sol#578) is not in mixedCase
Variable MinterUpgradeable._rewards_distributor (MinterUpgradeable.sol#579) 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 >> Pair.sol

```
PairFactory.setFee(bool,uint256) (Pair.sol#215-224) should emit an event for:
        - stableFee = _fee (Pair.sol#220)
        - volatileFee = _fee (Pair.sol#222)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

PairFees.constructor(address,address)._token0 (Pair.sol#108) lacks a zero-check on :
                - token0 = _token0 (Pair.sol#110)
PairFees.constructor(address,address)._token1 (Pair.sol#108) lacks a zero-check on :
                - token1 = _token1 (Pair.sol#111)
PairFactory.setPauser(address)._pauser (Pair.sol#178) lacks a zero-check on :
                - pendingPauser = _pauser (Pair.sol#180)
PairFactory.setFeeManager(address)._feeManager (Pair.sol#193) lacks a zero-check on :
                - pendingFeeManager = _feeManager (Pair.sol#195)
Pair.constructor()._token1 (Pair.sol#335) lacks a zero-check on :
                - (token0,token1,stable) = (_token0,_token1,_stable) (Pair.sol#336)
                - fees = address(new PairFees(_token0,_token1)) (Pair.sol#337)
Pair.constructor()._token0 (Pair.sol#335) lacks a zero-check on :
                - (token0,token1,stable) = (_token0,_token1,_stable) (Pair.sol#336)
                - fees = address(new PairFees(_token0,_token1)) (Pair.sol#337)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
PairFactory.pairCodeHash() (Pair.sol#230-232) uses literals with too many digits:
        - keccak256(bytes)(type()(Pair).creationCode) (Pair.sol#231)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Pair.name (Pair.sol#257) should be immutable
Pair.symbol (Pair.sol#258) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Pair.sol analyzed (9 contracts with 84 detectors), 77 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 version0.8.4 (PairFees.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in 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.sol analyzed (2 contracts with 84 detectors), 5 result(s) found
```

Slither log >> Router.sol

```
Router.constructor(address,address)._factory (Router.sol#125) lacks a zero-check on :
                - factory = _factory (Router.sol#126)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Math.cbrt(uint256) (Router.sol#23-34) is never used and should be removed
Math.max(uint256,uint256) (Router.sol#5-7) is never used and should be removed
Math.min(uint256,uint256) (Router.sol#8-10) is never used and should be removed
Math.sqrt(uint256) (Router.sol#11-22) is never used and should be removed
Router._safeTransfer(address,address,uint256) (Router.sol#205-210) is never used and should be removed
Router.quoteLiquidity(uint256,uint256,uint256) (Router.sol#154-158) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

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

Low level call in Router._safeTransferETH(address,uint256) (Router.sol#200-203):
        - (success) = to.call{value: value}(new bytes(0)) (Router.sol#201)
Low level call in Router._safeTransfer(address,address,uint256) (Router.sol#205-210):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20.transfer.selector,to,value)) (Router.sol#207-208)
Low level call in Router._safeTransferFrom(address,address,address,uint256) (Router.sol#212-217):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20.transferFrom.selector,from,to,value)) (Router.sol#214-215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Struct Router.route (Router.sol#107-111) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Router.MINIMUM_LIQUIDITY (Router.sol#115) is never used in Router (Router.sol#105-218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
Router.sol analyzed (7 contracts with 84 detectors), 17 result(s) found
```

Slither log >> RouterV2.sol

```
RouterV2.constructor(address,address)._factory (RouterV2.sol#98) lacks a zero-check on :
                - factory = _factory (RouterV2.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

RouterV2.getAmountsOut(uint256,RouterV2.route[]) (RouterV2.sol#155-165) has external calls inside a loop: IBaseV1Factory(facto
ry).isPair(pair) (RouterV2.sol#161)
RouterV2.getAmountsOut(uint256,RouterV2.route[]) (RouterV2.sol#155-165) has external calls inside a loop: amounts[i + 1] = IBa
seV1Pair(pair).getAmountOut(amounts[i],routes[i].from) (RouterV2.sol#162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
Pragma version0.8.4 (RouterV2.sol#7) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Contract erc20 (RouterV2.sol#28-36) is not in CapWords
Function IPairFactory.MAX_REFERRAL_FEE() (RouterV2.sol#40) is not in mixedCase
Struct RouterV2.route (RouterV2.sol#78-82) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

RouterV2.MINIMUM_LIQUIDITY (RouterV2.sol#86) is never used in RouterV2 (RouterV2.sol#74-174)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
RouterV2.sol analyzed (7 contracts with 84 detectors), 16 result(s) found
```

Slither log >> Spoon.sol

```
Spoon.setMinter(address)._minter (Spoon.sol#39) lacks a zero-check on :
                - minter = _minter (Spoon.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Pragma version0.8.4 (Spoon.sol#2) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter Spoon.setMinter(address)._minter (Spoon.sol#39) is not in mixedCase
Parameter Spoon.initialMint(address)._recipient (Spoon.sol#46) is not in mixedCase
Parameter Spoon.approve(address,uint256)._spender (Spoon.sol#52) is not in mixedCase
Parameter Spoon.approve(address,uint256)._value (Spoon.sol#52) is not in mixedCase
Parameter Spoon.transfer(address,uint256)._to (Spoon.sol#76) is not in mixedCase
Parameter Spoon.transfer(address,uint256)._value (Spoon.sol#76) is not in mixedCase
Parameter Spoon.transferFrom(address,address,uint256)._from (Spoon.sol#80) is not in mixedCase
Parameter Spoon.transferFrom(address,address,uint256)._to (Spoon.sol#80) is not in mixedCase
Parameter Spoon.transferFrom(address,address,uint256)._value (Spoon.sol#80) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Spoon.merkleClaim (Spoon.sol#28) should be constant
Spoon.redemptionReceiver (Spoon.sol#27) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
Spoon.sol analyzed (2 contracts with 84 detectors), 14 result(s) found
```

Slither log >> VeArtProxyUpgradeable.sol

```
Low level call in AddressUpgradeable.sendValue(address,uint256) (VeArtProxyUpgradeable.sol#119-124):
        - (success) = recipient.call{value: amount}() (VeArtProxyUpgradeable.sol#122)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (VeArtProxyUpgradeable.sol#183-192):
        - (success,returndata) = target.call{value: value}(data) (VeArtProxyUpgradeable.sol#190)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (VeArtProxyUpgradeable.sol#210-217):
        - (success,returndata) = target.staticcall(data) (VeArtProxyUpgradeable.sol#215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IVeArtProxy._tokenURI(uint256,uint256,uint256,uint256) (VeArtProxyUpgradeable.sol#61) is not in mixedCase
Parameter IVeArtProxy._tokenURI(uint256,uint256,uint256,uint256)._locked_end (VeArtProxyUpgradeable.sol#61) is not in mixedCas
e
Function ContextUpgradeable.__Context_init() (VeArtProxyUpgradeable.sol#386-387) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (VeArtProxyUpgradeable.sol#389-390) is not in mixedCase
Variable ContextUpgradeable.__gap (VeArtProxyUpgradeable.sol#404) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (VeArtProxyUpgradeable.sol#414-416) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (VeArtProxyUpgradeable.sol#418-420) is not in mixedCase
Variable OwnableUpgradeable.__gap (VeArtProxyUpgradeable.sol#479) is not in mixedCase
Function VeArtProxyUpgradeable._tokenURI(uint256,uint256,uint256,uint256) (VeArtProxyUpgradeable.sol#514-523) is not in mixedC
ase
Parameter VeArtProxyUpgradeable._tokenURI(uint256,uint256,uint256,uint256)._tokenId (VeArtProxyUpgradeable.sol#514) is not in
mixedCase
Parameter VeArtProxyUpgradeable._tokenURI(uint256,uint256,uint256,uint256)._balanceOf (VeArtProxyUpgradeable.sol#514) is not i
n mixedCase
Parameter VeArtProxyUpgradeable._tokenURI(uint256,uint256,uint256,uint256)._locked_end (VeArtProxyUpgradeable.sol#514) is not
in mixedCase
Parameter VeArtProxyUpgradeable._tokenURI(uint256,uint256,uint256,uint256)._value (VeArtProxyUpgradeable.sol#514) 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 >> VoterV2_1.sol

```
IVoter.votes(uint256,address).votes (VoterV2_1.sol#196) shadows:
        - IVoter.votes(uint256,address) (VoterV2_1.sol#196) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

VoterV2_1.initialize(address,address,address,address).__ve (VoterV2_1.sol#2272) lacks a zero-check on :
                - _ve = __ve (VoterV2_1.sol#2275)
                - base = IVotingEscrow(__ve).token() (VoterV2_1.sol#2277)
VoterV2_1.initialize(address,address,address,address)._factory (VoterV2_1.sol#2272) lacks a zero-check on :
                - factory = _factory (VoterV2_1.sol#2276)
VoterV2_1.initialize(address,address,address,address)._gauges (VoterV2_1.sol#2272) lacks a zero-check on :
                - gaugefactory = _gauges (VoterV2_1.sol#2278)
VoterV2_1.initialize(address,address,address,address)._bribes (VoterV2_1.sol#2272) lacks a zero-check on :
```

```
VoterV2_1.setEmergencyCouncil(address)._council (VoterV2_1.sol#2305) lacks a zero-check on :
                - emergencyCouncil = _council (VoterV2_1.sol#2307)
VoterV2_1.setBribeFactory(address)._bribeFactory (VoterV2_1.sol#2610) lacks a zero-check on :
                - bribefactory = _bribeFactory (VoterV2_1.sol#2612)
VoterV2_1.setGaugeFactory(address)._gaugeFactory (VoterV2_1.sol#2615) lacks a zero-check on :
                - gaugefactory = _gaugeFactory (VoterV2_1.sol#2617)
VoterV2_1.setPairFactory(address)._factory (VoterV2_1.sol#2620) lacks a zero-check on :
                - factory = _factory (VoterV2_1.sol#2622)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Parameter VoterV2_1.claimBribes(address[],address[][],uint256)._bribes (VoterV2_1.sol#2548) is not in mixedCase
Parameter VoterV2_1.claimBribes(address[],address[][],uint256)._tokens (VoterV2_1.sol#2548) is not in mixedCase
Parameter VoterV2_1.claimBribes(address[],address[][],uint256)._tokenId (VoterV2_1.sol#2548) is not in mixedCase
Parameter VoterV2_1.claimFees(address[],address[][],uint256)._fees (VoterV2_1.sol#2555) is not in mixedCase
Parameter VoterV2_1.claimFees(address[],address[][],uint256)._tokens (VoterV2_1.sol#2555) is not in mixedCase
Parameter VoterV2_1.claimFees(address[],address[][],uint256)._tokenId (VoterV2_1.sol#2555) is not in mixedCase
Parameter VoterV2_1.distributeFees(address[])._gauges (VoterV2_1.sol#2563) is not in mixedCase
Parameter VoterV2_1.distribute(address)._gauge (VoterV2_1.sol#2571) is not in mixedCase
Parameter VoterV2_1.distribute(address[])._gauges (VoterV2_1.sol#2597) is not in mixedCase
Parameter VoterV2_1.setBribeFactory(address)._bribeFactory (VoterV2_1.sol#2610) is not in mixedCase
Parameter VoterV2_1.setGaugeFactory(address)._gaugeFactory (VoterV2_1.sol#2615) is not in mixedCase
Parameter VoterV2_1.setPairFactory(address)._factory (VoterV2_1.sol#2620) is not in mixedCase
Parameter VoterV2_1.killGaugeTotally(address)._gauge (VoterV2_1.sol#2625) is not in mixedCase
Parameter VoterV2_1.whitelist(address[])._token (VoterV2_1.sol#2641) is not in mixedCase
Parameter VoterV2_1.initGauges(address[],address[])._gauges (VoterV2_1.sol#2649) is not in mixedCase
Parameter VoterV2_1.initGauges(address[],address[])._pools (VoterV2_1.sol#2649) is not in mixedCase
Parameter VoterV2_1.increaseGaugeApprovals(address)._gauge (VoterV2_1.sol#2669) is not in mixedCase
Parameter VoterV2_1.setNewBribe(address,address,address)._gauge (VoterV2_1.sol#2676) is not in mixedCase
Parameter VoterV2_1.setNewBribe(address,address,address)._internal (VoterV2_1.sol#2676) is not in mixedCase
Parameter VoterV2_1.setNewBribe(address,address,address)._external (VoterV2_1.sol#2676) is not in mixedCase
Variable VoterV2_1._ve (VoterV2_1.sol#2226) is not in mixedCase
Variable VoterV2_1.internal_bribes (VoterV2_1.sol#2246) is not in mixedCase
Variable VoterV2_1.external_bribes (VoterV2_1.sol#2247) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "i (VoterV2_1.sol#2652)" inVoterV2_1 (VoterV2_1.sol#2224-2684)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable VoterV2_1._vote(uint256,address[],uint256[])._usedWeight (VoterV2_1.sol#2365) is too similar to VoterV2_1.usedWeights
 (VoterV2_1.sol#2251)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

VoterV2_1.DURATION (VoterV2_1.sol#2231) is never used in VoterV2_1 (VoterV2_1.sol#2224-2684)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
VoterV2_1.sol analyzed (18 contracts with 84 detectors), 524 result(s) found
```

Slither log >> VotingEscrow.sol

```
VotingEscrow.constructor(address,address).token_addr (VotingEscrow.sol#383) lacks a zero-check on :
                - token = token_addr (VotingEscrow.sol#384)
VotingEscrow.constructor(address,address).art_proxy (VotingEscrow.sol#383) lacks a zero-check on :
                - artProxy = art_proxy (VotingEscrow.sol#387)
VotingEscrow.setTeam(address)._team (VotingEscrow.sol#426) lacks a zero-check on :
                - team = _team (VotingEscrow.sol#428)
VotingEscrow.setArtProxy(address)._proxy (VotingEscrow.sol#431) lacks a zero-check on :
                - artProxy = _proxy (VotingEscrow.sol#433)
VotingEscrow.setVoter(address)._voter (VotingEscrow.sol#1332) lacks a zero-check on :
                - voter = _voter (VotingEscrow.sol#1334)
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#993-1
026) has external calls inside a loop: assert(bool)(IERC20(token).transferFrom(from,address(this),_value)) (VotingEscrow.sol#1
021)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
Reentrancy in VotingEscrow.withdraw(uint256) (VotingEscrow.sol#1117-1141):
        External calls:
        - assert(bool)(IERC20(token).transfer(msg.sender,value)) (VotingEscrow.sol#1134)
        Event emitted after the call(s):
        - Approval(owner,_approved,_tokenId) (VotingEscrow.sol#521)
                - _burn(_tokenId) (VotingEscrow.sol#1137)
        - Supply(supply_before,supply_before - value) (VotingEscrow.sol#1140)
        - Transfer(owner,address(0),_tokenId) (VotingEscrow.sol#800)
                - _burn(_tokenId) (VotingEscrow.sol#1137)
        - Withdraw(msg.sender,_tokenId,value,block.timestamp) (VotingEscrow.sol#1139)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

VotingEscrow._checkpoint(uint256,VotingEscrow.LockedBalance,VotingEscrow.LockedBalance) (VotingEscrow.sol#850-985) uses timest
amp for comparisons
        Dangerous comparisons:
        - old_locked.end > block.timestamp && old_locked.amount > 0 (VotingEscrow.sol#864)
        - new_locked.end > block.timestamp && new_locked.amount > 0 (VotingEscrow.sol#868)
        - new_locked.end != 0 (VotingEscrow.sol#877)
        - new_locked.end == old_locked.end (VotingEscrow.sol#878)
        - block.timestamp > last_point.ts (VotingEscrow.sol#896)
        - t_i > block.timestamp (VotingEscrow.sol#910)
        - last_point.bias < 0 (VotingEscrow.sol#917)
        - last_point.slope < 0 (VotingEscrow.sol#921)
        - t_i == block.timestamp (VotingEscrow.sol#929)
        - last_point.slope < 0 (VotingEscrow.sol#946)
        - last_point.bias < 0 (VotingEscrow.sol#949)
        - old_locked.end > block.timestamp (VotingEscrow.sol#961)
        - new_locked.end == old_locked.end (VotingEscrow.sol#964)
        - new_locked.end > block.timestamp (VotingEscrow.sol#970)
        - new_locked.end > old_locked.end (VotingEscrow.sol#971)
VotingEscrow._deposit_for(uint256,uint256,uint256,VotingEscrow.LockedBalance,VotingEscrow.DepositType) (VotingEscrow.sol#993-1
026) uses timestamp for comparisons
```

```
Variable VotingEscrow._moveAllDelegates(address,address,address).tId_scope_1 (VotingEscrow.sol#1644) is too similar to VotingE
scrow._moveAllDelegates(address,address,address).tId_scope_3 (VotingEscrow.sol#1649)
Variable VotingEscrow._moveTokenDelegates(address,address,uint256).tId_scope_1 (VotingEscrow.sol#1573) is too similar to Votin
gEscrow._moveAllDelegates(address,address,address).tId_scope_3 (VotingEscrow.sol#1649)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
VotingEscrow.sol analyzed (9 contracts with 84 detectors), 152 result(s) found
```

Slither log >> BribeFactoryV2.sol

```
IVoter.votes(uint256,address).votes (BribeFactoryV2.sol#117) shadows:
        - IVoter.votes(uint256,address) (BribeFactoryV2.sol#117) (function)
BribeFactoryV2.createBribe(address,address,address,string)._owner (BribeFactoryV2.sol#2849) shadows:
        - OwnableUpgradeable._owner (BribeFactoryV2.sol#2760) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Bribe.setOwner(address) (BribeFactoryV2.sol#2400-2403) should emit an event for:
        - owner = _owner (BribeFactoryV2.sol#2402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

BribeFactoryV2.initialize(address)._voter (BribeFactoryV2.sol#2844) lacks a zero-check on :
                - voter = _voter (BribeFactoryV2.sol#2846)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Redundant expression "k (BribeFactoryV2.sol#2265)" inBribe (BribeFactoryV2.sol#2156-2414)
Redundant expression "i (BribeFactoryV2.sol#2867)" inBribeFactoryV2 (BribeFactoryV2.sol#2839-2887)
Redundant expression "i (BribeFactoryV2.sol#2877)" inBribeFactoryV2 (BribeFactoryV2.sol#2839-2887)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable Bribe.earned(uint256,address)._rewardToken (BribeFactoryV2.sol#2250) is too similar to Bribe.rewardTokens (BribeFacto
ryV2.sol#2172)
Variable Bribe.getReward(uint256,address[])._rewardToken (BribeFactoryV2.sol#2330) is too similar to Bribe.rewardTokens (Bribe
FactoryV2.sol#2172)
Variable Bribe.getRewardForOwner(uint256,address[])._rewardToken (BribeFactoryV2.sol#2347) is too similar to Bribe.rewardToken
s (BribeFactoryV2.sol#2172)
Variable Bribe._earned(uint256,address,uint256)._rewardToken (BribeFactoryV2.sol#2277) is too similar to Bribe.rewardTokens (B
ribeFactoryV2.sol#2172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Bribe.TYPE (BribeFactoryV2.sol#2179) should be immutable
Bribe.bribeFactory (BribeFactoryV2.sol#2174) should be immutable
Bribe.ve (BribeFactoryV2.sol#2176) 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

```
Function IBribe._deposit(uint256,uint256) (GaugeFactoryV2.sol#39) is not in mixedCase
Function IBribe._withdraw(uint256,uint256) (GaugeFactoryV2.sol#40) is not in mixedCase
Parameter GaugeV2.setDistribution(address)._distribution (GaugeFactoryV2.sol#863) is not in mixedCase
Parameter GaugeV2.setGaugeRewarder(address)._gaugeRewarder (GaugeFactoryV2.sol#870) is not in mixedCase
Parameter GaugeV2.setRewarderPid(uint256)._pid (GaugeFactoryV2.sol#877) is not in mixedCase
Function GaugeV2._periodFinish() (GaugeFactoryV2.sol#994-996) is not in mixedCase
Variable GaugeV2._VE (GaugeFactoryV2.sol#806) is not in mixedCase
Variable GaugeV2.TOKEN (GaugeFactoryV2.sol#807) is not in mixedCase
Variable GaugeV2.DISTRIBUTION (GaugeFactoryV2.sol#809) is not in mixedCase
Variable GaugeV2.internal_bribe (GaugeFactoryV2.sol#811) is not in mixedCase
Variable GaugeV2.external_bribe (GaugeFactoryV2.sol#812) is not in mixedCase
Variable GaugeV2.DURATION (GaugeFactoryV2.sol#815) is not in mixedCase
Variable GaugeV2._totalSupply (GaugeFactoryV2.sol#824) is not in mixedCase
Variable GaugeV2._balances (GaugeFactoryV2.sol#825) is not in mixedCase
```

```
GaugeV2.DURATION (GaugeFactoryV2.sol#815) should be immutable
GaugeV2.TOKEN (GaugeFactoryV2.sol#807) should be immutable
GaugeV2._VE (GaugeFactoryV2.sol#806) should be immutable
GaugeV2.external_bribe (GaugeFactoryV2.sol#812) should be immutable
GaugeV2.internal_bribe (GaugeFactoryV2.sol#811) should be immutable
GaugeV2.isForPair (GaugeFactoryV2.sol#802) should be immutable
GaugeV2.rewardToken (GaugeFactoryV2.sol#805) 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), 111 result(s) found
```

Slither log >> PairFactory.sol

```
PairFactory.setFee(bool,uint256) (PairFactory.sol#216-225) should emit an event for:
        - stableFee = _fee (PairFactory.sol#221)
        - volatileFee = _fee (PairFactory.sol#223)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

PairFees.constructor(address,address)._token0 (PairFactory.sol#108) lacks a zero-check on :
                - token0 = _token0 (PairFactory.sol#110)
PairFees.constructor(address,address)._token1 (PairFactory.sol#108) lacks a zero-check on :
                - token1 = _token1 (PairFactory.sol#111)
PairFactory.setPauser(address)._pauser (PairFactory.sol#179) lacks a zero-check on :
                - pendingPauser = _pauser (PairFactory.sol#181)
PairFactory.setFeeManager(address)._feeManager (PairFactory.sol#194) lacks a zero-check on :
                - pendingFeeManager = _feeManager (PairFactory.sol#196)
Pair.constructor()._token1 (PairFactory.sol#336) lacks a zero-check on :
                - (token0,token1,stable) = (_token0,_token1,_stable) (PairFactory.sol#337)
                - fees = address(new PairFees(_token0,_token1)) (PairFactory.sol#338)
Pair.constructor()._token0 (PairFactory.sol#336) lacks a zero-check on :
                - (token0,token1,stable) = (_token0,_token1,_stable) (PairFactory.sol#337)
                - fees = address(new PairFees(_token0,_token1)) (PairFactory.sol#338)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Pair.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (PairFactory.sol#744-767) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Pair: EXPIRED) (PairFactory.sol#745)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Math.cbrt(uint256) (PairFactory.sol#24-35) is never used and should be removed
Math.max(uint256,uint256) (PairFactory.sol#6-8) is never used and should be removed
Pair._safeApprove(address,address,uint256) (PairFactory.sol#805-812) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
PairFactory.pairCodeHash() (PairFactory.sol#231-233) uses literals with too many digits:
        - keccak256(bytes)(type()(Pair).creationCode) (PairFactory.sol#232)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Pair.name (PairFactory.sol#258) should be immutable
Pair.symbol (PairFactory.sol#259) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
PairFactory.sol analyzed (9 contracts with 84 detectors), 77 result(s) found
```

Slither log >> PairFactoryUpgradeable.sol

```
PairFactory.setFee(bool,uint256) (PairFactoryUpgradeable.sol#219-228) should emit an event for:
        - stableFee = _fee (PairFactoryUpgradeable.sol#224)
        - volatileFee = _fee (PairFactoryUpgradeable.sol#226)
PairFactoryUpgradeable.setFee(bool,uint256) (PairFactoryUpgradeable.sol#1309-1317) should emit an event for:
        - stableFee = _fee (PairFactoryUpgradeable.sol#1313)
        - volatileFee = _fee (PairFactoryUpgradeable.sol#1315)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

PairFees.constructor(address,address)._token0 (PairFactoryUpgradeable.sol#110) lacks a zero-check on :
                - token0 = _token0 (PairFactoryUpgradeable.sol#112)
PairFees.constructor(address,address)._token1 (PairFactoryUpgradeable.sol#110) lacks a zero-check on :
                - token1 = _token1 (PairFactoryUpgradeable.sol#113)
PairFactory.setPauser(address)._pauser (PairFactoryUpgradeable.sol#182) lacks a zero-check on :
                - pendingPauser = _pauser (PairFactoryUpgradeable.sol#184)
PairFactory.setFeeManager(address)._feeManager (PairFactoryUpgradeable.sol#197) lacks a zero-check on :
                - pendingFeeManager = _feeManager (PairFactoryUpgradeable.sol#199)
Pair.constructor()._token1 (PairFactoryUpgradeable.sol#339) lacks a zero-check on :
                - (token0,token1,stable) = (_token0,_token1,_stable) (PairFactoryUpgradeable.sol#340)
                - fees = address(new PairFees(_token0,_token1)) (PairFactoryUpgradeable.sol#341)
Pair.constructor()._token0 (PairFactoryUpgradeable.sol#339) lacks a zero-check on :
                - (token0,token1,stable) = (_token0,_token1,_stable) (PairFactoryUpgradeable.sol#340)
                - fees = address(new PairFees(_token0,_token1)) (PairFactoryUpgradeable.sol#341)
PairFactoryUpgradeable.setFeeManager(address)._feeManager (PairFactoryUpgradeable.sol#1290) lacks a zero-check on :
                - pendingFeeManager = _feeManager (PairFactoryUpgradeable.sol#1291)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
PairFactory.pairCodeHash() (PairFactoryUpgradeable.sol#234-236) uses literals with too many digits:
        - keccak256(bytes)(type()(Pair).creationCode) (PairFactoryUpgradeable.sol#235)
PairFactoryUpgradeable.pairCodeHash() (PairFactoryUpgradeable.sol#1323-1325) uses literals with too many digits:
        - keccak256(bytes)(type()(Pair).creationCode) (PairFactoryUpgradeable.sol#1324)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Pair.name (PairFactoryUpgradeable.sol#261) should be immutable
Pair.symbol (PairFactoryUpgradeable.sol#262) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
PairFactoryUpgradeable.sol analyzed (14 contracts with 84 detectors), 117 result(s) found
```

Slither log >> RewardsDistributor.sol

```
RewardsDistributor.constructor(address)._token (RewardsDistributor.sol#145) lacks a zero-check on :
                - token = _token (RewardsDistributor.sol#146)
RewardsDistributor.constructor(address)._voting_escrow (RewardsDistributor.sol#140) lacks a zero-check on :
                - voting_escrow = _voting_escrow (RewardsDistributor.sol#147)
RewardsDistributor.setDepositor(address)._depositor (RewardsDistributor.sol#422) lacks a zero-check on :
                - depositor = _depositor (RewardsDistributor.sol#424)
RewardsDistributor.setOwner(address)._owner (RewardsDistributor.sol#427) lacks a zero-check on :
                - owner = _owner (RewardsDistributor.sol#429)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
RewardsDistributor._claimable(uint256,address,uint256) (RewardsDistributor.sol#319-368) uses timestamp for comparisons
        Dangerous comparisons:
        - week_cursor == 0 (RewardsDistributor.sol#329)
        - week_cursor == 0 (RewardsDistributor.sol#339)
        - week_cursor >= last_token_time (RewardsDistributor.sol#340)
        - week_cursor < _start_time (RewardsDistributor.sol#341)
        - week_cursor >= _last_token_time (RewardsDistributor.sol#346)
        - week_cursor >= user_point.ts && user_epoch <= max_user_epoch (RewardsDistributor.sol#348)
        - balance_of == 0 && user_epoch > max_user_epoch (RewardsDistributor.sol#359)
        - balance_of != 0 (RewardsDistributor.sol#360)
RewardsDistributor.claim(uint256) (RewardsDistributor.sol#375-391) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= time_cursor (RewardsDistributor.sol#376)
        - _locked.end < block.timestamp (RewardsDistributor.sol#382)
RewardsDistributor.claim_many(uint256[]) (RewardsDistributor.sol#393-420) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= time_cursor (RewardsDistributor.sol#394)
        - _locked.end < block.timestamp (RewardsDistributor.sol#406)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Math.cbrt(uint256) (RewardsDistributor.sol#22-34) is never used and should be removed
Math.sqrt(uint256) (RewardsDistributor.sol#10-21) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
Function IRewardsDistributor.checkpoint_token() (RewardsDistributor.sol#53) is not in mixedCase
Function IRewardsDistributor.voting_escrow() (RewardsDistributor.sol#54) is not in mixedCase
Function IRewardsDistributor.checkpoint_total_supply() (RewardsDistributor.sol#55) is not in mixedCase
Function IVotingEscrow.create_lock_for(uint256,uint256,address) (RewardsDistributor.sol#73) is not in mixedCase
Parameter IVotingEscrow.create_lock_for(uint256,uint256,address)._lock_duration (RewardsDistributor.sol#73) is not in mixedCas
e
```

```
RewardsDistributor.start_time (RewardsDistributor.sol#125) should be immutable
RewardsDistributor.token (RewardsDistributor.sol#137) should be immutable
RewardsDistributor.voting_escrow (RewardsDistributor.sol#136) 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), 94 result(s) found
```

Solidity Static Analysis

Bribes.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Bribe.recoverERC20(address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2378:4:

Gas & Economy

Gas costs:

Gas requirement of function Bribe.getReward 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: 2323:4:

Gas costs:

Gas requirement of function Bribe.earned 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: 2250:4:

Miscellaneous

Constant/View/Pure functions:

SafeERC20._callOptionalReturn(contract IERC20,bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2012:4:

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: 2401:8:

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: 959:23:

Gas & Economy

Gas costs:

Gas requirement of function GaugeV2.claimFees 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: 963:4:

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: 956:8:

import.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: 532:8:

Gas & Economy

Gas costs:

Gas requirement of function TransparentUpgradeableProxy.upgradeToAndCall 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: 693:4:

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: 717:8:

MinterUpgradeable.sol

Security

Check-effects-interaction:

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

more

Pos: 700:4:

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: 613:4:

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: 622:12:

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: 746:8:

Pair.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: 744:28:

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: 526:4:

Gas costs:

Gas requirement of function Pair.sample 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: 540:4:

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: 762:8:

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: 35:44:

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: 40:4:

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: 41:8:

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: 414:33:

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: 196:4:

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: 431:8:

Router.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: 215:8:

Gas & Economy

Gas costs:

Gas requirement of function Router.swapExactTokensForETH 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: 184:4:

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: 213:8:

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: 94:28:

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: 159:8:

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: 156:8:

Spoon.sol

Gas & Economy

Gas costs:

Gas requirement of function Spoon.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: 88:4:

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: 89:8:

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: 183:4:

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: 514:4:

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: 189:8:

VoterV2_1.sol

Gas & Economy

Gas costs:

Gas requirement of function VoterV2_1.reset 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: 2310:4:

Gas costs:

Gas requirement of function VoterV2_1.poke 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: 2345:4:

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: 2669:8:

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: 2176:8:

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: 1713:12:

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: 1643:16:

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: 1122:8:

BribeFactoryV2.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in BribeFactoryV2.createBribe(address,address,address,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 2849:4:

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: 2873:4:

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: 2879:12:

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: 2859:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 2292:15:

GaugeFactoryV2.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: 1019:23:

Gas & Economy

Gas costs:

Gas requirement of function GaugeFactoryV2.setDistribution 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: 1498:4:

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: 1458:8:

PairFactory.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: 745:28:

Gas & Economy

Gas costs:

Gas requirement of function Pair.sample 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: 541:4:

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: 802:8:

PairFactoryUpgradeable.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: 748:28:

Gas & Economy

Gas costs:

Gas requirement of function PairFactoryUpgradeable.createPair 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: 1331:4:

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: 533:8:

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: 720:19:

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: 1311:8:

Solhint Linter

Bribes.sol

```
Bribes.sol:24:73: Error: Parse error: missing ';' at '{'
Bribes.sol:1998:18: Error: Parse error: missing ';' at '{'
```

GaugeV2.sol

```
GaugeV2.sol:23:73: Error: Parse error: missing ';' at '{'
GaugeV2.sol:86:18: Error: Parse error: missing ';' at '{'
GaugeV2.sol:99:18: Error: Parse error: missing ';' at '{'
GaugeV2.sol:111:18: Error: Parse error: missing ';' at '{'
GaugeV2.sol:128:18: Error: Parse error: missing ';' at '{'
GaugeV2.sol:140:18: Error: Parse error: missing ';' at '{'
GaugeV2.sol:232:18: Error: Parse error: missing ';' at '{'
GaugeV2.sol:251:18: Error: Parse error: missing ';' at '{'
GaugeV2.sol:273:18: Error: Parse error: missing ';' at '{'
GaugeV2.sol:567:18: Error: Parse error: missing ';' at '{'
```

import.sol

```
import.sol:2:1: Error: Compiler version ^0.8.0 does not satisfy the r
semver requirement
import.sol:182:51: Error: Avoid using low level calls.
import.sol:532:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
import.sol:594:49: Error: Code contains empty blocksimport.sol:725:5:
Error: Explicitly mark visibility in function (Set ignoreConstructors
to true if using solidity >=0.7.0)
import.sol:725:122: Error: Code contains empty blocks
```

MinterUpgradeable.sol

```
MinterUpgradeable.sol:22:73: Error: Parse error: missing ';' at '{'
```

Pair.sol

```
Pair.sol:24:73: Error: Parse error: missing ';' at '{'
```

```
Pair.sol:555:22: 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:27:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
PairFees.sol:35:45: Error: Avoid using low level calls.
```

RewardsDistributor.sol

```
RewardsDistributor.sol:22:73: Error: Parse error: missing ';' at '{'
```

Router.sol

```
Router.sol:23:73: Error: Parse error: missing ';' at '{'
```

RouterV2.sol

```
RouterV2.sol:7:1: Error: Compiler version 0.8.13 does not satisfy the
r semver requirementRouterV2.sol:40:5: Error: Function name must be
in mixedCase
RouterV2.sol:61:35: Error: Use double quotes for string literals
RouterV2.sol:78:5: Error: Contract name must be in CamelCase
RouterV2.sol:87:5: Error: Explicitly mark visibility of state
RouterV2.sol:94:29: Error: Avoid to make time-based decisions in your
business logic
RouterV2.sol:94:46: Error: Use double quotes for string literals
RouterV2.sol:98:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity
>=0.7.0)RouterV2.sol:128:47: Error: Use double quotes for string
literals
RouterV2.sol:156:37: Error: Use double quotes for string literals
```

Spoon.sol

```
Spoon.sol:60:18: Error: Parse error: missing ';' at '{'
Spoon.sol:69:18: Error: Parse error: missing ';' at '{'
```

VeArtProxyUpgradeable.sol

```
VeArtProxyUpgradeable.sol:2:1: Error: Compiler version 0.8.13 does
not satisfy the r semver requirement
VeArtProxyUpgradeable.sol:485:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
VeArtProxyUpgradeable.sol:485:19: Error: Code contains empty blocks
VeArtProxyUpgradeable.sol:487:39: Error: Visibility modifier must be
first in list of modifiers
VeArtProxyUpgradeable.sol:521:296: Error: Use double quotes for
string literals
VeArtProxyUpgradeable.sol:522:42: Error: Use double quotes for string
literals
```

VoterV2_1.sol

```
VoterV2_1.sol:22:73: Error: Parse error: missing ';' at '{'
```

VotingEscrow.sol

```
VotingEscrow.sol:2:1: Error: Compiler version 0.8.13 does not satisfy
the r semver requirementVotingEscrow.sol:56:56: Error: Variable name
must be in mixedCase
VotingEscrow.sol:309:1: Error: Contract has 26 states declarations
but allowed no more than 15VotingEscrow.sol:364:35: Error: Variable
name must be in mixedCase
VotingEscrow.sol:383:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity
>=0.7.0)VotingEscrow.sol:383:37: Error: Variable name must be in
mixedCase
VotingEscrow.sol:390:31: Error: Avoid to make time-based decisions in
your business logic
VotingEscrow.sol:1410:59: Error: Use double quotes for string
literalsVotingEscrow.sol:1713:13: Error: Avoid to make time-based
decisions in your business logic
```

BribeFactoryV2.sol

```
BribeFactoryV2.sol:24:73: Error: Parse error: missing ';' at '{'
BribeFactoryV2.sol:1998:18: Error: Parse error: missing ';' at '{'
```

GaugeFactoryV2.sol

```
GaugeFactoryV2.sol:23:73: Error: Parse error: missing ';' at '{'
GaugeFactoryV2.sol:86:18: Error: Parse error: missing ';' at '{'
GaugeFactoryV2.sol:99:18: Error: Parse error: missing ';' at '{'
GaugeFactoryV2.sol:111:18: Error: Parse error: missing ';' at '{'
GaugeFactoryV2.sol:128:18: Error: Parse error: missing ';' at '{'
GaugeFactoryV2.sol:140:18: Error: Parse error: missing ';' at '{'
GaugeFactoryV2.sol:232:18: Error: Parse error: missing ';' at '{'
GaugeFactoryV2.sol:251:18: Error: Parse error: missing ';' at '{'
GaugeFactoryV2.sol:273:18: Error: Parse error: missing ';' at '{'
GaugeFactoryV2.sol:627:18: Error: Parse error: missing ';' at '{'
```

PairFactory.sol

```
PairFactory.sol:24:73: Error: Parse error: missing ';' at '{'
PairFactory.sol:556:22: Error: Parse error: missing ';' at '{'
```

PairFactoryUpgradeable.sol

```
PairFactoryUpgradeable.sol:24:73: Error: Parse error: missing ';' at
'{'
PairFactoryUpgradeable.sol:559:22: Error: Parse error: missing ';' at
'{'
```

Software analysis result:

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

Email: audit@EtherAuthority.io