Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: BloomBox Protocol
Platform: AVAX Network
Language: Solidity
Date: June 23rd, 2022

Table of contents

Email: audit@EtherAuthority.io

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

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by BloomBox to perform the Security audit of the BloomBox Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 23rd, 2022.

The purpose of this audit was to address the following:

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

Project Background

The BloomBox Protocol is a Defi Program in which custom nodes can be created with customization and approved users or the owner of the bloom node can start auto compounding for the blooms by swapping their USDC.e tokens for some time period. It has functions like claim, airdrop, deposit, initialize, withdraw, setURI, burn, mint, mintBatch, addLiquidity, toggleSwap, liquidityReward, etc. The Bloomify contracts are ERC1155 smart contracts with treasury functionality. These contracts inherits the ERC721Upgradeable, OwnableUpgradeable, ReentrancyGuardUpgradeable, PausableUpgradeable, IERC20, SafeERC20, Ownable, Initializable, ERC20Upgradeable, IERC20Upgradeable, ERC20BurnableUpgradeable, SafeMathUpgradeable, ERC1155URIStorageUpgradeable standard smart contracts from the OpenZeppelin library. These OpenZeppelin contracts are considered community-audited and time-tested, and hence are not part of the audit scope.

Audit scope

Name	**Code Review and Security Analysis Report for BloomBox Protocol Smart Contracts**
Platform	**AVAX / Solidity**
File 1	BloomNFT.sol
File 1 MD5 Hash	AB3989A8FB12421E22821BEFA5845ECC

File 2	BloomsManagerUpgradeable.sol
File 2 MD5 Hash	3597608CA88FA5568F3E40CC00A7596E
File 3	LiquidityPoolManager.sol
File 3 MD5 Hash	9A5856B7996B214CFAECF31A5E8BA4C4
File 4	Nectar.sol
File 4 MD5 Hash	EA4F4C322AE91A3AB7EB975B3C46B6DB
File 5	OwnerRecovery.sol
File 5 MD5 Hash	8B0989C2653FEA5D3011C8A3F910E58C
File 6	OwnerRecoveryUpgradeable.sol
File 6 MD5 Hash	BB367961A2F5B2AF9952A9809FE244A5
File 7	BloomTiers.sol
File 7 MD5 Hash	63BDA048D7B6B887A8977A4A2D79A988
File 8	WalletObserverUpgradeable.sol
File 8 MD5 Hash	C3C5EDA54A383AE2327C807C33ABC4E8
File 9	Vault.sol
File 9 MD5 Hash	56BAD1A62944E50C5712D5A5410AE1EB
File 10	Bloomify-TreasuryUpgradeable.sol
File 10 MD5 Hash	a360f615bce3faf0367bad8042ff3dba
File 11	Bloomify-FlowerUpgradeable.sol
File 11 MD5 Hash	5ec7e70e33cb5cfd31757e60e5afee18
File 12	Bloomify-WhitelistUpgradeable.sol
File 12 MD5 Hash	a3d5598955989f8b190e0d2ec4216e98
Audit Date	June 23rd,2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 BloomNFT.sol • Name: BloomNFT • Symbol: Bloom	**YES, This is valid.**
File 2 BloomsManagerUpgradeable.sol • Standard Fee: 10 • Precision: 100 • Compound Delay: 24 hours • Creation Minimum Price: 52000	**YES, This is valid.**
File 3 LiquidityPoolManager.sol • LiquidityPoolManager has functions like: afterTokenTransfer, swapAndLiquify, etc.	**YES, This is valid.**
File 4 Nectar.sol • Tax on Transfers: 10%	**YES, This is valid.**
File 5 OwnerRecovery.sol • OwnerRecovery has functions like: recoverLostAVAX, recoverLostTokens.	**YES, This is valid.**
File 6 OwnerRecoveryUpgradeable.sol • OwnerRecoveryUpgradeable has functions like: recoverLostAVAX, recoverLostTokens.	**YES, This is valid.**
File 7 WalletObserverUpgradeable.sol • WalletObserverUpgradeable has functions like: changeNectarImplementation, changeLiquidityPoolManagerImplementation, etc.	**YES, This is valid.**
File 8 Vault.sol • Vault has functions like: initialize, withdraw, etc.	**YES, This is valid.**

File 9 BloomTiers.sol • BloomTiers has functions like: mint, mintBatch, etc.	**YES, This is valid.**
File 10 TreasuryUpgradeable.sol • TreasuryUpgradeable whitelisted users can withdraw the desired amount of NCTR from the Treasury to the desired address.	**YES, This is valid.**
File 11 FlowerUpgradeable.sol • Maximum Perc: 100 • Minimum Number Of Refer For Team Wallet: 5 • Minimum Tier Level: 1 • Maximum Tier Level: 15 • Number Of Tiers: 16 • Deposit Tax: 10% • Deposit Burn Percentage Nctr: 20% • Deposit Flower Percentage Nctr: 60% • Deposit Lp Percentage Nctr: 20% • Deposit Lp Percentage Usdce: 20% • Deposit Treasury Percusdce: 80% • Compound Tax: 10% • Compound Burn Percentage Nctr: 50% • Compound Upline Percentage Nctr: 45% • Compound Upline Percentage Usdce: 5% • Claim Tax: 10% • Team Wallet Downline Reward Percentage: 25%	**YES, This is valid.**
File 12 WhitelistUpgradeable.sol • The WhitelistUpgradeable owner can add and remove addresses to the whitelist.	**YES, This is valid.**

Audit Summary

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

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

We found 0 critical, 0 high, 0 medium and 3 low and some very low level issues.

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

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Moderated
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	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	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Moderated
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 14 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 BloomBox Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the BloomBox Protocol.

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

Some code parts are well commented on smart contracts. We suggest using Ethereum's NatSpec style for the commenting.

Documentation

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

As mentioned above, code parts are well commented. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

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

BloomNFT.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	__Ownable_init	internal	Passed	No Issue
7	__Ownable_init_unchained	internal	Passed	No Issue
8	onlyOwner	modifier	Passed	No Issue
9	owner	read	Passed	No Issue
10	_checkOwner	internal	Passed	No Issue
11	renounceOwnership	write	access only Owner	No Issue
12	transferOwnership	write	access only Owner	No Issue
13	_transferOwnership	internal	Passed	No Issue
14	__ERC721_init	internal	Passed	No Issue
15	__ERC721_init_unchained	internal	Passed	No Issue
16	supportsInterface	read	Passed	No Issue
17	balanceOf	read	Passed	No Issue
18	ownerOf	read	Passed	No Issue
19	name	read	Passed	No Issue
20	symbol	read	Passed	No Issue
21	tokenURI	read	Passed	No Issue
22	_baseURI	internal	Passed	No Issue
23	approve	write	Passed	No Issue
24	getApproved	read	Passed	No Issue
25	setApprovalForAll	write	Passed	No Issue
26	isApprovedForAll	read	Passed	No Issue
27	transferFrom	write	Passed	No Issue
28	safeTransferFrom	write	Passed	No Issue
29	safeTransferFrom	write	Passed	No Issue
30	_safeTransfer	internal	Passed	No Issue
31	_exists	internal	Passed	No Issue
32	_isApprovedOrOwner	internal	Passed	No Issue
33	_safeMint	internal	Passed	No Issue
34	_safeMint	internal	Passed	No Issue
35	_mint	internal	Passed	No Issue
36	_burn	internal	Passed	No Issue
37	_transfer	internal	Passed	No Issue
38	_approve	internal	Passed	No Issue

39	_setApprovalForAll	internal	Passed	No Issue
40	_requireMinted	internal	Passed	No Issue
41	_checkOnERC721Received	write	Passed	No Issue
42	_beforeTokenTransfer	internal	Passed	No Issue
43	_afterTokenTransfer	internal	Passed	No Issue
44	onlyBloomManagerOrOwner	modifier	Passed	No Issue
45	initialize	external	Passed	No Issue
46	mintBloom	external	access only Bloom Manager Or Owner	No Issue
47	burnBloom	external	access only Bloom Manager Or Owner	No Issue
48	isApprovedOrOwner	external	Passed	No Issue
49	setBloomNodes	external	access only Owner	No Issue
50	setBaseURI	external	access only Owner	No Issue
51	_baseURI	internal	Passed	No Issue

BloomsManagerUpgradeable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	__Ownable_init	internal	Passed	No Issue
7	__Ownable_init_unchained	internal	Passed	No Issue
8	onlyOwner	modifier	Passed	No Issue
9	owner	read	Passed	No Issue
10	_checkOwner	internal	Passed	No Issue
11	renounceOwnership	write	access only Owner	No Issue
12	transferOwnership	write	access only Owner	No Issue
13	_transferOwnership	internal	Passed	No Issue
14	__Pausable_init	internal	Passed	No Issue
15	__Pausable_init_unchained	internal	Passed	No Issue
16	whenPaused	modifier	Passed	No Issue
17	whenNotPaused	modifier	Passed	No Issue
18	paused	read	Passed	No Issue
19	_requireNotPaused	internal	Passed	No Issue
20	_requirePaused	internal	Passed	No Issue
21	_pause	internal	Passed	No Issue
22	_unpause	internal	Passed	No Issue

23	onlyNectar	modifier	Passed	No Issue
24	getNectarImplementation	read	Passed	No Issue
25	changeNectarImplementation	write	access only Owner	No Issue
26	onlyBloomOwner	modifier	Passed	No Issue
27	onlyApprovedOrOwnerOfBloom	modifier	Passed	No Issue
28	onlyValidName	modifier	Passed	No Issue
29	initialize	external	Passed	No Issue
30	renameBloom	external	access only Bloom Manager Or Owner	No Issue
31	createBloomWithTokens	external	access only Valid Name	No Issue
32	addValue	external	access only Bloom Manager Or Owner	No Issue
33	startAutoCompounding	external	access only Bloom Manager Or Owner	No Issue
34	autoCompound	external	access only Owner	No Issue
35	autoClaim	external	access only Owner	No Issue
36	emergencyClaim	external	access only Bloom Manager Or Owner	No Issue
37	burn	external	access only Bloom Manager Or Owner	No Issue
38	setNodeMinPrice	external	access only Owner	No Issue
39	setCompoundDelay	external	access only Owner	No Issue
40	setRewardPerDay	external	access only Owner	No Issue
41	changeTierSystem	external	access only Owner	No Issue
42	getBloomsByIds	external	Passed	No Issue
43	calculateTotalDailyEmission	external	Passed	No Issue
44	_emergencyReward	write	Passed	No Issue
45	_getRewardsAndCompound	write	Passed	No Issue
46	_cashoutReward	write	Passed	No Issue
47	_logTier	write	Passed	No Issue
48	_checkMultiplier	read	Passed	No Issue
49	_isProcessable	read	Passed	No Issue
50	_calculateReward	read	Passed	No Issue
51	_rewardPerDayFor	read	Passed	No Issue
52	_bloomExists	read	Passed	No Issue
53	_isOwnerOfBlooms	read	Passed	No Issue
54	_isApprovedOrOwnerOfBloom	read	Passed	No Issue
55	_swapAndBurn	write	Passed	No Issue
56	_routerSwap	write	Passed	No Issue
57	_routerAddLiquidity	write	Passed	No Issue
58	_deposit	write	Passed	No Issue
59	_whitelistedDeposit	write	Passed	No Issue

60	_nonWhitelistedDeposit	write	Passed	No Issue
61	_unsubscribeNodeFromAutoCompounding	write	Passed	No Issue
62	_removeNodeFromClaimable	write	Passed	No Issue
63	_updateEmergencyStatus	write	Passed	No Issue
64	_autoclaimRewards	write	Passed	No Issue
65	_resetRewardMultiplier	write	Passed	No Issue
66	_burn	internal	Passed	No Issue

LiquidityPoolManager.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	recoverLostAVAX	external	access only Owner	No Issue
8	recoverLostTokens	external	access only Owner	No Issue
9	onlyNectar	modifier	Passed	No Issue
10	getNectarImplementation	read	Passed	No Issue
11	changeNectarImplementation	write	access only Owner	No Issue
12	initializeManager	external	Passed	No Issue
13	afterTokenTransfer	external	access only Nectar	No Issue
14	isLiquidityAdded	external	Passed	No Issue
15	isLiquidityRemoved	external	Passed	No Issue
16	swapAndLiquify	write	Passed	No Issue
17	sendLPTokensTo	write	Passed	No Issue
18	createPairWith	write	Passed	No Issue
19	swapLeftSideForRightSide	write	Passed	No Issue
20	addLiquidityToken	write	Passed	No Issue
21	getRouter	external	Passed	No Issue
22	getPair	external	Passed	No Issue
23	getLeftSide	external	Passed	No Issue
24	getRightSide	external	Passed	No Issue
25	isPair	read	Passed	No Issue
26	isFeeReceiver	external	Passed	No Issue
27	isRouter	read	Passed	No Issue
28	getFeeAddresses	external	Passed	No Issue
29	getFeePercentages	external	Passed	No Issue
30	setAllowance	write	access only Owner	No Issue

31	shouldLiquify	write	access only Owner	No Issue
32	updateSwapTokensToLiquidityThreshold	write	access only Owner	No Issue
33	feesForwarder	write	access only Owner	No Issue

Nectar.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	Passed	No Issue
3	__Ownable_init_unchained	internal	Passed	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	__ERC20Burnable_init	internal	Passed	No Issue
11	__ERC20Burnable_init_unchained	internal	Passed	No Issue
12	burn	write	Passed	No Issue
13	burnFrom	write	Passed	No Issue
14	recoverLostAVAX	external	access only Owner	No Issue
15	recoverLostTokens	external	access only Owner	No Issue
16	onlyWalletObserver	modifier	Passed	No Issue
17	getWalletObserverImplementation	read	Passed	No Issue
18	changeWalletObserverImplementation	write	access only Owner	No Issue
19	canMint	modifier	Passed	No Issue
20	onlyFlowerManager	modifier	Passed	No Issue
21	initialize	external	Passed	No Issue
22	_beforeTokenTransfer	internal	Passed	No Issue
23	calculateTransactionTax	internal	Passed	No Issue
24	calculateTransferTaxes	read	Passed	No Issue
25	_afterTokenTransfer	internal	Passed	No Issue
26	burnNectar	external	access only Flower Manager	No Issue
27	mintNectar	external	access only Flower Manager	No Issue
28	liquidityReward	external	access only Flower Manager	No Issue
29	toggleSwap	write	Unused functions	Refer audit findings
30	mint	write	Passed	No Issue

31	finishMinting	write	access only Flower Manager	No Issue
32	setVaultAddress	write	access only Owner	No Issue
33	setDevWallet	external	access only Owner	No Issue
34	setFlowerManager	external	access only Flower Manager	No Issue
35	canMint	modifier	Passed	No Issue
36	onlyFlowerManager	modifier	Passed	No Issue
37	initialize	external	Passed	No Issue
38	_beforeTokenTransfer	internal	Passed	No Issue
39	calculateTransactionTax	internal	Passed	No Issue
40	calculateTransferTaxes	read	Passed	No Issue
41	_afterTokenTransfer	internal	Passed	No Issue
42	burnNectar	external	access only Flower Manager	No Issue
43	mintNectar	external	access only Flower Manager	No Issue
44	liquidityReward	external	access only Flower Manager	No Issue
45	toggleSwap	write	access only Owner	No Issue
46	mint	write	Condition has been checked twice	Refer audit findings
47	finishMinting	write	access only Flower Manager	No Issue
48	setVaultAddress	write	Function input parameters lack of check	Refer audit findings
49	setDevWallet	external	Function input parameters lack of check, Unused functions	Refer audit findings
50	setFlowerManager	external	Function input parameters lack of check	Refer audit findings

OwnerRecovery.sol

Functions

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

6	_transferOwnership	internal	Passed	No Issue
7	recoverLostAVAX	external	access only Owner	No Issue
8	recoverLostTokens	external	access only Owner	No Issue

OwnerRecoveryUpgradeable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	Passed	No Issue
3	__Ownable_init_unchained	internal	Passed	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	recoverLostAVAX	external	access only Owner	No Issue
11	recoverLostTokens	external	access only Owner	No Issue

BloomTiers.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	__Ownable_init	internal	Passed	No Issue
7	__Ownable_init_unchained	internal	Passed	No Issue
8	onlyOwner	modifier	Passed	No Issue
9	owner	read	Passed	No Issue
10	_checkOwner	internal	Passed	No Issue
11	renounceOwnership	write	access only Owner	No Issue
12	transferOwnership	write	access only Owner	No Issue
13	_transferOwnership	internal	Passed	No Issue
14	__ERC1155URIStorage_init	internal	Passed	No Issue
15	__ERC1155URIStorage_init_unchained	internal	Passed	No Issue
16	uri	read	Passed	No Issue
17	_setURI	internal	Passed	No Issue
18	_setBaseURI	internal	Passed	No Issue

19	onlyBloomReferral	modifier	Passed	No Issue
20	onlyExistingId	modifier	Passed	No Issue
21	initialize	external	Passed	No Issue
22	mint	external	onlyBloomReferral	No Issue
23	mintBatch	external	access only Owner	No Issue
24	burn	external	onlyExistingId	No Issue
25	burnBatch	external	access only Owner	No Issue
26	setBloomReferral	external	access only Owner	No Issue
27	setBaseURI	external	access only Owner	No Issue
28	setURI	external	access only Owner	No Issue
29	_beforeTokenTransfer	write	Passed	No Issue

WalletObserverUpgradeable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	__Ownable_init	internal	Passed	No Issue
7	__Ownable_init_unchained	internal	Passed	No Issue
8	onlyOwner	modifier	Passed	No Issue
9	owner	read	Passed	No Issue
10	_checkOwner	internal	Passed	No Issue
11	renounceOwnership	write	access only Owner	No Issue
12	transferOwnership	write	access only Owner	No Issue
13	_transferOwnership	internal	Passed	No Issue
14	recoverLostAVAX	external	access only Owner	No Issue
15	recoverLostTokens	external	access only Owner	No Issue
16	onlyNectar	modifier	Passed	No Issue
17	getNectarImplementation	read	Passed	No Issue
18	changeNectarImplementation	write	access only Owner	No Issue
19	onlyLiquidityPoolManager	modifier	Passed	No Issue
20	getLiquidityPoolManagerImplementation	read	Passed	No Issue
21	changeLiquidityPoolManagerImplementation	write	access only Owner	No Issue
22	initialize	external	Passed	No Issue
23	checkTimeframe	modifier	Passed	No Issue
24	isNotDenied	modifier	Passed	No Issue
25	changeNectarImplementation	write	access only Owner	No Issue

26	changeLiquidityPoolManagerImplementation	write	access only Owner	No Issue
27	isPair	internal	Passed	No Issue
28	beforeTokenTransfer	external	access only Nectar	No Issue
29	getMaxTokenPerWallet	read	Passed	No Issue
30	getTimeframeExpiresAfter	external	Passed	No Issue
31	getTimeframeCurrent	external	Passed	No Issue
32	getRemainingTransfersOut	read	Passed	No Issue
33	getRemainingTransfersOutWithSellAllowance	read	Passed	No Issue
34	getRemainingTransfersIn	read	Passed	No Issue
35	getOverviewOf	external	Passed	No Issue
36	getBoughtTokensOf	read	Passed	No Issue
37	isWalletFull	read	Passed	No Issue
38	isExcludedFromObserver	read	Passed	No Issue
39	setMaxTokenPerWalletPercent	write	Percentage limit is not set	Refer audit findings
40	resetBoughtTokensOf	external	access only Owner	No Issue
41	setTimeframeExpiresAfter	write	access only Owner	No Issue
42	setTimeframeQuotaIn	write	access only Owner	No Issue
43	setTimeframeQuotaOut	write	access only Owner	No Issue
44	denyMalicious	external	access only Owner	No Issue
45	excludeFromObserver	external	access only Owner	No Issue
46	totalSupply	external	Passed	No Issue

Vault.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initializer	modifier	Passed	No Issue
3	reinitializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_disableInitializers	internal	Passed	No Issue
6	initialize	external	Function input parameters lack of check	Refer audit findings
7	withdraw	write	Function input parameters lack of check	Refer audit findings

TreasuryUpgradeable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyWhitelisted	modifier	Passed	No Issue
3	__Whitelist_init	internal	Passed	No Issue
4	addAddressesToWhitelist	external	access only Owner	No Issue
5	removeAddressesFromWhitelist	external	access only Owner	No Issue
6	initialize	external	Passed	No Issue
7	withdrawNCTR	external	access only Whitelisted	No Issue
8	withdrawUSDCe	external	access only Whitelisted	No Issue

FlowerUpgradeable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	Passed	No Issue
3	__Ownable_init_unchained	internal	Passed	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	external	Passed	No Issue
11	updateDepositTax	external	access only Owner	No Issue
12	updateDepositDistributionPercentages	external	access only Owner	No Issue
13	updateCompoundTax	external	access only Owner	No Issue
14	updateCompoundDistributionPercentages	external	access only Owner	No Issue
15	updateClaimTax	external	access only Owner	No Issue
16	updateTeamWalletDownlineRewardPerc	external	access only Owner	No Issue
17	onlyBloomReferralNode	modifier	Passed	No Issue
18	_calculatePercentagePart	write	Passed	No Issue
19	_calculateTax	write	Passed	No Issue
20	_getTierLevel	read	Passed	No Issue
21	_getRewardEligibility	read	Passed	No Issue
22	_routerSwap	write	Passed	No Issue
23	_routerAddLiquidity	write	Passed	No Issue
24	creditsAndDebits	read	Passed	No Issue
25	isNetPositive	read	Passed	No Issue

26	setUSDCeWallet	external	Passed	No Issue
27	deposit	external	Passed	No Issue
28	compoundRewards	external	access only Bloom Referral Node	No Issue
29	claim	external	access only Bloom Referral Node	No Issue
30	airdrop	external	access only Bloom Referral Node	No Issue

WhitelistUpgradeable.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	__Ownable_init	internal	Passed	No Issue
3	__Ownable_init_unchained	internal	Passed	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	__Whitelist_init	internal	access by initializer	No Issue
11	onlyWhitelisted	modifier	Passed	No Issue
12	addAddressesToWhitelist	external	access only Owner	No Issue
13	removeAddressesFromWhitelist	external	access only Owner	No Issue

Severity Definitions

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

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Function input parameters lack of check:

Variable validation is not performed in below functions:

Nectar.sol

- setVaultAddress = _newVaultAddress
- setDevWallet = devWallet_
- setFlowerManager = _newFlowerManager

Vault.sol

- withdraw = _amount
- initialize = _nectar , _whitelist

Resolution: We advise to put validation : int type variables should not be empty and > 0 & address type variables should not be address(0).

(2) Percentage limit is not set:-

WalletObserverUpgradeable.sol

```
function setMaxTokenPerWalletPercent(uint8 _maxTokenPerWalletPercent)
    public
    onlyOwner
{
    require(
        _maxTokenPerWalletPercent > 0,
        "WalletObserverUpgradeable: Max token per wallet percentage cannot be 0"
    );

    // Modifying this with a lower value won't brick wallets
    // It will just prevent transferring / buys to be made for them
    maxTokenPerWalletPercent = _maxTokenPerWalletPercent;
    require(
        getMaxTokenPerWallet() >= timeframeQuotaIn,
        "WalletObserverUpgradeable: Max token per wallet must be above or equal to timeframeQuotaIn"
    );
}
```

The maxTokenPerWalletPercent 's max limit is not set. Owner can set it up to any number.

LiquidityPoolManager.sol

```
function feesForwarder(
    address[] memory _feeAddresses,
    uint8[] memory _feePercentages
) public onlyOwner {
    require(
        _feeAddresses.length > 0,
        "LiquidityPoolManager: Addresses array length must be greater than zero"
    );
    require(
        _feeAddresses.length == _feePercentages.length + 1,
        "LiquidityPoolManager: Addresses arrays length mismatch. Remember last address receive the remains."
    );
    feeAddresses = _feeAddresses;
    feePercentages = _feePercentages;
}
```

Owner can set the individual fee percentage to any number.

Resolution: We suggest adding a percentage max limit.

(3) Uninitialized variable:- **Nectar.sol**

```
mapping(address => uint8) private _customTaxRate;
mapping(address => bool) private _hasCustomTax;
mapping(address => bool) private _isExcluded;
```

These _customTaxRate ,_hasCustomTax , _isExcluded variables are used in contract but not Initialized.

Resolution: We suggest initializing variables with values.

Very Low / Informational / Best practices:

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

FlowerManager can mint unlimited tokens.

Resolution: We suggest putting a minting limit.

(2) Unused variables / event / functions:**- Nectar.sol**

Functions:

- toggleSwap
- setDevWallet

Events:

- SetTransferFee
- SetSellFee

Variables:

- devWallet
- swapEnabled
- _excluded

Resolution: Remove unused variables / events / functions from the code.

(3) Condition has been checked twice:- **Nectar.sol**

```
function mint(address _to, uint256 _amount)
    public
    onlyFlowerManager
    canMint
    nonReentrant
    returns (bool)
{
    require(!mintingFinished, "Minting is finished");
    //Never fail, just don't mint if over
    if (_amount == 0 || mintedSupply_.add(_amount) > TARGET_SUPPLY) {
        return false;
    }
```

In the mint function , canMint modifier is used and also requires which checks minting finished or not.

Resolution: We suggest removing either canMint modifier Or require to check minting is finished inside the mint function.

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:

- mintBloom: BloomNFT owner can mint bloom tokens.
- burnBloom: BloomNFT owner can burn bloom tokens.
- setBloomNodes: BloomNFT owner can set bloom nodes.
- setBaseURI: BloomNFT owner can set the base URI for Bloom nodes.
- renameBloom: BloomsManagerUpgradeable bloom owner can rename bloom.
- addValue: BloomsManagerUpgradeable bloom owner can add new value.
- startAutoCompounding: BloomsManagerUpgradeable bloom owner can start auto compounding.
- autoCompound: BloomsManagerUpgradeable owner can auto compound.
- autoClaim: BloomsManagerUpgradeable owner can auto claim.
- emergencyClaim: BloomsManagerUpgradeable owner can authority to emergency claim.
- burn: BloomsManagerUpgradeable bloom owner can burn tokens.
- setNodeMinPrice: BloomsManagerUpgradeable owner can set node minimum price.
- setCompoundDelay: BloomsManagerUpgradeable owner can set compound delay.
- setRewardPerDay: BloomsManagerUpgradeable owner can set reward per day amount.
- changeTierSystem: BloomsManagerUpgradeable owner can change tier system.
- setAllowance: LiquidityPoolManager owner can set allowance.
- shouldLiquify: LiquidityPoolManager owner should liquify.
- updateSwapTokensToLiquidityThreshold: LiquidityPoolManager owner can update swap tokens to liquidity threshold.
- feesForwarder: LiquidityPoolManager owner can fees forwarder.
- burnNectar: Nectar FlowerManager owner can burn nectar.
- mintNectar: Nectar FlowerManager owner can mint nectar.
- liquidityReward: Nectar FlowerManager owner can liquidity reward.
- toggleSwap: Nectar FlowerManager owner can toggle swap.

- mint: Nectar FlowerManager owner can mint a token.
- finishMinting: Nectar FlowerManager owner can stop minting new tokens.
- setVaultAddress: Nectar owner can set vault address.
- setDevWallet: Nectar owner can set dev wallet address.
- setFlowerManager: Nectar FlowerManager owner can set flower manager address.
- recoverLostAVAX: OwnerRecovery owner can recover lost AVAX.
- recoverLostTokens: OwnerRecovery owner can recover lost tokens.
- recoverLostTokens: OwnerRecoveryUpgradeable owner can recover lost tokens.
- mint: BloomTiers owner can mint tokens.
- mintBatch: BloomTiers owner can mint tokens batch vise.
- burnBatch: BloomTiers owner can burn batch vise.
- setBloomReferral: BloomTiers owner can set bloomReferral address to another contract.
- setBaseURI: BloomTiers owner can set base URI.
- setURI: BloomTiers owner can set tokenURI of _tokenId to _tokenURI.
- changeNectarImplementation: WalletObserverUpgradeable owner can change nectar implementations.
- changeLiquidityPoolManagerImplementation: WalletObserverUpgradeable owner can change liquidity pool manager implementations.
- setMaxTokenPerWalletPercent: WalletObserverUpgradeable owner can set maximum tokens per wallet percentage.
- resetBoughtTokensOf: WalletObserverUpgradeable owner can reset bought tokens.
- setTimeframeExpiresAfter: WalletObserverUpgradeable owner can set time frame expires.
- setTimeframeQuotaIn: WalletObserverUpgradeable owner can set time frame quota in.
- setTimeframeQuotaOut: WalletObserverUpgradeable owner can set time frame quota out.
- denyMalicious: WalletObserverUpgradeable owner can deny malicious.
- excludeFromObserver: WalletObserverUpgradeable owner can exclude from observer.
- addAddressesToWhitelist: WhitelistUpgradeable owner can add addresses to whitelist.

- removeAddressesFromWhitelist: WhitelistUpgradeable owner can remove addresses from the whitelist.
- updateDepositTax: FlowerUpgradeable can update deposit tax with only Owner rights.
- updateDepositDistributionPercentages: FlowerUpgradeable can update deposit distribution percentages with onlyOwner rights.
- updateCompoundTax: FlowerUpgradeable can update compound tax with onlyOwner rights.
- updateCompoundDistributionPercentages: FlowerUpgradeable can update compound distribution percentages with onlyOwner rights.
- updateClaimTax: FlowerUpgradeable can update claim tax with onlyOwner rights.
- updateTeamWalletDownlineRewardPerc: FlowerUpgradeable can update reward percentage for downline.
- compoundRewards: FlowerUpgradeable Bloom Referral Node can Distribute compound rewards to the upline with Round Robin system.
- claim: FlowerUpgradeable Bloom Referral Node can claim from Bloom Treasury, calculate taxes.
- airdrop: FlowerUpgradeable Bloom Referral Node can airdrop to any address and save airdrop to Treasury.
- withdrawNCTR: TreasuryUpgradeable Whitelisted owner can withdraw desired amount of NCTR from Treasury to desired address.
- withdrawUSDCe: TreasuryUpgradeable Whitelisted owner can withdraw desired amount of USDC.e from Treasury to desired address.

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 files. And we have used all possible tests based on given objects as files. We have not observed any major issues in smart contracts. **So, the smart contracts are ready for the mainnet deployment**.

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

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

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

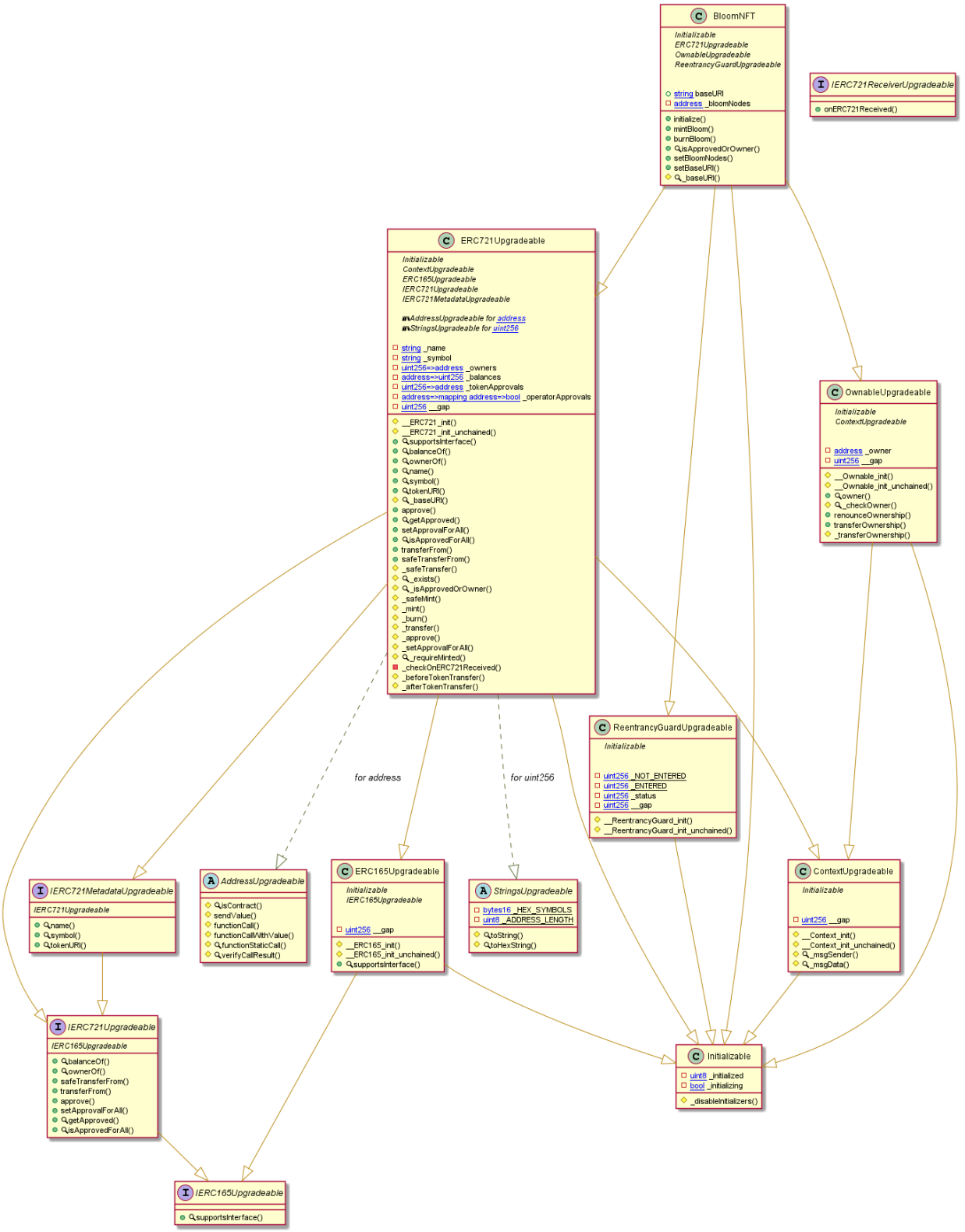
Technical Disclaimer

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

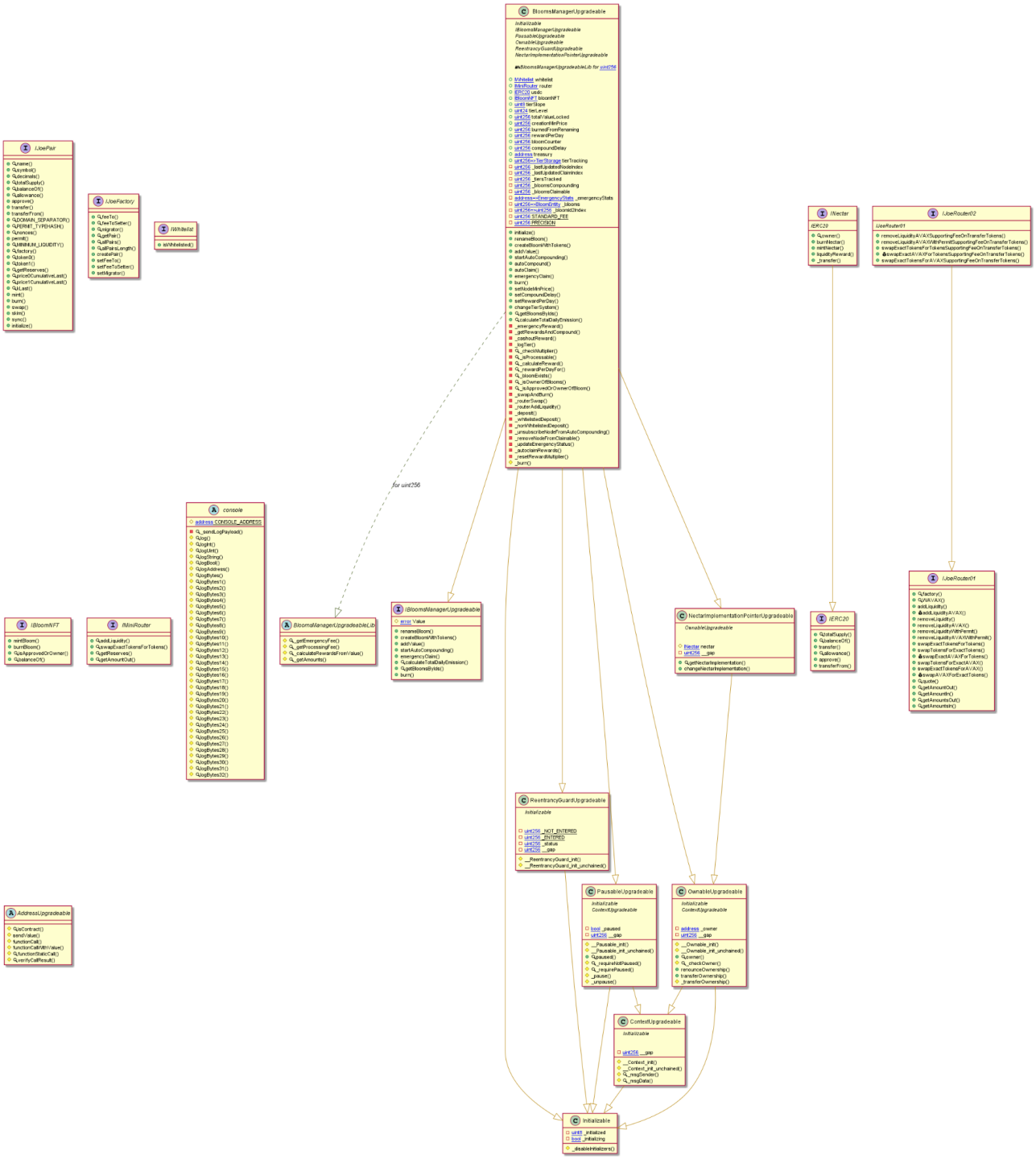
Appendix

Code Flow Diagram - BloomBox Protocol

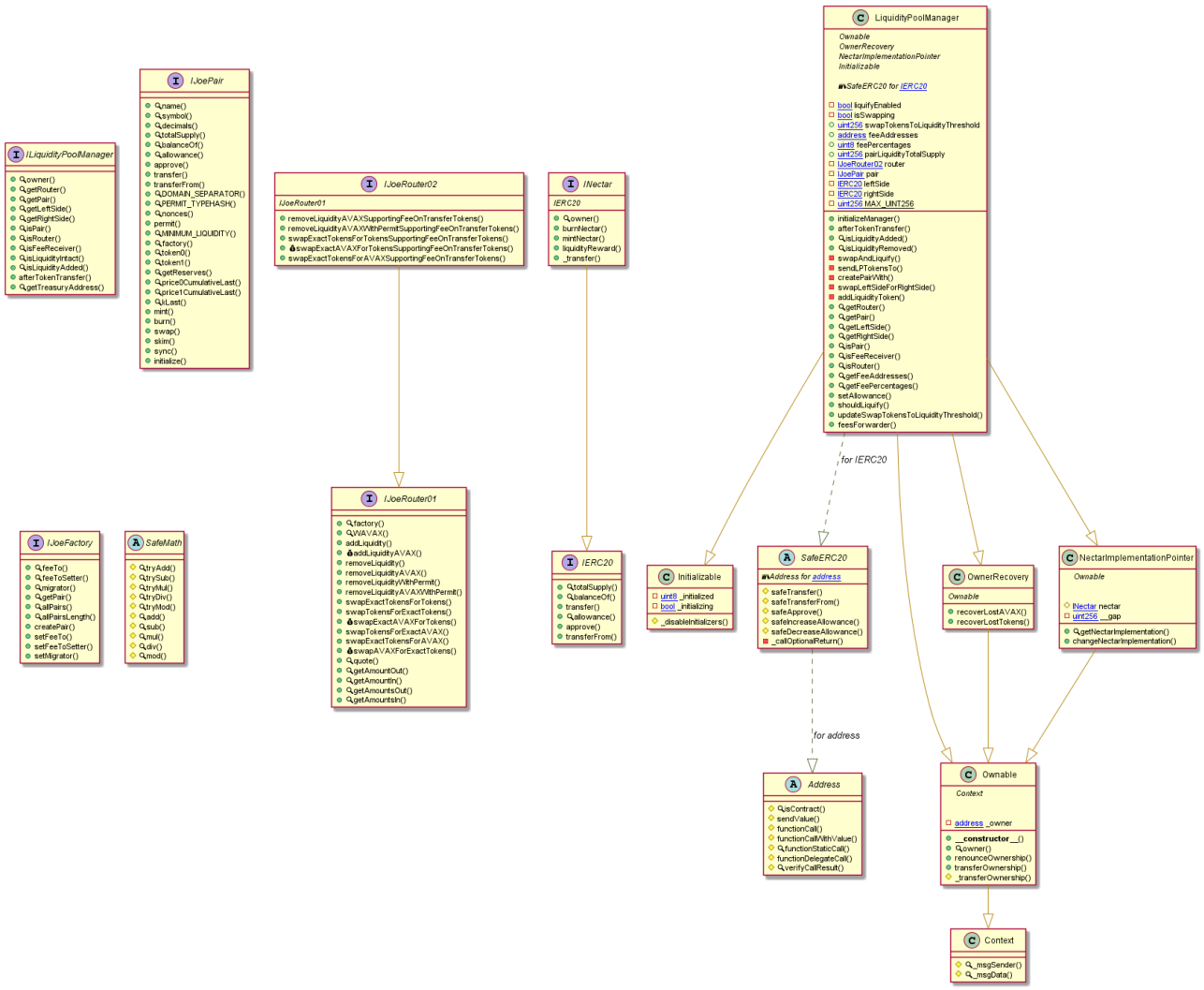
BloomNFT Diagram



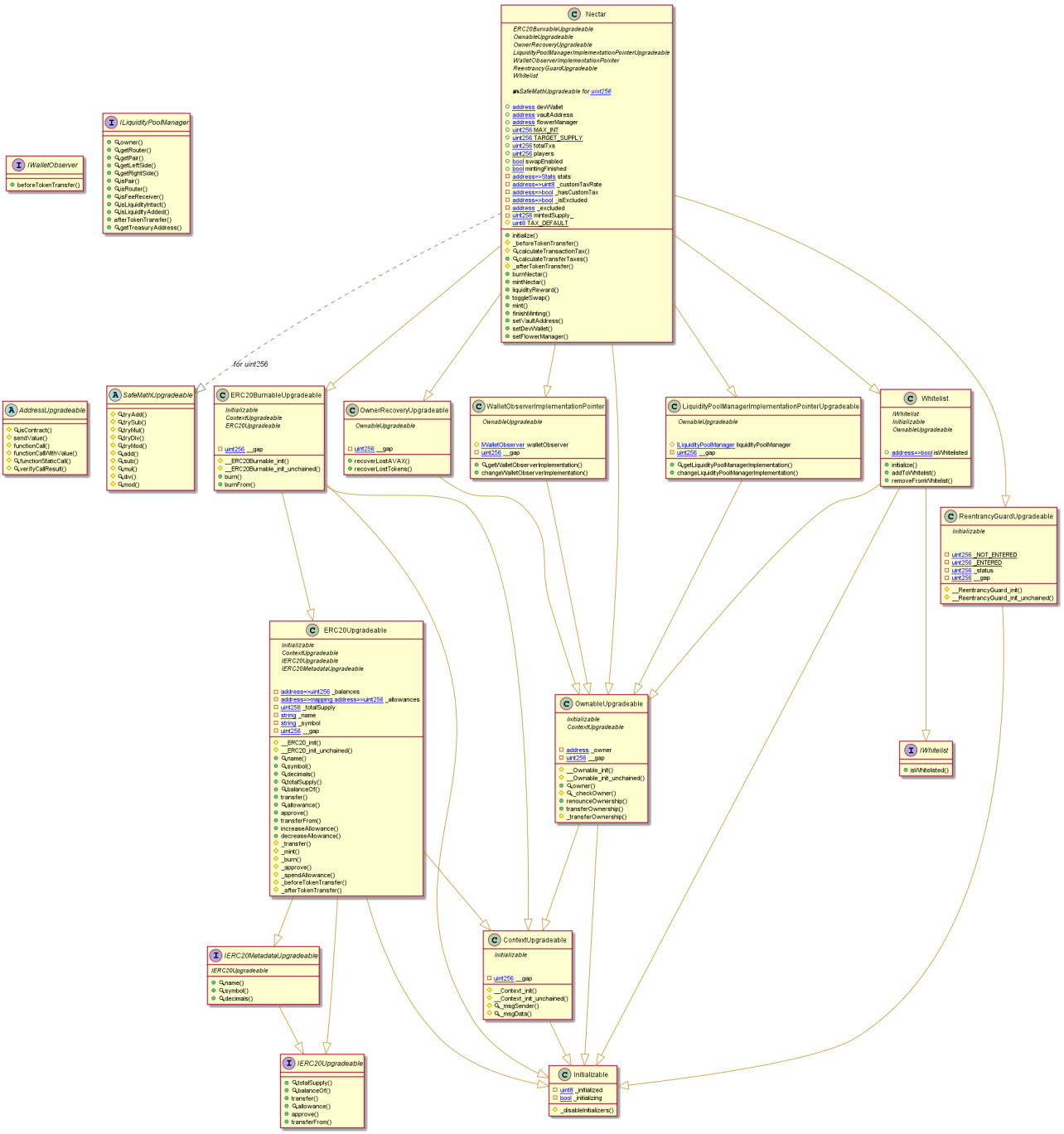
BloomsManagerUpgradeable Diagram



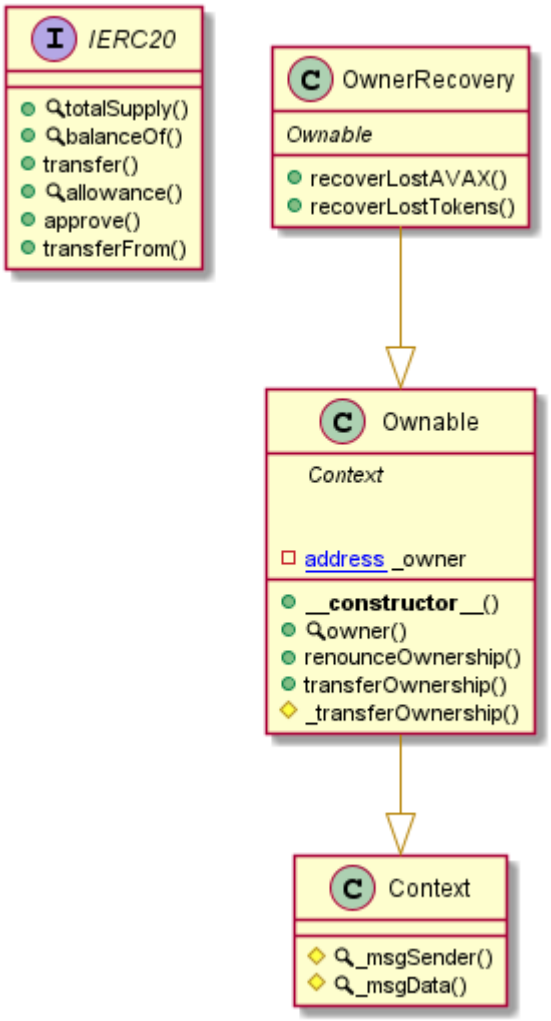
LiquidityPoolManager Diagram



Nectar Diagram

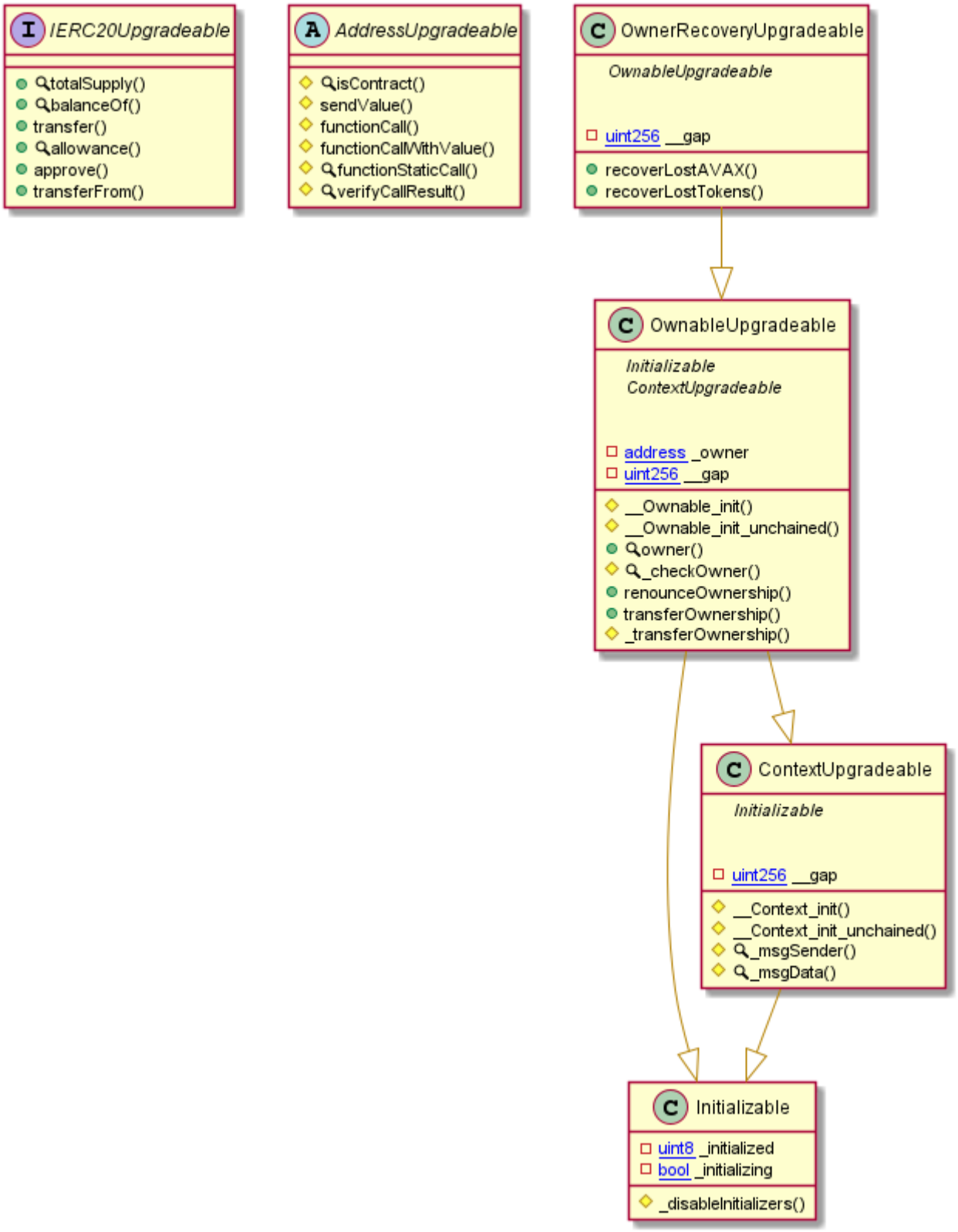


OwnerRecovery Diagram

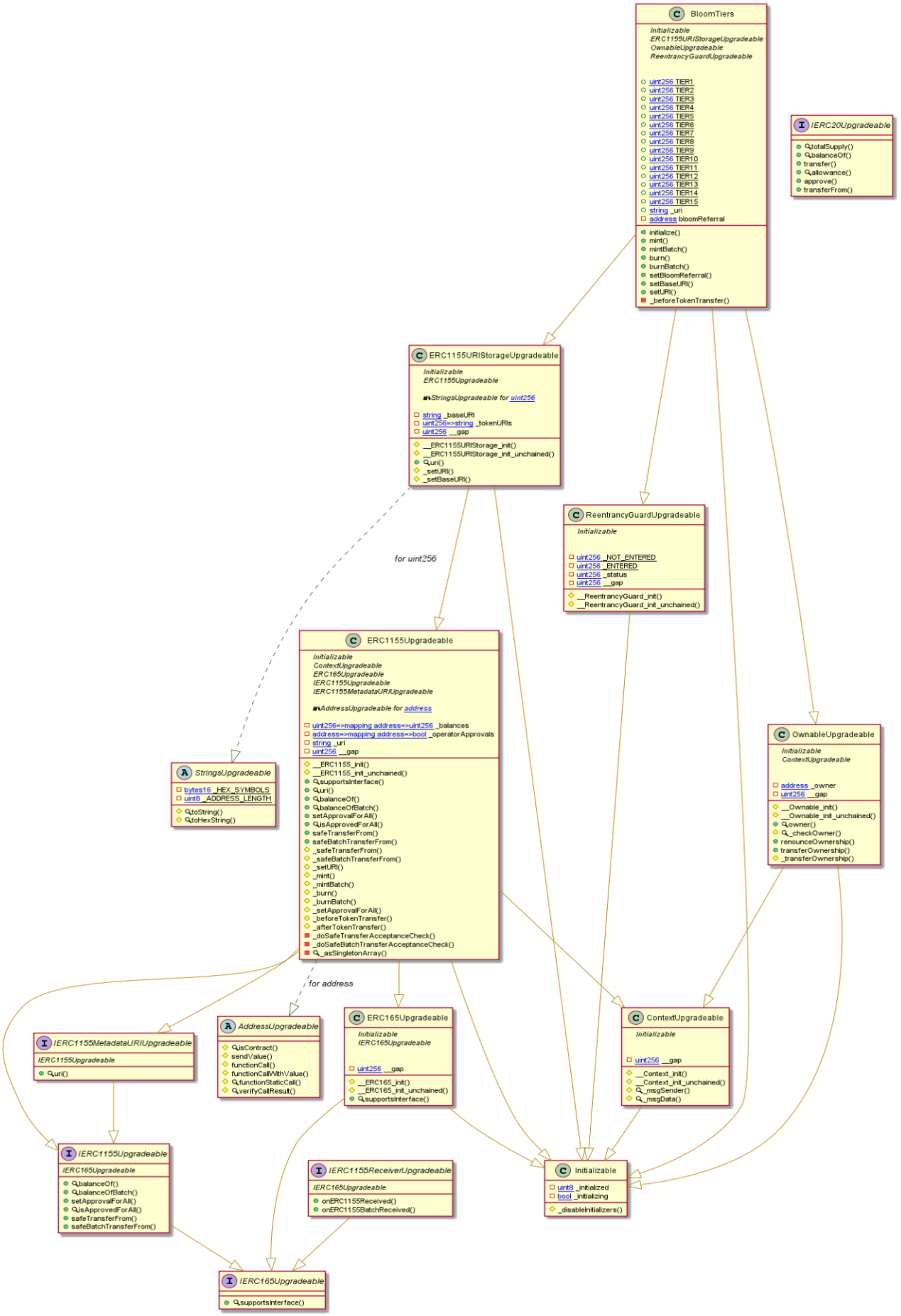


Email: audit@EtherAuthority.io

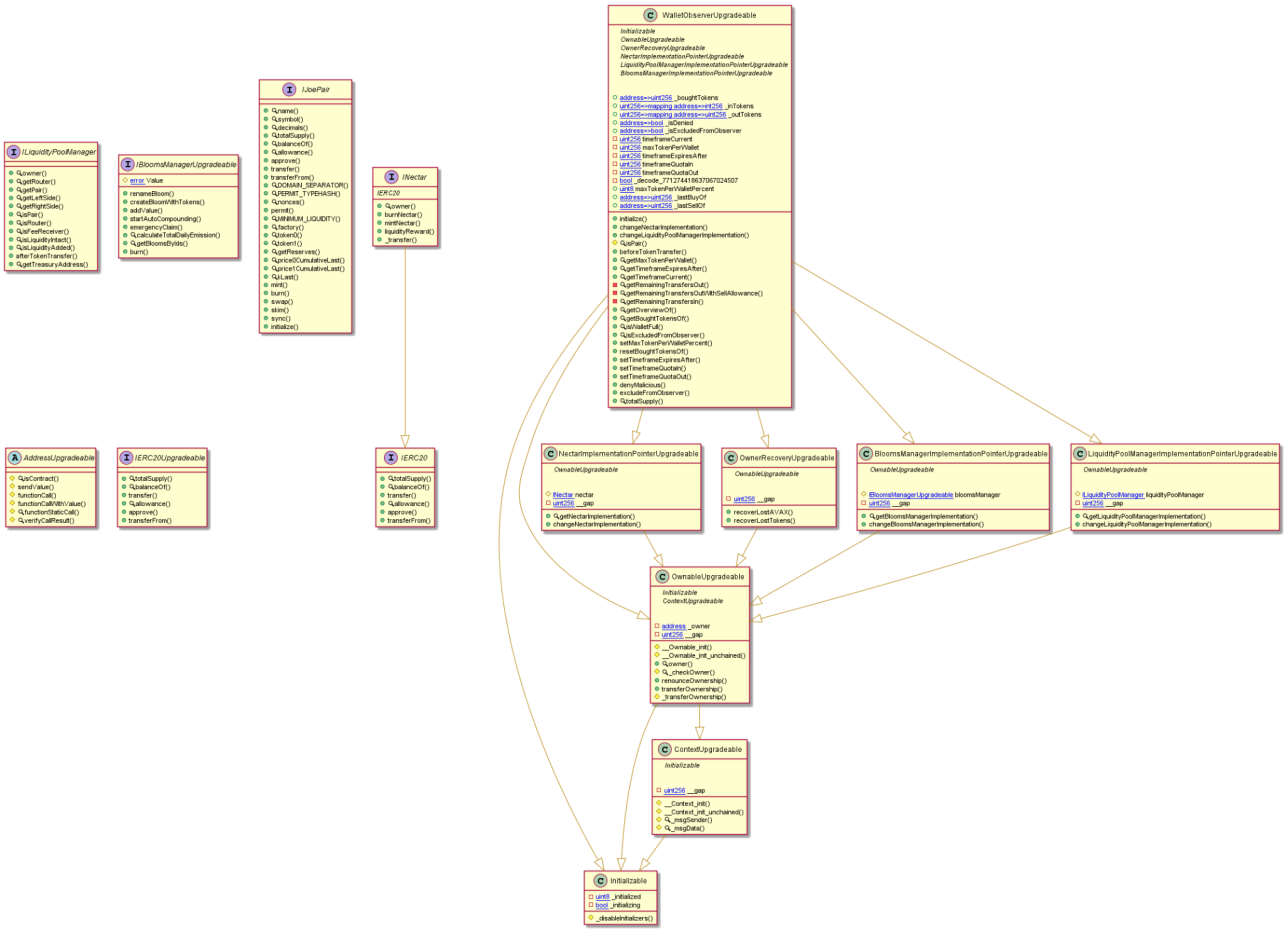
OwnerRecoveryUpgradeable Diagram



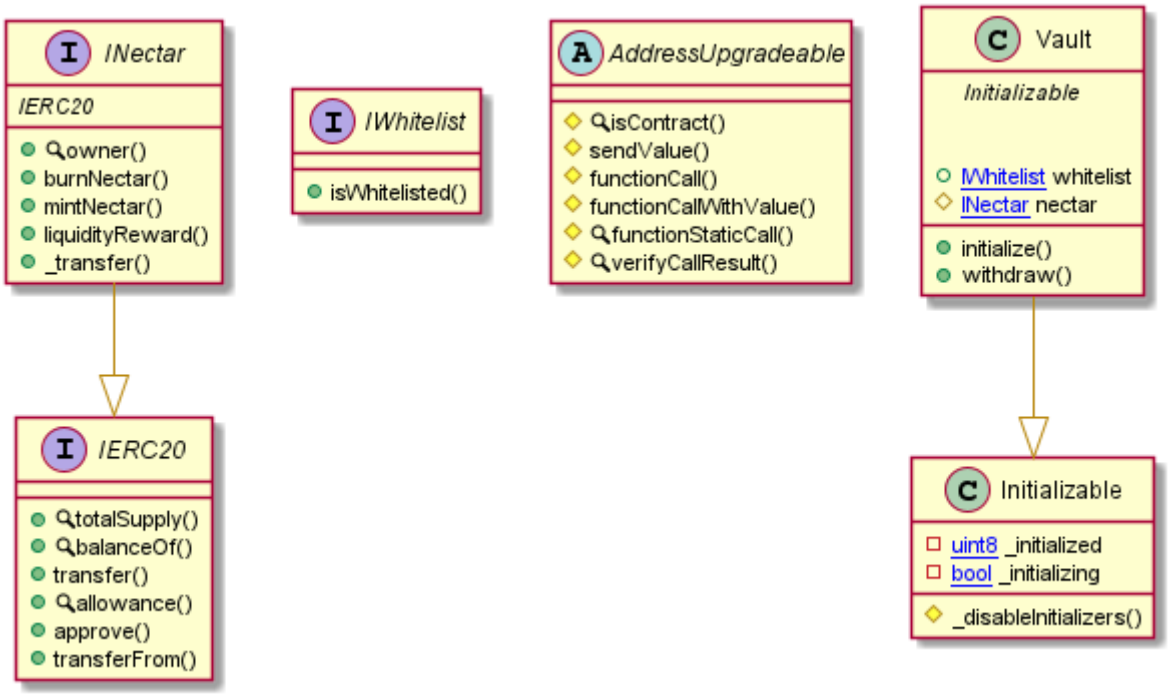
BloomTiers Diagram



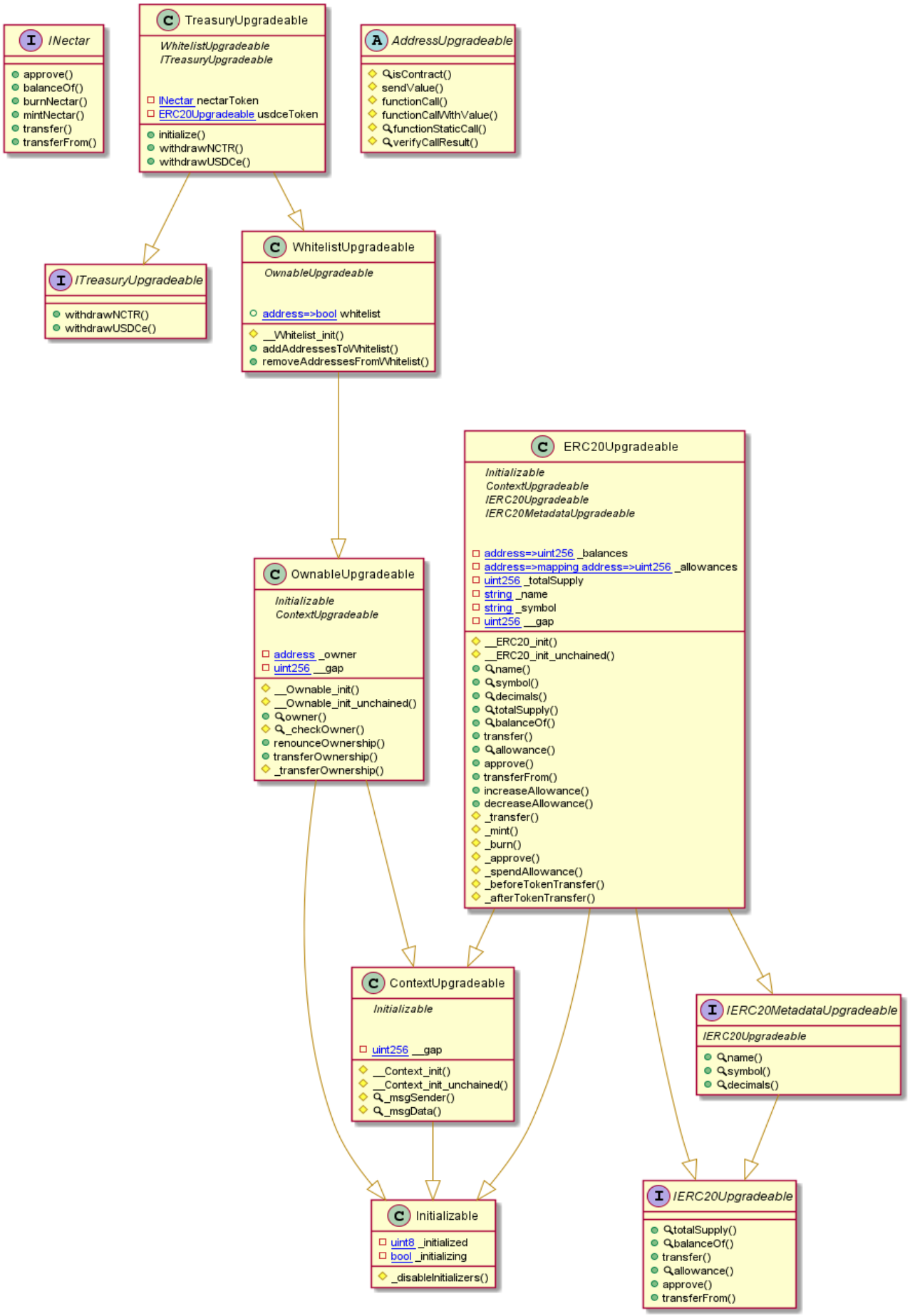
WalletObserverUpgradeable Diagram



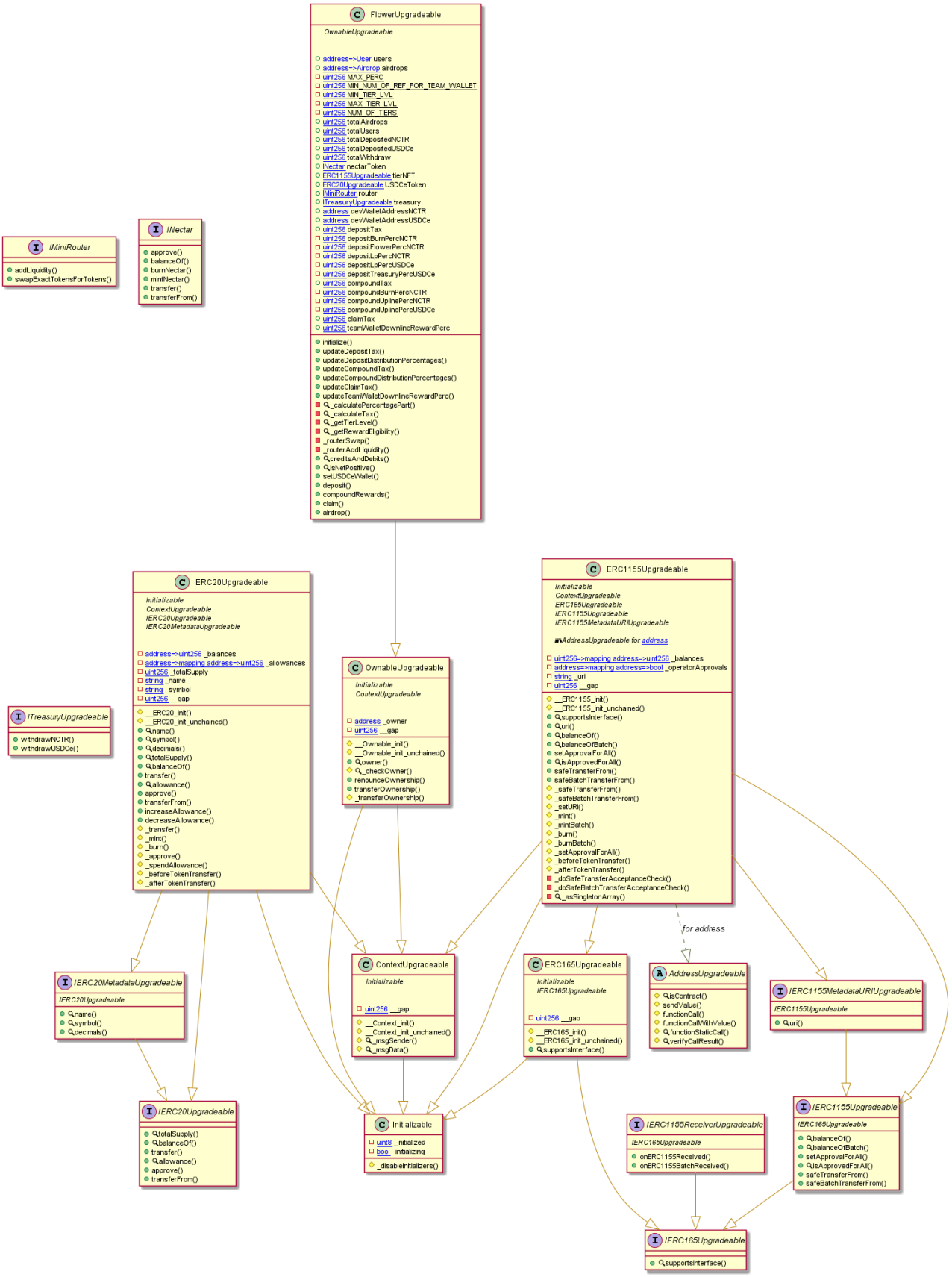
Vault Diagram



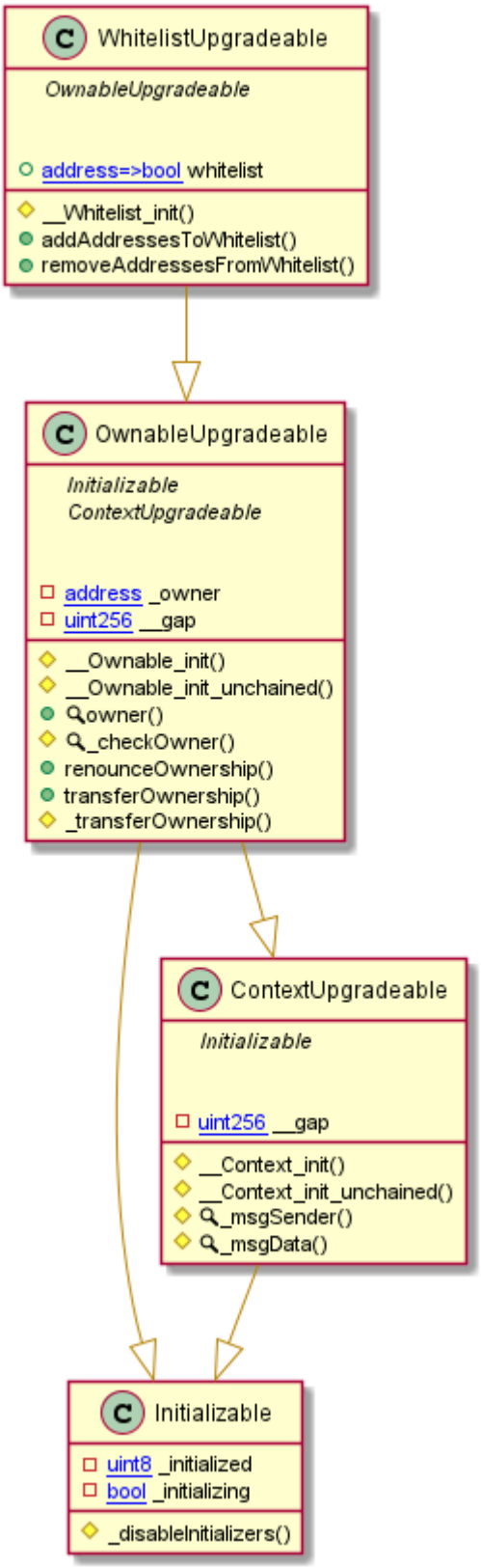
TreasuryUpgradeable Diagram



FlowerUpgradeable Diagram



WhitelistUpgradeable Diagram



Slither Results Log

Slither log >> BloomNFT.sol

```
INFO:Detectors:
BloomNFT.initialize(address,string).bloomNodes_ (BloomNFT.sol#1185) lacks a zero-check on :
                - _bloomNodes = bloomNodes_ (BloomNFT.sol#1192)
BloomNFT.setBloomNodes(address)._newBloomNodes (BloomNFT.sol#1233) lacks a zero-check on :
                - _bloomNodes = _newBloomNodes (BloomNFT.sol#1234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
```

```
INFO:Detectors:
Variable BloomNFT._bloomNodes (BloomNFT.sol#1161) is too similar to BloomNFT.initialize(address,string).bloomNodes_ (BloomNFT.
#1185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
ReentrancyGuardUpgradeable.__gap (BloomNFT.sol#557) is never used in BloomNFT (BloomNFT.sol#1145-1252)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (BloomNFT.sol#625-627)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (BloomNFT.sol#633-636)
balanceOf(address) should be declared external:
        - ERC721Upgradeable.balanceOf(address) (BloomNFT.sol#744-747)
name() should be declared external:
        - ERC721Upgradeable.name() (BloomNFT.sol#761-763)
symbol() should be declared external:
        - ERC721Upgradeable.symbol() (BloomNFT.sol#768-770)
tokenURI(uint256) should be declared external:
        - ERC721Upgradeable.tokenURI(uint256) (BloomNFT.sol#775-780)
approve(address,uint256) should be declared external:
        - ERC721Upgradeable.approve(address,uint256) (BloomNFT.sol#794-804)
setApprovalForAll(address,bool) should be declared external:
        - ERC721Upgradeable.setApprovalForAll(address,bool) (BloomNFT.sol#818-820)
transferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.transferFrom(address,address,uint256) (BloomNFT.sol#832-841)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.safeTransferFrom(address,address,uint256) (BloomNFT.sol#846-852)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:BloomNFT.sol analyzed (13 contracts with 75 detectors), 70 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> BloomsManagerUpgradeable.sol

```
INFO:Detectors:
BloomsManagerUpgradeable.initialize(address,address,address,address,address,address,uint256)._treasury (BloomsManagerUpgradeab
sol#2497) lacks a zero-check on :
                - treasury = _treasury (BloomsManagerUpgradeable.sol#2513)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
BloomsManagerUpgradeable.autoCompound(uint256) (BloomsManagerUpgradeable.sol#2638-2675) has external calls inside a loop: nect
liquidityReward(feeAmount) (BloomsManagerUpgradeable.sol#2665)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in BloomsManagerUpgradeable.createBloomWithTokens(string,uint256) (BloomsManagerUpgradeable.sol#2558-2592):
        External calls:
        - ! whitelist.isWhitelisted(_msgSender()) (BloomsManagerUpgradeable.sol#2564)
        State variables written after the call(s):
        - _blooms[++ bloomCounter] = BloomEntity(_msgSender(),bloomCounter,_bloomName,block.timestamp,0,tierLevel[0],bloomValu
,0,0,0,true) (BloomsManagerUpgradeable.sol#2574-2587)
        - _blooms[++ bloomCounter] = BloomEntity(_msgSender(),bloomCounter,_bloomName,block.timestamp,0,tierLevel[0],bloomValu
,0,0,0,true) (BloomsManagerUpgradeable.sol#2574-2587)
        - totalValueLocked += bloomValue (BloomsManagerUpgradeable.sol#2570)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in BloomsManagerUpgradeable.autoCompound(uint256) (BloomsManagerUpgradeable.sol#2638-2675):
        External calls:
        - nectar.liquidityReward(feeAmount) (BloomsManagerUpgradeable.sol#2665)
        Event emitted after the call(s):
        - Autocompound(msg.sender,bloomId,amountToCompound) (BloomsManagerUpgradeable.sol#2671)
Reentrancy in BloomsManagerUpgradeable.createBloomWithTokens(string,uint256) (BloomsManagerUpgradeable.sol#2558-2592):
        External calls:
        - ! whitelist.isWhitelisted(_msgSender()) (BloomsManagerUpgradeable.sol#2564)
        - bloomNFT.mintBloom(_msgSender(),bloomCounter) (BloomsManagerUpgradeable.sol#2589)
```

```
        - OwnableUpgradeable.renounceOwnership() (BloomsManagerUpgradeable.sol#2370-2372)
        - OwnableUpgradeable.transferOwnership(address) (BloomsManagerUpgradeable.sol#2374-2377)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
NectarImplementationPointerUpgradeable.__gap (BloomsManagerUpgradeable.sol#2425) is never used in BloomsManagerUpgradeable (Bl
sManagerUpgradeable.sol#2428-3132)
PausableUpgradeable._paused (BloomsManagerUpgradeable.sol#2299) is never used in BloomsManagerUpgradeable (BloomsManagerUpgrad
le.sol#2428-3132)
OwnableUpgradeable._owner (BloomsManagerUpgradeable.sol#2345) is never used in BloomsManagerUpgradeable (BloomsManagerUpgradea
.sol#2428-3132)
BloomsManagerUpgradeable._tiersTracked (BloomsManagerUpgradeable.sol#2461) is never used in BloomsManagerUpgradeable (BloomsMa
erUpgradeable.sol#2428-3132)
BloomsManagerUpgradeable._emergencyStats (BloomsManagerUpgradeable.sol#2465) is never used in BloomsManagerUpgradeable (Blooms
agerUpgradeable.sol#2428-3132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (BloomsManagerUpgradeable.sol#2370-2372)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (BloomsManagerUpgradeable.sol#2374-2377)
getNectarImplementation() should be declared external:
        - NectarImplementationPointerUpgradeable.getNectarImplementation() (BloomsManagerUpgradeable.sol#2406-2408)
changeNectarImplementation(address) should be declared external:
        - NectarImplementationPointerUpgradeable.changeNectarImplementation(address) (BloomsManagerUpgradeable.sol#2410-2423)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:BloomsManagerUpgradeable.sol analyzed (20 contracts with 75 detectors), 494 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> LiquidityPoolManager.sol

```
INFO:Detectors:
Reentrancy in LiquidityPoolManager.afterTokenTransfer(address) (LiquidityPoolManager.sol#1055-1100):
        External calls:
        - totalLP = swapAndLiquify(swapTokensToLiquidityThreshold) (LiquidityPoolManager.sol#1075)
                - router.addLiquidity(address(leftSide),address(rightSide),leftAmount,rightAmount,0,0,address(this),block.time
stamp) (LiquidityPoolManager.sol#1162-1171)
                - router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestam
p) (LiquidityPoolManager.sol#1150-1156)
        - pair.sync() (LiquidityPoolManager.sol#1090)
        State variables written after the call(s):
        - pairLiquidityTotalSupply = pair.totalSupply() (LiquidityPoolManager.sol#1097)
Reentrancy in LiquidityPoolManager.initializeManager(address,address[2],uint256) (LiquidityPoolManager.sol#1031-1053):
        External calls:
        - pair = createPairWith(path) (LiquidityPoolManager.sol#1042)
                - _pair = factory.createPair(path[0],path[1]) (LiquidityPoolManager.sol#1140)
        State variables written after the call(s):
        - leftSide = IERC20(path[0]) (LiquidityPoolManager.sol#1043)
        - changeNectarImplementation(address(leftSide)) (LiquidityPoolManager.sol#1050)
                - nectar = INectar(newImplementation) (LiquidityPoolManager.sol#985)
        - pairLiquidityTotalSupply = pair.totalSupply() (LiquidityPoolManager.sol#1045)
        - rightSide = IERC20(path[1]) (LiquidityPoolManager.sol#1044)
        - updateSwapTokensToLiquidityThreshold(_swapTokensToLiquidityThreshold) (LiquidityPoolManager.sol#1047)
                - swapTokensToLiquidityThreshold = _swapTokensToLiquidityThreshold (LiquidityPoolManager.sol#1241)
Reentrancy in LiquidityPoolManager.initializeManager(address,address[2],uint256) (LiquidityPoolManager.sol#1031-1053):
        External calls:
```

```
        Event emitted after the call(s):
        - SwapAndLiquify(half,initialRightBalance,newRightBalance) (LiquidityPoolManager.sol#1122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (LiquidityPoolManager.sol#692-710) uses assembly
        - INLINE ASM (LiquidityPoolManager.sol#702-705)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Variable IJoeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (LiquidityP
oolManager.sol#184) is too similar to IJoeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint25
6).amountBDesired (LiquidityPoolManager.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
NectarImplementationPointer.__gap (LiquidityPoolManager.sol#989) is never used in LiquidityPoolManager (LiquidityPoolManager.s
ol#991-1259)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (LiquidityPoolManager.sol#908-910)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (LiquidityPoolManager.sol#912-915)
getNectarImplementation() should be declared external:
        - NectarImplementationPointer.getNectarImplementation() (LiquidityPoolManager.sol#970-972)
feesForwarder(address[],uint8[]) should be declared external:
        - LiquidityPoolManager.feesForwarder(address[],uint8[]) (LiquidityPoolManager.sol#1244-1258)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LiquidityPoolManager.sol analyzed (16 contracts with 75 detectors), 67 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Nectar.sol

```
INFO:Detectors:
Nectar.setVaultAddress(address)._newVaultAddress (Nectar.sol#1541) lacks a zero-check on :
                - vaultAddress = _newVaultAddress (Nectar.sol#1542)
Nectar.setDevWallet(address).devWallet_ (Nectar.sol#1546) lacks a zero-check on :
                - devWallet = devWallet_ (Nectar.sol#1547)
Nectar.setFlowerManager(address)._newFlowerManager (Nectar.sol#1550) lacks a zero-check on :
                - flowerManager = _newFlowerManager (Nectar.sol#1554)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Nectar.mint(address,uint256) (Nectar.sol#1496-1529):
        External calls:
        - super._mint(_to,_amount) (Nectar.sol#1511)
                - liquidityPoolManager.afterTokenTransfer(_msgSender()) (Nectar.sol#1452)
                - walletObserver.beforeTokenTransfer(_msgSender(),from,to,amount) (Nectar.sol#1378)
        State variables written after the call(s):
        - players += 1 (Nectar.sol#1520)
        - stats[_to].txs += 1 (Nectar.sol#1523)
        - stats[_to].minted += _amount (Nectar.sol#1524)
        - totalTxs += 1 (Nectar.sol#1526)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Nectar.mint(address,uint256) (Nectar.sol#1496-1529):
        External calls:
        - super._mint(_to,_amount) (Nectar.sol#1511)
                - liquidityPoolManager.afterTokenTransfer(_msgSender()) (Nectar.sol#1452)
                - walletObserver.beforeTokenTransfer(_msgSender(),from,to,amount) (Nectar.sol#1378)
        Event emitted after the call(s):
        - MintingFinished() (Nectar.sol#1515)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
OwnableUpgradeable.__gap (Nectar.sol#763) is never used in Nectar (Nectar.sol#1302-1556)
Nectar._excluded (Nectar.sol#1333) is never used in Nectar (Nectar.sol#1302-1556)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (Nectar.sol#735-737)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (Nectar.sol#743-746)
name() should be declared external:
        - ERC20Upgradeable.name() (Nectar.sol#796-798)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (Nectar.sol#804-806)
decimals() should be declared external:
        - ERC20Upgradeable.decimals() (Nectar.sol#821-823)
totalSupply() should be declared external:
        - ERC20Upgradeable.totalSupply() (Nectar.sol#828-830)
balanceOf(address) should be declared external:
        - ERC20Upgradeable.balanceOf(address) (Nectar.sol#835-837)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (Nectar.sol#847-851)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (Nectar.sol#870-874)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (Nectar.sol#892-901)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (Nectar.sol#915-919)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (Nectar.sol#935-944)
burn(uint256) should be declared external:
        - ERC20BurnableUpgradeable.burn(uint256) (Nectar.sol#1137-1139)
burnFrom(address,uint256) should be declared external:
        - ERC20BurnableUpgradeable.burnFrom(address,uint256) (Nectar.sol#1152-1155)
getLiquidityPoolManagerImplementation() should be declared external:
        - LiquidityPoolManagerImplementationPointerUpgradeable.getLiquidityPoolManagerImplementation() (Nectar.sol#1186-1188)
```

```
getLiquidityPoolManagerImplementation() should be declared external:
        - LiquidityPoolManagerImplementationPointerUpgradeable.getLiquidityPoolManagerImplementation() (Nectar.sol#1186-1188)
changeLiquidityPoolManagerImplementation(address) should be declared external:
        - LiquidityPoolManagerImplementationPointerUpgradeable.changeLiquidityPoolManagerImplementation(address) (Nectar.sol#1
190-1203)
getWalletObserverImplementation() should be declared external:
        - WalletObserverImplementationPointer.getWalletObserverImplementation() (Nectar.sol#1281-1283)
changeWalletObserverImplementation(address) should be declared external:
        - WalletObserverImplementationPointer.changeWalletObserverImplementation(address) (Nectar.sol#1285-1298)
```

```
calculateTransferTaxes(address,uint256) should be declared external:
        - Nectar.calculateTransferTaxes(address,uint256) (Nectar.sol#1421-1443)
toggleSwap() should be declared external:
        - Nectar.toggleSwap() (Nectar.sol#1485-1488)
mint(address,uint256) should be declared external:
        - Nectar.mint(address,uint256) (Nectar.sol#1496-1529)
finishMinting() should be declared external:
        - Nectar.finishMinting() (Nectar.sol#1535-1539)
setVaultAddress(address) should be declared external:
        - Nectar.setVaultAddress(address) (Nectar.sol#1541-1543)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Nectar.sol analyzed (18 contracts with 75 detectors), 108 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> OwnerRecovery.sol

```
INFO:Detectors:
Context._msgData() (OwnerRecovery.sol#33-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.4 (OwnerRecovery.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter OwnerRecovery.recoverLostTokens(address,address,uint256)._token (OwnerRecovery.sol#79) is not in mixedCase
Parameter OwnerRecovery.recoverLostTokens(address,address,uint256)._to (OwnerRecovery.sol#80) is not in mixedCase
Parameter OwnerRecovery.recoverLostTokens(address,address,uint256)._amount (OwnerRecovery.sol#81) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (OwnerRecovery.sol#56-58)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (OwnerRecovery.sol#60-63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:OwnerRecovery.sol analyzed (4 contracts with 75 detectors), 10 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> OwnerRecoveryUpgradeable.sol

```
INFO:Detectors:
AddressUpgradeable.verifyCallResult(bool,bytes,string) (OwnerRecoveryUpgradeable.sol#243-263) uses assembly
        - INLINE ASM (OwnerRecoveryUpgradeable.sol#255-258)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AddressUpgradeable.functionCall(address,bytes) (OwnerRecoveryUpgradeable.sol#154-156) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (OwnerRecoveryUpgradeable.sol#164-170) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (OwnerRecoveryUpgradeable.sol#183-189) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (OwnerRecoveryUpgradeable.sol#197-208) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (OwnerRecoveryUpgradeable.sol#216-218) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (OwnerRecoveryUpgradeable.sol#226-235) is never used and should be removed
AddressUpgradeable.isContract(address) (OwnerRecoveryUpgradeable.sol#105-111) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (OwnerRecoveryUpgradeable.sol#129-134) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (OwnerRecoveryUpgradeable.sol#243-263) is never used and should be removed
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function ContextUpgradeable.__Context_init() (OwnerRecoveryUpgradeable.sol#349-350) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (OwnerRecoveryUpgradeable.sol#352-353) is not in mixedCase
Variable ContextUpgradeable.__gap (OwnerRecoveryUpgradeable.sol#367) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (OwnerRecoveryUpgradeable.sol#377-379) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (OwnerRecoveryUpgradeable.sol#381-383) is not in mixedCase
Variable OwnableUpgradeable.__gap (OwnerRecoveryUpgradeable.sol#442) is not in mixedCase
Parameter OwnerRecoveryUpgradeable.recoverLostTokens(address,address,uint256)._token (OwnerRecoveryUpgradeable.sol#452) is not in mixedCase
Parameter OwnerRecoveryUpgradeable.recoverLostTokens(address,address,uint256)._to (OwnerRecoveryUpgradeable.sol#453) is not in mixedCase
Parameter OwnerRecoveryUpgradeable.recoverLostTokens(address,address,uint256)._amount (OwnerRecoveryUpgradeable.sol#454) is not in mixedCase
Variable OwnerRecoveryUpgradeable.__gap (OwnerRecoveryUpgradeable.sol#459) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
OwnerRecoveryUpgradeable.__gap (OwnerRecoveryUpgradeable.sol#459) is never used in OwnerRecoveryUpgradeable (OwnerRecoveryUpgradeable.sol#446-461)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (OwnerRecoveryUpgradeable.sol#414-416)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (OwnerRecoveryUpgradeable.sol#422-425)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:OwnerRecoveryUpgradeable.sol analyzed (6 contracts with 75 detectors), 38 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> BloomTiers.sol

```
INFO:Detectors:
BloomTiers.initialize(string,address)._bloomReferral (BloomTiers.sol#1416) lacks a zero-check on :
                - bloomReferral = _bloomReferral (BloomTiers.sol#1424)
BloomTiers.setBloomReferral(address)._newBloomReferral (BloomTiers.sol#1515) lacks a zero-check on :
                - bloomReferral = _newBloomReferral (BloomTiers.sol#1516)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
AddressUpgradeable.verifyCallResult(bool,bytes,string) (BloomTiers.sol#314-334) uses assembly
        - INLINE ASM (BloomTiers.sol#326-329)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AddressUpgradeable.functionCall(address,bytes) (BloomTiers.sol#225-227) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (BloomTiers.sol#235-241) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (BloomTiers.sol#254-260) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (BloomTiers.sol#268-279) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (BloomTiers.sol#287-289) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (BloomTiers.sol#297-306) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (BloomTiers.sol#200-205) is never used and should be removed
```

```
INFO:Detectors:
ReentrancyGuardUpgradeable.__gap (BloomTiers.sol#752) is never used in BloomTiers (BloomTiers.sol#1350-1564)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
BloomTiers._uri (BloomTiers.sol#1380) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (BloomTiers.sol#670-672)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (BloomTiers.sol#678-681)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155Upgradeable.balanceOfBatch(address[],uint256[]) (BloomTiers.sol#842-858)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155Upgradeable.setApprovalForAll(address,bool) (BloomTiers.sol#863-865)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155Upgradeable.safeTransferFrom(address,address,uint256,uint256,bytes) (BloomTiers.sol#877-889)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155Upgradeable.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (BloomTiers.sol#894-906)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:BloomTiers.sol analyzed (15 contracts with 75 detectors), 85 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> WalletObserverUpgradeable.sol

```
INFO:Detectors:
AddressUpgradeable.verifyCallResult(bool,bytes,string) (WalletObserverUpgradeable.sol#432-452) uses assembly
        - INLINE ASM (WalletObserverUpgradeable.sol#444-447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AddressUpgradeable.functionCall(address,bytes) (WalletObserverUpgradeable.sol#343-345) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (WalletObserverUpgradeable.sol#353-359) is never used and should be remo
ved
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (WalletObserverUpgradeable.sol#372-378) is never used and shou
ld be removed
```

```
INFO:Detectors:
BloomsManagerImplementationPointerUpgradeable.__gap (WalletObserverUpgradeable.sol#806) is never used in WalletObserverUpgrade
able (WalletObserverUpgradeable.sol#851-1322)
WalletObserverUpgradeable.maxTokenPerWallet (WalletObserverUpgradeable.sol#885) is never used in WalletObserverUpgradeable (Wa
lletObserverUpgradeable.sol#851-1322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
WalletObserverUpgradeable.maxTokenPerWallet (WalletObserverUpgradeable.sol#885) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (WalletObserverUpgradeable.sol#603-605)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (WalletObserverUpgradeable.sol#611-614)
getNectarImplementation() should be declared external:
        - NectarImplementationPointerUpgradeable.getNectarImplementation() (WalletObserverUpgradeable.sol#652-654)
getBloomsManagerImplementation() should be declared external:
        - BloomsManagerImplementationPointerUpgradeable.getBloomsManagerImplementation() (WalletObserverUpgradeable.sol#787-78
9)
```

```
changeBloomsManagerImplementation(address) should be declared external:
        - BloomsManagerImplementationPointerUpgradeable.changeBloomsManagerImplementation(address) (WalletObserverUpgradeable.
sol#791-804)
getLiquidityPoolManagerImplementation() should be declared external:
        - LiquidityPoolManagerImplementationPointerUpgradeable.getLiquidityPoolManagerImplementation() (WalletObserverUpgradea
ble.sol#829-831)
isWalletFull(address) should be declared external:
        - WalletObserverUpgradeable.isWalletFull(address) (WalletObserverUpgradeable.sol#1226-1228)
setMaxTokenPerWalletPercent(uint8) should be declared external:
        - WalletObserverUpgradeable.setMaxTokenPerWalletPercent(uint8) (WalletObserverUpgradeable.sol#1242-1258)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:WalletObserverUpgradeable.sol analyzed (15 contracts with 75 detectors), 68 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Vault.sol

```
INFO:Detectors:
AddressUpgradeable.verifyCallResult(bool,bytes,string) (Vault.sol#260-280) uses assembly
        - INLINE ASM (Vault.sol#272-275)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AddressUpgradeable.functionCall(address,bytes) (Vault.sol#171-173) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (Vault.sol#181-187) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (Vault.sol#200-206) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (Vault.sol#214-225) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (Vault.sol#233-235) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (Vault.sol#243-252) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (Vault.sol#146-151) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (Vault.sol#260-280) is never used and should be removed
Initializable._disableInitializers() (Vault.sol#356-362) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.4 (Vault.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (Vault.sol#146-151):
        - (success) = recipient.call{value: amount}() (Vault.sol#149)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (Vault.sol#214-225):
        - (success,returndata) = target.call{value: value}(data) (Vault.sol#223)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (Vault.sol#243-252):
        - (success,returndata) = target.staticcall(data) (Vault.sol#250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function INectar._transfer(address,address,uint256) (Vault.sol#92) is not in mixedCase
Parameter Vault.initialize(address,address)._nectar (Vault.sol#375) is not in mixedCase
Parameter Vault.initialize(address,address)._whitelist (Vault.sol#375) is not in mixedCase
Parameter Vault.withdraw(uint256)._amount (Vault.sol#383) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
withdraw(uint256) should be declared external:
        - Vault.withdraw(uint256) (Vault.sol#383-389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Vault.sol analyzed (6 contracts with 75 detectors), 20 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> TreasuryUpgradeable.sol

```
INFO:Detectors:
AddressUpgradeable.verifyCallResult(bool,bytes,string) (TreasuryUpgradeable.sol#287-307) uses assembly
        - INLINE ASM (TreasuryUpgradeable.sol#299-302)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
ERC20Upgradeable.__gap (TreasuryUpgradeable.sol#873) is never used in ERC20Upgradeable (TreasuryUpgradeable.sol#515-874)
OwnableUpgradeable.__gap (TreasuryUpgradeable.sol#487) is never used in TreasuryUpgradeable (TreasuryUpgradeable.sol#937-986)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (TreasuryUpgradeable.sol#459-461)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (TreasuryUpgradeable.sol#467-470)
name() should be declared external:
        - ERC20Upgradeable.name() (TreasuryUpgradeable.sol#546-548)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (TreasuryUpgradeable.sol#554-556)
decimals() should be declared external:
        - ERC20Upgradeable.decimals() (TreasuryUpgradeable.sol#571-573)
totalSupply() should be declared external:
        - ERC20Upgradeable.totalSupply() (TreasuryUpgradeable.sol#578-580)
balanceOf(address) should be declared external:
        - ERC20Upgradeable.balanceOf(address) (TreasuryUpgradeable.sol#585-587)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (TreasuryUpgradeable.sol#597-601)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (TreasuryUpgradeable.sol#620-624)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (TreasuryUpgradeable.sol#642-651)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (TreasuryUpgradeable.sol#665-669)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (TreasuryUpgradeable.sol#685-694)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:TreasuryUpgradeable.sol analyzed (11 contracts with 75 detectors), 54 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> FlowerUpgradeable.sol

```
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (FlowerUpgradeable.sol#664-666)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (FlowerUpgradeable.sol#672-675)
name() should be declared external:
        - ERC20Upgradeable.name() (FlowerUpgradeable.sol#726-728)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (FlowerUpgradeable.sol#734-736)
decimals() should be declared external:
        - ERC20Upgradeable.decimals() (FlowerUpgradeable.sol#751-753)
totalSupply() should be declared external:
        - ERC20Upgradeable.totalSupply() (FlowerUpgradeable.sol#758-760)
balanceOf(address) should be declared external:
        - ERC20Upgradeable.balanceOf(address) (FlowerUpgradeable.sol#765-767)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (FlowerUpgradeable.sol#777-781)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (FlowerUpgradeable.sol#800-804)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (FlowerUpgradeable.sol#822-831)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (FlowerUpgradeable.sol#845-849)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (FlowerUpgradeable.sol#865-874)
uri(uint256) should be declared external:
        - ERC1155Upgradeable.uri(uint256) (FlowerUpgradeable.sol#1120-1122)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155Upgradeable.balanceOfBatch(address[],uint256[]) (FlowerUpgradeable.sol#1143-1159)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155Upgradeable.setApprovalForAll(address,bool) (FlowerUpgradeable.sol#1164-1166)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155Upgradeable.safeTransferFrom(address,address,uint256,uint256,bytes) (FlowerUpgradeable.sol#1178-1190)
```

```
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155Upgradeable.safeTransferFrom(address,address,uint256,uint256,bytes) (FlowerUpgradeable.sol#1178-1190)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155Upgradeable.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (FlowerUpgradeable.sol#1195-1207
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:FlowerUpgradeable.sol analyzed (17 contracts with 75 detectors), 137 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> WhitelistUpgradeable.sol

```
INFO:Detectors:
ContextUpgradeable.__Context_init() (WhitelistUpgradeable.sol#87-88) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (WhitelistUpgradeable.sol#90-91) is never used and should be removed
ContextUpgradeable._msgData() (WhitelistUpgradeable.sol#96-98) is never used and should be removed
Initializable._disableInitializers() (WhitelistUpgradeable.sol#77-83) is never used and should be removed
OwnableUpgradeable.__Ownable_init() (WhitelistUpgradeable.sol#116-118) is never used and should be removed
OwnableUpgradeable.__Ownable_init_unchained() (WhitelistUpgradeable.sol#120-122) is never used and should be removed
WhitelistUpgradeable.__Whitelist_init() (WhitelistUpgradeable.sol#190-192) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.4 (WhitelistUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6
/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function ContextUpgradeable.__Context_init() (WhitelistUpgradeable.sol#87-88) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (WhitelistUpgradeable.sol#90-91) is not in mixedCase
Variable ContextUpgradeable.__gap (WhitelistUpgradeable.sol#105) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (WhitelistUpgradeable.sol#116-118) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (WhitelistUpgradeable.sol#120-122) is not in mixedCase
Variable OwnableUpgradeable.__gap (WhitelistUpgradeable.sol#181) is not in mixedCase
Function WhitelistUpgradeable.__Whitelist_init() (WhitelistUpgradeable.sol#190-192) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
OwnableUpgradeable.__gap (WhitelistUpgradeable.sol#181) is never used in WhitelistUpgradeable (WhitelistUpgradeable.sol#184-24
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (WhitelistUpgradeable.sol#153-155)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (WhitelistUpgradeable.sol#161-164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:WhitelistUpgradeable.sol analyzed (4 contracts with 75 detectors), 20 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

BloomNFT.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: 355:4:

Gas & Economy

Gas costs:

Gas requirement of function BloomNFT.setBaseURI 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: 1241:4:

Miscellaneous

Constant/View/Pure functions:

ERC721Upgradeable._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1131:4:

Similar variable names:

BloomNFT.setBaseURI(string) : Variables have very similar names "baseURI" and "baseURI_". Note: Modifiers are currently not considered by this static analysis.
Pos: 1242:18:

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: 1168:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 994:8:

BloomsManagerUpgradeable.sol

Security

Check-effects-interaction:

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

more

Pos: 3546:4:

Block timestamp:

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

more

Pos: 3638:12:

Gas & Economy

Gas costs:

Gas requirement of function BloomsManagerUpgradeable.whitelist 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: 2712:4:

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: 3183:8:

ERC

ERC20:

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

more

Pos: 208:4:

Miscellaneous

Constant/View/Pure functions:

BloomsManagerUpgradeable._resetRewardMultiplier(uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 3666:4:

Similar variable names:

BloomsManagerUpgradeable._burn(uint256) : Variables have very similar names "_blooms" and "bloom". Note: Modifiers are currently not considered by this static analysis.
Pos: 3694:43:

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: 3652:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 3596:8:

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: 3464:33:

LiquidityPoolManager.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in LiquidityPoolManager.swapAndLiquify(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1111:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1170:12:

Gas & Economy

Gas costs:

Gas requirement of function LiquidityPoolManager.feesForwarder 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: 1244:4:

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: 1078:12:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 45:4:

Miscellaneous

Constant/View/Pure functions:

LiquidityPoolManager.setAllowance(bool) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1221:4:

Similar variable names:

LiquidityPoolManager.swapLeftSideForRightSide(uint256) : Variables have very similar names "pair" and "path". Note: Modifiers are currently not considered by this static analysis.
Pos: 1153:12:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1252:8:

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: 1112:23:

Nectar.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: 379:50:

Gas & Economy

Gas costs:

Gas requirement of function Nectar.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: 1496:4:

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: 1237:8:

Miscellaneous

Similar variable names:

Nectar.setDevWallet(address) : Variables have very similar names "devWallet" and "devWallet_".
Note: Modifiers are currently not considered by this static analysis.
Pos: 1547:20:

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: 1503:8:

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: 187:19:

OwnerRecovery.sol

Miscellaneous

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 14:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 61:8:

OwnerRecoveryUpgradeable.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: 197:4:

Miscellaneous

Constant/View/Pure functions:

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

more

Pos: 352:4:

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: 423:8:

BloomTiers.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: 268:4:

Gas & Economy

Gas costs:

Gas requirement of function BloomTiers.setURI 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: 1539:4:

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: 1498:8:

Miscellaneous

Constant/View/Pure functions:

ERC1155Upgradeable._asSingletonArray(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1271:4:

Similar variable names:

BloomTiers.setURI(uint256,string) : Variables have very similar names "_tokenURI" and "_tokenURIs". Note: Modifiers are currently not considered by this static analysis.
Pos: 1543:26:

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: 1478:8:

WalletObserverUpgradeable.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 1009:39:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in WalletObserverUpgradeable.beforeTokenTransfer(address,address,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1000:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 925:31:

Gas & Economy

Gas costs:

Gas requirement of function WalletObserverUpgradeable.totalSupply is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1302:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 171:4:

Miscellaneous

Constant/View/Pure functions:

WalletObserverUpgradeable.totalSupply() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1302:4:

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: 1284:8:

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: 1161:15:

Vault.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: 214:4:

Gas & Economy

Gas costs:

Gas requirement of function Vault.whitelist 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: 366:4:

Miscellaneous

Constant/View/Pure functions:

AddressUpgradeable.verifyCallResult(bool,bytes,string) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 260:4:

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: 385:12:

TreasuryUpgradeable.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: 241:4:

Gas & Economy

Gas costs:

Gas requirement of function TreasuryUpgradeable.withdrawUSDCe 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: 976:4:

Miscellaneous

Constant/View/Pure functions:

ERC20Upgradeable._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 862:4:

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: 981:8:

FlowerUpgradeable.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in FlowerUpgradeable.airdrop(address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2327:4:

Gas & Economy

Gas costs:

Gas requirement of function FlowerUpgradeable.airdrop 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: 2327:4:

Miscellaneous

Constant/View/Pure functions:

ERC1155Upgradeable._asSingletonArray(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1572:4:

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: 2108:12:

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: 2039:30:

WhitelistUpgradeable.sol

Gas & Economy

Gas costs:

Gas requirement of function WhitelistUpgradeable.addAddressesToWhitelist 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: 208:4:

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: 233:8:

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: 162:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 198:8:

Solhint Linter

BloomNFT.sol

```
BloomNFT.sol:2:1: Error: Compiler version ^0.8.13 does not satisfy
the r semver requirement
BloomNFT.sol:290:28: Error: Avoid using low level calls.
BloomNFT.sol:364:51: Error: Avoid using low level calls.
BloomNFT.sol:413:17: Error: Avoid using inline assembly. It is
acceptable only in rare cases
BloomNFT.sol:523:5: Error: Function name must be in mixedCase
BloomNFT.sol:527:5: Error: Function name must be in mixedCase
BloomNFT.sol:560:5: Error: Function name must be in mixedCase
BloomNFT.sol:560:57: Error: Code contains empty blocks
BloomNFT.sol:563:5: Error: Function name must be in mixedCase
BloomNFT.sol:563:67: Error: Code contains empty blocks
BloomNFT.sol:1090:21: Error: Avoid using inline assembly. It is
acceptable only in rare cases
BloomNFT.sol:1118:24: Error: Code contains empty blocks
BloomNFT.sol:1135:24: Error: Code contains empty blocks
```

BloomsManagerUpgradeable.sol

```
BloomsManagerUpgradeable.sol:104:15: Error: Parse error: mismatched
input '(' expecting {';', '='}
```

LiquidityPoolManager.sol

```
LiquidityPoolManager.sol:411:18: Error: Parse error: missing ';' at
'{'
LiquidityPoolManager.sol:424:18: Error: Parse error: missing ';' at
'{'
LiquidityPoolManager.sol:436:18: Error: Parse error: missing ';' at
'{'
LiquidityPoolManager.sol:453:18: Error: Parse error: missing ';' at
'{'
LiquidityPoolManager.sol:465:18: Error: Parse error: missing ';' at
'{'
LiquidityPoolManager.sol:561:18: Error: Parse error: missing ';' at
'{'
LiquidityPoolManager.sol:584:18: Error: Parse error: missing ';' at
'{'
LiquidityPoolManager.sol:610:18: Error: Parse error: missing ';' at
'{'
LiquidityPoolManager.sol:852:18: Error: Parse error: missing ';' at
'{'
```

Nectar.sol

```
Nectar.sol:12:18: Error: Parse error: missing ';' at '{'
Nectar.sol:25:18: Error: Parse error: missing ';' at '{'
Nectar.sol:37:18: Error: Parse error: missing ';' at '{'
Nectar.sol:54:18: Error: Parse error: missing ';' at '{'
```

OwnerRecovery.sol

```
OwnerRecovery.sol:2:1: Error: Compiler version ^0.8.13 does not
satisfy the r semver requirement
OwnerRecovery.sol:43:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

OwnerRecoveryUpgradeable.sol

```
OwnerRecoveryUpgradeable.sol:2:1: Error: Compiler version ^0.8.13
does not satisfy the r semver requirement
OwnerRecoveryUpgradeable.sol:132:28: Error: Avoid using low level
calls.
OwnerRecoveryUpgradeable.sol:206:51: Error: Avoid using low level
calls.
OwnerRecoveryUpgradeable.sol:255:17: Error: Avoid using inline
assembly. It is acceptable only in rare cases
OwnerRecoveryUpgradeable.sol:349:5: Error: Function name must be in
mixedCase
OwnerRecoveryUpgradeable.sol:349:57: Error: Code contains empty
blocks
OwnerRecoveryUpgradeable.sol:352:5: Error: Function name must be in
mixedCase
OwnerRecoveryUpgradeable.sol:352:67: Error: Code contains empty
blocks
OwnerRecoveryUpgradeable.sol:377:5: Error: Function name must be in
mixedCase
OwnerRecoveryUpgradeable.sol:381:5: Error: Function name must be in
mixedCase
```

BloomTiers.sol

```
BloomTiers.sol:937:18: Error: Parse error: missing ';' at '{'
BloomTiers.sol:979:22: Error: Parse error: missing ';' at '{'
BloomTiers.sol:1108:18: Error: Parse error: missing ';' at '{'
BloomTiers.sol:1144:22: Error: Parse error: missing ';' at '{'
```

WalletObserverUpgradeable.sol

```
WalletObserverUpgradeable.sol:67:15: Error: Parse error: mismatched
input '(' expecting {';', '='}
```

Vault.sol

```
Vault.sol:2:1: Error: Compiler version ^0.8.13 does not satisfy the r
semver requirement
Vault.sol:149:28: Error: Avoid using low level calls.
Vault.sol:223:51: Error: Avoid using low level calls.
Vault.sol:272:17: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

TreasuryUpgradeable.sol

```
TreasuryUpgradeable.sol:689:18: Error: Parse error: missing ';' at
'{'
TreasuryUpgradeable.sol:722:18: Error: Parse error: missing ';' at
'{'
TreasuryUpgradeable.sol:771:18: Error: Parse error: missing ';' at
'{'
TreasuryUpgradeable.sol:822:22: Error: Parse error: missing ';' at
'{'
```

FlowerUpgradeable.sol

```
FlowerUpgradeable.sol:869:18: Error: Parse error: missing ';' at '{'
FlowerUpgradeable.sol:902:18: Error: Parse error: missing ';' at '{'
FlowerUpgradeable.sol:951:18: Error: Parse error: missing ';' at '{'
FlowerUpgradeable.sol:1002:22: Error: Parse error: missing ';' at '{'
FlowerUpgradeable.sol:1238:18: Error: Parse error: missing ';' at '{'
FlowerUpgradeable.sol:1280:22: Error: Parse error: missing ';' at '{'
FlowerUpgradeable.sol:1409:18: Error: Parse error: missing ';' at '{'
FlowerUpgradeable.sol:1445:22: Error: Parse error: missing ';' at '{'
```

WhitelistUpgradeable.sol

```
WhitelistUpgradeable.sol:3:1: Error: Compiler version ^0.8.4 does not
satisfy the r semver requirement
WhitelistUpgradeable.sol:87:5: Error: Function name must be in
mixedCase
WhitelistUpgradeable.sol:87:57: Error: Code contains empty blocks
WhitelistUpgradeable.sol:90:5: Error: Function name must be in
mixedCase
```

```
WhitelistUpgradeable.sol:90:67: Error: Code contains empty blocks
WhitelistUpgradeable.sol:116:5: Error: Function name must be in
mixedCase
```

Software analysis result:

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