Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Waifu Protocol
Website: waifuverse.studio
Platform: FTM, AVAX and BSC
Language: Solidity
Date: August 23rd, 2022

Table of contents

Introduction 4
Project Background 4
Audit Scope 4
Claimed Smart Contract Features 6
Audit Summary 8
Technical Quick Stats 9
Code Quality 10
Documentation 10
Use of Dependencies 10
AS-IS overview 11
Severity Definitions 28
Audit Findings 29
Conclusion 39
Our Methodology 40
Disclaimers 42
Appendix
- Code Flow Diagram 43
- Slither Results Log 54
- Solhint Linter 62

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 Waifu to perform the Security audit of the Waifu 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 August 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

Waifu Protocol is a metaverse protocol using ERC1155 NFT tokens which has functions like initialize, mint, burn, grantRole, withdraw, hasRole, shares, release, payee, receive, pause, unpause, revokeRole, grantRole, etc. The Waifu contract inherits the SafeERC20Upgradeable, AccessControlEnumerableUpgradeable, IERC20Upgradeable, PausableUpgradeable, Initializable, UUPSUpgradeable, ERC1155Upgradeable, ERC20CappedUpgradeable, ERC721PausableUpgradeable, etc. 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 Waifu Protocol Smart Contracts**
Platform	**FTM, AVAX and BSC / Solidity**
File 1	PerkSaleHelper.sol
File 1 MD5 Hash	4ED409BABEBF156284DCD56D27635829
File 2	PresaleHelper.sol
File 2 MD5 Hash	A5CD6DED26A197836B2B6AECF74F2538
File 3	LiquidityManager.sol
File 3 MD5 Hash	08068C9C20EC746A7168A519BEF12383

File 4	WaifuCashier.sol
File 4 MD5 Hash	6448CC7A991B29AA8BF0D725D09872B1
File 5	WaifuManager.sol
File 5 MD5 Hash	5F559A1204638714BF26AAA0CD3CDDED
File 6	WaifuNodes.sol
File 6 MD5 Hash	0340946926AA1277187DB503E8B1CE48
File 7	WaifuPerks.sol
File 7 MD5 Hash	3CA4B9D84E20DF5621CE701564BA5FE6
File 8	EarlyWaifuHolders.sol
File 8 MD5 Hash	0C21DAB9B02377C329CAEB357B9F8078
File 9	RevenuePaymentSplitter.sol
File 9 MD5 Hash	E5A92EE503076F61F03611F038CD0144
File 10	PreLaunchToken.sol
File 10 MD5 Hash	ADCDEBB3090AB862677F626F0ECDFC14
File 11	WaifuToken.sol
File 11 MD5 Hash	116228396301C05BC4EB67181DFF6B5B
Audit Date	August 23rd,2022

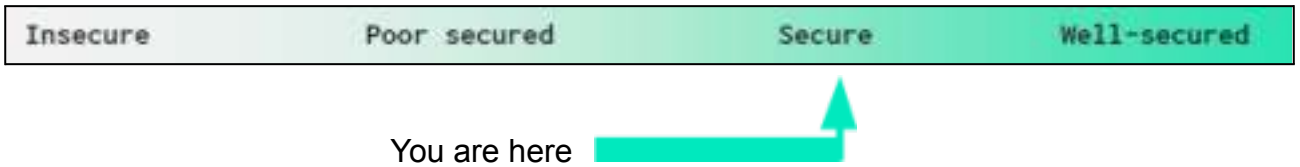
Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 PerkSaleHelper.sol • PerkSaleHelper has functions like: purchase, finishPresale, etc.	**YES, This is valid.**
File 2 PresaleHelper.sol • PresaleHelper has functions like: setPurchaseAllowed, purchase, etc.	**YES, This is valid.**
File 3 LiquidityManager.sol • Minimum Price: 0.9 Coin • Maximum Price: 1.15 Coin	**YES, This is valid.**
File 4 WaifuCashier.sol • Reclaim/Burn: 30% • Treasury: 20% • Liquidity: 30% • Company wallet: 10% • Escrow Account: 10% • WaifuCashier has functions like: grantRewardsFor, claimRewardsMax, etc.	**YES, This is valid.**
File 5 WaifuManager.sol • WaifuManager has functions like: buyNodes, buyNodesBatch, etc.	**YES, This is valid.**
File 6 WaifuNodes.sol • Snapshot Frequency: 1 Days	**YES, This is valid.**
File 7 WaifuPerks.sol • Tier Count: 4 • Maximum percentage: 2.5% • Maximum Wallet limit increase: 20,000	**YES, This is valid.**

• Precision: 10000	
File 8 EarlyWaifuHolders.sol • Name: EarlyWaifuHolders • Symbole: EWH	**YES, This is valid.**
File 9 RevenuePaymentSplitter.sol • RevenuePaymentSplitter has functions like: initialize, release, etc.	**YES, This is valid.**
File 10 PreLaunchToken.sol • PreLaunchToken has functions like: mint, withdrawTo, etc.	**YES, This is valid.**
File 11 WaifuToken.sol • Name: UWU Token • Symbole: UWU • Decimals: 18 • Precision: 10000	**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, 1 high, 2 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	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	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 11 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 Waifu 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 Waifu Protocol.

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

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

Documentation

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

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

Another source of information was its official website https://www.waifuverse.studio/ which provided rich information about the project architecture.

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

PerkSaleHelper.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	access by initializer	No Issue
3	getTotalPrice	read	Passed	No Issue
4	purchase	external	Critical operation lacks event log	Refer Audit Findings
5	finishPresale	external	access only Role	No Issue
6	setTypePrice	external	access only Role	No Issue
7	setTypePresalePrice	external	access only Role	No Issue
8	_authorizeUpgrade	internal	access only Role	No Issue
9	initializer	modifier	Passed	No Issue
10	reinitializer	modifier	Passed	No Issue
11	onlyInitializing	modifier	Passed	No Issue
12	_disableInitializers	internal	Passed	No Issue
13	__AccessControl_init	internal	access only Initializing	No Issue
14	__AccessControl_init_unchained	internal	access only Initializing	No Issue
15	onlyRole	modifier	Passed	No Issue
16	supportsInterface	read	Passed	No Issue
17	hasRole	read	Passed	No Issue
18	_checkRole	internal	Passed	No Issue
19	_checkRole	internal	Passed	No Issue
20	getRoleAdmin	read	Passed	No Issue
21	grantRole	write	access only Role	No Issue
22	revokeRole	write	access only Role	No Issue
23	renounceRole	write	Passed	No Issue
24	_setupRole	internal	Passed	No Issue
25	_setRoleAdmin	internal	Passed	No Issue
26	_grantRole	internal	Passed	No Issue
27	_revokeRole	internal	Passed	No Issue
28	__UUPSUpgradeable_init	internal	access only Initializing	No Issue
29	__UUPSUpgradeable_init_unchained	internal	access only Initializing	No Issue
30	onlyProxy	modifier	Passed	No Issue
31	notDelegated	modifier	Passed	No Issue
32	proxiableUUID	external	Passed	No Issue
33	upgradeTo	external	access only Proxy	No Issue
34	upgradeToAndCall	external	access only Proxy	No Issue
35	_authorizeUpgrade	internal	Passed	No Issue

PresaleHelper.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	__AccessControl_init	internal	access only Initializing	No Issue
7	__AccessControl_init_unchained	internal	access only Initializing	No Issue
8	onlyRole	modifier	Passed	No Issue
9	supportsInterface	read	Passed	No Issue
10	hasRole	read	Passed	No Issue
11	_checkRole	internal	Passed	No Issue
12	_checkRole	internal	Passed	No Issue
13	getRoleAdmin	read	Passed	No Issue
14	grantRole	write	access only Role	No Issue
15	revokeRole	write	access only Role	No Issue
16	renounceRole	write	Passed	No Issue
17	_setupRole	internal	Passed	No Issue
18	_setRoleAdmin	internal	Passed	No Issue
19	_grantRole	internal	Passed	No Issue
20	_revokeRole	internal	Passed	No Issue
21	__UUPSUpgradeable_init	internal	access only Initializing	No Issue
22	__UUPSUpgradeable_init_unchained	internal	access only Initializing	No Issue
23	onlyProxy	modifier	Passed	No Issue
24	notDelegated	modifier	Passed	No Issue
25	proxiableUUID	external	Passed	No Issue
26	upgradeTo	external	access only Proxy	No Issue
27	upgradeToAndCall	external	access only Proxy	No Issue
28	_authorizeUpgrade	internal	Passed	No Issue
29	initialize	write	access only Initializing	No Issue
30	getTotalPrice	read	Passed	No Issue
31	setPurchaseAllowed	external	access only Role	No Issue
32	setPurchaseAllowedBatch	external	access only Role	No Issue
33	purchase	external	Passed	No Issue
34	_authorizeUpgrade	internal	Passed	No Issue

LiquidityManager.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	Passed	No Issue
3	stabilize	write	Passed	No Issue
4	getLpPair	external	Passed	No Issue
5	_pullMainTokenFromLP	internal	Passed	No Issue
6	_sendMainTokensToLP	internal	Passed	No Issue
7	setPriceRange	write	access only Owner	No Issue
8	setIsEnabled	external	access only Owner	No Issue
9	adminWithdraw	external	access only Owner	No Issue
10	adminWithdrawETH	external	access only Owner	No Issue
11	__Ownable_init	internal	access only Initializing	No Issue
12	__Ownable_init_unchained	internal	access only Initializing	No Issue
13	onlyOwner	modifier	Passed	No Issue
14	owner	read	Passed	No Issue
15	_checkOwner	internal	Passed	No Issue
16	renounceOwnership	write	access only Owner	No Issue
17	transferOwnership	write	access only Owner	No Issue
18	_transferOwnership	internal	Passed	No Issue

WaifuCashier.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	DEFAULT_ADMIN_ROLE is Re-Assigned	Refer Audit Findings
3	getMintLimit	external	Passed	No Issue
4	getWaifuBalance	read	Passed	No Issue
5	getUsdBalance	read	Passed	No Issue
6	getClaimTaxOf	read	Passed	No Issue
7	getPaymentFrom	external	access only Role	No Issue
8	grantRewardsFor	external	access only Role	No Issue
9	claimRewardsMax	external	Passed	No Issue
10	claimRewards	write	Passed	No Issue
11	liquidateWaifu	external	Passed	No Issue
12	liquidateToken	external	access only Role	No Issue
13	setAnnualMintLimit	external	access only Role	No Issue
14	setDefaultClaimTax	external	Tax limit is not set	Refer Audit Findings

15	setTreasury	external	access only Role	No Issue
16	setCompanyWallet	external	access only Role	No Issue
17	setRevenueSplitter	external	access only Role	No Issue
18	setRouter	external	access only Role	No Issue
19	setUsdToken	external	access only Role	No Issue
20	setFees	external	access only Role	No Issue
21	pause	write	access only Role	No Issue
22	unpause	write	access only Role	No Issue
23	_checkAndUpdateMintLimit	write	Passed	No Issue
24	_updateAndGetMintLimit	write	Passed	No Issue
25	_getMintLimit	read	Passed	No Issue
26	_isMintLimitExpired	read	Passed	No Issue
27	_setFees	write	Passed	No Issue
28	_swap	write	Passed	No Issue
29	_swapWaifuToUsd	write	Passed	No Issue
30	_swapUsdToWaifu	write	Passed	No Issue
31	_swapTokenToUsd	write	Passed	No Issue
32	_addLiquidity	internal	Add Liquidity with External account	Refer Audit Findings
33	_liquidateWaifu	internal	Passed	No Issue
34	_liquidateUsd	internal	Passed	No Issue
35	initializer	modifier	Passed	No Issue
36	reinitializer	modifier	Passed	No Issue
37	onlyInitializing	modifier	Passed	No Issue
38	_disableInitializers	internal	Passed	No Issue
39	__AccessControl_init	internal	access only Initializing	No Issue
40	__AccessControl_init_unchained	internal	access only Initializing	No Issue
41	onlyRole	modifier	Passed	No Issue
42	supportsInterface	read	Passed	No Issue
43	hasRole	read	Passed	No Issue
44	_checkRole	internal	Passed	No Issue
45	_checkRole	internal	Passed	No Issue
46	getRoleAdmin	read	Passed	No Issue
47	grantRole	write	access only Role	No Issue
48	revokeRole	write	access only Role	No Issue
49	renounceRole	write	Passed	No Issue
50	_setupRole	internal	Passed	No Issue
51	_setRoleAdmin	internal	Passed	No Issue
52	_grantRole	internal	Passed	No Issue
53	_revokeRole	internal	Passed	No Issue
54	__UUPSUpgradeable_init	internal	access only Initializing	No Issue
55	__UUPSUpgradeable_init_unchained	internal	access only Initializing	No Issue
56	onlyProxy	modifier	Passed	No Issue

57	notDelegated	modifier	Passed	No Issue
58	proxiableUUID	external	Passed	No Issue
59	upgradeTo	external	access only Proxy	No Issue
60	upgradeToAndCall	external	access only Proxy	No Issue
61	_authorizeUpgrade	internal	Passed	No Issue
62	__Pausable_init	internal	access only Initializing	No Issue
63	__Pausable_init_unchained	internal	access only Initializing	No Issue
64	whenNotPaused	modifier	Passed	No Issue
65	whenPaused	modifier	Passed	No Issue
66	paused	read	Passed	No Issue
67	_requireNotPaused	internal	Passed	No Issue
68	_requirePaused	internal	Passed	No Issue
69	_pause	internal	Passed	No Issue
70	_unpause	internal	Passed	No Issue

WaifuManager.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	access by initializer	No Issue
3	getEpochCount	read	Passed	No Issue
4	getEpochStartSnapshots	read	Passed	No Issue
5	calculateNodesPrice	read	Passed	No Issue
6	getIncreasedPrice	read	Passed	No Issue
7	getRewardsIncreaseOf	read	Passed	No Issue
8	calculateUnclaimedRewardsFor	read	Passed	No Issue
9	calculateRewardsFor	read	Passed	No Issue
10	buyNodes	external	Passed	No Issue
11	buyNodesBatch	external	Passed	No Issue
12	upgradeNodes	external	Passed	No Issue
13	upgradeNodesBatch	external	Passed	No Issue
14	collectRewards	external	Passed	No Issue
15	collectRewardsUpTo	write	Passed	No Issue
16	setNodePrices	external	access only Role	No Issue
17	addNewNodeTier	external	access only Role	No Issue
18	addNewEpoch	external	access only Role	No Issue
19	pause	write	access only Role	No Issue
20	unpause	write	access only Role	No Issue
21	_setNodePrices	write	Passed	No Issue
22	_setNodeRewards	write	Passed	No Issue
23	_sqrt	write	Passed	No Issue
24	_roundPrice	write	Passed	No Issue

25	_authorizeUpgrade	internal	access only Role	No Issue
26	_authorizeUpgrade	internal	access only Role	No Issue
27	initializer	modifier	Passed	No Issue
28	reinitializer	modifier	Passed	No Issue
29	onlyInitializing	modifier	Passed	No Issue
30	_disableInitializers	internal	Passed	No Issue
31	__AccessControl_init	internal	access only Initializing	No Issue
32	__AccessControl_init_unchained	internal	access only Initializing	No Issue
33	onlyRole	modifier	Passed	No Issue
34	supportsInterface	read	Passed	No Issue
35	hasRole	read	Passed	No Issue
36	_checkRole	internal	Passed	No Issue
37	_checkRole	internal	Passed	No Issue
38	getRoleAdmin	read	Passed	No Issue
39	grantRole	write	access only Role	No Issue
40	revokeRole	write	access only Role	No Issue
41	renounceRole	write	Passed	No Issue
42	_setupRole	internal	Passed	No Issue
43	_setRoleAdmin	internal	Passed	No Issue
44	_grantRole	internal	Passed	No Issue
45	_revokeRole	internal	Passed	No Issue
46	__UUPSUpgradeable_init	internal	access only Initializing	No Issue
47	__UUPSUpgradeable_init_unchained	internal	access only Initializing	No Issue
48	onlyProxy	modifier	Passed	No Issue
49	notDelegated	modifier	Passed	No Issue
50	proxiableUUID	external	Passed	No Issue
51	upgradeTo	external	access only Proxy	No Issue
52	upgradeToAndCall	external	access only Proxy	No Issue
53	_authorizeUpgrade	internal	Passed	No Issue
54	__Pausable_init	internal	access only Initializing	No Issue
55	__Pausable_init_unchained	internal	access only Initializing	No Issue
56	whenNotPaused	modifier	Passed	No Issue
57	whenPaused	modifier	Passed	No Issue
58	paused	read	Passed	No Issue
59	_requireNotPaused	internal	Passed	No Issue
60	_requirePaused	internal	Passed	No Issue
61	_pause	internal	Passed	No Issue
62	_unpause	internal	Passed	No Issue

WaifuNodes.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	DEFAULT_ADMIN_ ROLE is Re-Assigned	Refer Audit Findings
3	getCurrentSnapshotId	read	Passed	No Issue
4	mint	external	access only Role	No Issue
5	mintBatch	external	access only Role	No Issue
6	upgradeNodesFor	external	access only Role	No Issue
7	upgradeNodesBatchFor	external	access only Role	No Issue
8	clearHistoryFor	external	access only Role	No Issue
9	setURI	external	access only Role	No Issue
10	setNodeTierCount	external	access only Role	No Issue
11	setTotalNodeLimit	external	access only Role	No Issue
12	setWalletLimit	external	access only Role	No Issue
13	setTransfersEnabled	external	access only Role	No Issue
14	pause	write	access only Role	No Issue
15	unpause	write	access only Role	No Issue
16	_getLastTokenId	internal	Passed	No Issue
17	_beforeTokenTransfer	internal	Passed	No Issue
18	_afterTokenTransfer	internal	Passed	No Issue
19	_authorizeUpgrade	internal	access only Role	No Issue
20	supportsInterface	read	Passed	No Issue
21	_authorizeUpgrade	internal	access only Role	No Issue
22	initializer	modifier	Passed	No Issue
23	reinitializer	modifier	Passed	No Issue
24	onlyInitializing	modifier	Passed	No Issue
25	_disableInitializers	internal	Passed	No Issue
26	__AccessControl_init	internal	access only Initializing	No Issue
27	__AccessControl_init_un chained	internal	access only Initializing	No Issue
28	onlyRole	modifier	Passed	No Issue
29	supportsInterface	read	Passed	No Issue
30	hasRole	read	Passed	No Issue
31	_checkRole	internal	Passed	No Issue
32	_checkRole	internal	Passed	No Issue
33	getRoleAdmin	read	Passed	No Issue
34	grantRole	write	access only Role	No Issue
35	revokeRole	write	access only Role	No Issue
36	renounceRole	write	Passed	No Issue
37	_setupRole	internal	Passed	No Issue
38	_setRoleAdmin	internal	Passed	No Issue
39	_grantRole	internal	Passed	No Issue
40	_revokeRole	internal	Passed	No Issue

41	__UUPSUpgradeable_init	internal	access only Initializing	No Issue
42	__UUPSUpgradeable_init_unchained	internal	access only Initializing	No Issue
43	onlyProxy	modifier	Passed	No Issue
44	notDelegated	modifier	Passed	No Issue
45	proxiableUUID	external	Passed	No Issue
46	upgradeTo	external	access only Proxy	No Issue
47	upgradeToAndCall	external	access only Proxy	No Issue
48	_authorizeUpgrade	internal	Passed	No Issue
49	__ERC1155_init	internal	access only Initializing	No Issue
50	__ERC1155_init_unchained	internal	access only Initializing	No Issue
51	supportsInterface	read	Passed	No Issue
52	uri	read	Passed	No Issue
53	balanceOf	read	Passed	No Issue
54	balanceOfBatch	read	Passed	No Issue
55	setApprovalForAll	write	Passed	No Issue
56	isApprovedForAll	read	Passed	No Issue
57	safeTransferFrom	write	Passed	No Issue
58	safeBatchTransferFrom	write	Passed	No Issue
59	_safeTransferFrom	internal	Passed	No Issue
60	_safeBatchTransferFrom	internal	Passed	No Issue
61	_setURI	internal	Passed	No Issue
62	_mint	internal	Passed	No Issue
63	_mintBatch	internal	Passed	No Issue
64	_burn	internal	Passed	No Issue
65	_burnBatch	internal	Passed	No Issue
66	_setApprovalForAll	internal	Passed	No Issue
67	_beforeTokenTransfer	internal	Passed	No Issue
68	_afterTokenTransfer	internal	Passed	No Issue
69	_doSafeTransferAcceptanceCheck	write	Passed	No Issue
70	_doSafeBatchTransferAcceptanceCheck	write	Passed	No Issue
71	_asSingletonArray	write	Passed	No Issue
72	__ERC1155Pausable_init	internal	access only Initializing	No Issue
73	__ERC1155Pausable_init_unchained	internal	access only Initializing	No Issue
74	_beforeTokenTransfer	internal	Passed	No Issue
75	__ERC1155AggregateSupply_init	internal	access only Initializing	No Issue
76	__ERC1155AggregateSupply_init_unchained	internal	access only Initializing	No Issue
77	aggregateSupply	read	Passed	No Issue
78	_beforeTokenTransfer	internal	Passed	No Issue

79	__ERC1155TempBalanceHistory_init	internal	access only Initializing	No Issue
80	__ERC1155TempBalanceHistory_init_unchained	internal	access only Initializing	No Issue
81	getCurrentSnapshotId	read	Passed	No Issue
82	getBalanceHistoryOf	read	Passed	No Issue
83	_snapshot	internal	Passed	No Issue
84	_getCurrentSnapshotId	internal	Passed	No Issue
85	_findSnapshotId	read	Passed	No Issue
86	_clearHistoryFor	internal	Passed	No Issue
87	_getLastTokenId	internal	Passed	No Issue
88	_beforeTokenTransfer	internal	Passed	No Issue
89	_updateAccountSnapshots	write	Passed	No Issue
90	_doubleUpdateAccountSnapshots	write	Passed	No Issue
91	_updateAccountSnapshot	write	Passed	No Issue
92	_updateSnapshot	write	Passed	No Issue
93	_lastSnapshotId	read	Passed	No Issue

WaifuPerks.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	DEFAULT_ADMIN_ROLE is Re-Assigned	Refer Audit Findings
3	getTierPercentages	read	Passed	No Issue
4	getTierWalletLimitIncrease	read	Passed	No Issue
5	getTaxReliefOf	read	Passed	No Issue
6	getTransferLimitIncreaseOf	read	Passed	No Issue
7	getRewardsIncreaseOf	read	Passed	No Issue
8	getWalletLimitIncreaseOf	read	Passed	No Issue
9	mint	write	access only Role	No Issue
10	mintBatch	write	Infinite Loop	Refer Audit Findings
11	releasePerkType	external	access only Role	No Issue
12	setURI	write	access only Role	No Issue
13	_getEffectivePercentageOf	read	Passed	No Issue
14	_isValidType	write	Passed	No Issue
15	_getRngSeed	write	Passed	No Issue
16	_randomTier	write	Passed	No Issue
17	_authorizeUpgrade	internal	access only Role	No Issue

18	supportsInterface	read	Passed	No Issue
19	_authorizeUpgrade	internal	access only Role	No Issue
20	initializer	modifier	Passed	No Issue
21	reinitializer	modifier	Passed	No Issue
22	onlyInitializing	modifier	Passed	No Issue
23	_disableInitializers	internal	Passed	No Issue
24	__AccessControl_init	internal	access only Initializing	No Issue
25	__AccessControl_init_unchained	internal	access only Initializing	No Issue
26	onlyRole	modifier	Passed	No Issue
27	supportsInterface	read	Passed	No Issue
28	hasRole	read	Passed	No Issue
29	_checkRole	internal	Passed	No Issue
30	_checkRole	internal	Passed	No Issue
31	getRoleAdmin	read	Passed	No Issue
32	grantRole	write	access only Role	No Issue
33	revokeRole	write	access only Role	No Issue
34	renounceRole	write	Passed	No Issue
35	_setupRole	internal	Passed	No Issue
36	_setRoleAdmin	internal	Passed	No Issue
37	_grantRole	internal	Passed	No Issue
38	_revokeRole	internal	Passed	No Issue
39	__UUPSUpgradeable_init	internal	access only Initializing	No Issue
40	__UUPSUpgradeable_init_unchained	internal	access only Initializing	No Issue
41	onlyProxy	modifier	Passed	No Issue
42	notDelegated	modifier	Passed	No Issue
43	proxiableUUID	external	Passed	No Issue
44	upgradeTo	external	access only Proxy	No Issue
45	upgradeToAndCall	external	access only Proxy	No Issue
46	_authorizeUpgrade	internal	Passed	No Issue

EarlyWaifuHolders.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	access only Initializing	No Issue
3	totalSupply	external	Passed	No Issue
4	safeMintNext	write	access only Role	No Issue
5	safeMintNextBatch	write	Infinite Loop	Refer Audit Findings
6	setRevealedURI	external	access only Role	No Issue

7	setUnrevealedURI	external	access only Role	No Issue
8	setIsRevealed	external	access only Role	No Issue
9	setTransfersEnabled	external	access only Role	No Issue
10	setAccountTransfersEnabled	external	access only Role	No Issue
11	setDefaultRoyalty	external	access only Role	No Issue
12	pause	write	access only Role	No Issue
13	unpause	write	access only Role	No Issue
14	supportsInterface	read	Passed	No Issue
15	_safeMintNext	write	Passed	No Issue
16	_isApprovedOrOwner	internal	Passed	No Issue
17	_baseURI	internal	Passed	No Issue
18	_beforeTokenTransfer	internal	Passed	No Issue
19	_authorizeUpgrade	internal	access only Role	No Issue
20	initializer	modifier	Passed	No Issue
21	reinitializer	modifier	Passed	No Issue
22	onlyInitializing	modifier	Passed	No Issue
23	_disableInitializers	internal	Passed	No Issue
24	__ERC721_init	internal	access only Initializing	No Issue
25	__ERC721_init_unchained	internal	access only Initializing	No Issue
26	supportsInterface	read	Passed	No Issue
27	balanceOf	read	Passed	No Issue
28	ownerOf	read	Passed	No Issue
29	name	read	Passed	No Issue
30	symbol	read	Passed	No Issue
31	tokenURI	read	Passed	No Issue
32	_baseURI	internal	Passed	No Issue
33	approve	write	Passed	No Issue
34	getApproved	read	Passed	No Issue
35	setApprovalForAll	write	Passed	No Issue
36	isApprovedForAll	read	Passed	No Issue
37	transferFrom	write	Passed	No Issue
38	safeTransferFrom	write	Passed	No Issue
39	safeTransferFrom	write	Passed	No Issue
40	_safeTransfer	internal	Passed	No Issue
41	_exists	internal	Passed	No Issue
42	_isApprovedOrOwner	internal	Passed	No Issue
43	_safeMint	internal	Passed	No Issue
44	_safeMint	write	Passed	No Issue
45	_mint	internal	Passed	No Issue
46	_burn	internal	Passed	No Issue
47	_transfer	internal	Passed	No Issue
48	_approve	internal	Passed	No Issue
49	_setApprovalForAll	internal	Passed	No Issue
50	_requireMinted	internal	Passed	No Issue

51	_checkOnERC721Received	write	Passed	No Issue
52	_beforeTokenTransfer	internal	Passed	No Issue
53	__ERC721Pausable_init	internal	access only Initializing	No Issue
54	_afterTokenTransfer	internal	Passed	No Issue
55	__ERC721Pausable_init_unchained	internal	access only Initializing	No Issue
56	_beforeTokenTransfer	internal	Passed	No Issue
57	__ERC2981_init	internal	access only Initializing	No Issue
58	__ERC2981_init_unchained	internal	access only Initializing	No Issue
59	supportsInterface	read	Passed	No Issue
60	royaltyInfo	read	Passed	No Issue
61	_feeDenominator	internal	Passed	No Issue
62	_setDefaultRoyalty	internal	Passed	No Issue
63	_deleteDefaultRoyalty	internal	Passed	No Issue
64	_setTokenRoyalty	internal	Passed	No Issue
65	_resetTokenRoyalty	internal	Passed	No Issue
66	__AccessControl_init	internal	access only Initializing	No Issue
67	__AccessControl_init_unchained	internal	access only Initializing	No Issue
68	onlyRole	modifier	Passed	No Issue
69	supportsInterface	read	Passed	No Issue
70	hasRole	read	Passed	No Issue
71	_checkRole	internal	Passed	No Issue
72	_checkRole	internal	Passed	No Issue
73	getRoleAdmin	read	Passed	No Issue
74	grantRole	write	access only Role	No Issue
75	revokeRole	write	access only Role	No Issue
76	renounceRole	write	Passed	No Issue
77	_setupRole	internal	Passed	No Issue
78	_setRoleAdmin	internal	Passed	No Issue
79	_grantRole	internal	Passed	No Issue
80	_revokeRole	internal	Passed	No Issue
81	__UUPSUpgradeable_init	internal	access only Initializing	No Issue
82	__UUPSUpgradeable_init_unchained	internal	access only Initializing	No Issue
83	onlyProxy	modifier	Passed	No Issue
84	notDelegated	modifier	Passed	No Issue
85	proxiableUUID	external	Passed	No Issue
86	upgradeTo	external	access only Proxy	No Issue
87	upgradeToAndCall	external	access only Proxy	No Issue
88	_authorizeUpgrade	internal	Passed	No Issue

RevenuePaymentSplitter.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	access only Initializing	No Issue
3	receive	external	Passed	No Issue
4	totalShares	read	Passed	No Issue
5	totalReleased	read	Passed	No Issue
6	totalReleased	read	Passed	No Issue
7	shares	read	Passed	No Issue
8	released	read	Passed	No Issue
9	released	read	Passed	No Issue
10	payee	read	Passed	No Issue
11	release	write	Passed	No Issue
12	release	write	Passed	No Issue
13	_pendingPayment	read	Passed	No Issue
14	_addPayee	write	Passed	No Issue
15	_authorizeUpgrade	internal	access only Role	No Issue
16	initializer	modifier	Passed	No Issue
17	reinitializer	modifier	Passed	No Issue
18	onlyInitializing	modifier	Passed	No Issue
19	_disableInitializers	internal	Passed	No Issue
20	__Context_init	internal	access only Initializing	No Issue
21	__Context_init_unchained	internal	access only Initializing	No Issue
22	_msgSender	internal	Passed	No Issue
23	_msgData	internal	Passed	No Issue
24	__AccessControl_init	internal	access only Initializing	No Issue
25	__AccessControl_init_unchained	internal	access only Initializing	No Issue
26	onlyRole	modifier	Passed	No Issue
27	supportsInterface	read	Passed	No Issue
28	hasRole	read	Passed	No Issue
29	_checkRole	internal	Passed	No Issue
30	_checkRole	internal	Passed	No Issue
31	getRoleAdmin	read	Passed	No Issue
32	grantRole	write	access only Role	No Issue
33	revokeRole	write	access only Role	No Issue
34	renounceRole	write	Passed	No Issue
35	__UUPSUpgradeable_init	internal	access only Initializing	No Issue
36	__UUPSUpgradeable_init_unchained	internal	access only Initializing	No Issue
37	onlyProxy	modifier	Passed	No Issue

38	notDelegated	modifier	Passed	No Issue
39	proxiableUUID	external	Passed	No Issue
40	upgradeTo	external	access only Proxy	No Issue
41	upgradeToAndCall	external	access only Proxy	No Issue

PreLaunchToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	access only Initializing	No Issue
3	mint	write	Passed	No Issue
4	withdrawTo	write	Passed	No Issue
5	withdrawFrom	write	access only Role	No Issue
6	setMainToken	external	access only Role	No Issue
7	_withdraw	write	Passed	No Issue
8	_beforeTokenTransfer	internal	Passed	No Issue
9	_authorizeUpgrade	internal	access only Role	No Issue
10	_mint	internal	Passed	No Issue
11	initializer	modifier	DEFAULT_ADMIN_ ROLE is Re-Assigned	Refer Audit Findings
12	reinitializer	modifier	Passed	No Issue
13	onlyInitializing	modifier	Passed	No Issue
14	_disableInitializers	internal	Passed	No Issue
15	__AccessControl_init	internal	access only Initializing	No Issue
16	__AccessControl_init_unchained	internal	access only Initializing	No Issue
17	onlyRole	modifier	Passed	No Issue
18	supportsInterface	read	Passed	No Issue
19	hasRole	read	Passed	No Issue
20	_checkRole	internal	Passed	No Issue
21	_checkRole	internal	Passed	No Issue
22	getRoleAdmin	read	Passed	No Issue
23	grantRole	write	access only Role	No Issue
24	revokeRole	write	access only Role	No Issue
25	renounceRole	write	Passed	No Issue
26	_setupRole	internal	Passed	No Issue
27	_setRoleAdmin	internal	Passed	No Issue
28	_grantRole	internal	Passed	No Issue
29	_revokeRole	internal	Passed	No Issue
30	__UUPSUpgradeable_init	internal	access only Initializing	No Issue

31	__UUPSUpgradeable_init _unchained	internal	access only Initializing	No Issue
32	onlyProxy	modifier	Passed	No Issue
33	notDelegated	modifier	Passed	No Issue
34	proxiableUUID	external	Passed	No Issue
35	upgradeTo	external	access only Proxy	No Issue
36	upgradeToAndCall	external	access only Proxy	No Issue
37	_authorizeUpgrade	internal	Passed	No Issue
38	__ERC20_init	internal	access only Initializing	No Issue
39	__ERC20_init_unchained	internal	access only Initializing	No Issue
40	name	read	Passed	No Issue
41	symbol	read	Passed	No Issue
42	decimals	read	Passed	No Issue
43	totalSupply	read	Passed	No Issue
44	balanceOf	write	Passed	No Issue
45	transfer	write	Passed	No Issue
46	allowance	read	Passed	No Issue
47	approve	write	Passed	No Issue
48	transferFrom	write	Passed	No Issue
49	increaseAllowance	write	Passed	No Issue
50	decreaseAllowance	write	Passed	No Issue
51	_transfer	internal	Passed	No Issue
52	_mint	internal	Passed	No Issue
53	_burn	internal	Passed	No Issue
54	_approve	internal	Passed	No Issue
55	_spendAllowance	internal	Passed	No Issue
56	_beforeTokenTransfer	internal	Passed	No Issue
57	_afterTokenTransfer	internal	Passed	No Issue

WaifuToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	DEFAULT_ADMIN_ ROLE is Re-Assigned	Refer Audit Findings
3	getTransferLimitPercentageOf	read	Passed	No Issue
4	getTransferTaxOf	read	Passed	No Issue
5	getWalletLimitOf	read	Passed	No Issue
6	mint	write	Unlimited Minting	Refer Audit Findings
7	burn	write	access only Role	No Issue

8	setWaifuCashier	external	access only Role	No Issue
9	setLiquidityManager	external	access only Role	No Issue
10	setDefaultWalletLimit	external	access only Role	No Issue
11	setTransferLimitDuration	external	access only Role	No Issue
12	setDefaultTransferLimitPercentage	external	access only Role	No Issue
13	setDefaultTransferTax	external	Tax limit is not set	Refer Audit Findings
14	setAccountLimitsDisabled	external	access only Role	No Issue
15	setAccountTaxDisabled	external	access only Role	No Issue
16	approveForLiquidityManger	external	Passed	No Issue
17	_checkAndUpdateTransferLimitOf	write	Passed	No Issue
18	_applyTransferTax	write	Passed	No Issue
19	_checkWalletLimit	read	Passed	No Issue
20	_beforeTokenTransfer	internal	Passed	No Issue
21	_afterTokenTransfer	internal	Passed	No Issue
22	_authorizeUpgrade	internal	access only Role	No Issue
23	__ERC20_init	internal	access only Initializing	No Issue
24	__ERC20_init_unchained	internal	access only Initializing	No Issue
25	name	read	Passed	No Issue
26	symbol	read	Passed	No Issue
27	decimals	read	Passed	No Issue
28	totalSupply	read	Passed	No Issue
29	balanceOf	write	Passed	No Issue
30	transfer	write	Passed	No Issue
31	allowance	read	Passed	No Issue
32	approve	write	Passed	No Issue
33	transferFrom	write	Passed	No Issue
34	increaseAllowance	write	Passed	No Issue
35	decreaseAllowance	write	Passed	No Issue
36	_transfer	internal	Passed	No Issue
37	_mint	internal	Passed	No Issue
38	_burn	internal	Passed	No Issue
39	_approve	internal	Passed	No Issue
40	_spendAllowance	internal	Passed	No Issue
41	_beforeTokenTransfer	internal	Passed	No Issue
42	_afterTokenTransfer	internal	Passed	No Issue
43	__AccessControl_init	internal	access only Initializing	No Issue
44	__AccessControl_init_unchained	internal	access only Initializing	No Issue
45	onlyRole	modifier	Passed	No Issue
46	supportsInterface	read	Passed	No Issue
47	hasRole	read	Passed	No Issue

48	_checkRole	internal	Passed	No Issue
49	_checkRole	internal	Passed	No Issue
50	getRoleAdmin	read	Passed	No Issue
51	grantRole	write	access only Role	No Issue
52	revokeRole	write	access only Role	No Issue
53	renounceRole	write	Passed	No Issue
54	_setupRole	internal	Passed	No Issue
55	_setRoleAdmin	internal	Passed	No Issue
56	_grantRole	internal	Passed	No Issue
57	_revokeRole	internal	Passed	No Issue
58	__UUPSUpgradeable_init	internal	access only Initializing	No Issue
59	__UUPSUpgradeable_init_unchained	internal	access only Initializing	No Issue
60	onlyProxy	modifier	Passed	No Issue
61	notDelegated	modifier	Passed	No Issue
62	proxiableUUID	external	Passed	No Issue
63	upgradeTo	external	access only Proxy	No Issue
64	upgradeToAndCall	external	access only Proxy	No Issue
65	_authorizeUpgrade	internal	Passed	No Issue

Severity Definitions

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

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

(1) Add Liquidity with External account: **- WaifuCashier.sol**

```
function _addLiquidity(uint256 waifuBalance, uint256 usdBalance) internal {
    if (waifuBalance == 0 || usdBalance == 0) {
        return;
    }

    waifuToken.approve(address(router), waifuBalance);
    usdToken.approve(address(router), usdBalance);
    if (waifuBalance != 0 && usdBalance != 0) {
        router.addLiquidity(
            address(waifuToken),
            address(usdToken),
            waifuBalance,
            usdBalance,
            0,
            0,
            treasury,
            block.timestamp
        );
    }
}
```

addLiquidity function of pancakeswapRouter with the address specified as treasury for acquiring the generated LP tokens from the WaifuToken-WBNB pool. As a result, over time the treasury address will accumulate a significant portion of LP tokens. If the treasury is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Resolution: We advise the address of the addLiquidity function call to be replaced by the contract itself, i.e. address(this), and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the treasury account is compromised.

Medium

(1) Tax limit is not set:

WaifuToken.sol

```
function setDefaultTransferTax(uint256 tax)
    external
    onlyRole(LIMITS_ADMIN_ROLE)
{
    defaultTransferTax = tax;

    emit NewDefaultTransferTax(tax);
}
```

WaifuCashier.sol

```
function setDefaultClaimTax(uint256 tax)
    external
    onlyRole(FEES_ADMIN_ROLE)
{
    defaultClaimTax = tax;

    emit NewDefaultClaimTax(tax);
}
```

Operators can set the tax to any variable. This might deter investors as they could be wary that these taxes might one day be set to 100% to force transfers to go to the contract admin role.

Resolution: Consider adding an explicit limit while setting the defaultTransferTax value.

(2) DEFAULT_ADMIN_ROLE is Re-Assigned:-

PreLaunchToken.sol

```
function initialize(
    uint256 cap,
    uint256 _earlyLimit,
    address admin
) public initializer {
    __ERC20_init("PreLaunch UwU Token", "$PUT");
    __AccessControlEnumerable_init();
    __ERC20Capped_init(cap);
    __UUPSUpgradeable_init();

    earlyLimit = _earlyLimit;

    _grantRole(DEFAULT_ADMIN_ROLE, admin);
    _grantRole(MINTER_ROLE, admin);
    _grantRole(TOKEN_SETTER_ROLE, admin);
    _grantRole(UPGRADER_ROLE, admin);

    if (admin != _msgSender()) {
        _grantRole(DEFAULT_ADMIN_ROLE, _msgSender());
    }
}
```

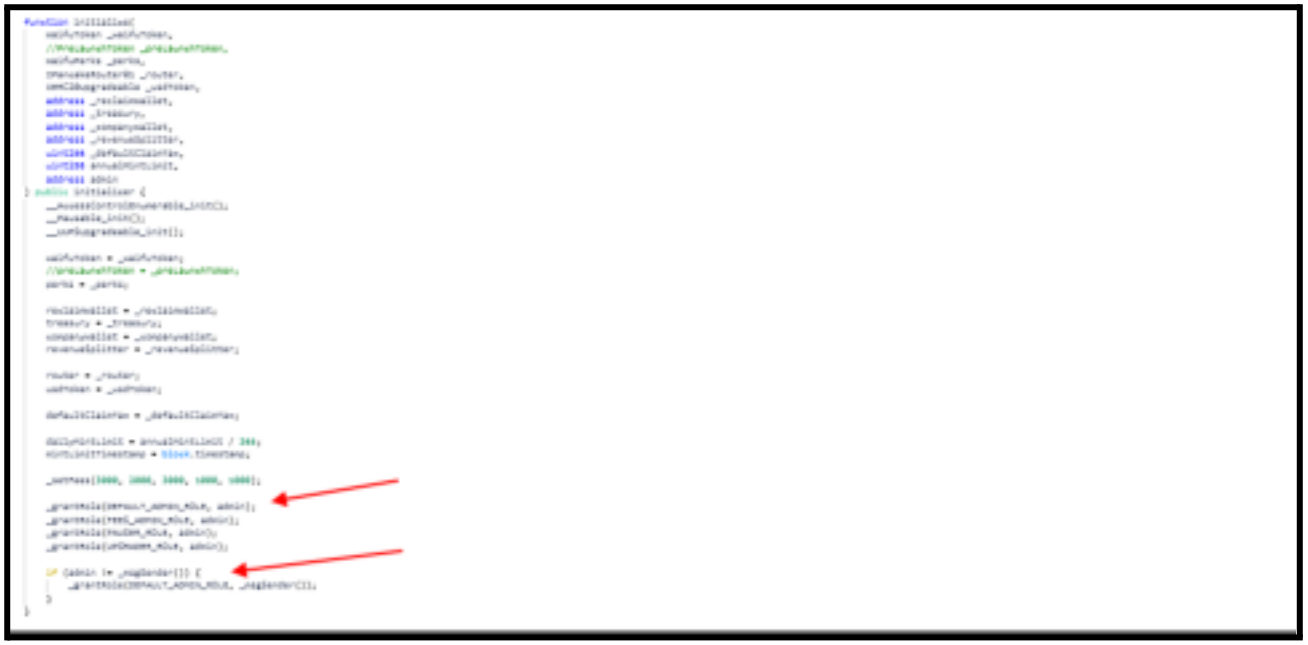
WaifuNodes.sol

```
function initialize(
    string memory _uri,
    uint256 _nodeTierCount,
    uint256 _totalNodeLimit,
    uint256 _walletLimit,
    address admin
) public initializer {
    __ERC1155_init(_uri);
    __AccessControl_init();
    __ERC1155Pausable_init();
    __ERC1155TempBalanceHistory_init();
    __ERC1155AggregateSupply_init();
    __UUPSUpgradeable_init();

    nodeTierCount = _nodeTierCount;
    walletLimit = _walletLimit;
    totalNodeLimit = _totalNodeLimit;

    deployTime = block.timestamp;

    _grantRole(DEFAULT_ADMIN_ROLE, admin);
    _grantRole(LIMITS_ADMIN_ROLE, admin);
    _grantRole(URI_SETTER_ROLE, admin);
    _grantRole(PAUSER_ROLE, admin);
    _grantRole(UPGRADER_ROLE, admin);

    if (admin != _msgSender()) {
        _grantRole(DEFAULT_ADMIN_ROLE, _msgSender());
    }
```

WaifuToken.sol

```
function initialize(
    WaifuPerks _perks,
    uint256 _defaultTransferTax,
    uint256 _defaultWalletLimit,
    uint256 _defaultTransferLimitPercentage,
    uint256 _transferLimitDuration,
    address admin
) public initializer {
    __ERC20_init("UWU Token", "UWU");
    __AccessControlEnumerable_init();
    __UUPSUpgradeable_init();

    require(
        _perks.PRECISION() == PRECISION,
        "WaifuToken: invalid precision"
    );

    perks = _perks;
    defaultTransferTax = _defaultTransferTax;
    defaultWalletLimit = _defaultWalletLimit;
    defaultTransferLimitPercentage = _defaultTransferLimitPercentage;
    transferLimitDuration = _transferLimitDuration;

    _grantRole(DEFAULT_ADMIN_ROLE, admin);
    _grantRole(MINTER_ROLE, admin);
    _grantRole(LIMITS_ADMIN_ROLE, admin);
    _grantRole(ADDRESS_ADMIN_ROLE, admin);
    _grantRole(UPGRADER_ROLE, admin);

    if (admin != _msgSender()) {
        _grantRole(ADDRESS_ADMIN_ROLE, _msgSender());
    }
}
```

WaifuCashier.sol



WaifuPerks.sol

```
function initialize(
    string memory uri,
    uint256[TIER_COUNT] calldata tierPercentages,
    uint256[TIER_COUNT] calldata tierWalletLimitIncrease,
    address admin
) public initializer {
    __ERC1155_init(uri);
    __AccessControl_init();
    __UUPSUpgradeable_init();

    for (uint256 i = 0; i < tierPercentages.length; i++) {
        require(
            tierPercentages[i] <= MAX_PERCENTAGE,
            "WaifuPerks: percentage > max"
        );
    }

    _tierPercentages = tierPercentages;
    _tierWalletLimitIncrease = tierWalletLimitIncrease;

    _grantRole(DEFAULT_ADMIN_ROLE, admin);
    _grantRole(URI_SETTER_ROLE, admin);
    _grantRole(RELEASER_ROLE, admin);

    if (admin != _msgSender()) {
        _grantRole(DEFAULT_ADMIN_ROLE, _msgSender());
    }
}
```

The DEFAULT_ADMIN_ROLE will be re-assigned to the caller of function initialize().

Resolution: We suggest to re-check the logic.If this is a desired feature, then please ignore this point.

Low

(1) Infinite Loop:

WaifuPerks.sol

```
function mintBatch(
    address to,
    uint256 perkType,
    uint256 amount,
    bytes memory data
) public onlyRole(MINTER_ROLE) {
    require(perkTypeReleased[perkType], "WaifuPerks: not released");

    bytes32 seed = _getRngSeed();
    uint256[] memory ids = new uint256[](amount);
    uint256[] memory amounts = new uint256[](amount);

    for (uint256 i = 0; i < amount; i++) {
        uint256 tier = _randomTier(seed);
        seed = keccak256(abi.encodePacked(seed));

        ids[i] = perkType + tier;
        amounts[i] = 1;
    }

    _mintBatch(to, ids, amounts, data);
}
```

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

EarlyWaifuHolders.sol

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

Resolution: upper bound should have a certain limit for loops.

(2) Critical operation lacks event log:**- PerkSaleHelper.sol**

Missing event log for: purchase.

Resolution: Write an event log for listed events.

Very Low / Informational / Best practices:

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

Operators can mint unlimited tokens.

Resolution: We suggest putting a minting limit.

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:

- finishPresale: PerkSaleHelper owner can set the finish presale.
- setTypePrice: PerkSaleHelper owner can set type price.
- setTypePresalePrice: PerkSaleHelper owner can set type presale price.
- _authorizeUpgrade: PerkSaleHelper owner can set authorize upgrade address.
- setPurchaseAllowed: PresaleHelper owner can set purchase allowed address.
- setPurchaseAllowedBatch: PresaleHelper owner can set purchase allowed addresses batch vise.
- _authorizeUpgrade: PresaleHelper owner can set authorize upgrade address.
- setPriceRange: LiquidityManager owner can set price range.
- setIsEnabled: LiquidityManager owner can set its enabled status.
- adminWithdraw: LiquidityManager owner can set admin withdraw address.
- adminWithdrawETH: LiquidityManager owner can set admin payable withdraw ETH address.
- getPaymentFrom: WaifuCashier owner can get the payment address and amount.
- grantRewardsFor: WaifuCashier owner can grant rewards address.
- liquidateToken: WaifuCashier owner can liquidate tokens.
- setAnnualMintLimit: WaifuCashier owner can set annual mint limit
- setDefaultClaimTax: WaifuCashier owner can set default claim tax.
- setTreasury: WaifuCashier owner can set a new treasury address.
- setCompanyWallet: WaifuCashier owner can set a new company wallet address.
- setRevenueSplitter: WaifuCashier owner can set a new revenue splitter address.
- setRouter: WaifuCashier owner can set a new router address.
- setUsdToken: WaifuCashier owner can set new USD tokens.
- setFees: WaifuCashier owner can set fees like: ReclaimFee, TreasuryFee, LiquidityFee, CompanyFee, RevenueSplitterFee.
- pause: WaifuCashier owner can trigger a stopped state.
- unpause: WaifuCashier owner can return to normal state.
- _authorizeUpgrade: WaifuCashier owner can set authorize upgrade address.

- setNodePrices: WaifuManager owner can set node prices.
- addNewNodeTier: WaifuManager owner can add a new node tier.
- addNewEpoch: WaifuManager owner can add a new epoch.
- pause: WaifuManager owner can trigger a stopped state.
- unpause: WaifuManager owner can return to normal state.
- _authorizeUpgrade: WaifuManager owner can set authorize upgrade address.
- mint: WaifuNodes owner can mint a token.
- mintBatch: WaifuNodes owner can mint a token batch vise.
- upgradeNodesFor: WaifuNodes owner can upgrade nodes.
- upgradeNodesBatchFor: WaifuNodes owner can upgrade nodes batch vise.
- clearHistoryFor: WaifuNodes owner can clear history address.
- setURI: WaifuNodes owner can set the URI.
- setNodeTierCount: WaifuNodes owner can set node tier count.
- setTotalNodeLimit: WaifuNodes owner can set the total node limit.
- setWalletLimit: WaifuNodes owner can set wallet limit.
- setTransfersEnabled: WaifuNodes owner can set transfers enabled status.
- pause: WaifuNodes owner can trigger a stopped state.
- unpause: WaifuNodes owner can return to normal state.
- _authorizeUpgrade: WaifuNodes owner can set authorize upgrade address.
- mint: WaifuPerks owner can mint tokens.
- mintBatch: WaifuPerks owner can mint tokens batch.
- releasePerkType: WaifuPerks owner can release perk type.
- setURI: WaifuPerks owner can set the URI.
- _authorizeUpgrade: WaifuPerks owner can set authorize upgrade address.
- safeMintNext: EarlyWaifuHolders owner can safe mint next token address.
- safeMintNextBatch: EarlyWaifuHolders owner can safe mint next token batch vise.
- setRevealedURI: EarlyWaifuHolders owner can set revealed URI.
- setUnrevealedURI: EarlyWaifuHolders owner can set unrevealed URI.
- setIsRevealed: EarlyWaifuHolders owner can set IS revealed status.
- setTransfersEnabled: EarlyWaifuHolders owner can set transfers enabled status.
- setAccountTransfersEnabled: EarlyWaifuHolders owner can set account transfers enabled status.
- setDefaultRoyalty: EarlyWaifuHolders owner can set default royalty address.
- pause: EarlyWaifuHolders owner can trigger a stopped state.

- unpause: EarlyWaifuHolders owner can return to normal state.
- _authorizeUpgrade: EarlyWaifuHolders owner can set authorize upgrade address.
- _authorizeUpgrade: RevenuePaymentSplitter owner can set authorize upgrade address.
- withdrawFrom: PreLaunchToken owner can withdraw amount from address.
- setMainToken: PreLaunchToken owner can set main token address.
- _authorizeUpgrade: PreLaunchToken owner can set authorize upgrade address.
- mint: WaifuToken owner can mint a token.
- burn: WaifuToken owner can burn a token.
- setWaifuCashier: WaifuToken owner can set the waifu cashier address.
- setLiquidityManager: WaifuToken owner can set the liquidity manager address.
- setDefaultWalletLimit: WaifuToken owner can set default wallet limit.
- setTransferLimitDuration: WaifuToken owner can set transfer limit duration.
- setDefaultTransferLimitPercentage: WaifuToken owner can set default transfer limit percentage.
- setDefaultTransferTax: WaifuToken owner can set default transfer tax.
- setAccountLimitsDisabled: WaifuToken owner can set account disabled limits.
- setAccountTaxDisabled: WaifuToken owner can set account tax disabled.
- approveForLiquidityManger: WaifuToken owner can approve the liquidity manager address.
- _authorizeUpgrade: WaifuToken owner can set authorize upgrade 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 observed 1 High Severity issue, 2 Medium Severity issue, 3 low Severity issue and some Informational issues in the smart contracts. **So, the smart contracts are ready for the mainnet deployment**.

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

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

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

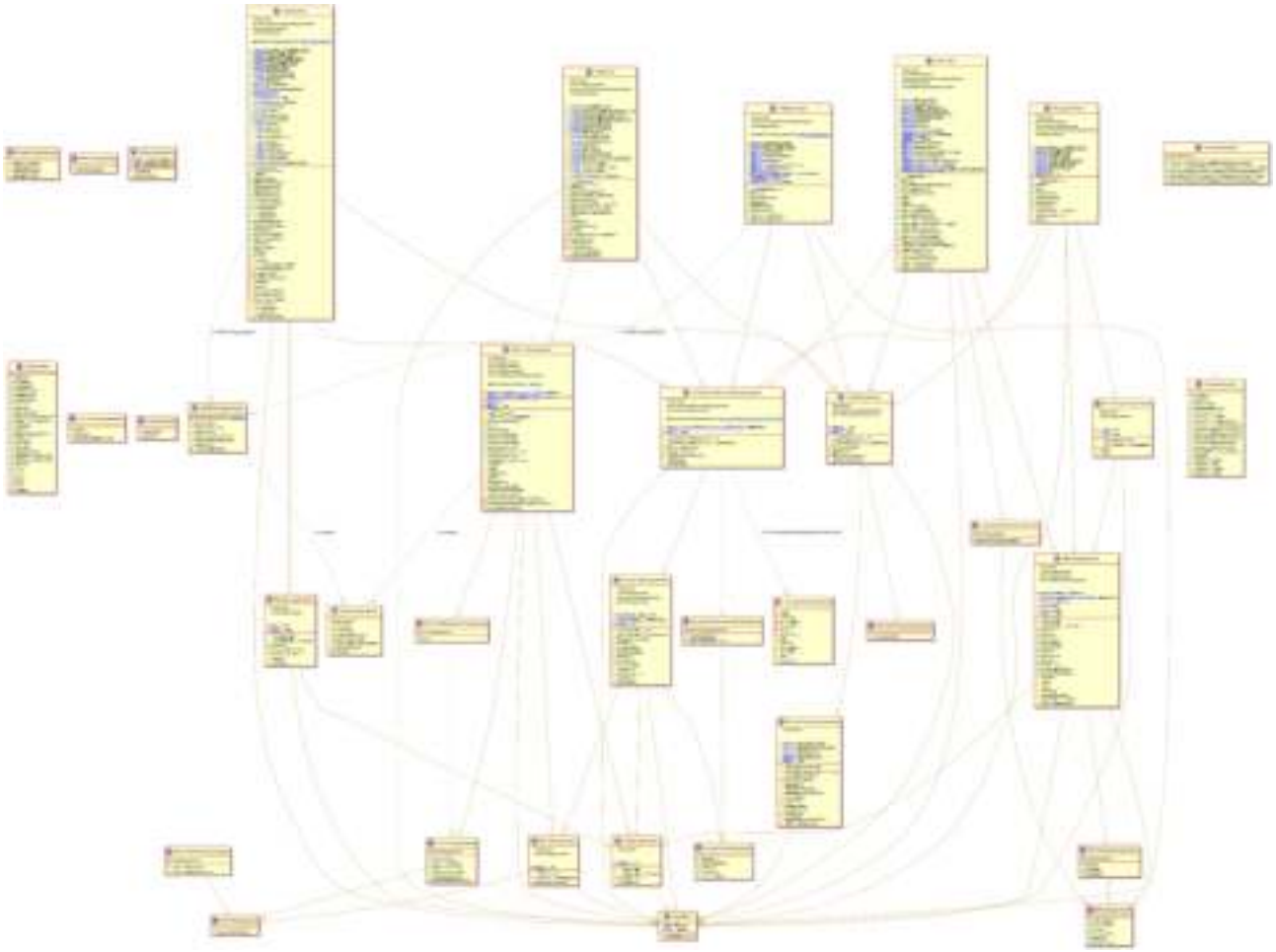
Technical Disclaimer

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

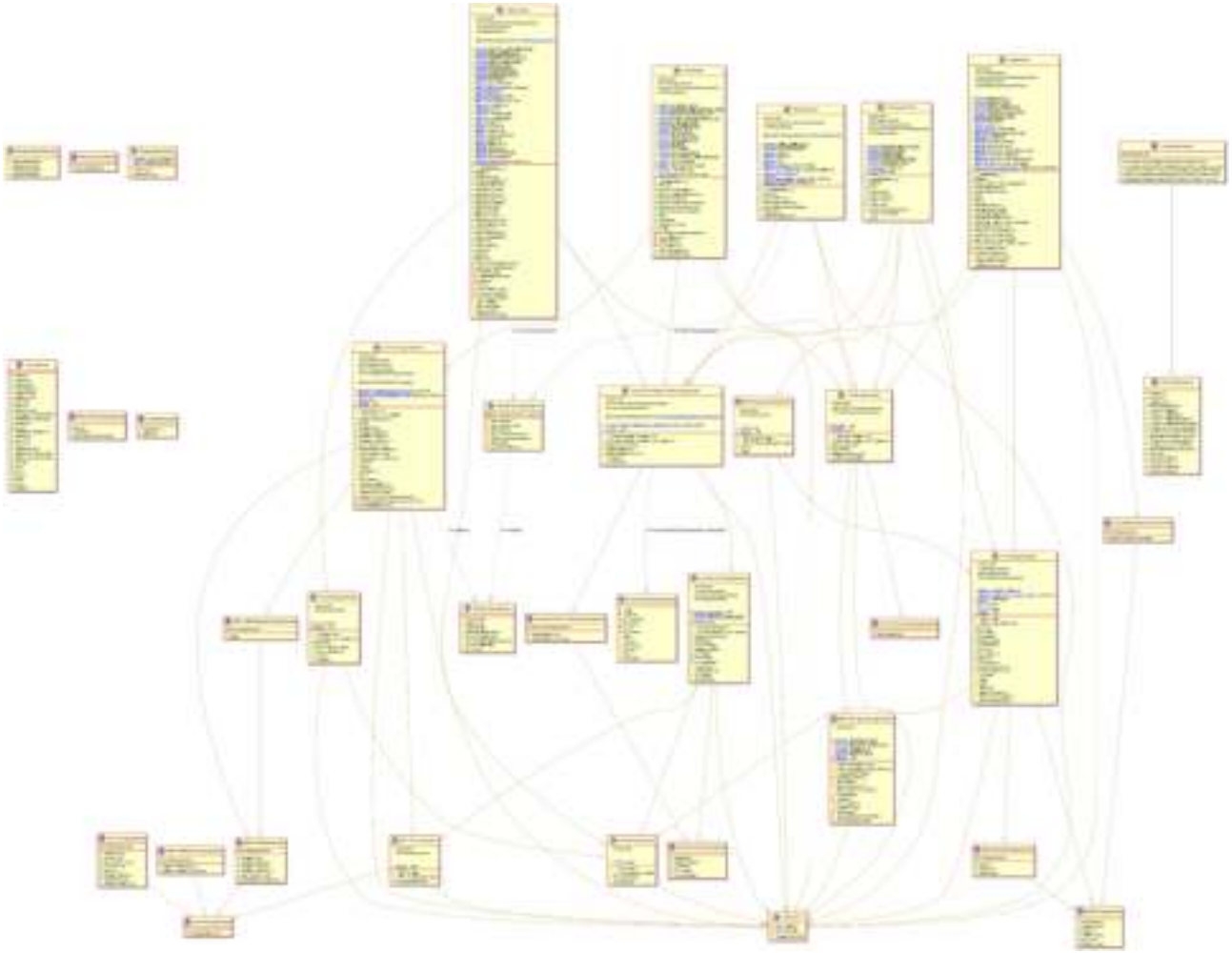
Appendix

Code Flow Diagram - Waifu Protocol

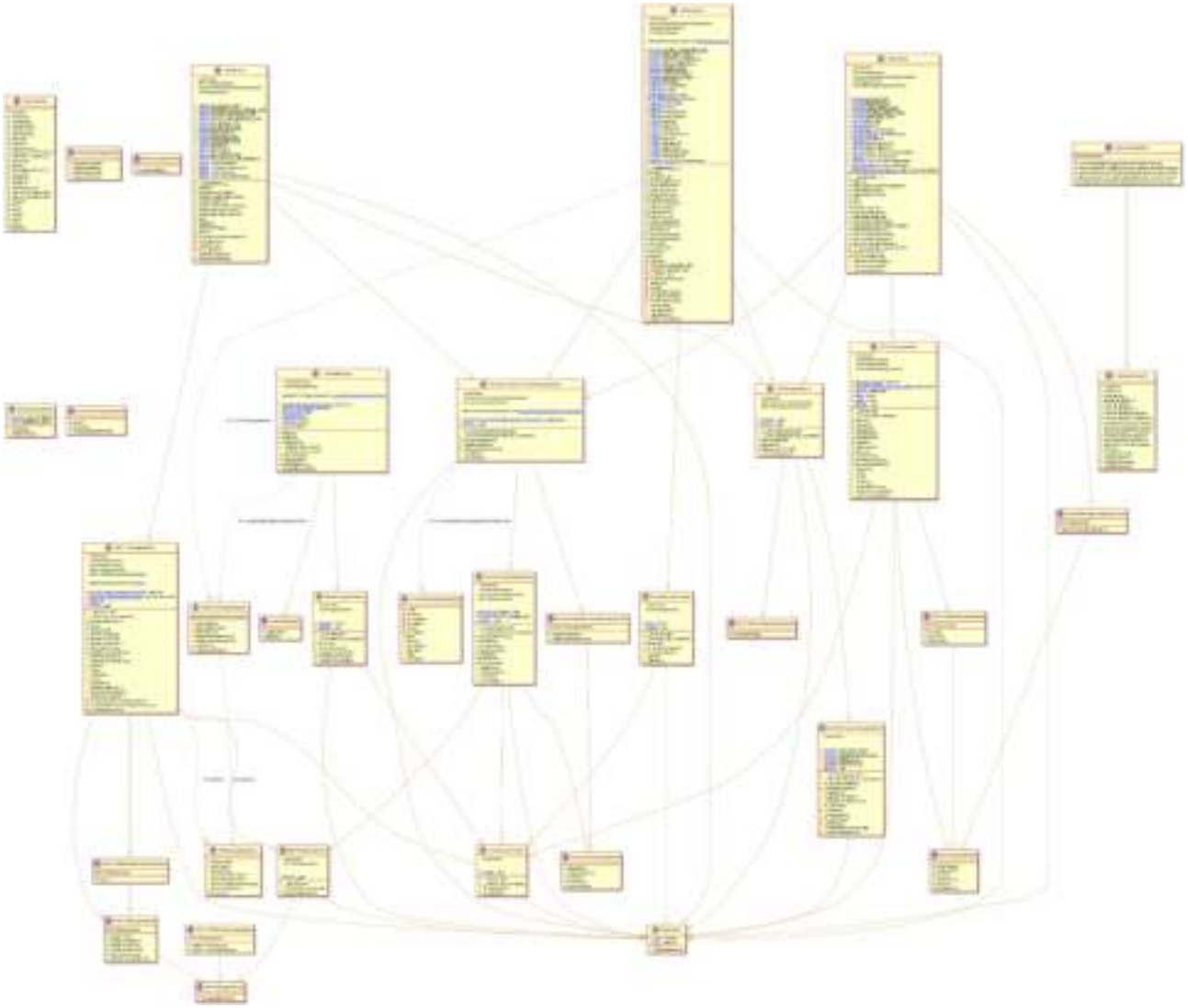
PerkSaleHelper Diagram



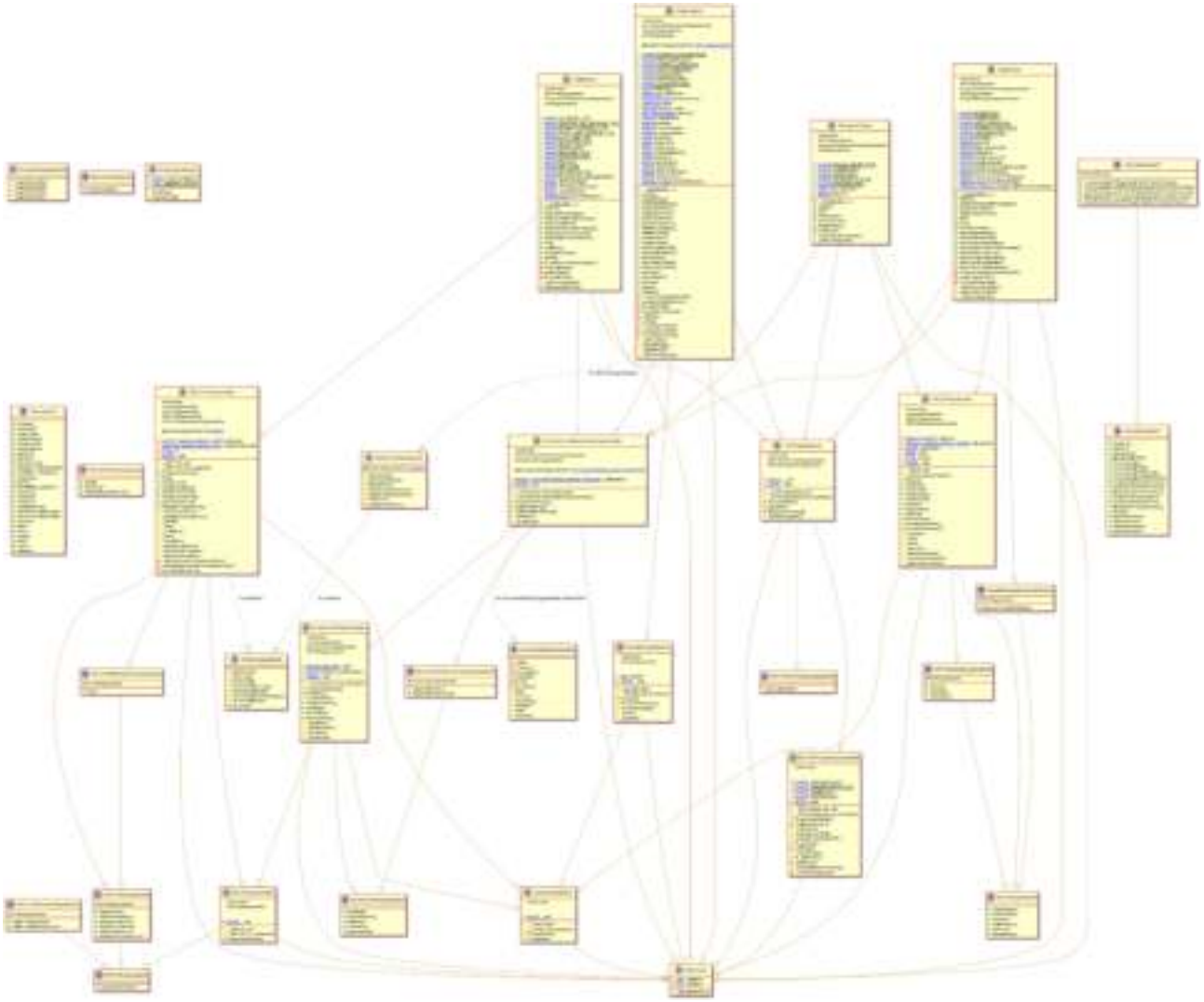
PresaleHelper Diagram



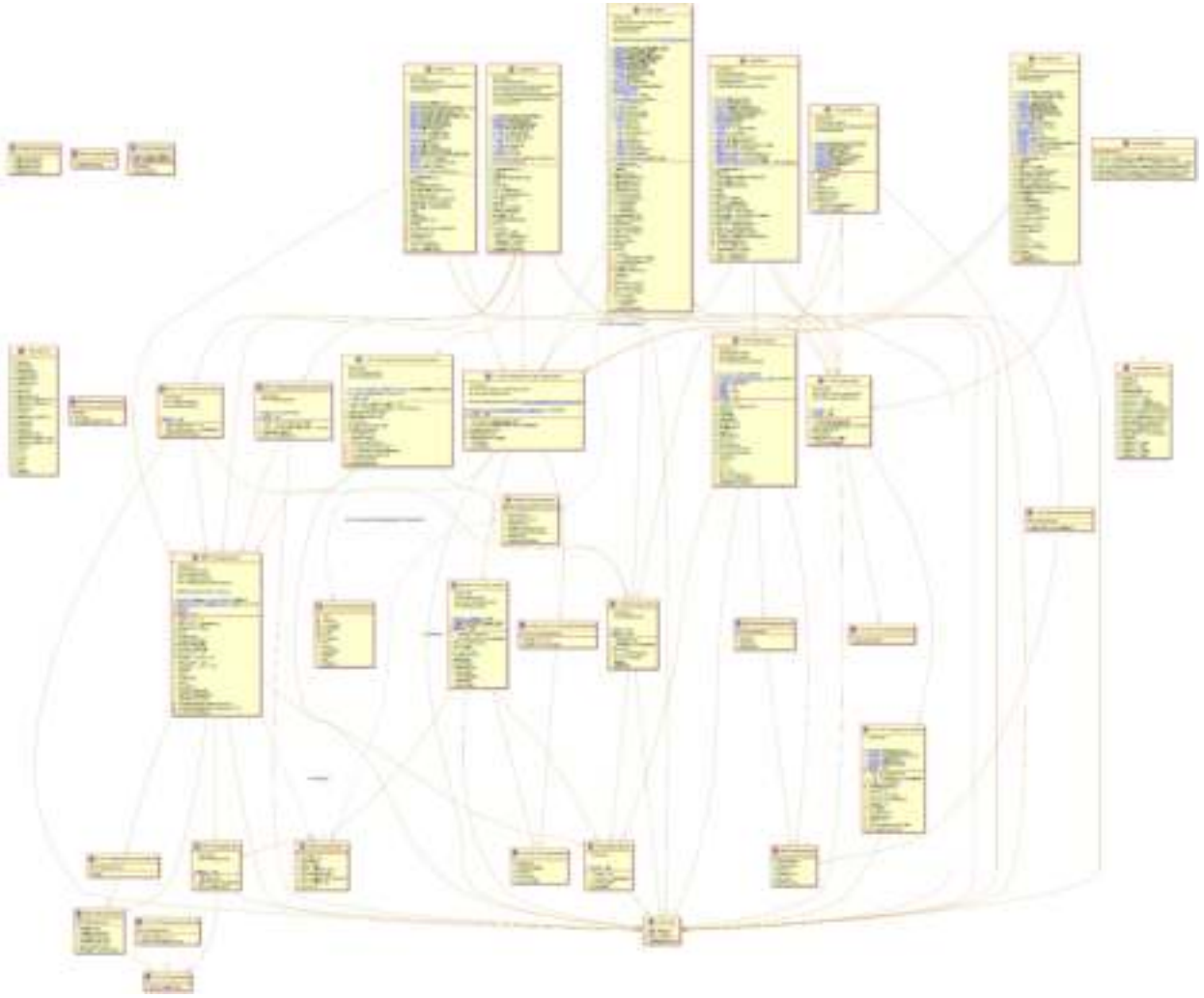
LiquidityManager Diagram



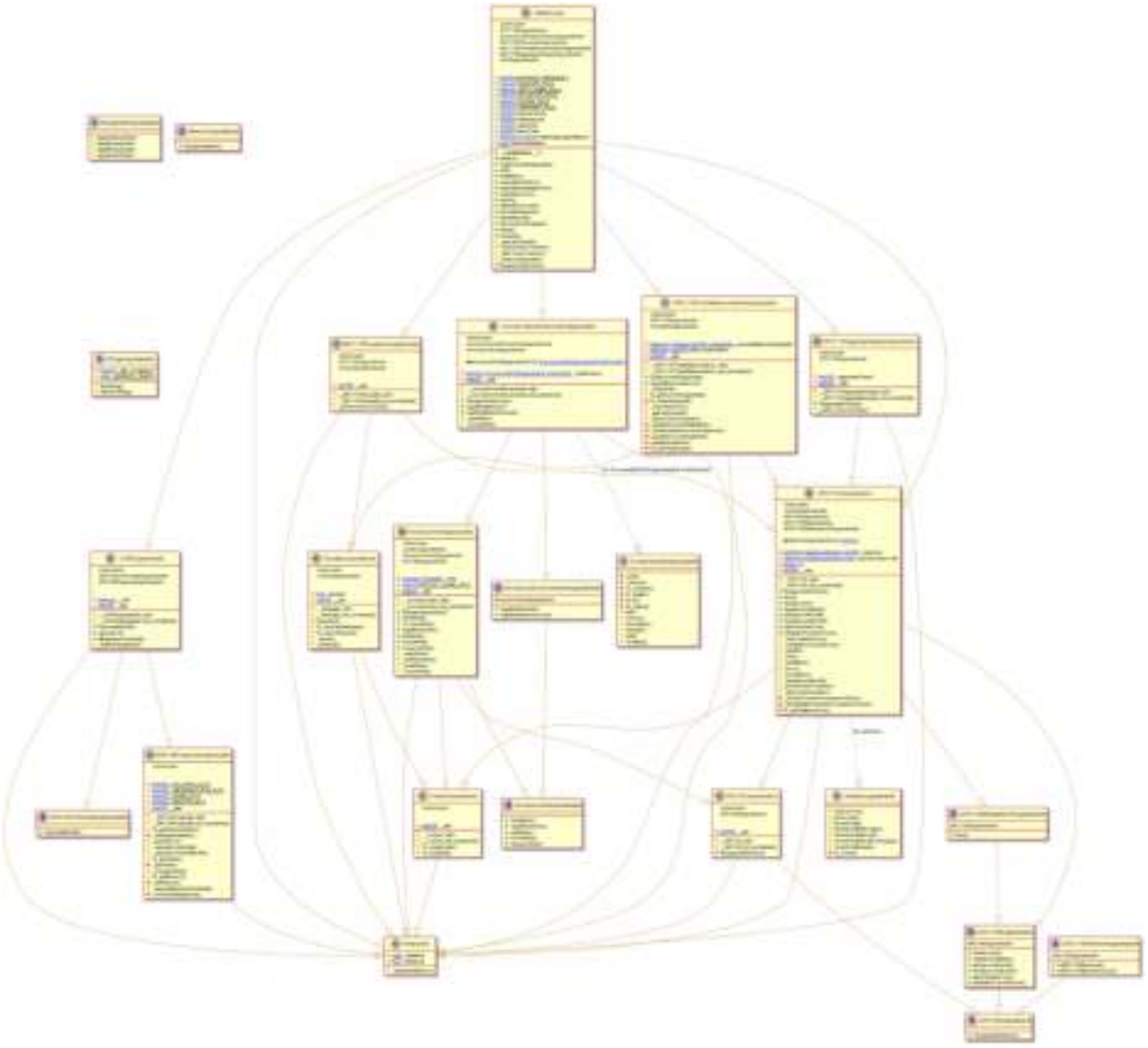
WaifuCashier Diagram



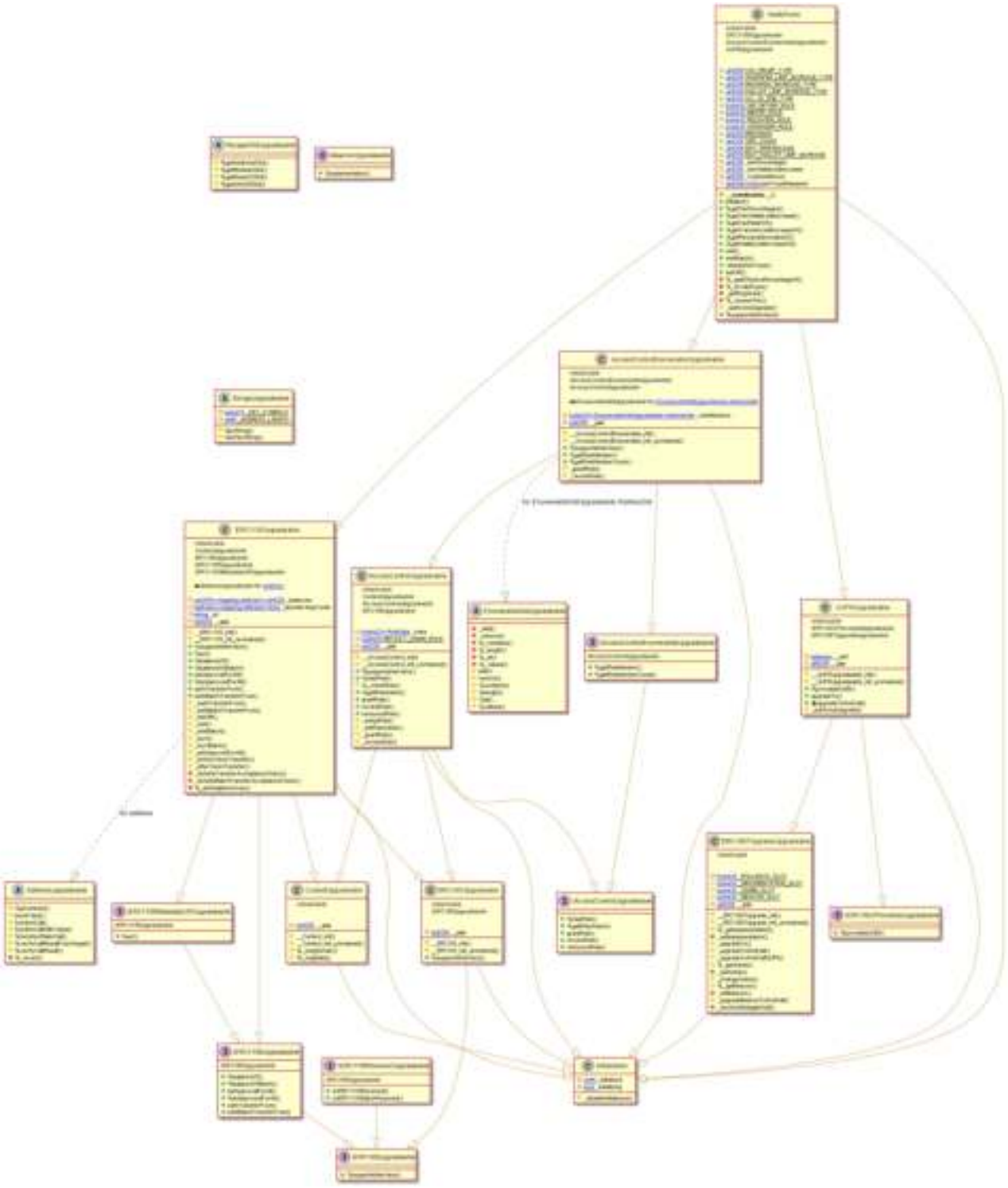
WaifuManager Diagram



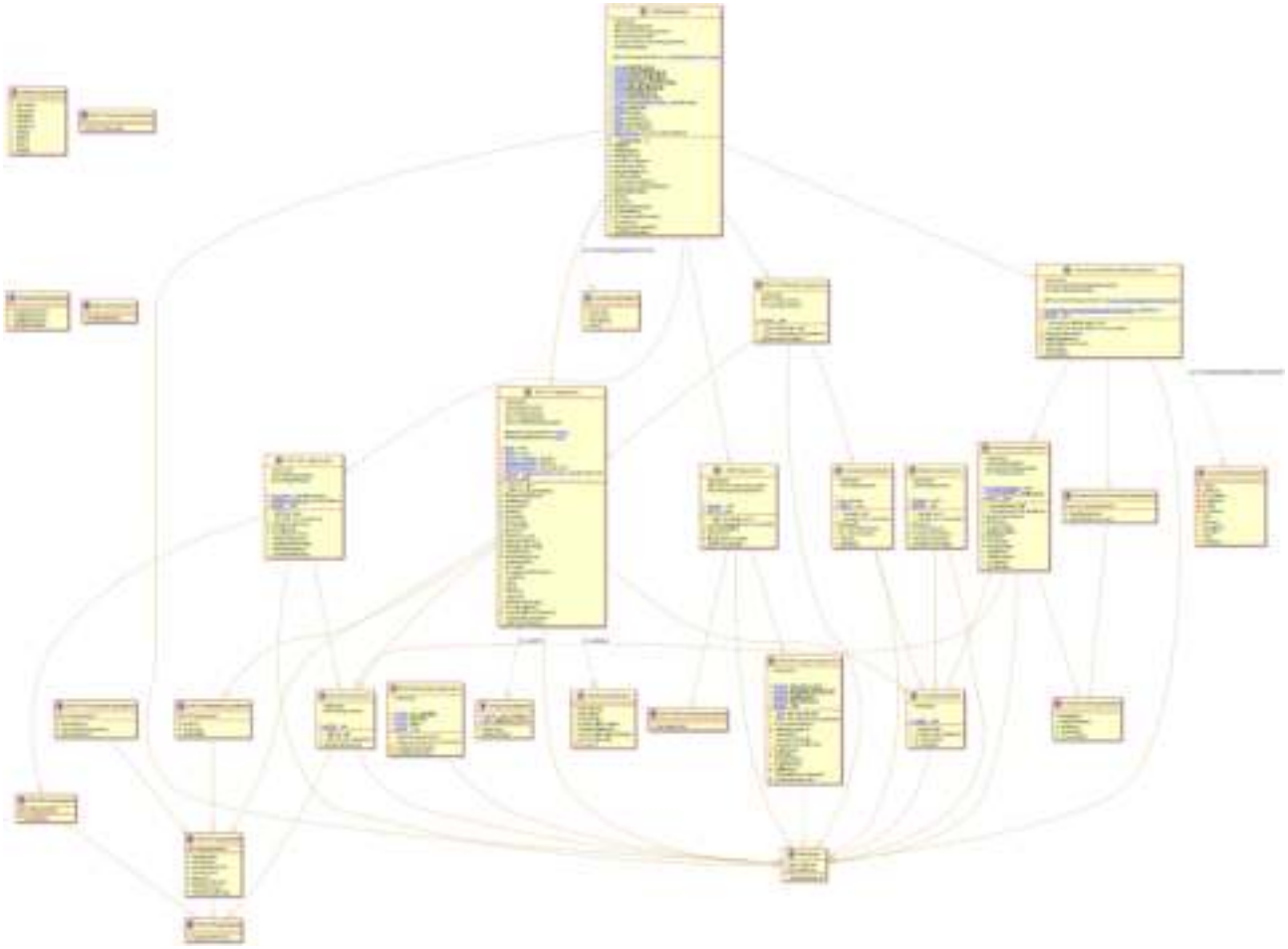
WaifuNodes Diagram



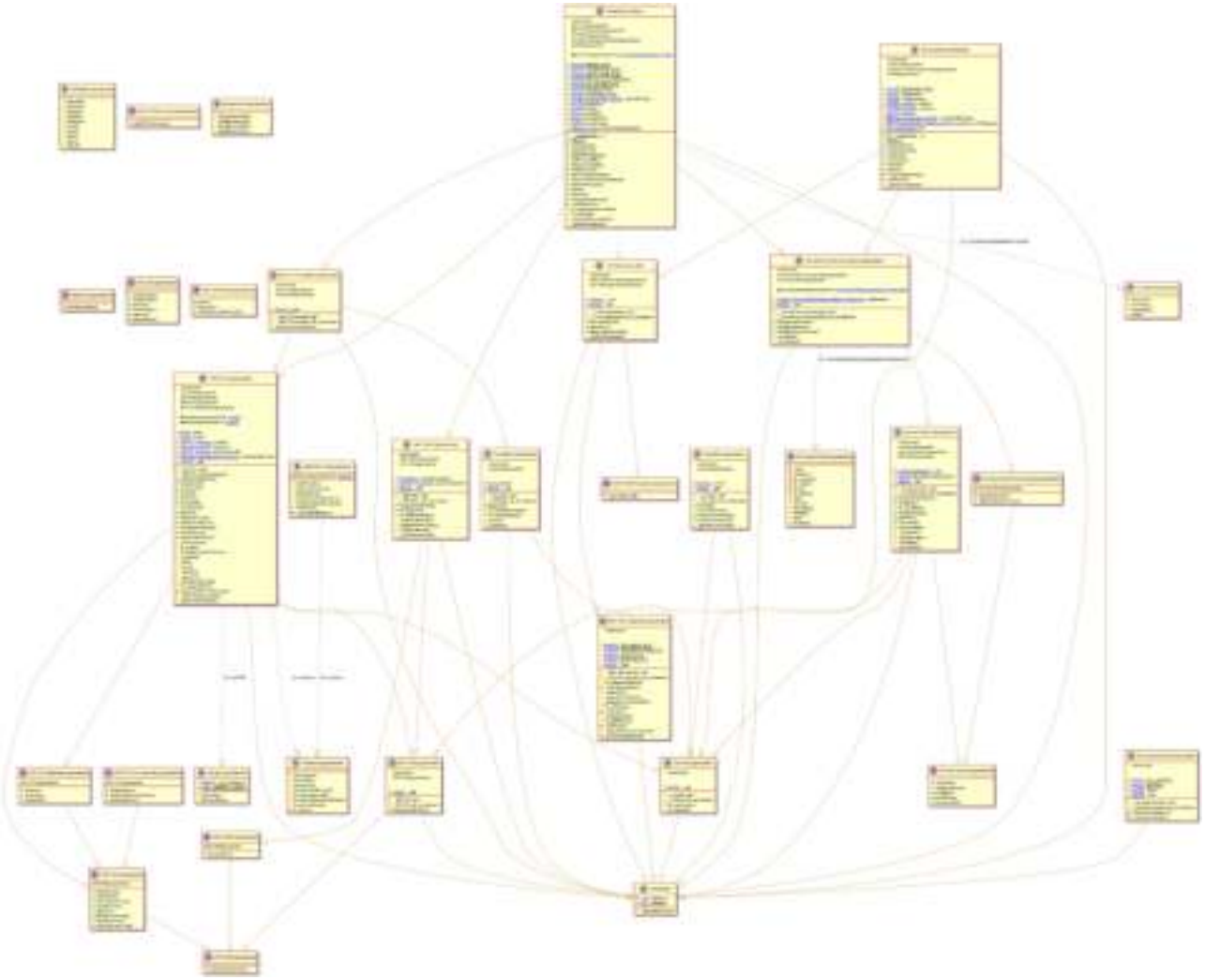
WaifuPerks Diagram



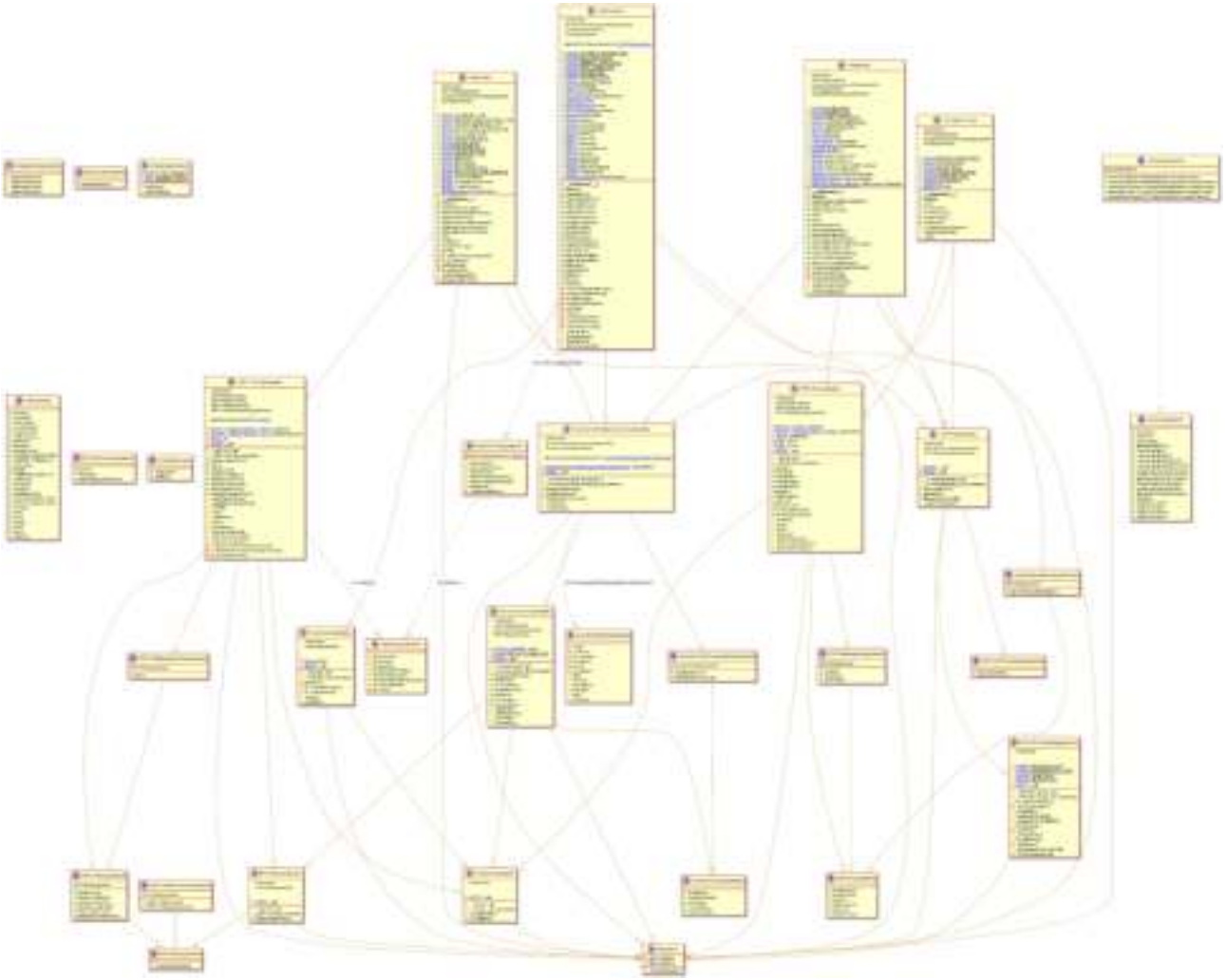
EarlyWaifuHolders Diagram



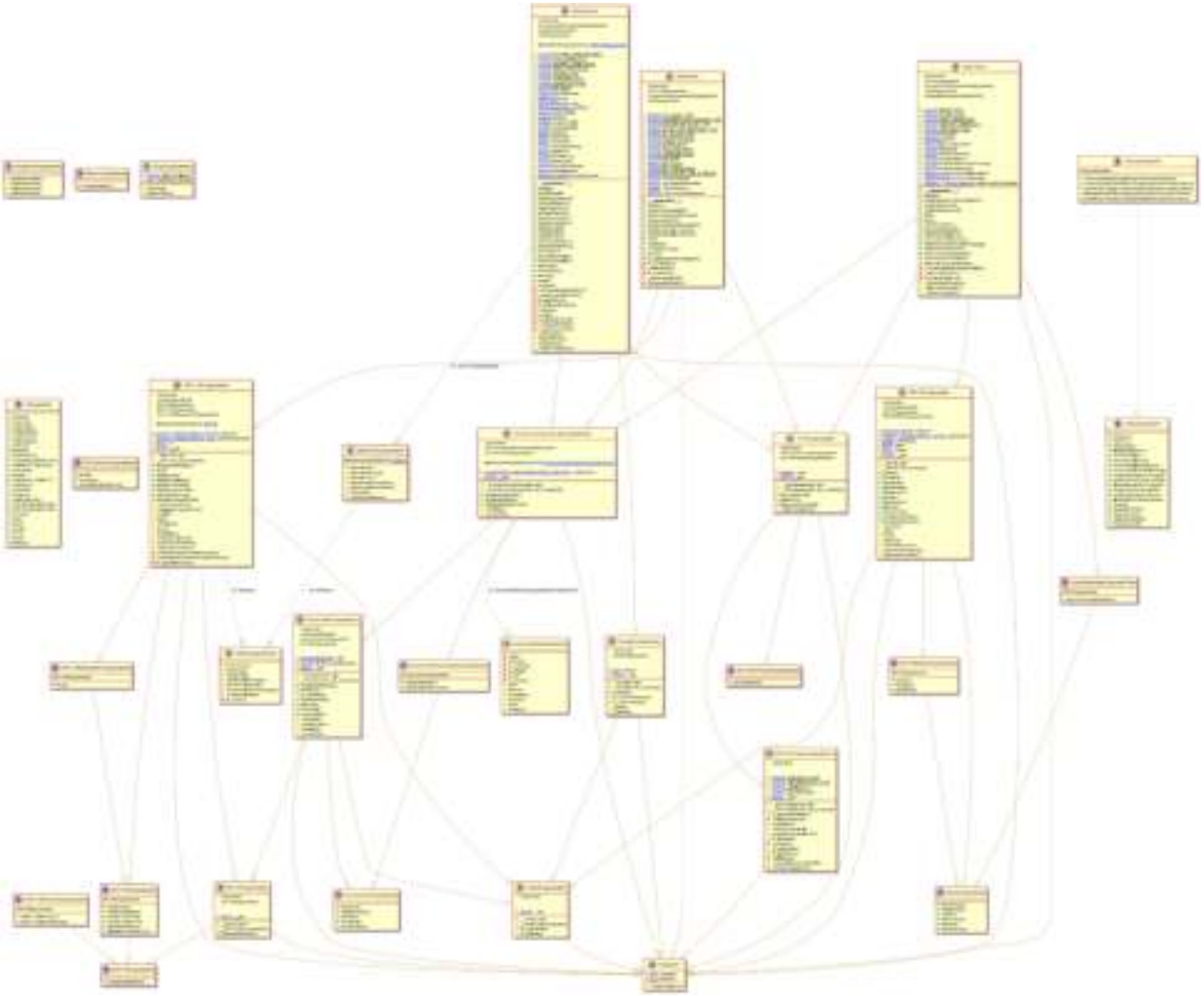
RevenuePaymentSplitter Diagram



PreLaunchToken Diagram

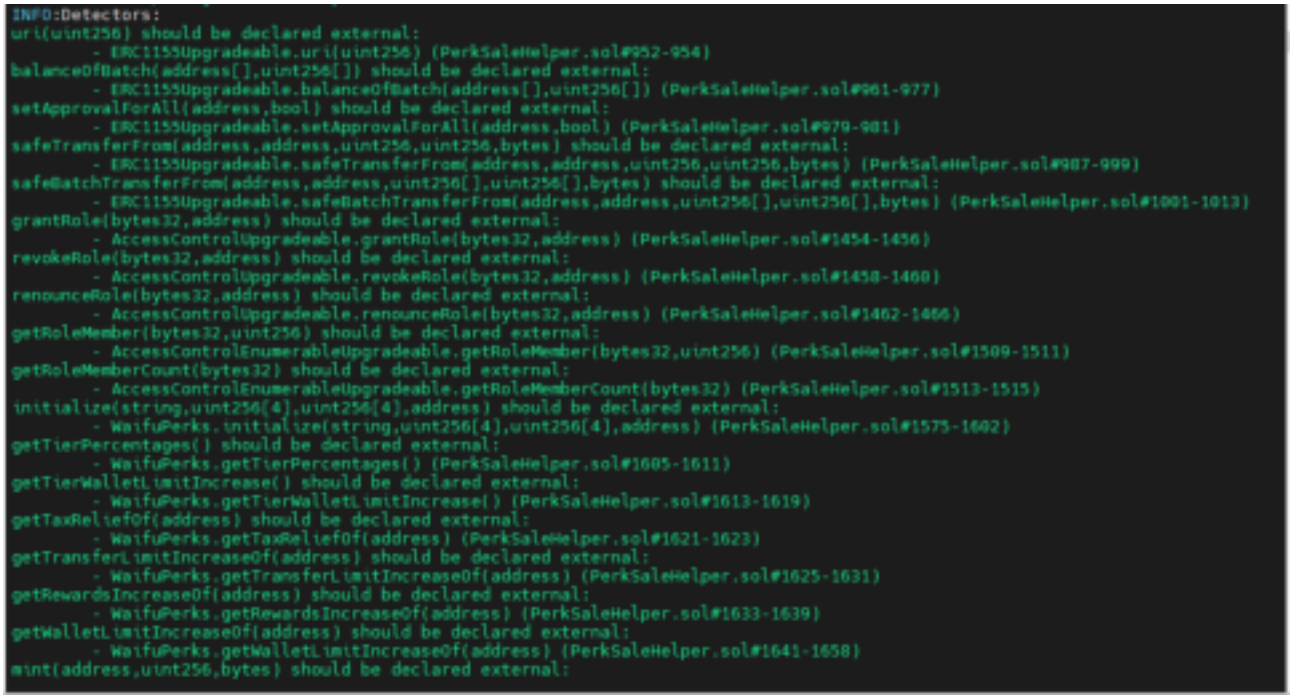


WaifuToken Diagram



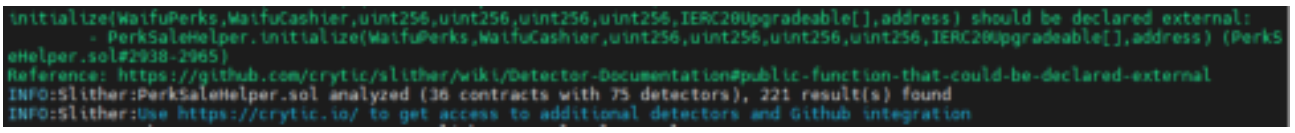
Slither Results Log

Slither log >> PerkSaleHelper.sol



```
INFO:Detectors:
uri(uint256) should be declared external:
        - ERC1155Upgradeable.uri(uint256) (PerkSaleHelper.sol#952-954)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155Upgradeable.balanceOfBatch(address[],uint256[]) (PerkSaleHelper.sol#961-977)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155Upgradeable.setApprovalForAll(address,bool) (PerkSaleHelper.sol#979-981)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155Upgradeable.safeTransferFrom(address,address,uint256,uint256,bytes) (PerkSaleHelper.sol#987-999)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155Upgradeable.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (PerkSaleHelper.sol#1001-1013)
grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (PerkSaleHelper.sol#1454-1456)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (PerkSaleHelper.sol#1458-1460)
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (PerkSaleHelper.sol#1462-1466)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerableUpgradeable.getRoleMember(bytes32,uint256) (PerkSaleHelper.sol#1509-1511)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlEnumerableUpgradeable.getRoleMemberCount(bytes32) (PerkSaleHelper.sol#1513-1515)
initialize(string,uint256[4],uint256[4],address) should be declared external:
        - WaifuPerks.initialize(string,uint256[4],uint256[4],address) (PerkSaleHelper.sol#1575-1602)
getTierPercentages() should be declared external:
        - WaifuPerks.getTierPercentages() (PerkSaleHelper.sol#1605-1611)
getTierWalletLimitIncrease() should be declared external:
        - WaifuPerks.getTierWalletLimitIncrease() (PerkSaleHelper.sol#1613-1619)
getTaxReliefOf(address) should be declared external:
        - WaifuPerks.getTaxReliefOf(address) (PerkSaleHelper.sol#1621-1623)
getTransferLimitIncreaseOf(address) should be declared external:
        - WaifuPerks.getTransferLimitIncreaseOf(address) (PerkSaleHelper.sol#1625-1631)
getRewardsIncreaseOf(address) should be declared external:
        - WaifuPerks.getRewardsIncreaseOf(address) (PerkSaleHelper.sol#1633-1639)
getWalletLimitIncreaseOf(address) should be declared external:
        - WaifuPerks.getWalletLimitIncreaseOf(address) (PerkSaleHelper.sol#1641-1658)
mint(address,uint256,bytes) should be declared external:
```

```
mint(address,uint256,bytes) should be declared external:
        - WaifuPerks.mint(address,uint256,bytes) (PerkSaleHelper.sol#1661-1672)
mintBatch(address,uint256,uint256,bytes) should be declared external:
        - WaifuPerks.mintBatch(address,uint256,uint256,bytes) (PerkSaleHelper.sol#1674-1695)
setURI(string) should be declared external:
        - WaifuPerks.setURI(string) (PerkSaleHelper.sol#1709-1711)
initialize(WaifuToken,WaifuPerks,IPancakeRouter01,IERC20Upgradeable,address,address,address,address,uint256,uint256,address) sh
ld be declared external:
        - WaifuCashier.initialize(WaifuToken,WaifuPerks,IPancakeRouter01,IERC20Upgradeable,address,address,address,address,uint
6,uint256,address) (PerkSaleHelper.sol#1913-1950)
pause() should be declared external:
        - WaifuCashier.pause() (PerkSaleHelper.sol#2167-2169)
unpause() should be declared external:
        - WaifuCashier.unpause() (PerkSaleHelper.sol#2171-2173)
name() should be declared external:
        - ERC20Upgradeable.name() (PerkSaleHelper.sol#2399-2401)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (PerkSaleHelper.sol#2403-2405)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (PerkSaleHelper.sol#2419-2423)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (PerkSaleHelper.sol#2429-2433)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (PerkSaleHelper.sol#2435-2444)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (PerkSaleHelper.sol#2446-2450)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (PerkSaleHelper.sol#2452-2461)
initialize(WaifuPerks,uint256,uint256,uint256,uint256,address) should be declared external:
        - WaifuToken.initialize(WaifuPerks,uint256,uint256,uint256,uint256,address) (PerkSaleHelper.sol#2611-2643)
mint(address,uint256) should be declared external:
        - WaifuToken.mint(address,uint256) (PerkSaleHelper.sol#2670-2672)
burn(uint256) should be declared external:
        - WaifuToken.burn(uint256) (PerkSaleHelper.sol#2674-2676)
initialize(WaifuPerks,WaifuCashier,uint256,uint256,uint256,uint256,IERC20Upgradeable[],address) should be declared external:
        - PerkSaleHelper.initialize(WaifuPerks,WaifuCashier,uint256,uint256,uint256,uint256,IERC20Upgradeable[],address) (PerkS
```



```
initialize(WaifuPerks,WaifuCashier,uint256,uint256,uint256,uint256,IERC20Upgradeable[],address) should be declared external:
        - PerkSaleHelper.initialize(WaifuPerks,WaifuCashier,uint256,uint256,uint256,uint256,IERC20Upgradeable[],address) (PerkS
eHelper.sol#2938-2965)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:PerkSaleHelper.sol analyzed (36 contracts with 75 detectors), 221 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> PresaleHelper.sol

```
INFO:Detectors:
uri(uint256) should be declared external:
        - ERC1155Upgradeable.uri(uint256) (PresaleHelper.sol#953-955)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155Upgradeable.balanceOfBatch(address[],uint256[]) (PresaleHelper.sol#962-978)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155Upgradeable.setApprovalForAll(address,bool) (PresaleHelper.sol#980-982)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155Upgradeable.safeTransferFrom(address,address,uint256,uint256,bytes) (PresaleHelper.sol#988-1000)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155Upgradeable.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (PresaleHelper.sol#1002-1014)
grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (PresaleHelper.sol#1455-1457)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (PresaleHelper.sol#1459-1461)
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (PresaleHelper.sol#1463-1467)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerableUpgradeable.getRoleMember(bytes32,uint256) (PresaleHelper.sol#1510-1512)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlEnumerableUpgradeable.getRoleMemberCount(bytes32) (PresaleHelper.sol#1514-1516)
initialize(string,uint256[4],uint256[4],address) should be declared external:
        - WaifuPerks.initialize(string,uint256[4],uint256[4],address) (PresaleHelper.sol#1575-1602)
getTierPercentages() should be declared external:
        - WaifuPerks.getTierPercentages() (PresaleHelper.sol#1605-1611)
getTierWalletLimitIncrease() should be declared external:
        - WaifuPerks.getTierWalletLimitIncrease() (PresaleHelper.sol#1613-1619)
getTaxReliefOf(address) should be declared external:
        - WaifuPerks.getTaxReliefOf(address) (PresaleHelper.sol#1621-1623)
getTransferLimitIncreaseOf(address) should be declared external:
        - WaifuPerks.getTransferLimitIncreaseOf(address) (PresaleHelper.sol#1625-1631)
getRewardsIncreaseOf(address) should be declared external:
```

```
burn(uint256) should be declared external:
        - WaifuToken.burn(uint256) (PresaleHelper.sol#2546-2548)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:PresaleHelper.sol analyzed (37 contracts with 75 detectors), 201 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> LiquidityManager.sol

```
INFO:Detectors:
uri(uint256) should be declared external:
        - ERC1155Upgradeable.uri(uint256) (LiquidityManager.sol#390-392)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155Upgradeable.balanceOfBatch(address[],uint256[]) (LiquidityManager.sol#399-415)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155Upgradeable.setApprovalForAll(address,bool) (LiquidityManager.sol#417-419)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155Upgradeable.safeTransferFrom(address,address,uint256,uint256,bytes) (LiquidityManager.sol#425-437)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155Upgradeable.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (LiquidityManager.sol#439-451)
grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (LiquidityManager.sol#1186-1188)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (LiquidityManager.sol#1190-1192)
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (LiquidityManager.sol#1194-1198)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerableUpgradeable.getRoleMember(bytes32,uint256) (LiquidityManager.sol#1241-1243)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlEnumerableUpgradeable.getRoleMemberCount(bytes32) (LiquidityManager.sol#1245-1247)
```

```
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (LiquidityManager.sol#2485-2489)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (LiquidityManager.sol#2495-2499)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (LiquidityManager.sol#2501-2510)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (LiquidityManager.sol#2512-2516)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (LiquidityManager.sol#2518-2527)
initialize(WaifuPerks,uint256,uint256,uint256,uint256,address) should be declared external:
        - WaifuToken.initialize(WaifuPerks,uint256,uint256,uint256,uint256,address) (LiquidityManager.sol#2676-2708)
mint(address,uint256) should be declared external:
        - WaifuToken.mint(address,uint256) (LiquidityManager.sol#2735-2737)
burn(uint256) should be declared external:
        - WaifuToken.burn(uint256) (LiquidityManager.sol#2739-2741)
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (LiquidityManager.sol#2966-2968)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (LiquidityManager.sol#2970-2973)
initialize(ILiquidityManagerSupportedToken,IERC20Upgradeable,IPancakePair) should be declared external:
        - LiquidityManager.initialize(ILiquidityManagerSupportedToken,IERC20Upgradeable,IPancakePair) (LiquidityManager.sol#3016-3040)
setPriceRange(uint256,uint256) should be declared external:
        - LiquidityManager.setPriceRange(uint256,uint256) (LiquidityManager.sol#3096-3105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LiquidityManager.sol analyzed (36 contracts with 75 detectors), 232 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> WaifuCashier.sol

```
INFO:Detectors:
uri(uint256) should be declared external:
        - ERC1155Upgradeable.uri(uint256) (WaifuCashier.sol#271-273)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155Upgradeable.balanceOfBatch(address[],uint256[]) (WaifuCashier.sol#288-296)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155Upgradeable.setApprovalForAll(address,bool) (WaifuCashier.sol#298-300)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155Upgradeable.safeTransferFrom(address,address,uint256,uint256,bytes) (WaifuCashier.sol#306-318)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155Upgradeable.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (WaifuCashier.sol#320-332)
grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (WaifuCashier.sol#1063-1065)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (WaifuCashier.sol#1067-1069)
```

```
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (WaifuCashier.sol#1950-1959)
initialize(WaifuPerks,uint256,uint256,uint256,uint256,address) should be declared external:
        - WaifuToken.initialize(WaifuPerks,uint256,uint256,uint256,uint256,address) (WaifuCashier.sol#2109-2141)
mint(address,uint256) should be declared external:
        - WaifuToken.mint(address,uint256) (WaifuCashier.sol#2168-2170)
burn(uint256) should be declared external:
        - WaifuToken.burn(uint256) (WaifuCashier.sol#2172-2174)
initialize(uint256,address) should be declared external:
        - PreLaunchToken.initialize(uint256,address) (WaifuCashier.sol#2382-2400)
mint(address,uint256) should be declared external:
        - PreLaunchToken.mint(address,uint256) (WaifuCashier.sol#2403-2417)
withdrawTo(address,uint256) should be declared external:
        - PreLaunchToken.withdrawTo(address,uint256) (WaifuCashier.sol#2419-2421)
withdrawFrom(address,uint256) should be declared external:
        - PreLaunchToken.withdrawFrom(address,uint256) (WaifuCashier.sol#2423-2428)
initialize(WaifuToken,WaifuPerks,IPancakeRouter01,IERC20Upgradeable,address,address,address,address,uint256,uint256,address) sh
ould be declared external:
        - WaifuCashier.initialize(WaifuToken,WaifuPerks,IPancakeRouter01,IERC20Upgradeable,address,address,address,address,uint
256,uint256,address) (WaifuCashier.sol#2560-2603)
pause() should be declared external:
        - WaifuCashier.pause() (WaifuCashier.sol#2810-2812)
unpause() should be declared external:
        - WaifuCashier.unpause() (WaifuCashier.sol#2814-2816)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:WaifuCashier.sol analyzed (34 contracts with 75 detectors), 220 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> WaifuManager.sol

```
INFO:Detectors:
uri(uint256) should be declared external:
        - ERC1155Upgradeable.uri(uint256) (WaifuManager.sol#270-272)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155Upgradeable.balanceOfBatch(address[],uint256[]) (WaifuManager.sol#279-295)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155Upgradeable.setApprovalForAll(address,bool) (WaifuManager.sol#297-299)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155Upgradeable.safeTransferFrom(address,address,uint256,uint256,bytes) (WaifuManager.sol#305-317)
```

```
name() should be declared external:
        - ERC20Upgradeable.name() (WaifuManager.sol#2875-2877)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (WaifuManager.sol#2879-2881)
decimals() should be declared external:
        - ERC20Upgradeable.decimals() (WaifuManager.sol#2883-2885)
totalSupply() should be declared external:
        - ERC20Upgradeable.totalSupply() (WaifuManager.sol#2887-2889)
balanceOf(address) should be declared external:
        - ERC20Upgradeable.balanceOf(address) (WaifuManager.sol#2891-2893)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (WaifuManager.sol#2895-2899)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (WaifuManager.sol#2905-2909)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (WaifuManager.sol#2911-2920)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (WaifuManager.sol#2922-2926)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (WaifuManager.sol#2928-2937)
initialize(WaifuNodes,WaifuCashier,WaifuPerks,uint256[],uint256[],address) should be declared external:
        - WaifuManager.initialize(WaifuNodes,WaifuCashier,WaifuPerks,uint256[],uint256[],address) (WaifuManager.sol#3109-3140)
getEpochStartSnapshots() should be declared external:
        - WaifuManager.getEpochStartSnapshots() (WaifuManager.sol#3147-3153)
getRewardsIncreaseOf(address) should be declared external:
        - WaifuManager.getRewardsIncreaseOf(address) (WaifuManager.sol#3205-3211)
calculateUnclaimedRewardsFor(address) should be declared external:
        - WaifuManager.calculateUnclaimedRewardsFor(address) (WaifuManager.sol#3213-3219)
pause() should be declared external:
        - WaifuManager.pause() (WaifuManager.sol#3377-3379)
unpause() should be declared external:
        - WaifuManager.unpause() (WaifuManager.sol#3381-3383)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:WaifuManager.sol analyzed (37 contracts with 75 detectors), 240 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> WaifuPerks.sol

```
INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (WaifuPerks.sol#157-166) uses assembly
        - INLINE ASM (WaifuPerks.sol#159-162)
StorageSlotUpgradeable.getAddressSlot(bytes32) (WaifuPerks.sol#601-605) uses assembly
        - INLINE ASM (WaifuPerks.sol#602-604)
StorageSlotUpgradeable.getBooleanSlot(bytes32) (WaifuPerks.sol#607-611) uses assembly
        - INLINE ASM (WaifuPerks.sol#608-610)
StorageSlotUpgradeable.getBytes32Slot(bytes32) (WaifuPerks.sol#613-617) uses assembly
        - INLINE ASM (WaifuPerks.sol#614-616)
StorageSlotUpgradeable.getUint256Slot(bytes32) (WaifuPerks.sol#619-623) uses assembly
        - INLINE ASM (WaifuPerks.sol#620-622)
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.AddressSet) (WaifuPerks.sol#998-1007) uses assembly
        - INLINE ASM (WaifuPerks.sol#1002-1004)
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.UintSet) (WaifuPerks.sol#1070-1079) uses assembly
        - INLINE ASM (WaifuPerks.sol#1074-1076)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
```

```
INFO:Detectors:
WaifuPerks (WaifuPerks.sol#1527-1780) does not implement functions:
        - UUPSUpgradeable._authorizeUpgrade(address) (WaifuPerks.sol#772)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
UUPSUpgradeable.__gap (WaifuPerks.sol#774) is never used in WaifuPerks (WaifuPerks.sol#1527-1780)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
```

Slither log >> EarlyWaifuHolders.sol

```
INFO:Detectors:
royaltyInfo(uint256,uint256) should be declared external:
        - ERC2981Upgradeable.royaltyInfo(uint256,uint256) (EarlyWaifuHolders.sol#135-145)
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (EarlyWaifuHolders.sol#845-847)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (EarlyWaifuHolders.sol#849-852)
balanceOf(address) should be declared external:
        - ERC721Upgradeable.balanceOf(address) (EarlyWaifuHolders.sol#906-909)
name() should be declared external:
        - ERC721Upgradeable.name() (EarlyWaifuHolders.sol#923-925)
symbol() should be declared external:
        - ERC721Upgradeable.symbol() (EarlyWaifuHolders.sol#930-932)
tokenURI(uint256) should be declared external:
        - ERC721Upgradeable.tokenURI(uint256) (EarlyWaifuHolders.sol#937-942)
```

```
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (EarlyWaifuHolders.sol#2046-2050)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerableUpgradeable.getRoleMember(bytes32,uint256) (EarlyWaifuHolders.sol#2183-2185)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlEnumerableUpgradeable.getRoleMemberCount(bytes32) (EarlyWaifuHolders.sol#2191-2193)
initialize(uint256,address) should be declared external:
        - EarlyWaifuHolders.initialize(uint256,address) (EarlyWaifuHolders.sol#2264-2287)
safeMintNext(address) should be declared external:
        - EarlyWaifuHolders.safeMintNext(address) (EarlyWaifuHolders.sol#2297-2299)
safeMintNextBatch(address,uint256) should be declared external:
        - EarlyWaifuHolders.safeMintNextBatch(address,uint256) (EarlyWaifuHolders.sol#2301-2308)
pause() should be declared external:
        - EarlyWaifuHolders.pause() (EarlyWaifuHolders.sol#2360-2362)
unpause() should be declared external:
        - EarlyWaifuHolders.unpause() (EarlyWaifuHolders.sol#2364-2366)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:EarlyWaifuHolders.sol analyzed (30 contracts with 75 detectors), 182 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> RevenuePaymentSplitter.sol

```
initialize(EarlyWaifuHolders,address) should be declared external:
        - RevenuePaymentSplitter.initialize(EarlyWaifuHolders,address) (RevenuePaymentSplitter.sol#2586-2596)
totalShares() should be declared external:
        - RevenuePaymentSplitter.totalShares() (RevenuePaymentSplitter.sol#2614-2616)
shares(uint256) should be declared external:
        - RevenuePaymentSplitter.shares(uint256) (RevenuePaymentSplitter.sol#2640-2642)
payee(uint256) should be declared external:
        - RevenuePaymentSplitter.payee(uint256) (RevenuePaymentSplitter.sol#2666-2668)
release(uint256) should be declared external:
        - RevenuePaymentSplitter.release(uint256) (RevenuePaymentSplitter.sol#2675-2702)
release(IERC20Upgradeable,uint256) should be declared external:
        - RevenuePaymentSplitter.release(IERC20Upgradeable,uint256) (RevenuePaymentSplitter.sol#2709-2737)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:RevenuePaymentSplitter.sol analyzed (34 contracts with 75 detectors), 198 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> PreLaunchToken.sol

```
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (PreLaunchToken.sol#2932-2936)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (PreLaunchToken.sol#2938-2947)
getWalletLimitOf(address) should be declared external:
        - WaifuToken.getWalletLimitOf(address) (PreLaunchToken.sol#3125-3127)
getTransferTaxOf(address) should be declared external:
        - WaifuToken.getTransferTaxOf(address) (PreLaunchToken.sol#3129-3133)
mint(address,uint256) should be declared external:
        - PreLaunchToken.mint(address,uint256) (PreLaunchToken.sol#3205-3209)
withdrawTo(address,uint256) should be declared external:
        - PreLaunchToken.withdrawTo(address,uint256) (PreLaunchToken.sol#3211-3213)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:PreLaunchToken.sol analyzed (35 contracts with 75 detectors), 205 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> WaifuToken.sol

```
name() should be declared external:
        - ERC20Upgradeable.name() (WaifuToken.sol#2934-2936)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (WaifuToken.sol#2938-2940)
totalSupply() should be declared external:
        - ERC20Upgradeable.totalSupply() (WaifuToken.sol#2946-2948)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (WaifuToken.sol#2954-2958)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (WaifuToken.sol#2964-2968)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (WaifuToken.sol#2970-2979)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (WaifuToken.sol#2981-2985)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (WaifuToken.sol#2987-2996)
initialize(WaifuPerks,uint256,uint256,uint256,uint256,address) should be declared external:
        - WaifuToken.initialize(WaifuPerks,uint256,uint256,uint256,uint256,address) (WaifuToken.sol#3157-3189)
mint(address,uint256) should be declared external:
        - WaifuToken.mint(address,uint256) (WaifuToken.sol#3218-3220)
burn(uint256) should be declared external:
        - WaifuToken.burn(uint256) (WaifuToken.sol#3222-3224)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:WaifuToken.sol analyzed (33 contracts with 75 detectors), 212 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Email: audit@EtherAuthority.io