Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: PlanetMoon
Website: https://www.planetmoon.io
Platform: Binance Smart Chain
Language: Solidity
Date: June 9th, 2023

Table of contents

Email: audit@EtherAuthority.io

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

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contacted by PlanetMoon to perform the Security audit of the PlanetMoon token 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 9th, 2023.

The purpose of this audit was to address the following:

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

Project Background

- PlanetMoon is a staking contract on the Binance Smart Chain blockchain.
- The PlanetMoon Contracts handle multiple contracts, and all contracts have different functions.
    - StakingPoolFactory: This contract is used for creating a staking pool.
    - StakingPool: This contract has functions like: stake tokens, unstake tokens.
    - PMRewardDistributor: This contract is used to distribute rewards and emergency withdrawal tokens.
    - MembershipFeeManager: This contract manages membership fees and sets distribution schemes.
    - CampaignFeeManager: This contract manages campaign fees.
    - PMTeamManager: This contract is used to create a team and add team members to a team.
    - PMMembershipManager: This contract is used to manage membership management.
- There are 12 smart contracts and 10 Interfaces, which were included in the audit scope. And there were some standard library codes, such as OpenZepelin, that were excluded. Because those standard library code is considered as time tested and community audited, so we can safely ignore them.

Audit scope

Name	Code Review and Security Analysis Report for PlanetMoon Smart Contracts
Platform	**BSC / Solidity**
File 1	CreatorContract.sol
File 1 MD5 Hash	C4EEA3C6734402C92378F1D64795C10A
Updated File 1 MD5 Hash	38188592B7CE7AEF0BF9C7A8F27481F1
File 2	CreatorManager.sol
File 2 MD5 Hash	734D984618CC705D82FEB698CB09C651
Updated File 2 MD5 Hash	01FF54E0D4324547E737CAFF77E59645
File 3	PMMembershipManager.sol
File 3 MD5 Hash	2292C9FFEA2B7C39DFC6323A6A43F88D
Updated File 3 MD5 Hash	ACF59DB9925B330CC28BD9AA998248EB
File 4	PMTeamManager.sol
File 4 MD5 Hash	32C8F6A546CE93BF3F736320594DB913
Updated File 4 MD5 Hash	83A59CAB9A2761A643D6B033A7368978
File 5	CampaignFeeManager.sol
File 5 MD5 Hash	39B80FDF00D4BED3BE035588D6A35E53
Updated File 5 MD5 Hash	594A482F2CCA0C0AF1614FFD977B4F8E
File 6	MembershipFeeManager.sol
File 6 MD5 Hash	5A0D763A3F86113A686F9E3FD637FD43
Updated File 6 MD5 Hash	65C512973A0C3C02D1841BD8BF9AD689
File 7	PMRewardDistributor.sol
File 7 MD5 Hash	B49079168252FC6905CBF9768A98F046
Updated File 7 MD5 Hash	43BF55FB736B61B5E73A5F8EC279DCB6
File 8	StakingPool.sol
File 8 MD5 Hash	9673A432913EDA0D403CD646C1F90C34

Updated File 8 MD5 Hash	6FF229F3DE8B978750DB6DC4803C2D0B
File 9	StakingPoolFactory.sol
File 9 MD5 Hash	F51E1D5CE85D5F83E9B6DC32B99987EF
Updated File 9 MD5 Hash	EAE1130FA355323409A419C441266E15
File 10	SwapETHForTokens.sol
File 10 MD5 Hash	66081D74E1C2179D40E7A10CA474A465
File 11	PriceFeed.sol
File 11 MD5 Hash	2D9F8A045E7792533EA18B4A5BCC3329
Audit Date	June 9th, 2023
Revised Date	June 13th, 2023

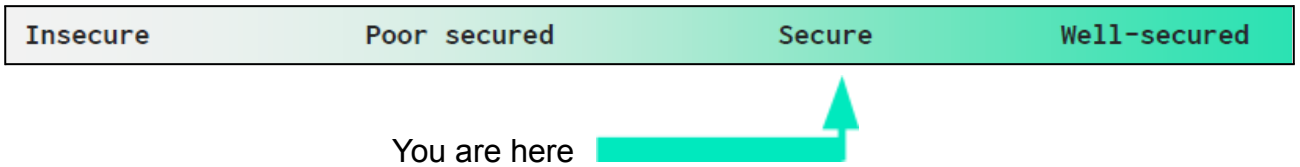
Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 CreatorManager.sol • This contract is access to CreatorContract.	**YES, This is valid.**
File 2 PMMembershipManager.sol • Name: PlanetMoon Membership Manager • Symbol: PMM **Owner has control over following functions:** • Set the membership fee manager addresses. • Subscribe as a member. • Set the give away membership. • Update pause status.	**YES, This is valid.**
File 3 PMTeamManager.sol • Name: PlanetMoon Team Manager • Symbol: PTM **Owner has control over following functions:** • Set the membership fee manager address. • Add/update team members. • Update pause status.	**YES, This is valid.**
File 4 CampaignFeeManager.sol **Owner has control over following functions:** • Set the campaign fees. • Set the fee distribution wallet addresses. • Emergency withdrawal token. • Split funds.	**YES, This is valid.**
File 5 MembershipFeeManager.sol **Owner has control over following functions:** • Set the membership fee values.	**YES, This is valid.**

• Set the distribution scheme. • Set the fee distribution wallets. • Split funds. • Emergency withdrawal token.	
File 6 PMRewardDistributor.sol **<u>Owner has control over following functions:</u>** • Set the distribute reward address. • Reward to campaign address can be applied. • Emergency withdrawal token. • Update pause status. • Set the giveaway manager address.	**YES, This is valid.**
File 7 StakingPool.sol • UnstakeTokens can only be called by creator contract of the token holder.	**YES, This is valid.**
File 8 StakingPoolFactory.sol **<u>Owner has control over following functions:</u>** • Set the pause status. • Set the campaign fee manager address.	**YES, This is valid.**
File 9 PriceFeed.sol • PriceFeed contract is used to get the latest price of one USD.	**YES, This is valid.**
File 10 SwapETHForTokens.sol **<u>Owner has control over following functions:</u>** • Set the router addresses. • Current owner can transfer ownership of the contract to a new account. • Deleting ownership will leave the contract without an owner, removing any owner-only functionality.	**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 0 low and 8 very low level issues.
We confirm that 8 very low severity issues are fixed in the revised smart contract code.

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

Technical Quick Stats

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

Overall Audit Result: PASSED

Code Quality

This audit scope has 12 smart contracts and 10 interfaces files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the PlanetMoon 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 PlanetMoon Protocol.

The PlanetMoon team has provided unit test scripts, which helped to determine the integrity of the code in an automated way.

Code parts are not well commented on smart contracts.

Documentation

We were given a PlanetMoon 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.planetmoon.io which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

CreatorContract.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onERC721Received	write	Passed	No Issue
3	sendTokensBackToOwner	write	Passed	No Issue
4	getPoolAddresses	read	Passed	No Issue
5	onlyOwner	modifier	Passed	No Issue
6	owner	read	Passed	No Issue
7	_checkOwner	internal	Passed	No Issue
8	renounceOwnership	write	access only Owner	No Issue
9	transferOwnership	write	access only Owner	No Issue
10	_transferOwnership	internal	Passed	No Issue
11	onERC721Received	write	Passed	No Issue
12	removePoolAddress	write	Passed	Fixed

CreatorManager.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	createACreator	write	Passed	No Issue
3	getCreatorAddress	read	Passed	No Issue
4	getPoolAddressesOfCreator	read	Passed	No Issue

PMMembershipManager.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getUserTokenData	read	Passed	No Issue
3	becomeMember	write	Passed	No Issue
4	upgradeToPremium	write	Passed	No Issue
5	becomePremiumMember	write	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	_beforeTokenTransfer	internal	Passed	No Issue
8	tokenURI	read	Passed	No Issue

9	updateMembershipFeeManager	write	access only Owner	No Issue
10	giveAwayMembership	write	access only Owner	No Issue
11	changePauseStatus	write	access only Owner	No Issue
12	supportsInterface	read	Passed	No Issue
13	balanceOf	read	Passed	No Issue
14	ownerOf	read	Passed	No Issue
15	name	read	Passed	No Issue
16	symbol	read	Passed	No Issue
17	tokenURI	read	Passed	No Issue
18	_baseURI	internal	Passed	No Issue
19	approve	write	Passed	No Issue
20	getApproved	read	Passed	No Issue
21	setApprovalForAll	write	Passed	No Issue
21	isApprovedForAll	read	Passed	No Issue
22	transferFrom	write	Passed	No Issue
23	safeTransferFrom	write	Passed	No Issue
24	safeTransferFrom	write	Passed	No Issue
25	_safeTransfer	internal	Passed	No Issue
26	_ownerOf	internal	Passed	No Issue
27	_exists	internal	Passed	No Issue
28	_isApprovedOrOwner	internal	Passed	No Issue
29	_safeMint	internal	Passed	No Issue
30	_safeMint	write	Passed	No Issue
31	_mint	internal	Passed	No Issue
32	_burn	internal	Passed	No Issue
33	_transfer	internal	Passed	No Issue
34	_approve	internal	Passed	No Issue
35	_setApprovalForAll	internal	Passed	No Issue
36	_requireMinted	internal	Passed	No Issue
37	_checkOnERC721Received	internal	Passed	No Issue
38	_beforeTokenTransfer	internal	Passed	No Issue
39	_afterTokenTransfer	internal	Passed	No Issue
40	_beforeConsecutiveTokenTransfer	internal	Passed	No Issue
41	_afterConsecutiveTokenTransfer	internal	Passed	No Issue
42	onlyOwner	modifier	Passed	No Issue
43	owner	read	Passed	No Issue
44	_checkOwner	internal	Passed	No Issue
45	renounceOwnership	write	access only Owner	No Issue
46	transferOwnership	write	access only Owner	No Issue
47	_transferOwnership	internal	Passed	No Issue

PMTeamManager.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	createATeam	write	Passed	No Issue
9	updateTeamMembers	write	Passed	No Issue
10	getTeamData	read	Passed	No Issue
11	getTeamsDataByRange	read	Passed	No Issue
12	tokenURI	read	Passed	No Issue
13	updateMembershipFeeManager	write	access only Owner	No Issue
14	changePauseStatus	write	access only Owner	No Issue
15	supportsInterface	read	Passed	No Issue
16	tokenOfOwnerByIndex	read	Passed	No Issue
17	totalSupply	read	Passed	No Issue
18	tokenByIndex	read	Passed	No Issue
19	_beforeTokenTransfer	internal	Passed	No Issue
20	_beforeConsecutiveTokenTransfer	internal	Passed	No Issue
21	_addTokenToOwnerEnumeration	write	Passed	No Issue
21	_addTokenToAllTokensEnumeration	write	Passed	No Issue
22	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
23	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue

CampaignFeeManager.sol

Functions

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

8	getCampaignFee	read	Passed	No Issue
9	getAllCampaignFees	read	Passed	No Issue
10	setCampaignFees	write	access only Owner	No Issue
11	getUnstakingFee	read	Passed	No Issue
12	getAllUnstakingFees	read	Passed	No Issue
13	setUnstakingFees	write	access only Owner	No Issue
14	setDistributionScheme	write	access only Owner	No Issue
15	setFeeDistributionWallets	write	access only Owner	No Issue
16	SplitFunds	write	Passed	Fixed
17	emergencyWithdraw	write	access only Owner	No Issue
18	receive	external	Passed	No Issue
19	setFeeDistributionShares	write	access only Owner	No Issue

MembershipFeeManager.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	getMembershipFee	read	Passed	No Issue
9	getAllFees	read	Passed	No Issue
10	setMembershipFee	write	access only Owner	No Issue
11	setFeeDistributionShares	write	access only Owner	No Issue
12	setFeeDistributionWallets	write	access only Owner	No Issue
13	SplitFunds	write	Passed	Fixed
14	emergencyWithdraw	write	access only Owner	No Issue
15	receive	external	Passed	No Issue

PMRewardDistributor.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue

4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	onlyGiveAwayManager	modifier	Passed	No Issue
9	distributeReward	write	access only Give Away Manager	No Issue
10	applyRewardToACampaing	write	access only Give Away Manager	No Issue
11	emergencyWithdraw	write	access only Owner	No Issue
12	changePauseStatus	write	access only Owner	No Issue
13	updateGiveAwayManager	write	access only Owner	No Issue
14	receive	external	Passed	No Issue

StakingPool.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	supportsInterface	read	Passed	No Issue
3	tokenOfOwnerByIndex	read	Passed	No Issue
4	totalSupply	read	Passed	No Issue
5	tokenByIndex	read	Passed	No Issue
6	_beforeTokenTransfer	internal	Passed	No Issue
7	_beforeConsecutiveTokenTransfer	internal	Passed	No Issue
8	_addTokenToOwnerEnumeration	write	Passed	No Issue
9	_addTokenToAllTokensEnumeration	write	Passed	No Issue
10	_removeTokenFromOwnerEnumeration	write	Passed	No Issue
11	_removeTokenFromAllTokensEnumeration	write	Passed	No Issue
12	stakeTokens	write	Passed	No Issue
13	unstakeTokens	write	Passed	No Issue
14	findRedeemableReward	read	Passed	No Issue
15	checkTokenReward	read	Passed	No Issue
16	getProjectInfo	read	Passed	No Issue
17	getTokenData	read	Passed	No Issue
18	getUserTokens	read	Passed	No Issue

Email: audit@EtherAuthority.io

StakingPoolFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	owner	read	Passed	No Issue
4	_checkOwner	internal	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	_transferOwnership	internal	Passed	No Issue
8	createAStakingPool	write	Passed	No Issue
9	getPoolsByToken	read	Passed	No Issue
10	getPoolByID	read	Passed	No Issue
11	getPoolIdsOfAUser	read	Passed	No Issue
12	getPoolIdsOfATeam	read	Passed	No Issue
13	changePauseStatus	write	access only Owner	No Issue
14	updateCampaignFeeManager	write	access only Owner	No Issue

SwapETHForTokens.sol

Functions

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

PriceFeed.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getLatestPriceOfOneUSD	read	Passed	No Issue

Severity Definitions

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

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No critical severity vulnerabilities were found in the contract code.

High Severity

No high severity vulnerabilities were found in the contract code.

Medium

No medium severity vulnerabilities were found in the contract code.

Low

No low severity vulnerabilities were found in the contract code.

Very Low / Informational / Best practices:

8 very low severity issues were found which are fixed in the revised smart contract code.

Email: audit@EtherAuthority.io

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:

PMMembershipManager.sol

- updateMembershipFeeManager: Membership fee manager address can be set by the owner.
- giveAwayMembership: Give away membership can be set by the owner.
- changePauseStatus: Pause status can be set by the owner.

PMTeamManager.sol

- updateMembershipFeeManager: Membership fee manager address can be set by the owner.
- changePauseStatus: Pause status can be set by the owner.

CampaignFeeManager.sol

- setCampaignFees: CampaignFees can be set by the owner.
- setUnstakingFees: UnstakingFees can be set by the owner.
- setDistributionShares: DistributionScheme can be set by the owner.
- setFeeDistributionWallets: Fee distribution wallet addresses can be set by the owner.
- SplitFunds: Split funds can be set by the owner.
- emergencyWithdraw: Emergency withdrawal token by the owner.

MembershipFeeManager.sol

- setMembershipFee: Membership Fee values can be set by the owner.
- setDistributionShares: Distribution scheme can be set by the owner.
- setFeeDistributionWallets: Fee Distribution Wallets can be set by the owner.
- SplitFunds: Split funds can be set by the owner.
- emergencyWithdraw: Emergency withdrawal token by the owner.

PMRewardDistributor.sol

- distributeReward: Distribute reward address can be set by the owner.
- applyRewardToACampaing: Reward to campaign address can be applied by the owner.
- emergencyWithdraw: Emergency withdrawal token by the owner.
- changePauseStatus: Pause status can be set by the owner.
- updateGiveAwayManager: Give away manager address can be set by the owner.

StakingPoolFactory.sol

- changePauseStatus: Pause status can be set by the owner.
- updateCampaignFeeManager: Campaign fee manager address can be set by the owner.

SwapETHForTokens.sol

- setRouter: Router address can be set by the owner.

Ownable.sol

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

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

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We had observed 8 very low severity issues in the smart contracts. We confirm that 8 very low severity issues are fixed in the revised smart contract code. **So, the smart contracts are ready for the mainnet deployment.**

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

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

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

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

Email: audit@EtherAuthority.io

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

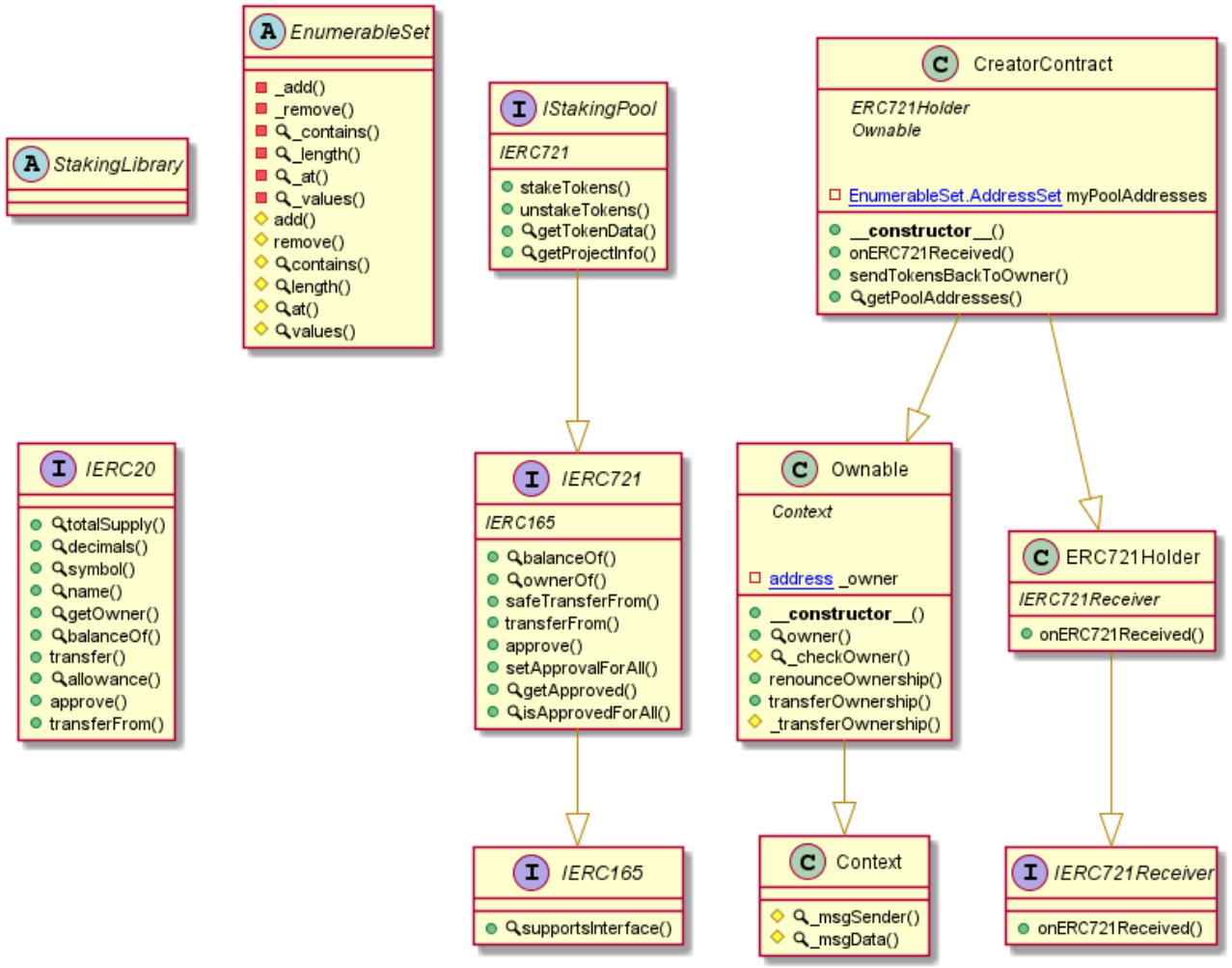
Technical Disclaimer

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

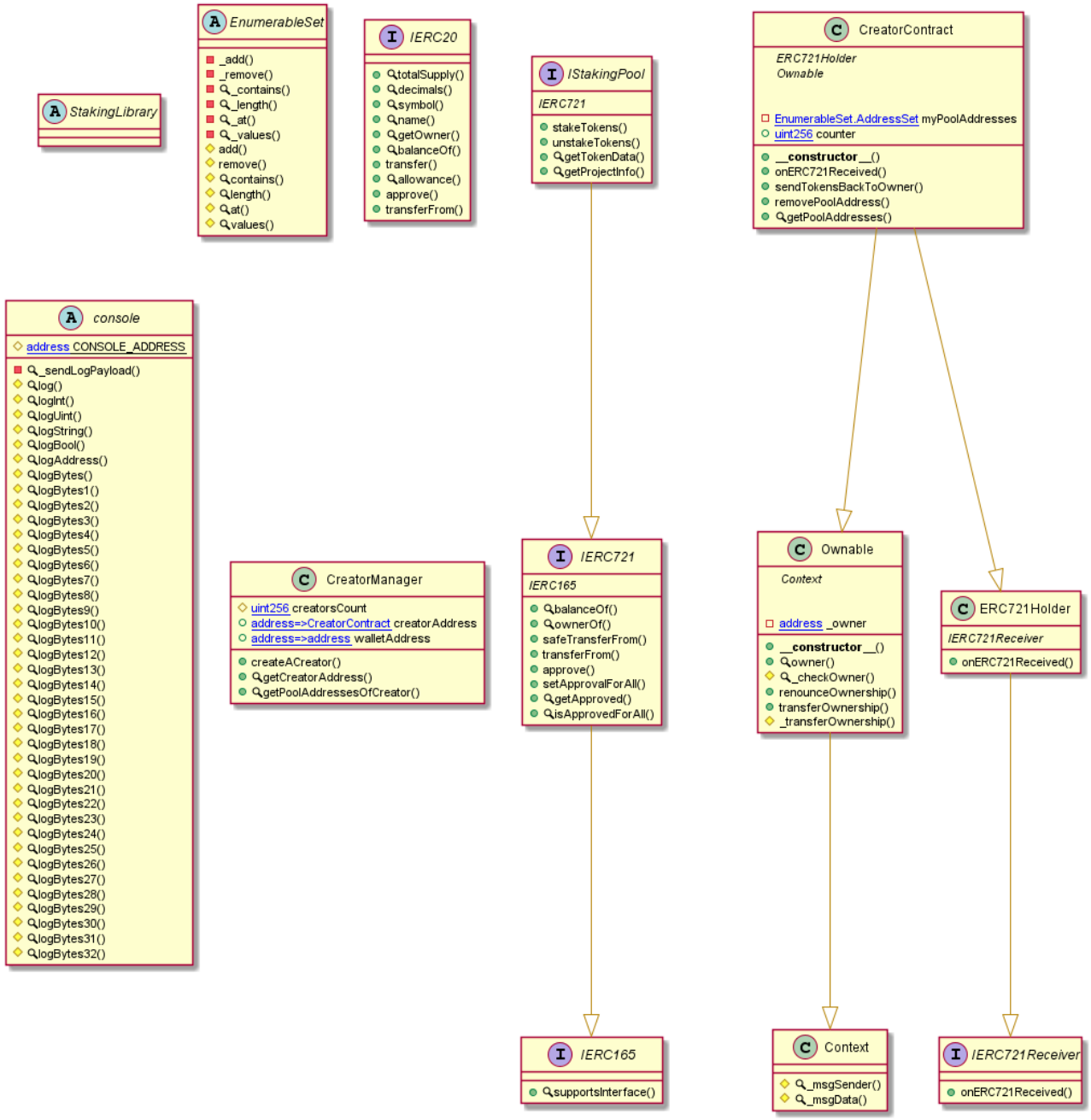
Appendix

Code Flow Diagram - PlanetMoon

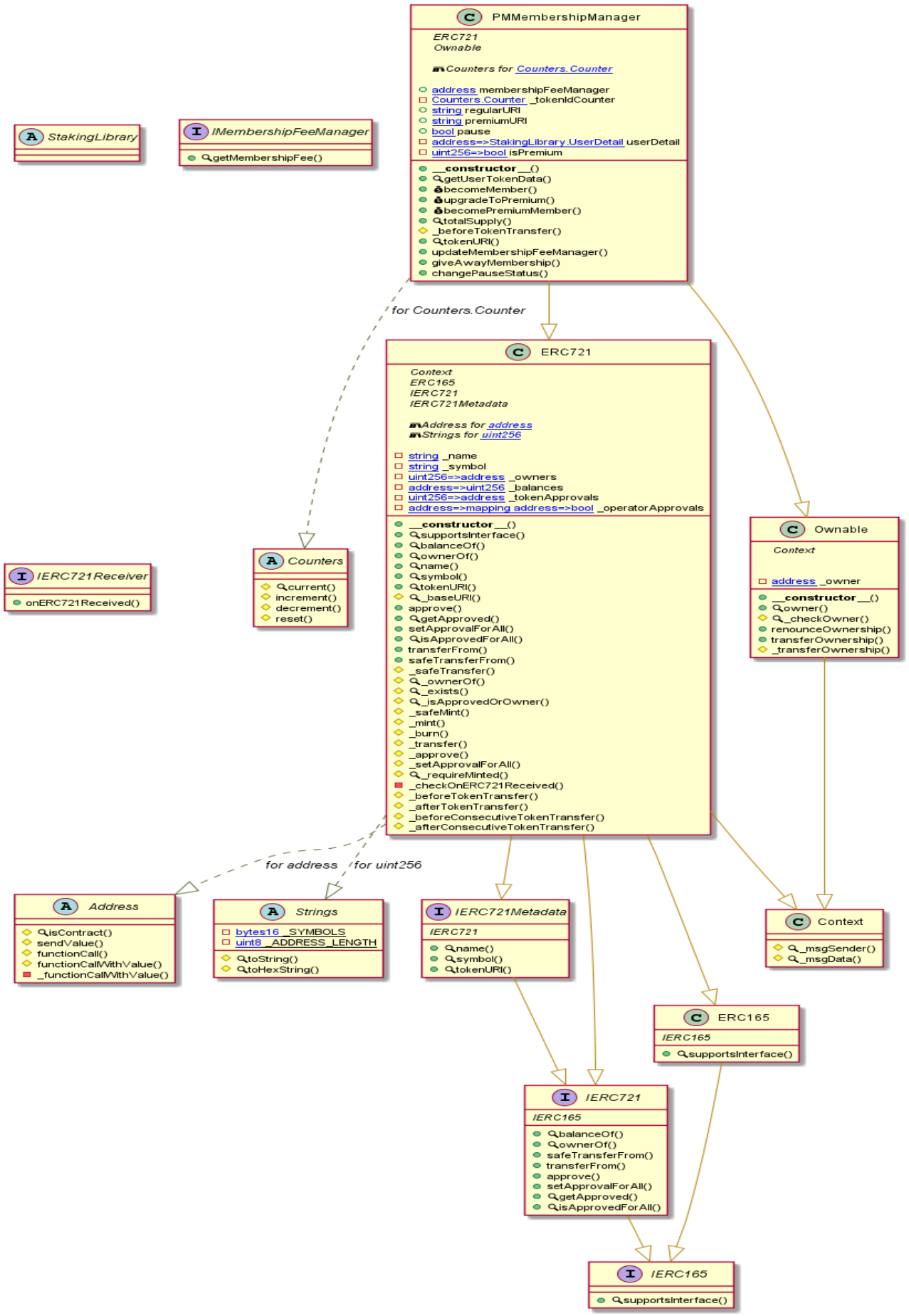
CreatorContract Diagram



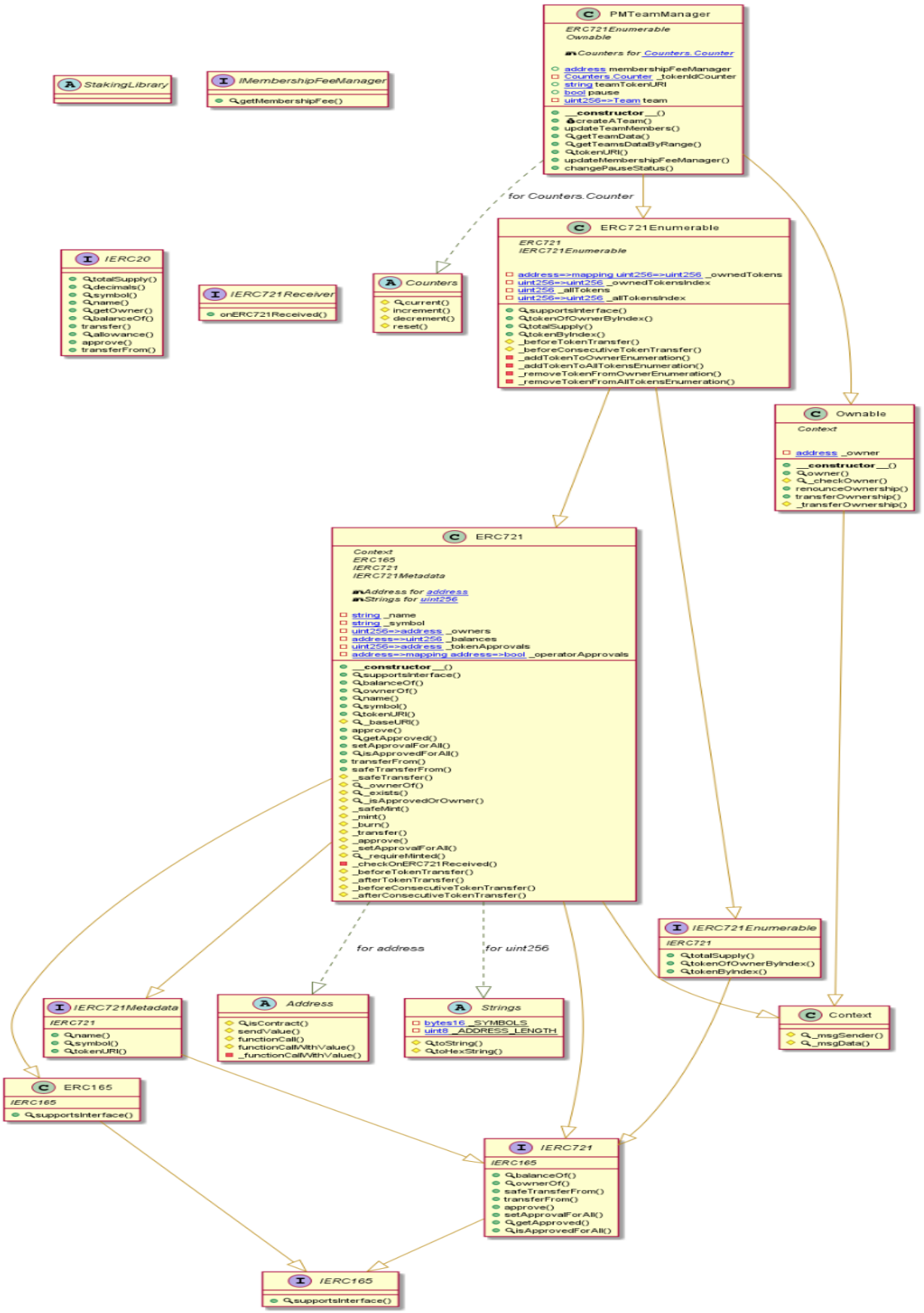
CreatorManager Diagram



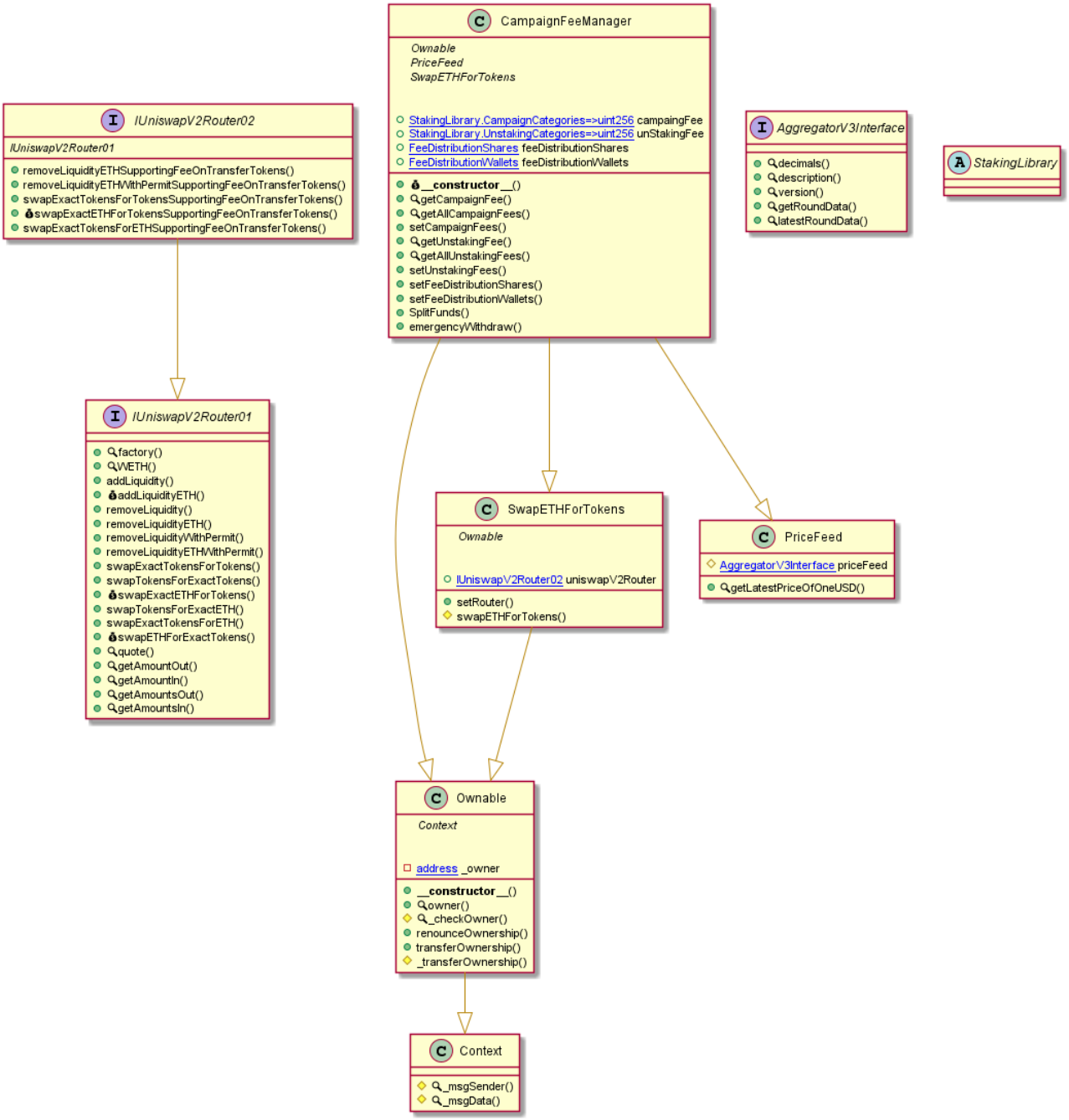
PMMembershipManager Diagram



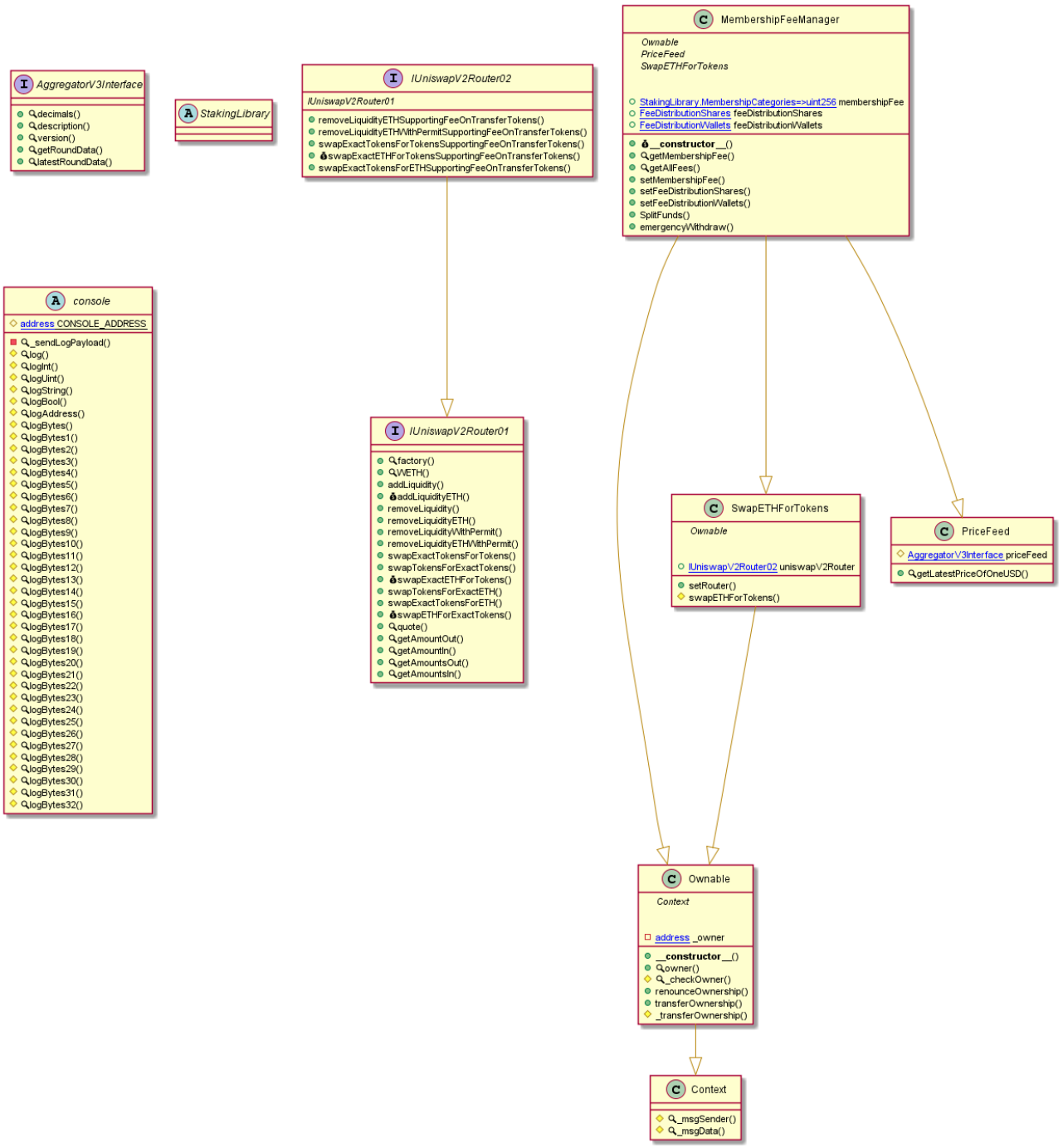
PMTeamManager Diagram



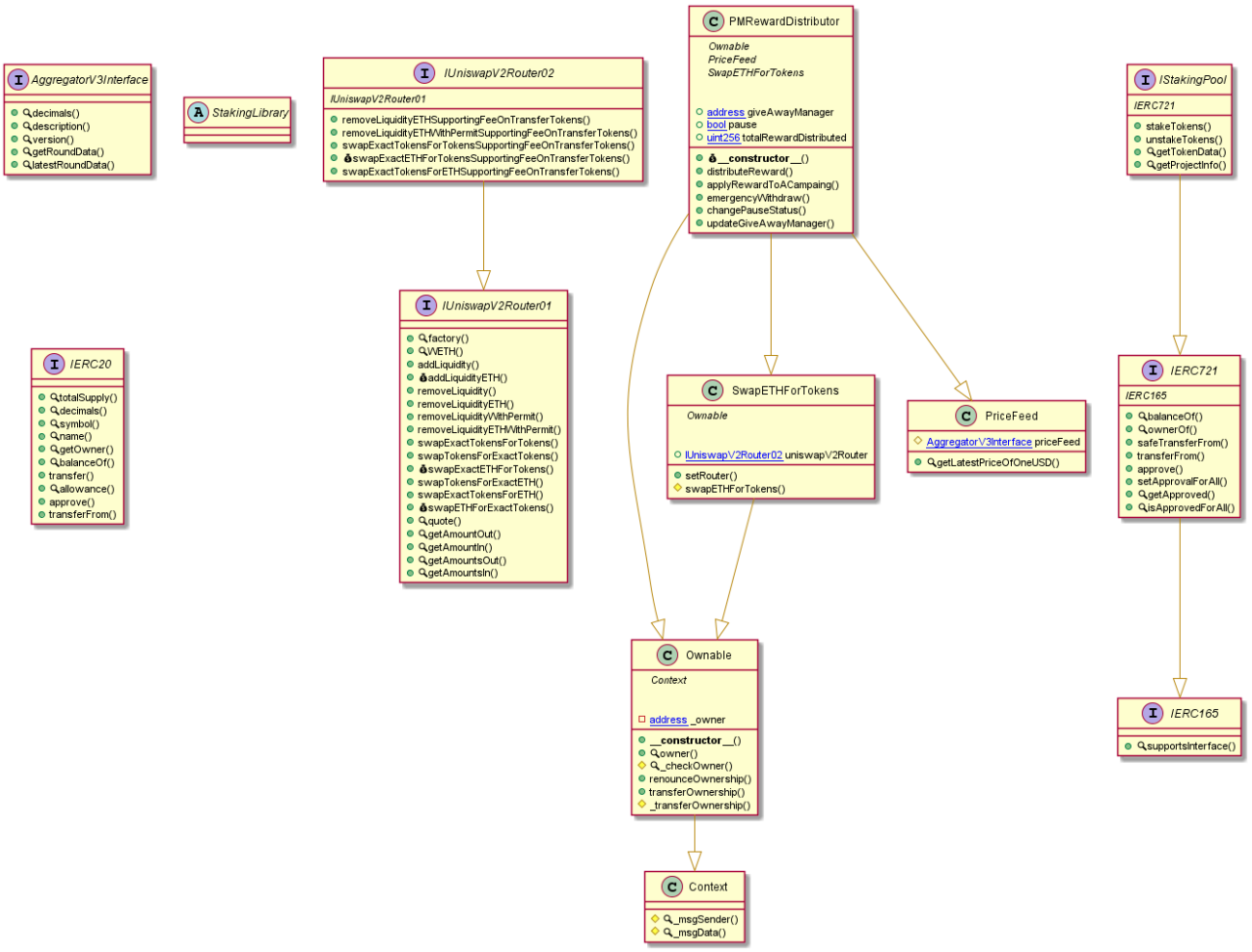
CampaignFeeManager Diagram



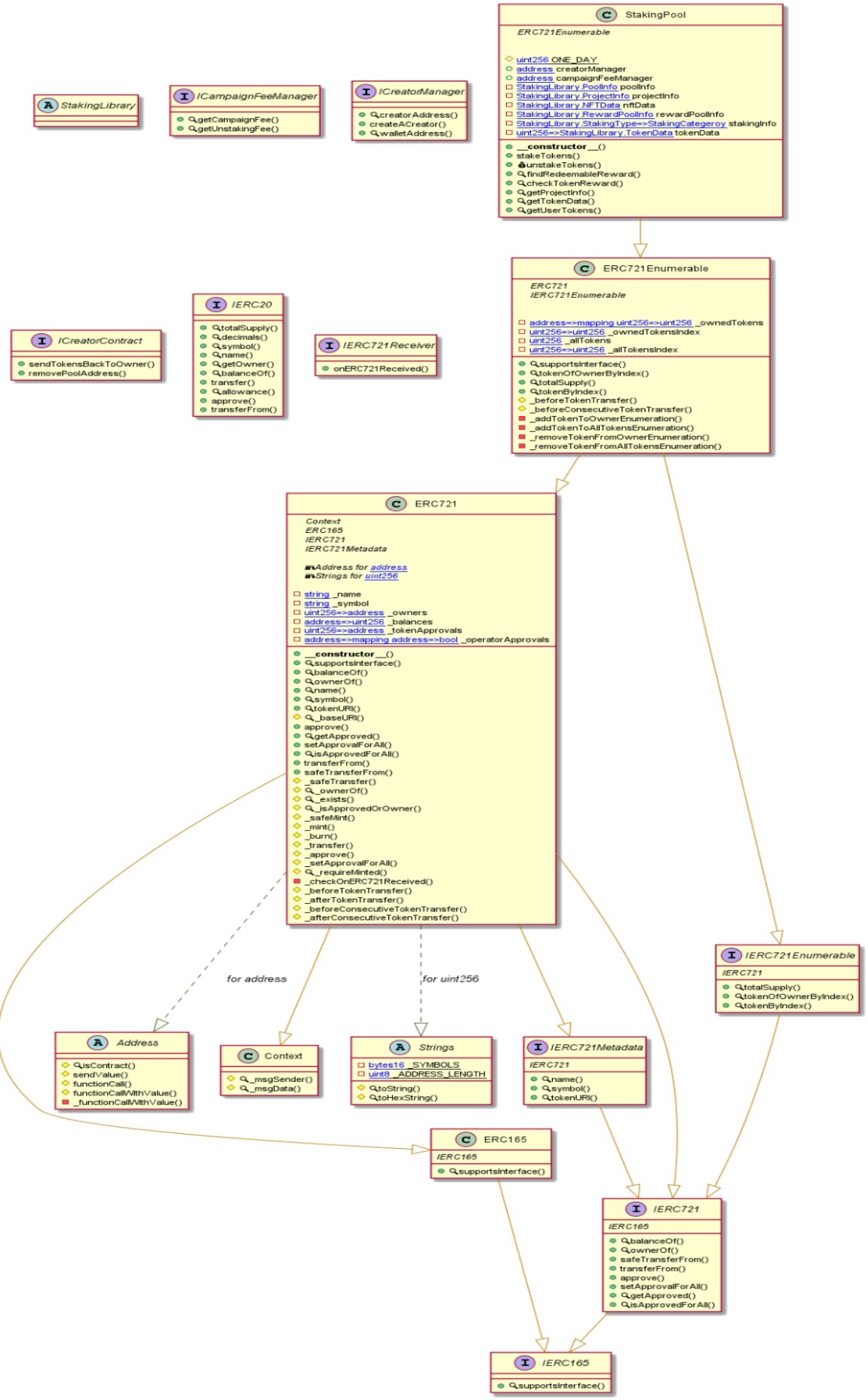
MembershipFeeManager Diagram



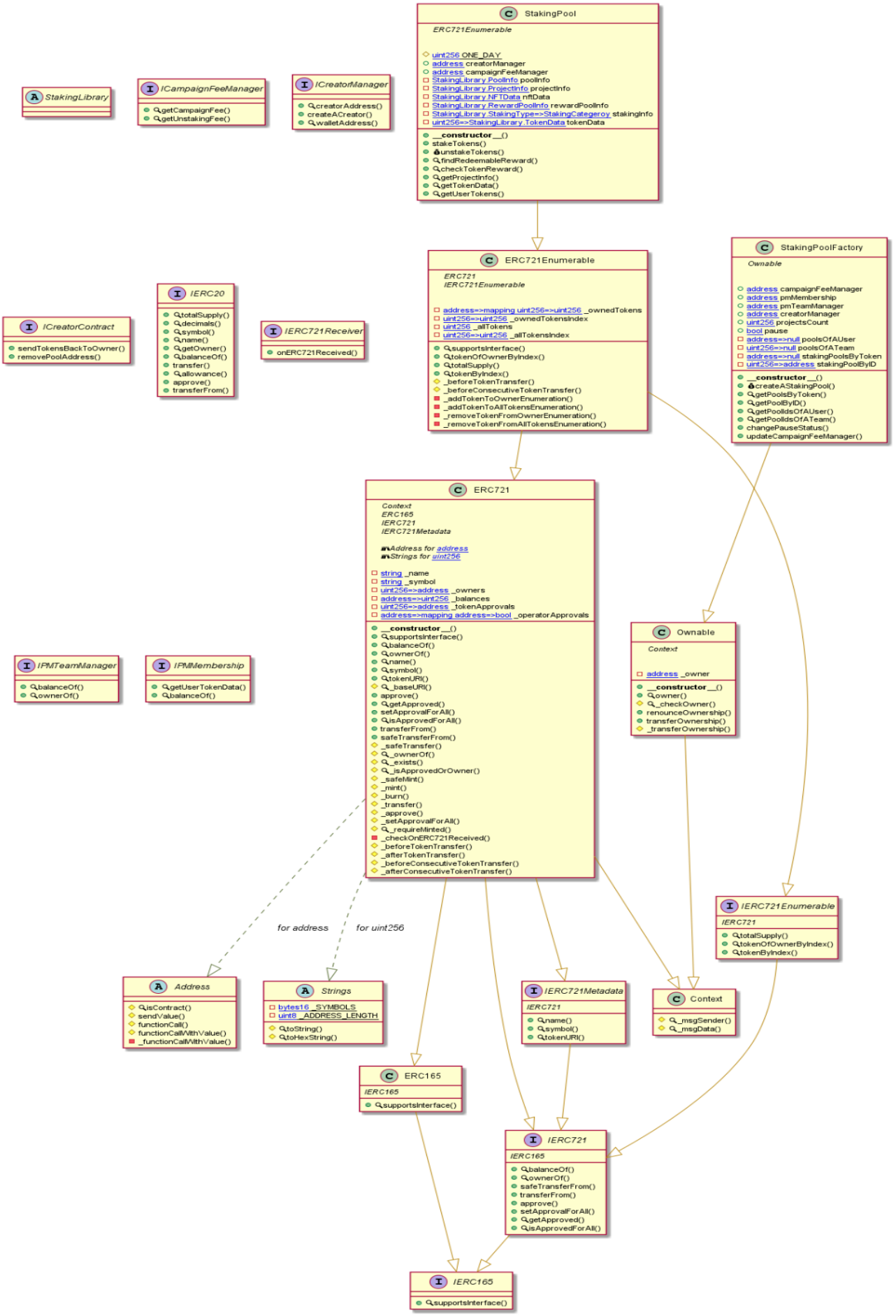
PMRewardDistributor Diagram



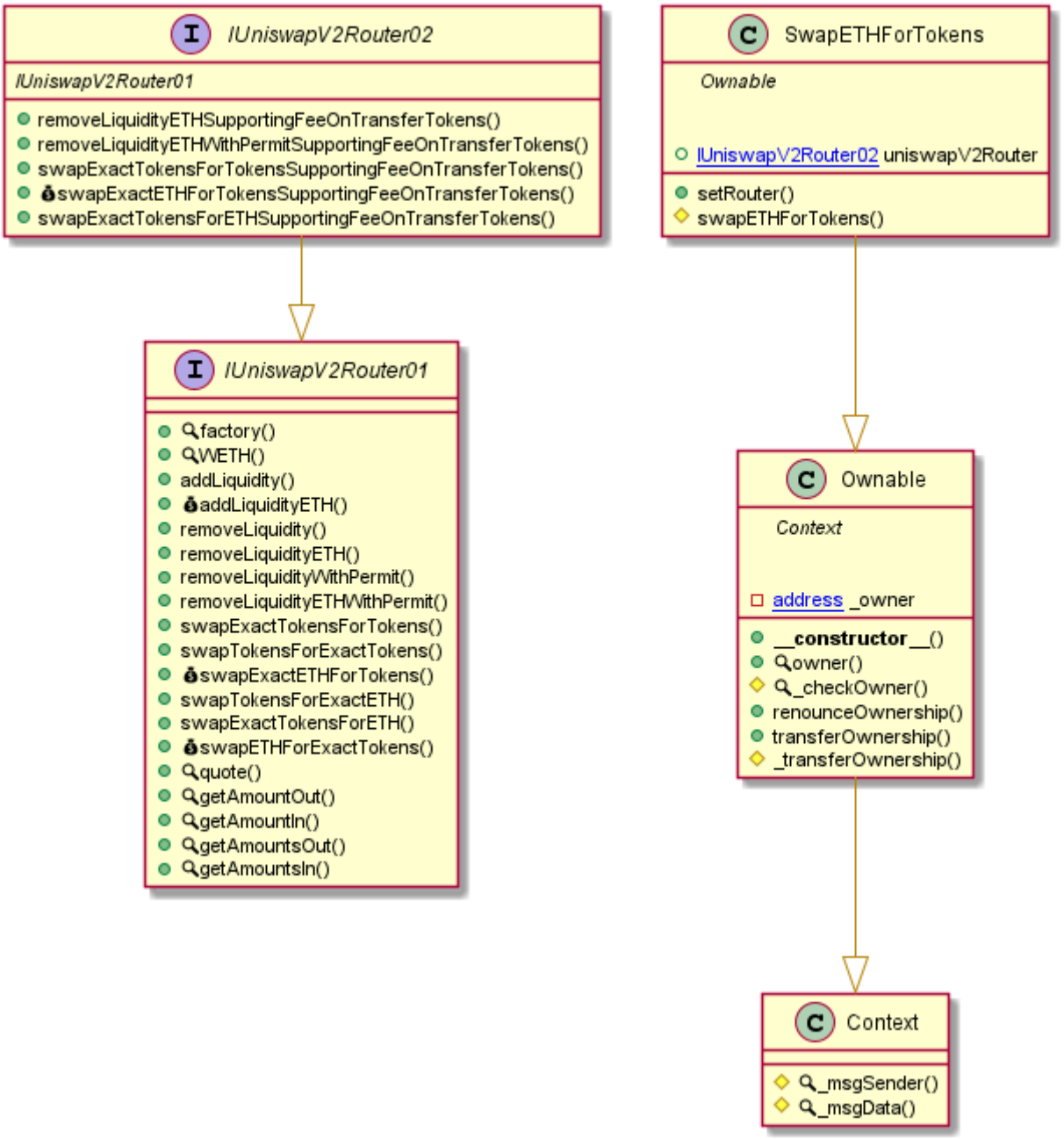
StakingPool Diagram



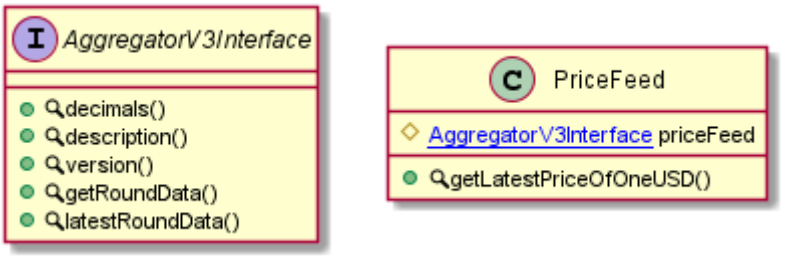
StakingPoolFactory Diagram



SwapETHForTokens Diagram



PriceFeed Diagram



Slither Results Log

Slither log >> CreatorContract.sol

```
CreatorContract.constructor(address)._owner (CreatorContract.sol#639) shadows:
        - Ownable._owner (CreatorContract.sol#542) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

EnumerableSet.values(EnumerableSet.Bytes32Set) (CreatorContract.sol#340-350) uses assembly
        - INLINE ASM (CreatorContract.sol#345-347)
EnumerableSet.values(EnumerableSet.AddressSet) (CreatorContract.sol#414-424) uses assembly
        - INLINE ASM (CreatorContract.sol#419-421)
EnumerableSet.values(EnumerableSet.UintSet) (CreatorContract.sol#488-498) uses assembly
        - INLINE ASM (CreatorContract.sol#493-495)
CreatorContract.onERC721Received(address,address,uint256,bytes) (CreatorContract.sol#643-657) uses assembly
        - INLINE ASM (CreatorContract.sol#647-649)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version^0.8.14 (CreatorContract.sol#2) allows old versions
solc-0.8.14 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter CreatorContract.sendTokensBackToOwner(address,uint256)._tokenId (CreatorContract.sol#659) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
CreatorContract.sol analyzed (11 contracts with 84 detectors), 28 result(s) found
```

Slither log >> CreatorManager.sol

```
CreatorContract.constructor(address)._owner (CreatorManager.sol#638) shadows:
        - Ownable._owner (CreatorManager.sol#540) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

EnumerableSet.values(EnumerableSet.Bytes32Set) (CreatorManager.sol#338-348) uses assembly
        - INLINE ASM (CreatorManager.sol#343-345)
EnumerableSet.values(EnumerableSet.AddressSet) (CreatorManager.sol#412-422) uses assembly
        - INLINE ASM (CreatorManager.sol#417-419)
EnumerableSet.values(EnumerableSet.UintSet) (CreatorManager.sol#486-496) uses assembly
        - INLINE ASM (CreatorManager.sol#491-493)
CreatorContract.onERC721Received(address,address,uint256,bytes) (CreatorManager.sol#642-656) uses assembly
        - INLINE ASM (CreatorManager.sol#646-648)
console._sendLogPayload(bytes) (CreatorManager.sol#682-689) uses assembly
        - INLINE ASM (CreatorManager.sol#685-688)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version0.8.9 (CreatorManager.sol#2) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter CreatorContract.sendTokensBackToOwner(address,uint256)._tokenId (CreatorManager.sol#658) is not in mixedCase
Parameter CreatorContract.removePoolAddress(address)._poolAddress (CreatorManager.sol#669) is not in mixedCase
Contract console (CreatorManager.sol#679-2207) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions




CreatorManager.creatorsCount (CreatorManager.sol#2214) is never used in CreatorManager (CreatorManager.sol#2211-2245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

CreatorContract.counter (CreatorManager.sol#636) should be constant
CreatorManager.creatorsCount (CreatorManager.sol#2214) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
CreatorManager.sol analyzed (13 contracts with 84 detectors), 415 result(s) found
```

Slither log >> PMMembershipManager.sol

```
PMMembershipManager.updateMembershipFeeManager(address)._membershipFeeManager (PMMembershipManager.sol#1216) lacks a zero-chec
k on :
                - membershipFeeManager = address(_membershipFeeManager) (PMMembershipManager.sol#1217)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

ERC721._checkOnERC721Received(address,address,uint256,bytes) (PMMembershipManager.sol#942-964) has external calls inside a loo
p: IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,data) (PMMembershipManager.sol#949-960)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (PMMembershipManager.sol#949)' in ERC721._checkO
nERC721Received(address,address,uint256,bytes) (PMMembershipManager.sol#942-964) potentially used before declaration: retval =
= IERC721Receiver.onERC721Received.selector (PMMembershipManager.sol#950)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (PMMembershipManager.sol#951)' in ERC721._checkO
nERC721Received(address,address,uint256,bytes) (PMMembershipManager.sol#942-964) potentially used before declaration: reason.l
ength == 0 (PMMembershipManager.sol#952)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (PMMembershipManager.sol#951)' in ERC721._checkO
nERC721Received(address,address,uint256,bytes) (PMMembershipManager.sol#942-964) potentially used before declaration: revert(u
int256,uint256)(32 + reason,mload(uint256)(reason)) (PMMembershipManager.sol#957)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in PMMembershipManager.giveAwayMembership(address[]) (PMMembershipManager.sol#1220-1231):
        External calls:
        - _safeMint(to[i],tokenId) (PMMembershipManager.sol#1227)
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,data) (PMMembershipManager.sol#949-960)
        State variables written after the call(s):
        - userDetail[to[i]] = StakingLibrary.UserDetail(block.timestamp,tokenId,true) (PMMembershipManager.sol#1226)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

PMMembershipManager.upgradeToPremium(address) (PMMembershipManager.sol#1158-1182) uses timestamp for comparisons
        Dangerous comparisons:
        - userDetail[userAddress].memberId == 0 (PMMembershipManager.sol#1160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
ERC721._burn(uint256) (PMMembershipManager.sol#834-855) is never used and should be removed
PMMembershipManager._beforeTokenTransfer(address,address,uint256,uint256) (PMMembershipManager.sol#1194-1197) is never used an
d should be removed
Strings.toHexString(address) (PMMembershipManager.sol#323-325) is never used and should be removed
Strings.toHexString(uint256) (PMMembershipManager.sol#299-303) is never used and should be removed
Strings.toHexString(uint256,uint256) (PMMembershipManager.sol#308-318) is never used and should be removed
Strings.toString(uint256) (PMMembershipManager.sol#291-294) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.9 (PMMembershipManager.sol#2) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (PMMembershipManager.sol#176-182):
        - (success) = recipient.call{value: amount}() (PMMembershipManager.sol#180)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (PMMembershipManager.sol#255-281):
        - (success,returndata) = target.call{value: weiValue}(data) (PMMembershipManager.sol#264)
Low level call in PMMembershipManager.becomeMember(address) (PMMembershipManager.sol#1126-1156):
        - (sent) = membershipFeeManager.call{value: msg.value}() (PMMembershipManager.sol#1150)
Low level call in PMMembershipManager.upgradeToPremium(address) (PMMembershipManager.sol#1158-1182):
        - (sent) = membershipFeeManager.call{value: msg.value}() (PMMembershipManager.sol#1176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter PMMembershipManager.updateMembershipFeeManager(address)._membershipFeeManager (PMMembershipManager.sol#1216) is not
in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (PMMembershipManager.sol#518)" inContext (PMMembershipManager.sol#512-521)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

PMMembershipManager.isPremium (PMMembershipManager.sol#1116) is never used in PMMembershipManager (PMMembershipManager.sol#110
3-1238)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```

```
PMMembershipManager.premiumURI (PMMembershipManager.sol#1111) should be constant
PMMembershipManager.regularURI (PMMembershipManager.sol#1110) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
PMMembershipManager.sol analyzed (14 contracts with 84 detectors), 41 result(s) found
```

Slither log >> PMTeamManager.sol

```
PMTeamManager.updateMembershipFeeManager(address)._membershipFeeManager (PMTeamManager.sol#1436) lacks a zero-check on :
                - membershipFeeManager = address(_membershipFeeManager) (PMTeamManager.sol#1437)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Reentrancy in PMTeamManager.createATeam(address) (PMTeamManager.sol#1357-1388):
        External calls:
        - _safeMint(to,tokenId) (PMTeamManager.sol#1377)
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,data) (PMTeamManager.sol#989-1000)
        - (sent) = membershipFeeManager.call{value: msg.value}() (PMTeamManager.sol#1380)
        External calls sending eth:
        - (sent) = membershipFeeManager.call{value: msg.value}() (PMTeamManager.sol#1380)
        Event emitted after the call(s):
        - TeamCreated(tokenId,createdAt) (PMTeamManager.sol#1386)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.isContract(address) (PMTeamManager.sol#194-205) uses assembly
        - INLINE ASM (PMTeamManager.sol#201-203)
Address._functionCallWithValue(address,bytes,uint256,string) (PMTeamManager.sol#280-306) uses assembly
        - INLINE ASM (PMTeamManager.sol#298-301)
Strings.toString(uint256) (PMTeamManager.sol#316-334) uses assembly
        - INLINE ASM (PMTeamManager.sol#326-328)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (PMTeamManager.sol#982-1004) uses assembly
        - INLINE ASM (PMTeamManager.sol#996-998)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version0.8.9 (PMTeamManager.sol#2) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (PMTeamManager.sol#206-212):
        - (success) = recipient.call{value: amount}() (PMTeamManager.sol#210)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (PMTeamManager.sol#280-306):
        - (success,returndata) = target.call{value: weiValue}(data) (PMTeamManager.sol#289)
Low level call in PMTeamManager.createATeam(address) (PMTeamManager.sol#1357-1388):
        - (sent) = membershipFeeManager.call{value: msg.value}() (PMTeamManager.sol#1380)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter PMTeamManager.updateMembershipFeeManager(address)._membershipFeeManager (PMTeamManager.sol#1436) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (PMTeamManager.sol#558)" inContext (PMTeamManager.sol#552-561)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

PMTeamManager.teamTokenURI (PMTeamManager.sol#1341) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
PMTeamManager.sol analyzed (17 contracts with 84 detectors), 36 result(s) found
```

Slither log >> CampaignFeeManager.sol

```
Context._msgData() (CampaignFeeManager.sol#145-148) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.14 (CampaignFeeManager.sol#2) allows old versions
solc-0.8.14 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IUniswapV2Router01.WETH() (CampaignFeeManager.sol#6) is not in mixedCase
Parameter SwapETHForTokens.setRouter(IUniswapV2Router02)._uniswapV2Router (CampaignFeeManager.sol#222) is not in mixedCase
Parameter CampaignFeeManager.setUnstakingFees(uint256,uint256,uint256,uint256).reward_0pc (CampaignFeeManager.sol#502) is not
 mixedCase
Parameter CampaignFeeManager.setUnstakingFees(uint256,uint256,uint256,uint256).reward_30pc (CampaignFeeManager.sol#503) is not
n mixedCase
Parameter CampaignFeeManager.setUnstakingFees(uint256,uint256,uint256,uint256).reward_50pc (CampaignFeeManager.sol#504) is not
n mixedCase
Parameter CampaignFeeManager.setUnstakingFees(uint256,uint256,uint256,uint256).reward_100pc (CampaignFeeManager.sol#505) is no
in mixedCase
Function CampaignFeeManager.SplitFunds() (CampaignFeeManager.sol#543-576) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (CampaignFeeManager.sol#146)" inContext (CampaignFeeManager.sol#140-149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

```
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Camp
gnFeeManager.sol#11) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address
int256).amountBDesired (CampaignFeeManager.sol#12)
Variable CampaignFeeManager.constructor(uint256,uint256,uint256,uint256,uint256,uint256,uint256).reward_30pc (CampaignFeeManag
.sol#431) is too similar to CampaignFeeManager.setUnstakingFees(uint256,uint256,uint256,uint256).reward_50pc (CampaignFeeManag
.sol#504)
Variable CampaignFeeManager.setUnstakingFees(uint256,uint256,uint256,uint256).reward_30pc (CampaignFeeManager.sol#503) is too
milar to CampaignFeeManager.getAllUnstakingFees(CampaignFeeManager.FeesType).reward_50pc (CampaignFeeManager.sol#482)
Variable CampaignFeeManager.setUnstakingFees(uint256,uint256,uint256,uint256).reward_30pc (CampaignFeeManager.sol#503) is too
milar to CampaignFeeManager.constructor(uint256,uint256,uint256,uint256,uint256,uint256,uint256).reward_50pc (CampaignFeeManag
.sol#431)
Variable CampaignFeeManager.constructor(uint256,uint256,uint256,uint256,uint256,uint256,uint256).reward_30pc (CampaignFeeManag
.sol#431) is too similar to CampaignFeeManager.constructor(uint256,uint256,uint256,uint256,uint256,uint256,uint256).reward_50p
(CampaignFeeManager.sol#431)
Variable CampaignFeeManager.setUnstakingFees(uint256,uint256,uint256,uint256).reward_30pc (CampaignFeeManager.sol#503) is too
milar to CampaignFeeManager.setUnstakingFees(uint256,uint256,uint256,uint256).reward_50pc (CampaignFeeManager.sol#504)
Variable CampaignFeeManager.constructor(uint256,uint256,uint256,uint256,uint256,uint256,uint256).reward_30pc (CampaignFeeManag
.sol#431) is too similar to CampaignFeeManager.getAllUnstakingFees(CampaignFeeManager.FeesType).reward_50pc (CampaignFeeManage
sol#482)
Variable CampaignFeeManager.getAllUnstakingFees(CampaignFeeManager.FeesType).reward_30pc (CampaignFeeManager.sol#481) is too s
ilar to CampaignFeeManager.setUnstakingFees(uint256,uint256,uint256,uint256).reward_50pc (CampaignFeeManager.sol#504)
Variable CampaignFeeManager.getAllUnstakingFees(CampaignFeeManager.FeesType).reward_30pc (CampaignFeeManager.sol#481) is too s
ilar to CampaignFeeManager.constructor(uint256,uint256,uint256,uint256,uint256,uint256,uint256).reward_50pc (CampaignFeeManage
sol#431)
Variable CampaignFeeManager.getAllUnstakingFees(CampaignFeeManager.FeesType).reward_30pc (CampaignFeeManager.sol#481) is too s
ilar to CampaignFeeManager.getAllUnstakingFees(CampaignFeeManager.FeesType).reward_50pc (CampaignFeeManager.sol#482)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

PriceFeed.priceFeed (CampaignFeeManager.sol#271) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
CampaignFeeManager.sol analyzed (9 contracts with 84 detectors), 23 result(s) found
```

Slither log >> MembershipFeeManager.sol

```
console._sendLogPayload(bytes) (MembershipFeeManager.sol#390-397) uses assembly
        - INLINE ASM (MembershipFeeManager.sol#393-396)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version^0.8.14 (MembershipFeeManager.sol#2) allows old versions
solc-0.8.14 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IUniswapV2Router01.WETH() (MembershipFeeManager.sol#6) is not in mixedCase
Parameter SwapETHForTokens.setRouter(IUniswapV2Router02)._uniswapV2Router (MembershipFeeManager.sol#222) is not in mixedCase
Contract console (MembershipFeeManager.sol#387-1915) is not in CapWords
Function MembershipFeeManager.SplitFunds() (MembershipFeeManager.sol#2015-2048) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (MembershipFeeManager.sol#146)" inContext (MembershipFeeManager.sol#140-149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements




Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Memb
ershipFeeManager.sol#11) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,add
ress,uint256).amountBDesired (MembershipFeeManager.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

PriceFeed.priceFeed (MembershipFeeManager.sol#269) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
MembershipFeeManager.sol analyzed (10 contracts with 84 detectors), 392 result(s) found
```

Slither log >> PMRewardDistributor.sol

```
PMRewardDistributor.updateGiveAwayManager(address) (PMRewardDistributor.sol#564-566) should emit an event for:
        - giveAwayManager = _giveAwayManager (PMRewardDistributor.sol#565)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

PMRewardDistributor.distributeReward(address,uint8).winner (PMRewardDistributor.sol#496) lacks a zero-check on :
                - address(winner).transfer(prizeBNB) (PMRewardDistributor.sol#513)
PMRewardDistributor.updateGiveAwayManager(address)._giveAwayManager (PMRewardDistributor.sol#564) lacks a zero-check on :
                - giveAwayManager = _giveAwayManager (PMRewardDistributor.sol#565)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in PMRewardDistributor.applyRewardToACampaing(address,address,uint8,StakingLibrary.StakingType) (PMRewardDistributo
r.sol#517-550):
        External calls:
        - boughtTokens = swapETHForTokens(tokenAddress,address(this),amount) (PMRewardDistributor.sol#540)
                - amounts = uniswapV2Router.swapExactETHForTokens{value: amount}(0,path,toAddress,block.timestamp + 500) (PMRe
wardDistributor.sol#236-243)
        - IERC20(tokenAddress).approve(campaign,boughtTokens) (PMRewardDistributor.sol#543)
        - IStakingPool(campaign).stakeTokens(user,boughtTokens,_type) (PMRewardDistributor.sol#546)
        External calls sending eth:
        - boughtTokens = swapETHForTokens(tokenAddress,address(this),amount) (PMRewardDistributor.sol#540)
                - amounts = uniswapV2Router.swapExactETHForTokens{value: amount}(0,path,toAddress,block.timestamp + 500) (PMRe
wardDistributor.sol#236-243)
        Event emitted after the call(s):
        - RewardApplied(campaign,user,boughtTokens,uint8(_type)) (PMRewardDistributor.sol#548)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Context._msgData() (PMRewardDistributor.sol#145-148) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
Pragma version^0.8.14 (PMRewardDistributor.sol#2) allows old versions
solc-0.8.14 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IUniswapV2Router01.WETH() (PMRewardDistributor.sol#6) is not in mixedCase
Parameter SwapETHForTokens.setRouter(IUniswapV2Router02)._uniswapV2Router (PMRewardDistributor.sol#222) is not in mixedCase
Parameter PMRewardDistributor.applyRewardToACampaing(address,address,uint8,StakingLibrary.StakingType)._type (PMRewardDistribu
tor.sol#517) is not in mixedCase
Parameter PMRewardDistributor.updateGiveAwayManager(address)._giveAwayManager (PMRewardDistributor.sol#564) is not in mixedCas
e
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (PMRewardDistributor.sol#146)" inContext (PMRewardDistributor.sol#140-149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PMRe
wardDistributor.sol#11) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,addr
ess,uint256).amountBDesired (PMRewardDistributor.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

PriceFeed.priceFeed (PMRewardDistributor.sol#270) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
PMRewardDistributor.sol analyzed (13 contracts with 84 detectors), 16 result(s) found
```

Slither log >> StakingPool.sol

```
StakingPool.constructor(uint256,StakingLibrary.ProjectInfo,StakingLibrary.RewardPoolInfo,StakingLibrary.NFTData,address,addres
s,address)._creatorManager (StakingPool.sol#1270) lacks a zero-check on :
                - creatorManager = _creatorManager (StakingPool.sol#1300)
StakingPool.constructor(uint256,StakingLibrary.ProjectInfo,StakingLibrary.RewardPoolInfo,StakingLibrary.NFTData,address,addres
s,address)._campaignFeeManager (StakingPool.sol#1271) lacks a zero-check on :
                - campaignFeeManager = _campaignFeeManager (StakingPool.sol#1301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (StakingPool.sol#968)' in ERC721._checkOnERC721R
eceived(address,address,uint256,bytes) (StakingPool.sol#961-983) potentially used before declaration: retval == IERC721Receive
r.onERC721Received.selector (StakingPool.sol#969)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (StakingPool.sol#970)' in ERC721._checkOnERC721R
eceived(address,address,uint256,bytes) (StakingPool.sol#961-983) potentially used before declaration: reason.length == 0 (Stak
ingPool.sol#971)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (StakingPool.sol#970)' in ERC721._checkOnERC721R
eceived(address,address,uint256,bytes) (StakingPool.sol#961-983) potentially used before declaration: revert(uint256,uint256)(
32 + reason,mload(uint256)(reason)) (StakingPool.sol#976)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
Reentrancy in StakingPool.stakeTokens(address,uint256,StakingLibrary.StakingType) (StakingPool.sol#1306-1380):
        External calls:
        - creator = ICreatorManager(creatorManager).createACreator(onBehalf) (StakingPool.sol#1346)
        State variables written after the call(s):
        - tokenData[poolInfo.tokenCounter] = StakingLibrary.TokenData(address(this),poolInfo.poolId,amount,projectInfo.tokenAd
dress,address(onBehalf),address(creator),poolInfo.tokenCounter,tokenUri,uint8(_type),block.timestamp,block.timestamp + categor
y.duration,reward,false,0,0) (StakingPool.sol#1350-1366)
Reentrancy in StakingPool.unstakeTokens(uint256) (StakingPool.sol#1383-1444):
        External calls:
        - (sent) = campaignFeeManager.call{value: msg.value}() (StakingPool.sol#1420)
        - transfered = ICreatorContract(creator).sendTokensBackToOwner(address(this),_tokenId) (StakingPool.sol#1427)
        - transfered = IERC20(projectInfo.tokenAddress).transfer(_tokenData.owner,redeemableReward) (StakingPool.sol#1432)
        External calls sending eth:
        - (sent) = campaignFeeManager.call{value: msg.value}() (StakingPool.sol#1420)
        State variables written after the call(s):
        - _burn(_tokenData.tokenId) (StakingPool.sol#1437)
                - _allTokens.push(tokenId) (StakingPool.sol#1189)
                - _allTokens[tokenIndex] = lastTokenId (StakingPool.sol#1237)
                - _allTokens.pop() (StakingPool.sol#1242)
        - _burn(_tokenData.tokenId) (StakingPool.sol#1437)
                - _allTokensIndex[tokenId] = _allTokens.length (StakingPool.sol#1188)
                - _allTokensIndex[lastTokenId] = tokenIndex (StakingPool.sol#1238)
                - delete _allTokensIndex[tokenId] (StakingPool.sol#1241)
        - _burn(_tokenData.tokenId) (StakingPool.sol#1437)
                - _balances[owner] -= 1 (StakingPool.sol#867)
        - _burn(_tokenData.tokenId) (StakingPool.sol#1437)
                - _ownedTokens[to][length] = tokenId (StakingPool.sol#1179)
                - _ownedTokens[from][tokenIndex] = lastTokenId (StakingPool.sol#1211)
                - delete _ownedTokens[from][lastTokenIndex] (StakingPool.sol#1217)
        - _burn(_tokenData.tokenId) (StakingPool.sol#1437)
                - _ownedTokensIndex[tokenId] = length (StakingPool.sol#1180)
                - _ownedTokensIndex[lastTokenId] = tokenIndex (StakingPool.sol#1212)
                - delete _ownedTokensIndex[tokenId] (StakingPool.sol#1216)
        - _burn(_tokenData.tokenId) (StakingPool.sol#1437)
                - delete _owners[tokenId] (StakingPool.sol#869)
```

```
        - _burn(_tokenData.tokenId) (StakingPool.sol#1437)
                - delete _owners[tokenId] (StakingPool.sol#869)
        - _burn(_tokenData.tokenId) (StakingPool.sol#1437)
                - delete _tokenApprovals[tokenId] (StakingPool.sol#862)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Pragma version0.8.9 (StakingPool.sol#2) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (StakingPool.sol#194-200):
        - (success) = recipient.call{value: amount}() (StakingPool.sol#198)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (StakingPool.sol#273-299):
        - (success,returndata) = target.call{value: weiValue}(data) (StakingPool.sol#282)
Low level call in StakingPool.unstakeTokens(uint256) (StakingPool.sol#1383-1444):
        - (sent) = campaignFeeManager.call{value: msg.value}() (StakingPool.sol#1420)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter StakingPool.stakeTokens(address,uint256,StakingLibrary.StakingType)._type (StakingPool.sol#1306) is not in mixedCase
Parameter StakingPool.unstakeTokens(uint256)._tokenId (StakingPool.sol#1383) is not in mixedCase
Parameter StakingPool.findRedeemableReward(uint256,uint256,uint256)._expectedReward (StakingPool.sol#1449) is not in mixedCase
Parameter StakingPool.findRedeemableReward(uint256,uint256,uint256)._stakingTime (StakingPool.sol#1449) is not in mixedCase
Parameter StakingPool.findRedeemableReward(uint256,uint256,uint256)._unlockTime (StakingPool.sol#1449) is not in mixedCase
Parameter StakingPool.checkTokenReward(uint256)._tokenId (StakingPool.sol#1481) is not in mixedCase
Parameter StakingPool.getTokenData(uint256)._tokenId (StakingPool.sol#1499) is not in mixedCase
Parameter StakingPool.getUserTokens(address)._user (StakingPool.sol#1503) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (StakingPool.sol#537)" inContext (StakingPool.sol#531-540)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

StakingPool.campaignFeeManager (StakingPool.sol#1249) should be immutable
StakingPool.creatorManager (StakingPool.sol#1248) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
StakingPool.sol analyzed (17 contracts with 84 detectors), 48 result(s) found
```

Slither log >> StakingPoolFactory.sol

```
Pragma version0.8.9 (StakingPoolFactory.sol#2) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Parameter StakingPool.stakeTokens(address,uint256,StakingLibrary.StakingType)._type (StakingPoolFactory.sol#1323) is not in mi
xedCase
Parameter StakingPool.unstakeTokens(uint256)._tokenId (StakingPoolFactory.sol#1400) is not in mixedCase
Parameter StakingPool.findRedeemableReward(uint256,uint256,uint256)._expectedReward (StakingPoolFactory.sol#1466) is not in mi
xedCase
Parameter StakingPool.findRedeemableReward(uint256,uint256,uint256)._stakingTime (StakingPoolFactory.sol#1466) is not in mixed
Case
Parameter StakingPool.findRedeemableReward(uint256,uint256,uint256)._unlockTime (StakingPoolFactory.sol#1466) is not in mixedC
ase
Parameter StakingPool.checkTokenReward(uint256)._tokenId (StakingPoolFactory.sol#1498) is not in mixedCase
Parameter StakingPool.getTokenData(uint256)._tokenId (StakingPoolFactory.sol#1516) is not in mixedCase
Parameter StakingPool.getUserTokens(address)._user (StakingPoolFactory.sol#1520) is not in mixedCase
Parameter StakingPoolFactory.updateCampaignFeeManager(address)._campaignFeeManager (StakingPoolFactory.sol#1749) is not in mix
edCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (StakingPoolFactory.sol#553)" inContext (StakingPoolFactory.sol#547-556)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

StakingPool.campaignFeeManager (StakingPoolFactory.sol#1266) should be immutable
StakingPool.creatorManager (StakingPoolFactory.sol#1265) should be immutable
StakingPoolFactory.creatorManager (StakingPoolFactory.sol#1636) should be immutable
StakingPoolFactory.pmMembership (StakingPoolFactory.sol#1634) should be immutable
StakingPoolFactory.pmTeamManager (StakingPoolFactory.sol#1635) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
StakingPoolFactory.sol analyzed (21 contracts with 84 detectors), 56 result(s) found
```

Slither log >> SwapETHForTokens.sol

```
Context._msgData() (SwapETHForTokens.sol#145-148) is never used and should be removed
SwapETHForTokens.swapETHForTokens(address,address,uint256) (SwapETHForTokens.sol#226-246) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.14 (SwapETHForTokens.sol#2) allows old versions
solc-0.8.14 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IUniswapV2Router01.WETH() (SwapETHForTokens.sol#6) is not in mixedCase
Parameter SwapETHForTokens.setRouter(IUniswapV2Router02)._uniswapV2Router (SwapETHForTokens.sol#222) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (SwapETHForTokens.sol#146)" inContext (SwapETHForTokens.sol#140-149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Swap
ETHForTokens.sol#11) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address
,uint256).amountBDesired (SwapETHForTokens.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
SwapETHForTokens.sol analyzed (5 contracts with 84 detectors), 9 result(s) found
```

Slither log >> PriceFeed.sol

```
Pragma version^0.8.14 (PriceFeed.sol#2) allows old versions
solc-0.8.14 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

PriceFeed.priceFeed (PriceFeed.sol#23) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
PriceFeed.sol analyzed (2 contracts with 84 detectors), 3 result(s) found
```

Solidity Static Analysis

CreatorContract.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: 56:19:

Check-effects-interaction:

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

more

Pos: 44:6:

Gas & Economy

Gas costs:

Gas requirement of function CreatorContract.getPoolAddresses 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: 60:6:

This on local calls:

Use of "this" for local functions: Never use "this" to call functions in the same contract, it only consumes more gas than normal local calls.

more

Pos: 40:17:

Miscellaneous

Email: audit@EtherAuthority.io

Guard conditions:

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

more

Pos: 56:10:

CreatorManager.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: 56:19:

Gas & Economy

Gas costs:

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

Pos: 19:4:

Miscellaneous

Constant/View/Pure functions:

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

more

Pos: 60:6:

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: 56:10:

Email: audit@EtherAuthority.io

Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 169:62:

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: 125:23:

Gas & Economy

Gas costs:

Gas requirement of function PMMembershipManager.giveAwayMembership 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: 163:4:

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: 165:8:

Miscellaneous

Constant/View/Pure functions:

PMMembershipManager._beforeTokenTransfer(address,address,uint256,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 137:4:

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: 138:8:

PMTeamManager.sol

Security

Check-effects-interaction:

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

more

Pos: 43:4:

Gas & Economy

Gas costs:

Gas requirement of function PMTeamManager.getTeamsDataByRange 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: 93:4:

Miscellaneous

Constant/View/Pure functions:

PMTeamManager.getTeamData(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 85:4:

CampaignFeeManager.sol

Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 196:12:

Gas & Economy

Gas costs:

Gas requirement of function CampaignFeeManager.feeDistributionWallets 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: 45:4:

Miscellaneous

Similar variable names:

CampaignFeeManager.setUnstakingFees(uint256,uint256,uint256,uint256) : Variables have very similar names "reward_30pc" and "reward_100pc". Note: Modifiers are currently not considered by this static analysis.
Pos: 134:72:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 176:8:

MembershipFeeManager.sol

Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 157:12:

Gas & Economy

Gas costs:

Gas requirement of function MembershipFeeManager.emergencyWithdraw 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: 131:4:

Miscellaneous

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 169:16:

Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 125:12:

Gas & Economy

Gas costs:

Gas requirement of function PMRewardDistributor.emergencyWithdraw 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: 145:4:

Miscellaneous

Similar variable names:

PMRewardDistributor.swapETHForTokensNoFee(address,address,uint256) : Variables have very similar names "amount" and "amounts". Note: Modifiers are currently not considered by this static analysis.
Pos: 127:31:

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: 147:8:

StakingPool.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: 204:23:

Gas & Economy

Gas costs:

Gas requirement of function StakingPool.getUserTokens 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: 287:4:

Miscellaneous

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 249:31:

StakingPoolFactory.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: 204:23:

Gas & Economy

Gas costs:

Gas requirement of function StakingPool.getUserTokens 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: 287:4:

Miscellaneous

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 249:31:

SwapETHForTokens.sol

Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 32:12:

Gas & Economy

Gas costs:

Gas requirement of function SwapETHForTokens.uniswapV2Router 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: 9:4:

Miscellaneous

Similar variable names:

SwapETHForTokens.swapETHForTokens(address,address,uint256) : Variables have very similar names "amount" and "amounts". Note: Modifiers are currently not considered by this static analysis.
Pos: 34:31:

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: 70:8:

PriceFeed.sol

Gas & Economy

Gas costs:

Gas requirement of function PriceFeed.getLatestPriceOfOneUSD 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: 24:4:

Miscellaneous

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 31:16:

Solhint Linter

CreatorContract.sol

```
CreatorContract.sol:13:6: Error: Parse error: missing 'constant' at
'INVALID_POOLID'
CreatorContract.sol:13:20: Error: Parse error: missing '=' at '('
CreatorContract.sol:14:6: Error: Parse error: missing 'constant' at
'INVALID_TOKEN_ID'
CreatorContract.sol:14:22: Error: Parse error: missing '=' at '('
CreatorContract.sol:15:6: Error: Parse error: missing 'constant' at
'ALLREADY_UNSTAKED'
CreatorContract.sol:15:23: Error: Parse error: missing '=' at '('
CreatorContract.sol:16:6: Error: Parse error: missing 'constant' at
'NOT_AUTHERIZED'
CreatorContract.sol:16:20: Error: Parse error: missing '=' at '('
CreatorContract.sol:48:33: Error: Parse error: mismatched input '('
expecting {';', '='}
```

CreatorManager.sol

```
CreatorManager.sol:8:6: Error: Parse error: missing 'constant' at
'ALREADY_EXIST'
CreatorManager.sol:8:19: Error: Parse error: missing '=' at '('
CreatorManager.sol:9:6: Error: Parse error: missing 'constant' at
'NOT_EXIST'
CreatorManager.sol:9:15: Error: Parse error: missing '=' at '('
CreatorManager.sol:22:32: Error: Parse error: mismatched input '('
expecting {';', '='}
CreatorManager.sol:39:28: Error: Parse error: mismatched input '('
expecting {';', '='}
CreatorManager.sol:48:28: Error: Parse error: mismatched input '('
expecting {';', '='}
```

PMMembershipManager.sol

```
PMMembershipManager.sol:13:6: Error: Parse error: missing 'constant'
at 'ALREADY_A_MEMBER'
PMMembershipManager.sol:13:22: Error: Parse error: missing '=' at '('
PMMembershipManager.sol:14:6: Error: Parse error: missing 'constant'
at 'INSUFFICIENT_FUNDS'
PMMembershipManager.sol:14:24: Error: Parse error: missing '=' at '('
PMMembershipManager.sol:15:6: Error: Parse error: missing 'constant'
at 'FAILED_TO_TRANSFER_BNBS'
PMMembershipManager.sol:15:29: Error: Parse error: missing '=' at '('
PMMembershipManager.sol:16:6: Error: Parse error: missing 'constant'
at 'NOT_A_MEMBER'
PMMembershipManager.sol:16:18: Error: Parse error: missing '=' at '('
```

```
PMMembershipManager.sol:17:6: Error: Parse error: missing 'constant'
at 'ALREADY_A_PREMIUM_MEMBER'
PMMembershipManager.sol:17:30: Error: Parse error: missing '=' at '('
PMMembershipManager.sol:18:6: Error: Parse error: missing 'constant'
at 'TOKEN_DONT_EXIST'
PMMembershipManager.sol:18:22: Error: Parse error: missing '=' at '('
PMMembershipManager.sol:19:6: Error: Parse error: missing 'constant'
at 'CONTRACT_IS_PAUSED'
PMMembershipManager.sol:19:24: Error: Parse error: missing '=' at '('
PMMembershipManager.sol:47:37: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:51:35: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:58:37: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:70:42: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:79:31: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:82:43: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:91:37: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:96:42: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:105:37: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:109:35: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:116:37: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:127:42: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMMembershipManager.sol:145:35: Error: Parse error: mismatched input
'(' expecting {';', '='}
```

PMTeamManager.sol

```
PMTeamManager.sol:14:6: Error: Parse error: missing 'constant' at
'INSUFFICIENT_FUNDS'
PMTeamManager.sol:14:24: Error: Parse error: missing '=' at '('
PMTeamManager.sol:15:6: Error: Parse error: missing 'constant' at
'FAILED_TO_TRANSFER_BNBS'
PMTeamManager.sol:15:29: Error: Parse error: missing '=' at '('
PMTeamManager.sol:16:6: Error: Parse error: missing 'constant' at
'NOT_OWNER_OF_TEAM'
PMTeamManager.sol:16:23: Error: Parse error: missing '=' at '('
PMTeamManager.sol:17:6: Error: Parse error: missing 'constant' at
'TOKEN_DONT_EXIST'
PMTeamManager.sol:17:22: Error: Parse error: missing '=' at '('
PMTeamManager.sol:18:6: Error: Parse error: missing 'constant' at
'CONTRACT_IS_PAUSED'
PMTeamManager.sol:18:24: Error: Parse error: missing '=' at '('
PMTeamManager.sol:46:37: Error: Parse error: mismatched input '('
expecting {';', '='}
```

```
PMTeamManager.sol:53:37: Error: Parse error: mismatched input '('
expecting {';', '='}
PMTeamManager.sol:68:42: Error: Parse error: mismatched input '('
expecting {';', '='}
PMTeamManager.sol:79:36: Error: Parse error: mismatched input '('
expecting {';', '='}
PMTeamManager.sol:88:35: Error: Parse error: mismatched input '('
expecting {';', '='}
PMTeamManager.sol:114:35: Error: Parse error: mismatched input '('
expecting {';', '='}
```

CampaignFeeManager.sol

```
CampaignFeeManager.sol:2:1: Error: Compiler version 0.8.9 does not
satisfy the r semver requirement
CampaignFeeManager.sol:53:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
CampaignFeeManager.sol:55:9: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:55:29: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:55:50: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:55:71: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:104:9: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:105:9: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:106:9: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:107:9: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:126:9: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:127:9: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:128:9: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:129:9: Error: Variable name must be in
mixedCase
CampaignFeeManager.sol:150:5: Error: Function name must be in
mixedCase
CampaignFeeManager.sol:196:13: Error: Avoid to make time-based
decisions in your business logic
CampaignFeeManager.sol:207:9: Error: Variable name must be in
mixedCase
```

MembershipFeeManager.sol

```
MembershipFeeManager.sol:2:1: Error: Compiler version 0.8.9 does not
satisfy the r semver requirement
```

```
MembershipFeeManager.sol:54:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
MembershipFeeManager.sol:108:5: Error: Function name must be in
mixedCase
MembershipFeeManager.sol:157:13: Error: Avoid to make time-based
decisions in your business logic
MembershipFeeManager.sol:168:9: Error: Variable name must be in
mixedCase
```

PMRewardDistributor.sol

```
PMRewardDistributor.sol:13:6: Error: Parse error: missing 'constant'
at 'NOT_ENOUGH_BALANCE'
PMRewardDistributor.sol:13:24: Error: Parse error: missing '=' at '('
PMRewardDistributor.sol:14:6: Error: Parse error: missing 'constant'
at 'CONTRACT_IS_PAUSED'
PMRewardDistributor.sol:14:24: Error: Parse error: missing '=' at '('
PMRewardDistributor.sol:56:37: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMRewardDistributor.sol:68:37: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMRewardDistributor.sol:77:37: Error: Parse error: mismatched input
'(' expecting {';', '='}
PMRewardDistributor.sol:89:37: Error: Parse error: mismatched input
'(' expecting {';', '='}
```

StakingPool.sol

```
StakingPool.sol:13:6: Error: Parse error: missing 'constant' at
'POOL_NOT_STARTED'
StakingPool.sol:13:22: Error: Parse error: missing '=' at '('
StakingPool.sol:14:6: Error: Parse error: missing 'constant' at
'NOT_ENOUGH_REWARD'
StakingPool.sol:14:23: Error: Parse error: missing '=' at '('
StakingPool.sol:15:6: Error: Parse error: missing 'constant' at
'OWNER_ONLY'
StakingPool.sol:15:16: Error: Parse error: missing '=' at '('
StakingPool.sol:16:6: Error: Parse error: missing 'constant' at
'NOT_ALLOWED'
StakingPool.sol:16:17: Error: Parse error: missing '=' at '('
StakingPool.sol:17:6: Error: Parse error: missing 'constant' at
'NOT_AUTHERIZED'
StakingPool.sol:17:20: Error: Parse error: missing '=' at '('
StakingPool.sol:18:6: Error: Parse error: missing 'constant' at
'NOTHING_TO_UNSTAKE'
StakingPool.sol:18:24: Error: Parse error: missing '=' at '('
StakingPool.sol:19:6: Error: Parse error: missing 'constant' at
'ALREADY_UNSTAKED'
StakingPool.sol:19:22: Error: Parse error: missing '=' at '('
StakingPool.sol:20:6: Error: Parse error: missing 'constant' at
'INSUFFICIENT_FUNDS'
StakingPool.sol:20:24: Error: Parse error: missing '=' at '('
```

```
StakingPool.sol:21:6: Error: Parse error: missing 'constant' at
'FAILED_TO_TRANSFER_BNBS'
StakingPool.sol:21:29: Error: Parse error: missing '=' at '('
StakingPool.sol:22:6: Error: Parse error: missing 'constant' at
'FAILED_TO_TRANSFER_TOEKNS'
StakingPool.sol:22:31: Error: Parse error: missing '=' at '('
StakingPool.sol:23:6: Error: Parse error: missing 'constant' at
'FAILED_TO_TRANSFER_ORIGNAL_TOEKNS'
StakingPool.sol:23:39: Error: Parse error: missing '=' at '('
StakingPool.sol:24:6: Error: Parse error: missing 'constant' at
'FAILED_TO_TRANSFER_REWARD_TOEKNS'
StakingPool.sol:24:38: Error: Parse error: missing '=' at '('
StakingPool.sol:25:6: Error: Parse error: missing 'constant' at
'NO_CREATOR_CONTRACT_FOUND'
StakingPool.sol:25:31: Error: Parse error: missing '=' at '('
StakingPool.sol:26:6: Error: Parse error: missing 'constant' at
'NOT_A_VALID_STAKING_TYPE'
StakingPool.sol:26:30: Error: Parse error: missing '=' at '('
StakingPool.sol:94:35: Error: Parse error: mismatched input '('
expecting {';', '='}
StakingPool.sol:102:43: Error: Parse error: mismatched input '('
expecting {';', '='}
StakingPool.sol:110:36: Error: Parse error: mismatched input '('
expecting {';', '='}
StakingPool.sol:155:44: Error: Parse error: mismatched input '('
expecting {';', '='}
StakingPool.sol:173:33: Error: Parse error: mismatched input '('
expecting {';', '='}
StakingPool.sol:177:37: Error: Parse error: mismatched input '('
expecting {';', '='}
StakingPool.sol:182:35: Error: Parse error: mismatched input '('
expecting {';', '='}
StakingPool.sol:200:37: Error: Parse error: mismatched input '('
expecting {';', '='}
StakingPool.sol:206:42: Error: Parse error: mismatched input '('
expecting {';', '='}
StakingPool.sol:213:52: Error: Parse error: mismatched input '('
expecting {';', '='}
StakingPool.sol:218:51: Error: Parse error: mismatched input '('
expecting {';', '='}
StakingPool.sol:293:44: Error: Parse error: mismatched input '('
expecting {';', '='}
```

StakingPoolFactory.sol

```
StakingPoolFactory.sol:15:6: Error: Parse error: missing 'constant'
at 'NOT_PREMIUM_OR_TEAM'
StakingPoolFactory.sol:15:25: Error: Parse error: missing '=' at '('
StakingPoolFactory.sol:16:6: Error: Parse error: missing 'constant'
at 'NOT_OWNER_OF_TEAM'
StakingPoolFactory.sol:16:23: Error: Parse error: missing '=' at '('
StakingPoolFactory.sol:17:6: Error: Parse error: missing 'constant'
at 'START_TIME_SHOULD_BE_FUTURE'
StakingPoolFactory.sol:17:33: Error: Parse error: missing '=' at '('
StakingPoolFactory.sol:18:6: Error: Parse error: missing 'constant'
```

```
at 'PROFILE_IS_ALREADY_SET'
StakingPoolFactory.sol:18:28: Error: Parse error: missing '=' at '('
StakingPoolFactory.sol:19:6: Error: Parse error: missing 'constant'
at 'NOT_THE_CAMPAIGN_OWNER'
StakingPoolFactory.sol:19:28: Error: Parse error: missing '=' at '('
StakingPoolFactory.sol:20:6: Error: Parse error: missing 'constant'
at 'FAILED_TO_TRANSFER_TOKENS'
StakingPoolFactory.sol:20:31: Error: Parse error: missing '=' at '('
StakingPoolFactory.sol:21:6: Error: Parse error: missing 'constant'
at 'CONTRACT_IS_PAUSED'
StakingPoolFactory.sol:21:24: Error: Parse error: missing '=' at '('
StakingPoolFactory.sol:58:37: Error: Parse error: mismatched input
'(' expecting {';', '='}
StakingPoolFactory.sol:65:38: Error: Parse error: mismatched input
'(' expecting {';', '='}
StakingPoolFactory.sol:71:40: Error: Parse error: mismatched input
'(' expecting {';', '='}
StakingPoolFactory.sol:78:37: Error: Parse error: mismatched input
'(' expecting {';', '='}
StakingPoolFactory.sol:82:46: Error: Parse error: mismatched input
'(' expecting {';', '='}
StakingPoolFactory.sol:105:44: Error: Parse error: mismatched input
'(' expecting {';', '='}
StakingPoolFactory.sol:110:42: Error: Parse error: mismatched input
'(' expecting {';', '='}
```

SwapETHForTokens.sol

```
SwapETHForTokens.sol:2:1: Error: Compiler version ^0.8.14 does not
satisfy the r semver requirement
SwapETHForTokens.sol:32:13: Error: Avoid to make time-based decisions
in your business logic
```

PriceFeed.sol

```
PriceFeed.sol:2:1: Error: Compiler version ^0.8.14 does not satisfy
the r semver requirement
PriceFeed.sol:30:9: Error: Variable 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.

Email: audit@EtherAuthority.io