Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Starbank Protocol
Platform: Astar Network
Language: Solidity
Date: April 5th, 2022

Table of contents

Email: audit@EtherAuthority.io

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

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the Starbank team to perform the Security audit of the Starbank 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 April 5th, 2022.

The purpose of this audit was to address the following:

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

Project Background

The Starbank Contracts have functions like adding a new pool and updating LP, batchSwap, swap, flashLoan, deposit, withdraw, mint, burn, etc. The Starbank contracts also inherits ERC20Burnable, Math, IERC20, SafeERC20, ReentrancyGuard, SafeMath standard smart contracts from the openzepelin library.

Audit scope

Name	**Code Review and Security Analysis Report for Starbank Protocol Smart Contracts**
Platform	**Astar Network / Solidity**
File 1	Authorizer.sol
File 1 MD5 Hash	89FBC1ACC09B9EE2AA0337A7CB94D469
File 2	InvestmentPoolFactory.sol
File 2 MD5 Hash	5735312F2BC1FDD5C0B57695C92853E9
File 3	MetaStablePoolFactory.sol
File 3 MD5 Hash	2E3C6BDF059C3D6A8A86F2E6BE580ABF
File 4	Multicall2.sol
File 4 MD5 Hash	A5539355CC6AB06E648A358E6A3CF27F

File 5	NoProtocolFeeLiquidityBootstrappingPoolFactory.sol
File 5 MD5 Hash	177A8B4CCA7EEFDB9CC5ED5A52537A01
File 6	ProtocolFeesCollector.sol
File 6 MD5 Hash	EC904D61244952C1A004444B0E48AD66
File 7	StablePhantomPoolFactory.sol
File 7 MD5 Hash	F9D53F12E31CCDF316C63F128EB06478
File 8	StablePoolFactory.sol
File 8 MD5 Hash	FD84480294719501B87B5AD179E3A69D
File 9	Vault.sol
File 9 MD5 Hash	668D62234B36AFCB7B3811D76068E137
File 10	WeightedPool.sol
File 10 MD5 Hash	65F45CC467D4C918C7C09432B214330C
File 11	WeightedPool2TokensFactory.sol
File 11 MD5 Hash	096ADD93DD8EF6366B4A9474BC09731A
File 12	MasterChef.sol
File 12 MD5 Hash	EC393C79AC1B07DD836B04ABDF67D728
File 13	SBXToken.sol
File 13 MD5 Hash	465E7DC40093C1900142DFEEF27E6724
Audit Date	April 5th,2022

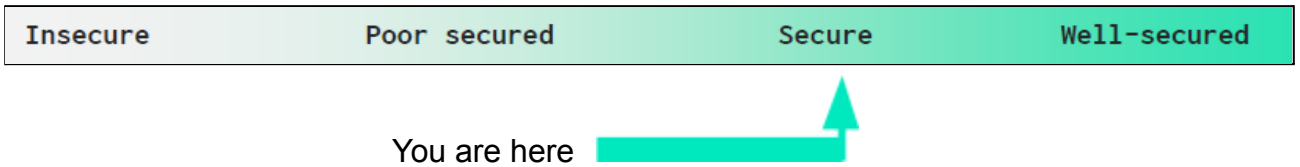
Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 Authorizer.sol • The Authorizer can access functions like: renouncePermissions, revokePermissions, grantPermissions, cancel, execute, schedule, etc.	**YES, This is valid.**
File 2 InvestmentPoolFactory.sol • The InvestmentPoolFactory can create a new pool.	**YES, This is valid.**
File 3 MetaStablePoolFactory.sol • The MetaStablePoolFactory can create a new pool.	**YES, This is valid.**
File 4 Multicall2.sol • The Multicall can access functions like: aggregate, blockAndAggregate, tryAggregate, etc.	**YES, This is valid.**
File 5 NoProtocolFeeLiquidityBootstrappingPoolFactory.sol • The NoProtocolFeeLiquidityBootstrappingPoolFactory access functions like: disable, create, _canPerform, etc.	**YES, This is valid.**
File 6 ProtocolFeesCollector.sol • Maximum Protocol Swap Fee: 50% • Maximum Protocol Flash Loan Fee: 1%	**YES, This is valid. Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**
File 7 StablePhantomPoolFactory.sol • The StablePhantomPoolFactory can access functions like: create a Stable Phantom Pool.	**YES, This is valid.**

File 8 StablePoolFactory.sol • The StablePoolFactory can access functions like: create a Stable Pool.	**YES, This is valid.**
File 9 Vault.sol • The Vault contract can access functions like: setPaused, WETH.	**YES, This is valid.**
File 10 WeightedPool.sol • The WeightedPool can access functions like: getInvariant, _onSwapGivenIn, _onJoinPool,etc.	**YES, This is valid.**
File 11 WeightedPool2TokensFactory.sol • The WeightedPool2TokensFactory can access functions like:create, etc.	**YES, This is valid.**
File 12 MasterChef.sol • LP Mining Reward: 6% • Community Growth: 4% • Dev: 18% • Treasury: 6% • Reserve for Potential Investors: 11.40% • Distribution percentages: 100% • Bonus Multiplier: 1 • Maximum Allocate Point: 4000	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**
File 13 SBXToken.sol • Name: SBXToken • Symbol: SBX • Maximum Supply: 250 million SBX	**YES, This is valid.**

Audit Summary

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



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

We found 0 critical, 0 high, 0 medium and 4 low and some very low level issues.

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

Technical Quick Stats

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

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 13 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 Starbank 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 Starbank Protocol.

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

Code parts are **not** well commented on smart contracts.

Documentation

We were given a Starbank Protocol smart contract code in the form of a blockscout astar Web Link and Github weblink. The hash of that code is mentioned above in the table.

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

Use of Dependencies

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

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

AS-IS overview

Authorizer.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	permissionId	write	Passed	No Issue
3	hasPermission	read	Passed	No Issue
4	canPerform	read	Passed	No Issue
5	setDelay	external	Passed	No Issue
6	scheduleDelayChange	external	Passed	No Issue
7	schedule	external	Passed	No Issue
8	execute	external	Passed	No Issue
9	cancel	external	Passed	No Issue
10	grantPermissions	external	Passed	No Issue
11	revokePermissions	external	Passed	No Issue
12	renouncePermissions	external	Passed	No Issue
13	_grantPermission	external	Passed	No Issue
14	_revokePermission	external	Passed	No Issue
15	_schedule	write	Passed	No Issue
16	_authenticate	internal	Passed	No Issue
17	_executeActionId	internal	Passed	No Issue
18	_decodeSelector	internal	Passed	No Issue

InvestmentPoolFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	create	external	Passed	No Issue
3	getCreationCodeContracts	read	Passed	No Issue
4	getCreationCode	read	Passed	No Issue
5	_getCreationCodeWithArgs	read	Passed	No Issue
6	_create	internal	Passed	No Issue
7	_memcpy	write	Passed	No Issue
8	getVault	read	Passed	No Issue
9	isPoolFromFactory	external	Passed	No Issue
10	_create	internal	Passed	No Issue
11	getPauseConfiguration	read	Passed	No Issue

MetaStablePoolFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	create	external	Passed	No Issue
3	getCreationCodeContracts	read	Passed	No Issue
4	getCreationCode	read	Passed	No Issue
5	_getCreationCodeWithArgs	read	Passed	No Issue
6	_create	internal	Passed	No Issue
7	_memcpy	write	Passed	No Issue
8	getVault	read	Passed	No Issue
9	isPoolFromFactory	external	Passed	No Issue
10	_create	internal	Passed	No Issue
11	getPauseConfiguration	read	Passed	No Issue

Multicall2.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	aggregate	read	Passed	No Issue
3	blockAndAggregate	write	Passed	No Issue
4	getBlockHash	read	Passed	No Issue
5	getBlockNumber	read	Passed	No Issue
6	getCurrentBlockCoinbase	read	Passed	No Issue
7	getCurrentBlockDifficulty	read	Passed	No Issue
8	getCurrentBlockGasLimit	read	Passed	No Issue
9	getCurrentBlockTimestamp	read	Passed	No Issue
10	getEthBalance	read	Passed	No Issue
11	getLastBlockHash	read	Passed	No Issue
12	tryAggregate	write	Passed	No Issue
13	tryBlockAndAggregate	write	Passed	No Issue

NoProtocolFeeLiquidityBootstrappingPoolFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getCreationCodeContracts	read	Passed	No Issue
3	getCreationCode	read	Passed	No Issue
4	_getCreationCodeWithArgs	read	Passed	No Issue
5	_create	internal	Passed	No Issue
6	_memcpy	write	Passed	No Issue
7	getVault	read	Passed	No Issue
8	isPoolFromFactory	external	Passed	No Issue

9	_create	internal	Passed	No Issue
10	getPauseConfiguration	read	Passed	No Issue
11	isDisabled	read	Passed	No Issue
12	disable	external	access by authenticate	No Issue
13	create	external	Passed	No Issue
14	_canPerform	internal	Passed	No Issue
15	authenticate	modifier	Passed	No Issue
16	_authenticateCaller	internal	Passed	No Issue
17	getActionId	read	Passed	No Issue
18	_canPerform	internal	Passed	No Issue

ProtocolFeesCollector.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	authenticate	modifier	Passed	No Issue
3	_authenticateCaller	internal	Passed	No Issue
4	getActionId	read	Passed	No Issue
5	_canPerform	internal	Passed	No Issue
6	nonReentrant	modifier	Passed	No Issue
7	_enterNonReentrant	write	Passed	No Issue
8	_exitNonReentrant	write	Passed	No Issue
9	withdrawCollectedFees	external	Function input parameters lack of check	Refer Audit Findings
10	setSwapFeePercentage	external	access by authenticate	No Issue
11	setFlashLoanFeePercentage	external	access by authenticate	No Issue
12	getSwapFeePercentage	external	Passed	No Issue
13	getFlashLoanFeePercentage	external	Passed	No Issue
14	getCollectedFeeAmounts	external	Passed	No Issue
15	getAuthorizer	external	Passed	No Issue
16	_canPerform	internal	Passed	No Issue
17	_getAuthorizer	internal	Passed	No Issue

StablePhantomPoolFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getCreationCodeContracts	read	Passed	No Issue
3	getCreationCode	read	Passed	No Issue

4	_getCreationCodeWithArgs	read	Passed	No Issue
5	_create	internal	Passed	No Issue
6	_memcpy	write	Passed	No Issue
7	getVault	read	Passed	No Issue
8	isPoolFromFactory	external	Passed	No Issue
9	_create	internal	Passed	No Issue
10	getPauseConfiguration	read	Passed	No Issue
11	create	external	Passed	No Issue

StablePoolFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getCreationCodeContracts	read	Passed	No Issue
3	getCreationCode	read	Passed	No Issue
4	_getCreationCodeWithArgs	read	Passed	No Issue
5	_create	internal	Passed	No Issue
6	_memcpy	write	Passed	No Issue
7	getVault	read	Passed	No Issue
8	isPoolFromFactory	external	Passed	No Issue
9	_create	internal	Passed	No Issue
10	getPauseConfiguration	read	Passed	No Issue
11	create	external	Passed	No Issue

Vault.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	setPaused	external	access by authenticate	No Issue
3	WETH	external	Passed	No Issue
4	swap	external	access by authenticate for	No Issue
5	batchSwap	external	access by authenticate for	No Issue
6	_tokenGiven	write	Passed	No Issue
7	_tokenCalculated	write	Passed	No Issue
8	_getAmounts	write	Passed	No Issue
9	_swapWithPools	write	Passed	No Issue
10	_swapWithPool	write	Passed	No Issue
11	_processTwoTokenPoolSwapRequest	write	Passed	No Issue
12	_processMinimalSwapInfoPoolSwapRequest	write	Passed	No Issue

13	_callMinimalSwapInfoPoolOnSwapHook	internal	Passed	No Issue
14	_processGeneralPoolSwapRequest	write	Passed	No Issue
15	queryBatchSwap	external	Passed	No Issue
16	flashLoan	external	Passed	No Issue
17	setAuthorizer	external	access by authenticate for	No Issue
18	_setAuthorizer	write	Passed	No Issue
19	getAuthorizer	external	Passed	No Issue
20	setRelayerApproval	external	access by authenticate for	No Issue
21	hasApprovedRelayer	external	Passed	No Issue
22	_authenticateFor	internal	Passed	No Issue
23	_hasApprovedRelayer	internal	Passed	No Issue
24	_canPerform	internal	Passed	No Issue
25	_typeHash	internal	Passed	No Issue

WeightedPool.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_normalizedWeight	internal	Passed	No Issue
3	_normalizedWeights	internal	Passed	No Issue
4	getLastInvariant	external	Passed	No Issue
5	getInvariant	read	Passed	No Issue
6	getNormalizedWeights	external	Passed	No Issue
7	_onSwapGivenIn	internal	Passed	No Issue
8	_onSwapGivenOut	internal	Passed	No Issue
9	_onInitializePool	internal	Passed	No Issue
10	_onJoinPool	internal	Passed	No Issue
11	_doJoin	read	Passed	No Issue
12	_joinExactTokensInForBPTOut	read	Passed	No Issue
13	_joinTokenInForExactBPTOut	read	Passed	No Issue
14	_onExitPool	internal	Passed	No Issue
15	_doExit	read	Passed	No Issue
16	_exitExactBPTInForTokenOut	read	Passed	No Issue
17	_exitExactBPTInForTokensOut	write	Passed	No Issue
18	_exitBPTInForExactTokensOut	read	Passed	No Issue
19	_getDueProtocolFeeAmounts	read	Passed	No Issue

20	_invariantAfterJoin	read	Passed	No Issue
21	_invariantAfterExit	read	Passed	No Issue
22	_mutateAmounts	read	Passed	No Issue
23	getRate	read	Passed	No Issue
24	onSwap	external	Passed	No Issue
25	_onSwapGivenIn	internal	Passed	No Issue
26	_register	internal	Passed	No Issue
27	_calculateInvariant	internal	Passed	No Issue
28	_calcOutGivenIn	internal	Passed	No Issue
29	_calcInGivenOut	internal	Passed	No Issue
30	_calcBptOutGivenExactTokensIn	internal	Passed	No Issue
31	_calcTokenInGivenExactBptOut	internal	Passed	No Issue
32	_calcBptInGivenExactTokensOut	internal	Passed	No Issue
33	_calcTokenOutGivenExactBptIn	internal	Passed	No Issue
34	_calcTokensOutGivenExactBptIn	internal	Passed	No Issue
35	_calcDueTokenProtocolSwapFeeAmount	internal	Passed	No Issue

WeightedPool2TokensFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	create	external	Passed	No Issue
3	getVault	read	Passed	No Issue
4	isPoolFromFactory	external	Passed	No Issue
5	_register	internal	Passed	No Issue
6	getPauseConfiguration	read	Passed	No Issue

MasterChef.sol

Functions

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

9	poolLength	external	Passed	No Issue
10	addPool	write	LP Token and reward token	Refer Audit Findings
11	setPool	write	access only Owner	No Issue
12	pendingReward	external	Passed	No Issue
13	getMultiplier	read	Passed	No Issue
14	massUpdatePools	write	Passed	No Issue
15	updatePools	external	Passed	No Issue
16	updatePool	write	Passed	No Issue
17	deposit	write	Passed	No Issue
18	withdraw	write	Passed	No Issue
19	emergencyWithdraw	write	Emergency Withdrawal	Refer Audit Findings
20	harvest	write	Passed	No Issue
21	harvestAll	external	Passed	No Issue
22	harvestSome	external	Passed	No Issue
23	safeRewardTransfer	internal	Passed	No Issue
24	setRewardPerSecond	external	access only Owner	No Issue
25	dev	write	access only Owner	No Issue
26	treasury	write	access only Owner	No Issue
27	reserve1	write	access only Owner	No Issue
28	reserve2	write	access only Owner	No Issue
29	reserve3	write	access only Owner	No Issue
30	communtyGrowth	write	access only Owner	No Issue
31	setStartTime	external	access only Owner	No Issue

SBXToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	write	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	name	read	Passed	No Issue
8	symbol	read	Passed	No Issue
9	decimals	read	Passed	No Issue
10	totalSupply	read	Passed	No Issue
11	balanceOf	read	Passed	No Issue
12	transfer	write	Passed	No Issue
13	allowance	read	Passed	No Issue
14	approve	write	Passed	No Issue
15	transferFrom	write	Passed	No Issue
16	increaseAllowance	write	Passed	No Issue
17	decreaseAllowance	write	Passed	No Issue
18	_transfer	internal	Passed	No Issue
19	_mint	internal	Passed	No Issue
20	_burn	internal	Passed	No Issue
21	_approve	internal	Passed	No Issue
22	_spendAllowance	internal	Passed	No Issue
23	_beforeTokenTransfer	internal	Passed	No Issue
24	_afterTokenTransfer	internal	Passed	No Issue
25	burnFrom	write	Passed	No Issue
26	burn	write	Passed	No Issue
27	mint	write	access only Owner	No Issue

Severity Definitions

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

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Function input parameters lack of check:
Variable validation is not performed in the functions below:

ProtocolFeesCollector.sol

- withdrawCollectedFees = recipient

.

Resolution: We advise using validation like address type variables should not be address(0).

(2) Emergency Withdrawal: **MasterChef.sol**
There is no validation for the user for emergency withdrawal. Users who have not deposited can also execute the emergencyWithdraw function.

Resolution: We suggest checking whether the pool id belongs to the caller before executing the transfer.

(3) Using experimental ABIEncoderV2: **InvestmentPoolFactory.sol, MultiCall.sol**
Because the ABIEncoderV2 is experimental, it would be risky to release the project using it. Moreover, the recent findings show that it is likely that other important bugs are yet to be found.
https://blog.ethereum.org/2019/03/26/solidity-optimizer-and-abiencoderv2-bug/

Resolution: We suggest avoiding using this if logically possible.

(4) LP Token and reward token: **MasterChef.sol**
In the addPool function , there is a condition that rewardToken is not equal to IpToken. But if we execute the addPool function without initializing, the above condition will always be true and so the LP token can be the same as the reward token.

Resolution: We suggest checking whether the Initialize function has been executed or not for all other functions.

Very Low / Informational / Best practices:

(1) rewardToken should be made immutable: **MasterChef.sol**
Variables that are defined within the constructor but further remain unchanged should be marked as immutable to save gas and to ease the reviewing process of third-parties.

Resolution: Consider marking this variable as immutable.

(2) SPDX license identifier is Missing: **Multicall2.sol, InvestmentPoolFactory.sol**
The SPDX license identifier is missing for the mentioned files.

Resolution: We suggest adding SPDX license identifier.

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:

- init: The MasterChef owner can initialize the reward address.
- addPool: The MasterChef owner can add a new lp to the pool.
- setPool: The MasterChef owner can update the given pool's reward allocation point.
- setRewardPerSecond: The MasterChef owner can set rewards per second.
- dev: The MasterChef owner can update the dev address by the previous dev.
- treasury: The MasterChef owner can update the treasury address by the owner.
- reserve1: The MasterChef owner can update the reserve1 address by the owner.
- reserve2:The MasterChef owner can update the reserve2 address by the owner.
- reserve3: The MasterChef owner can update the reserve3 address by the owner.
- communtyGrowth: The MasterChef owner can update communtyGrowth address by the owner.
- setStartTime: The MasterChef owner can set start time.
- mint: The SBXToken owner can mint an amount from the address.

Conclusion

We were given a contract code in the form of files. And we have used all possible tests based on given objects as files. We have not observed any major issues in the smart contracts. So, **it's good to go to production**.

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

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

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

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

Email: audit@EtherAuthority.io

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

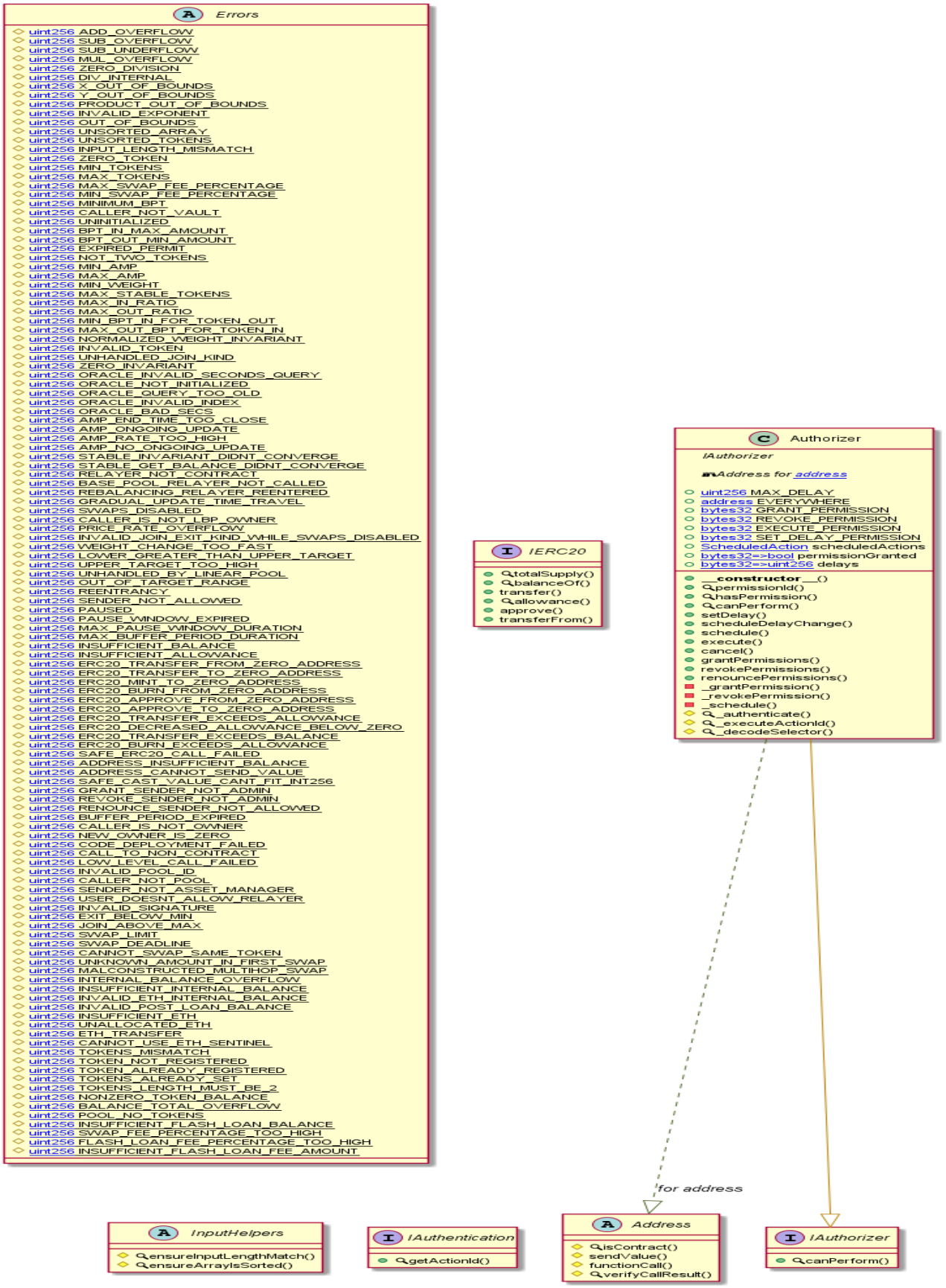
Technical Disclaimer

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

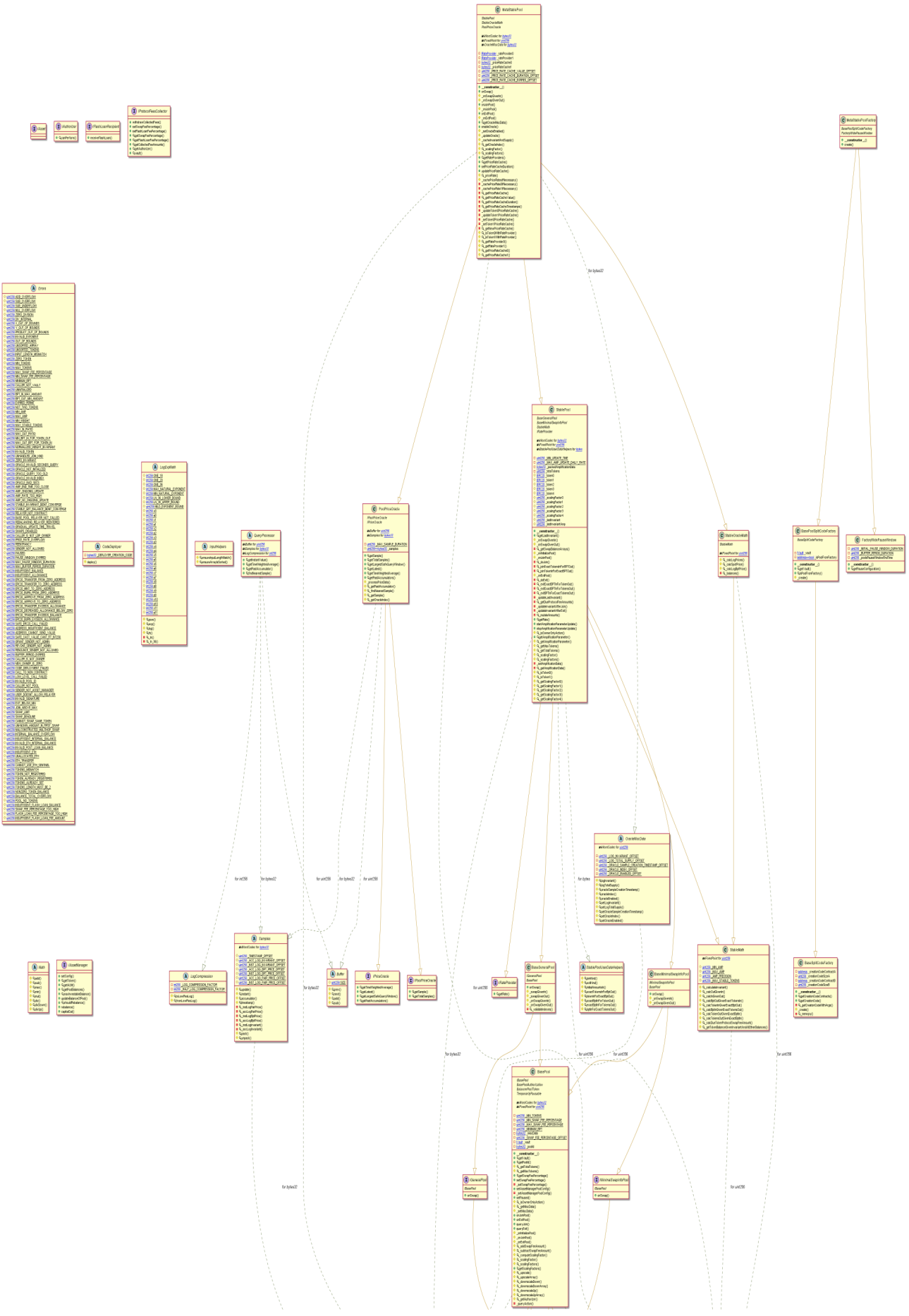
Appendix

Code Flow Diagram - Starbank Protocol

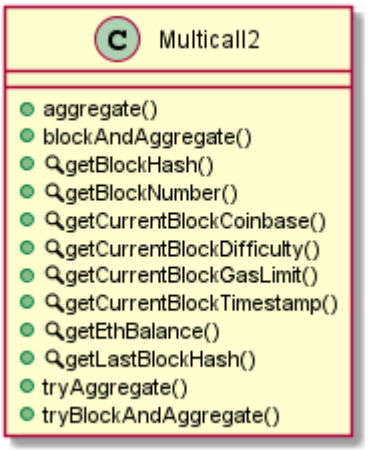
Authorizer Diagram



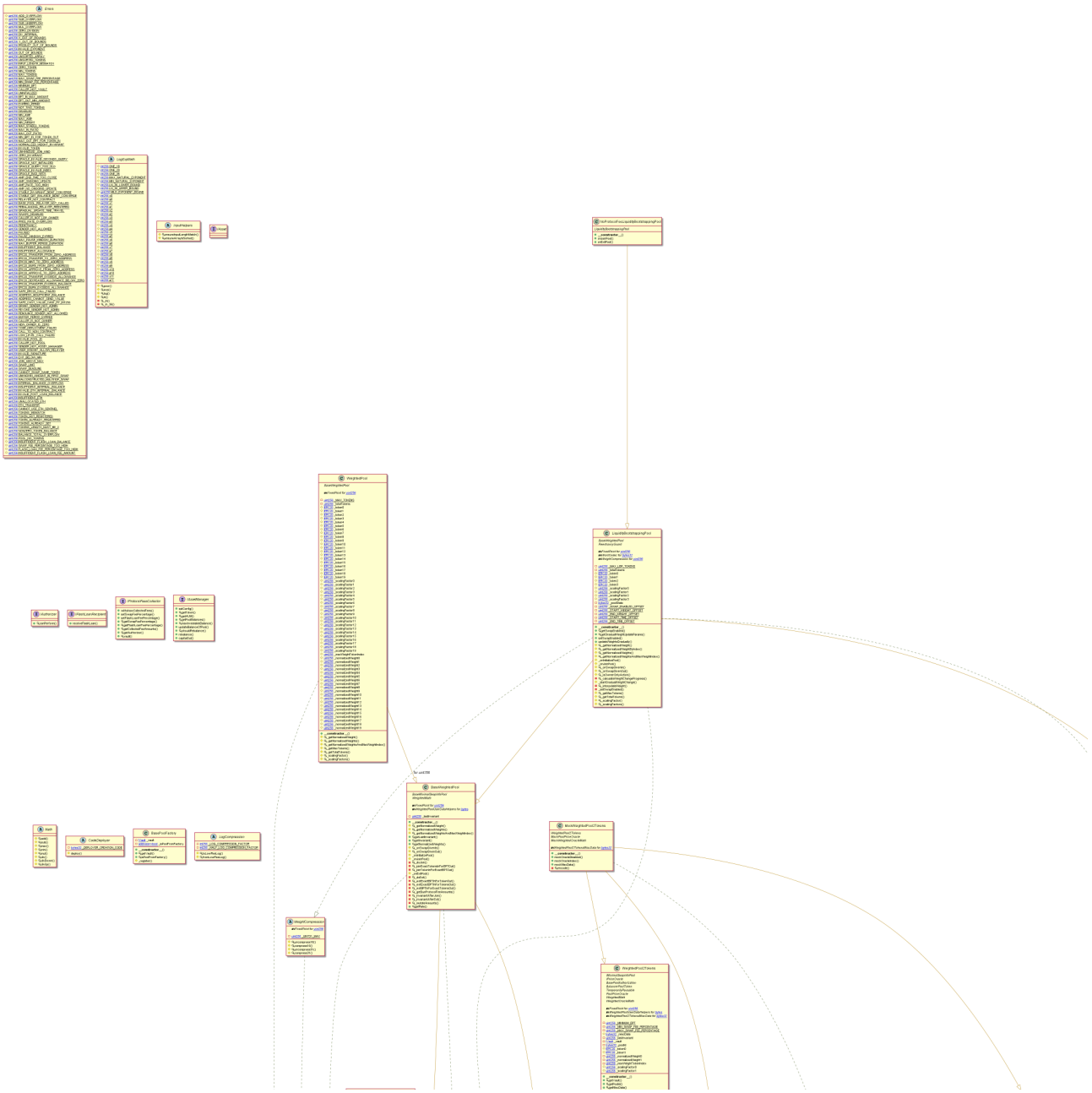
MetaStablePoolFactory Diagram



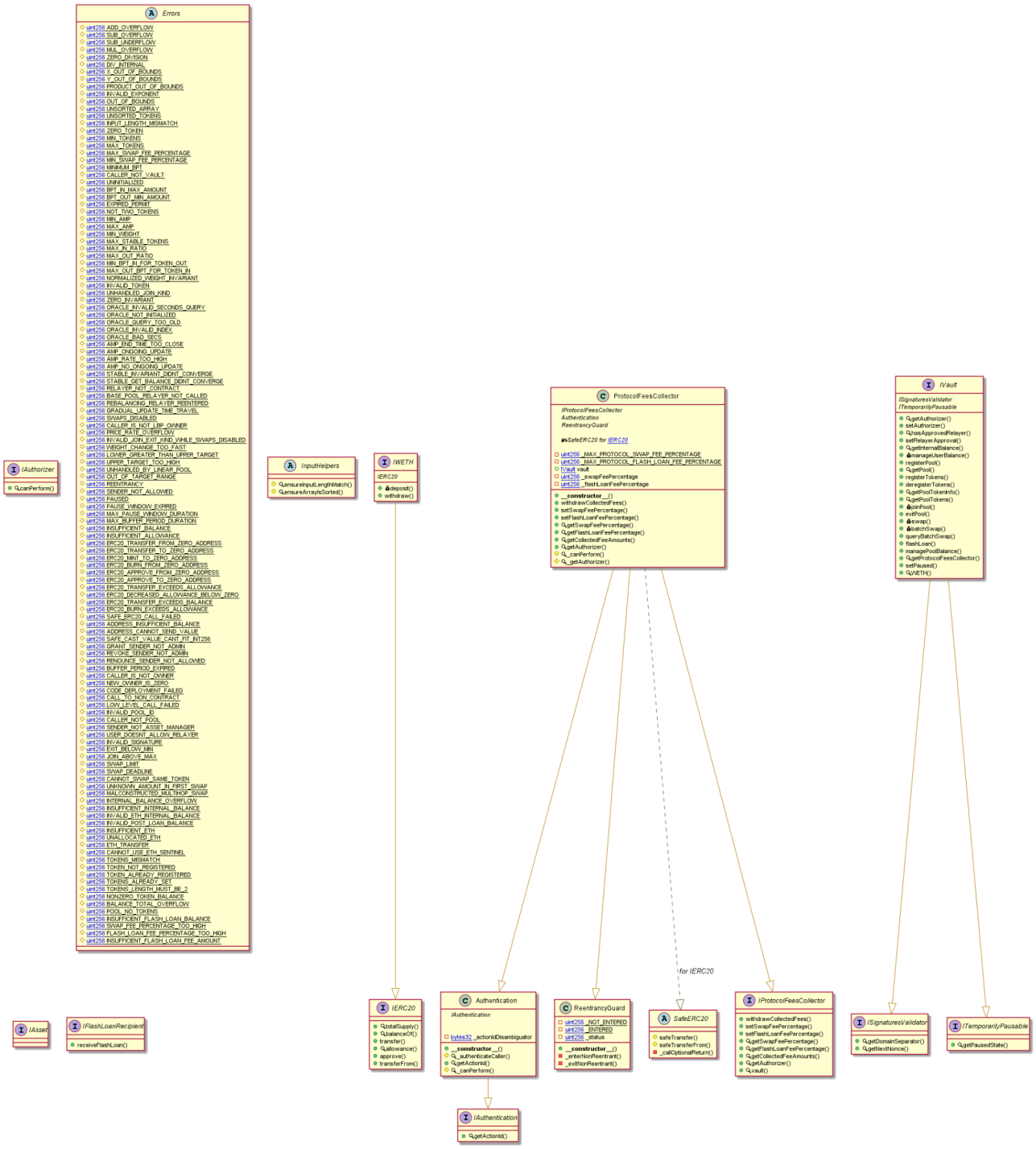
Multicall2 Diagram



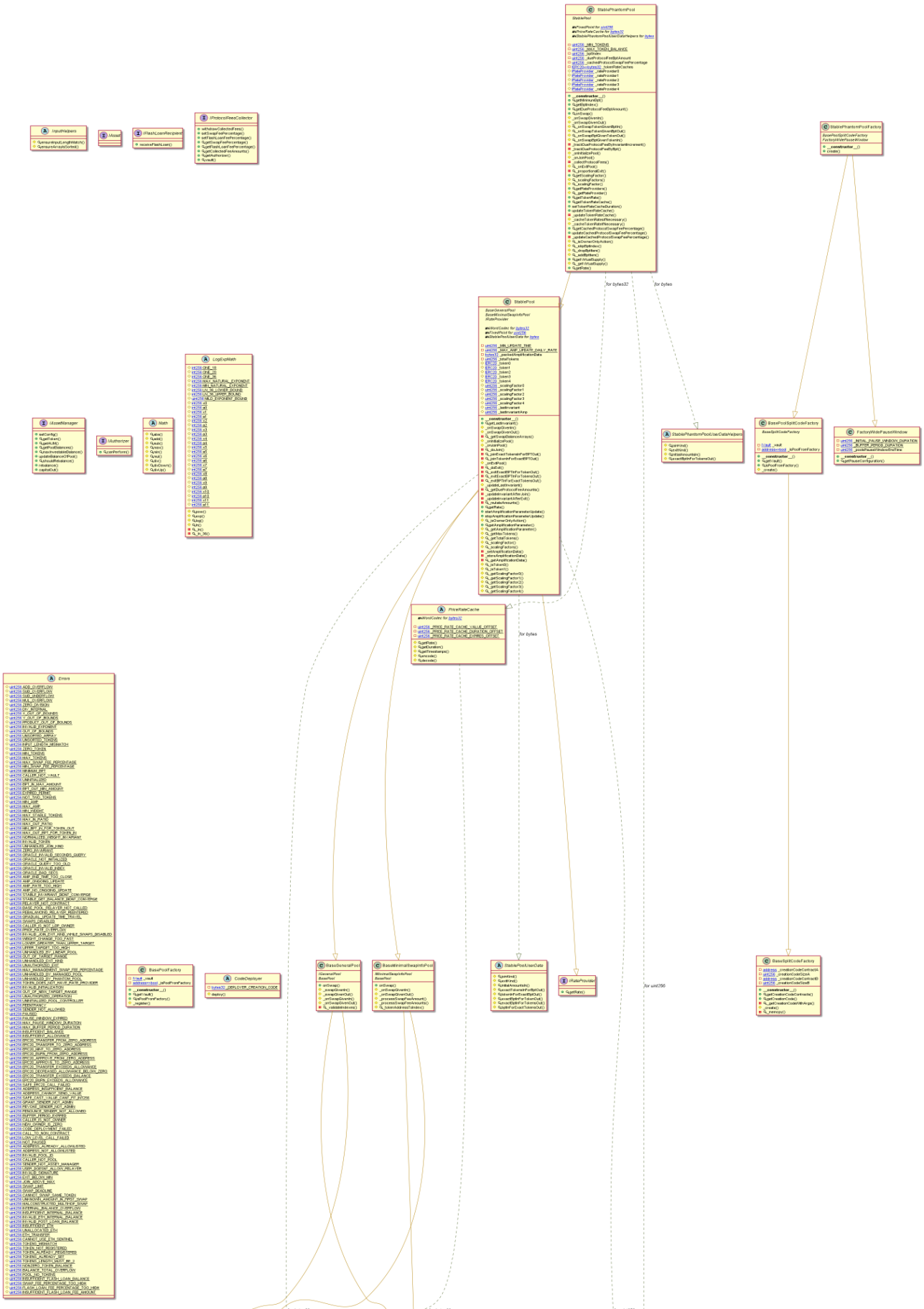
NoProtocolFeeLiquidityBootstrappingPoolFactory Diagram



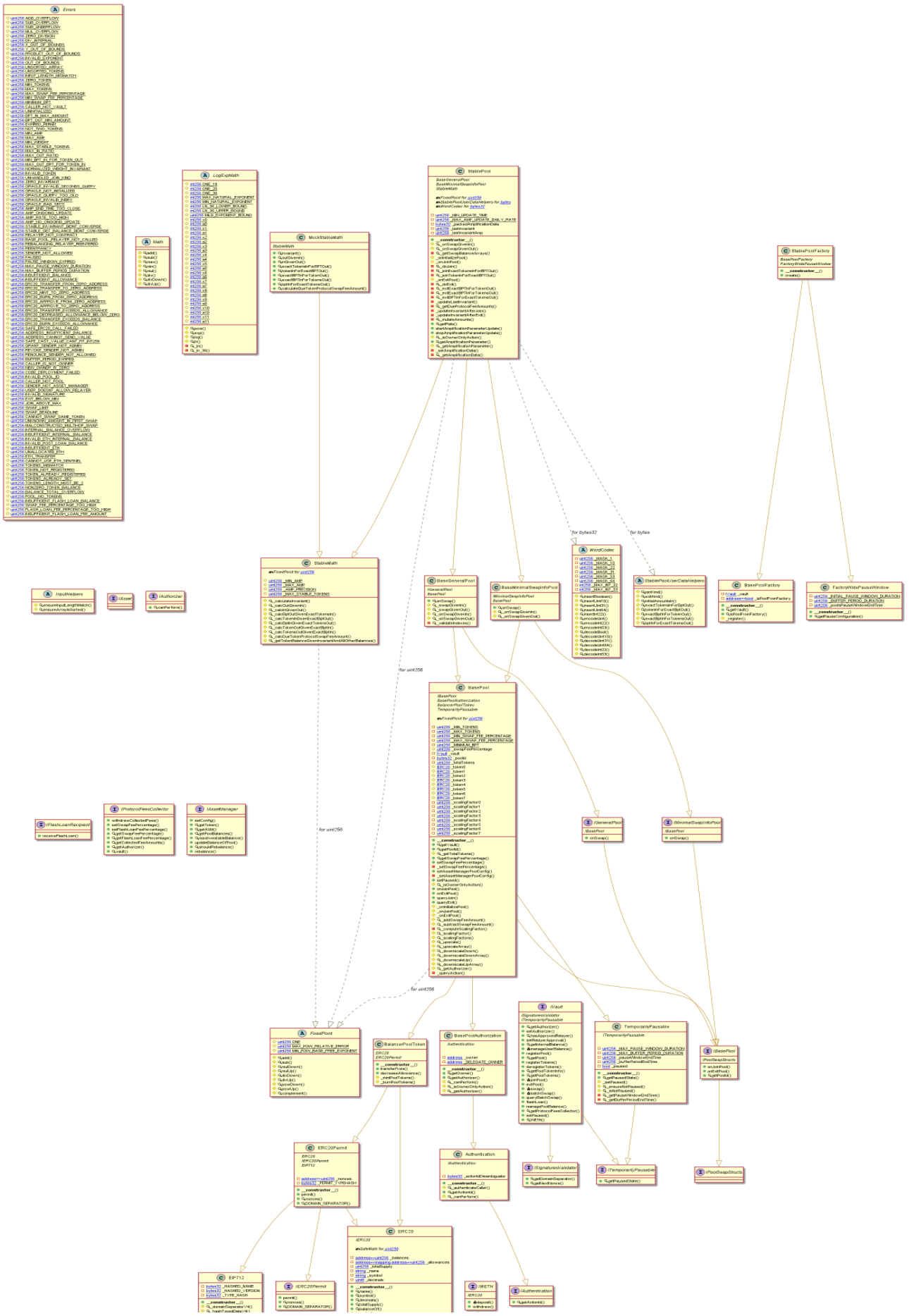
ProtocolFeesCollector Diagram



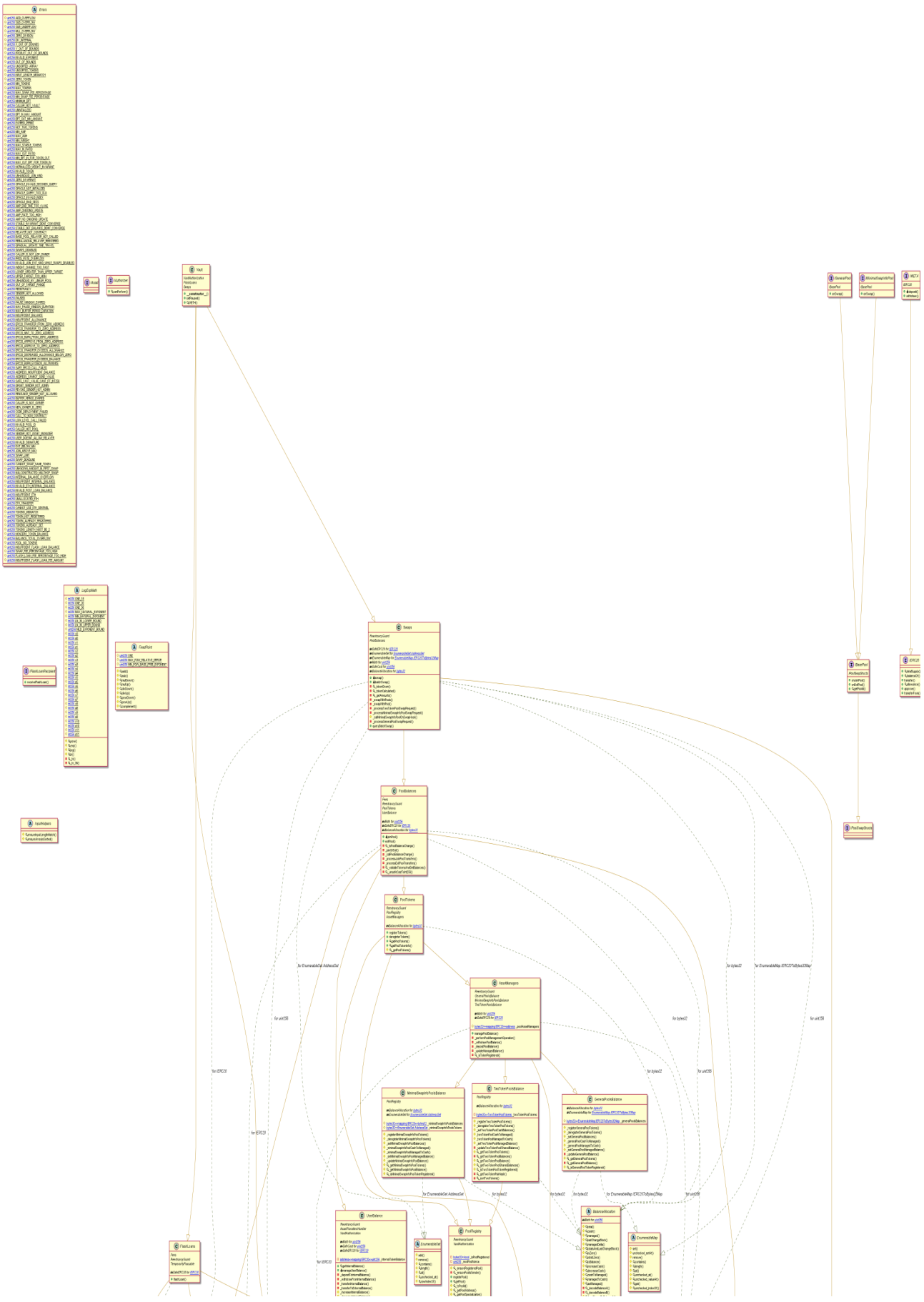
StablePhantomPoolFactory Diagram



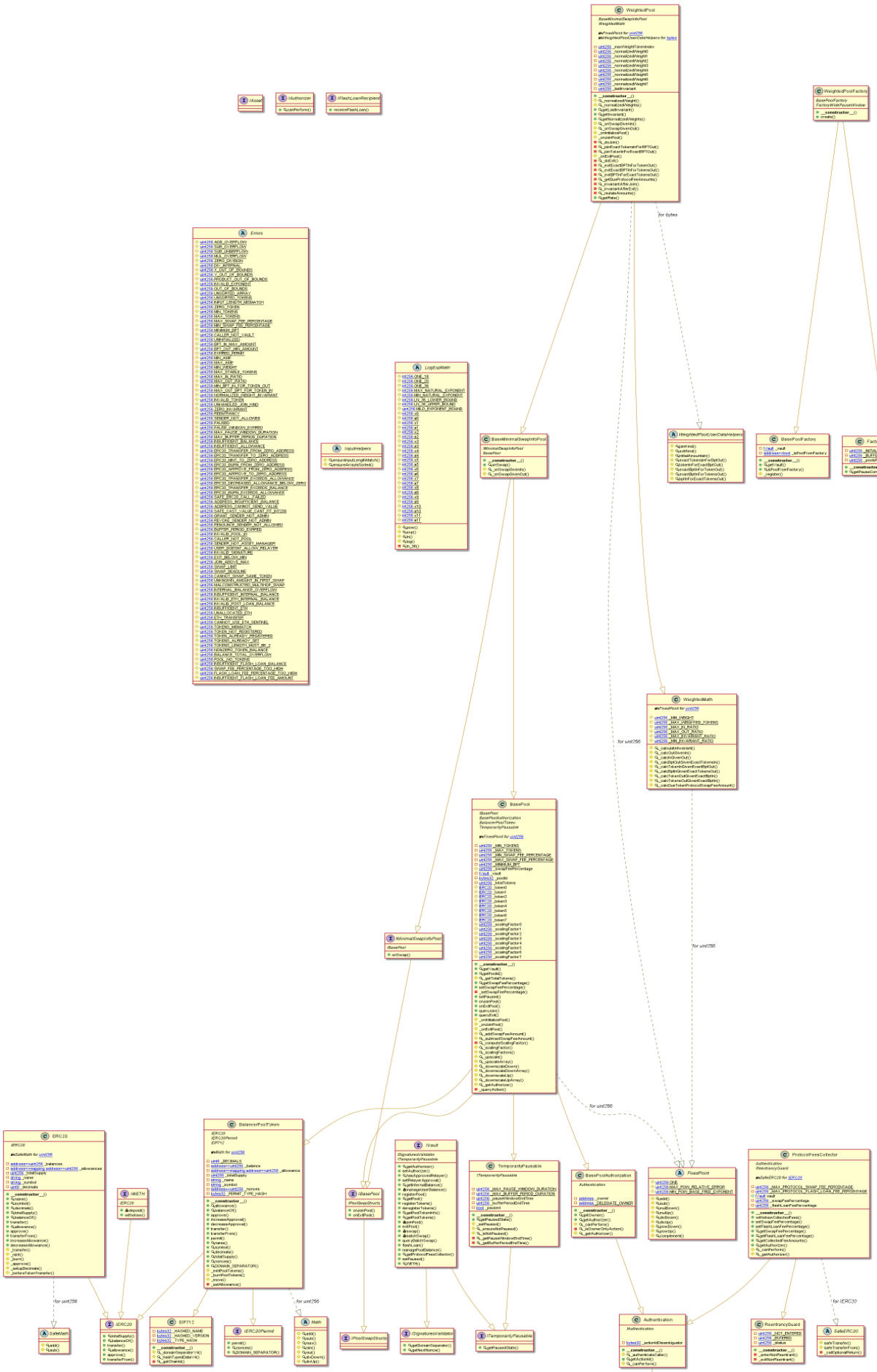
StablePoolFactory Diagram



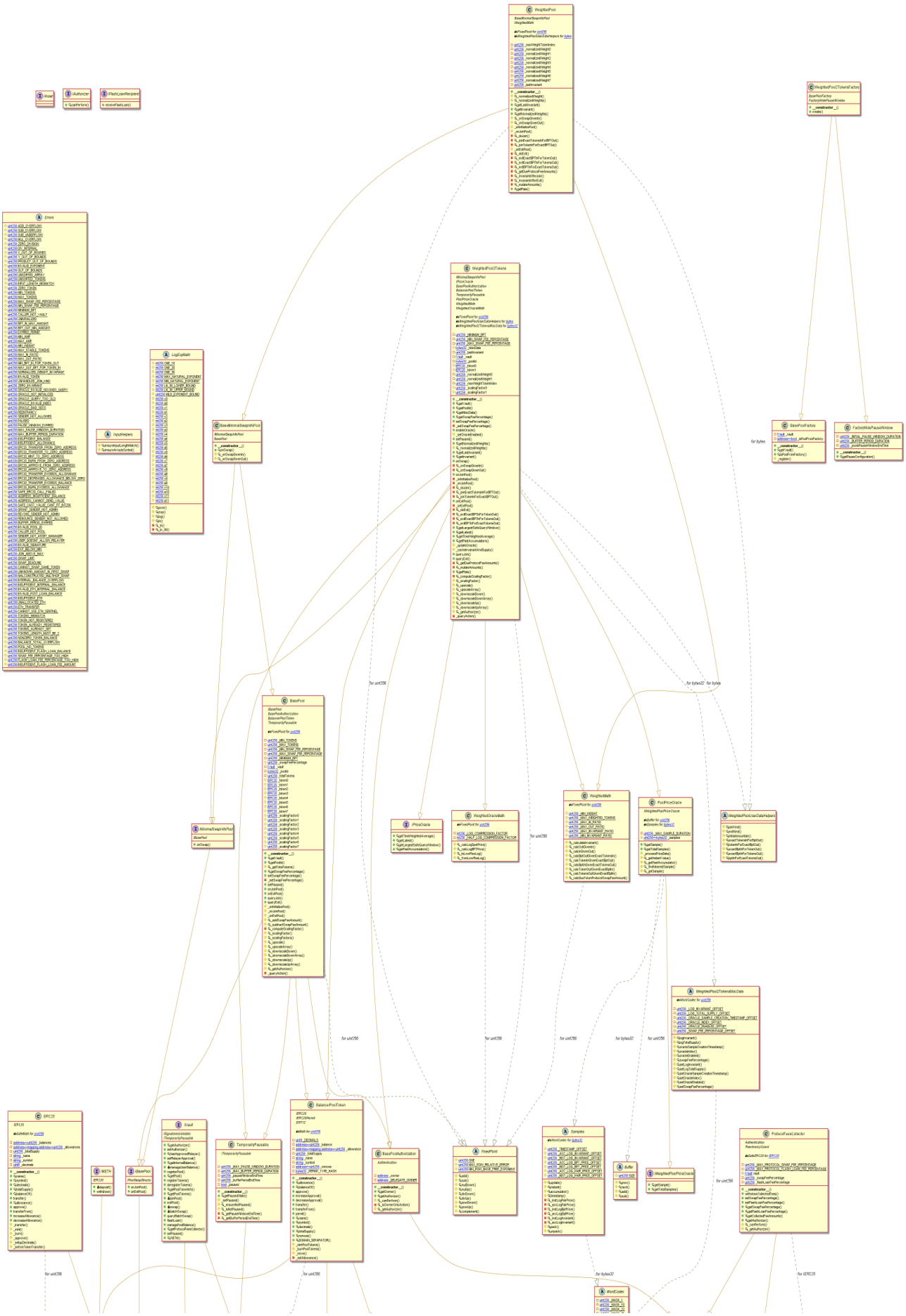
Vault Diagram



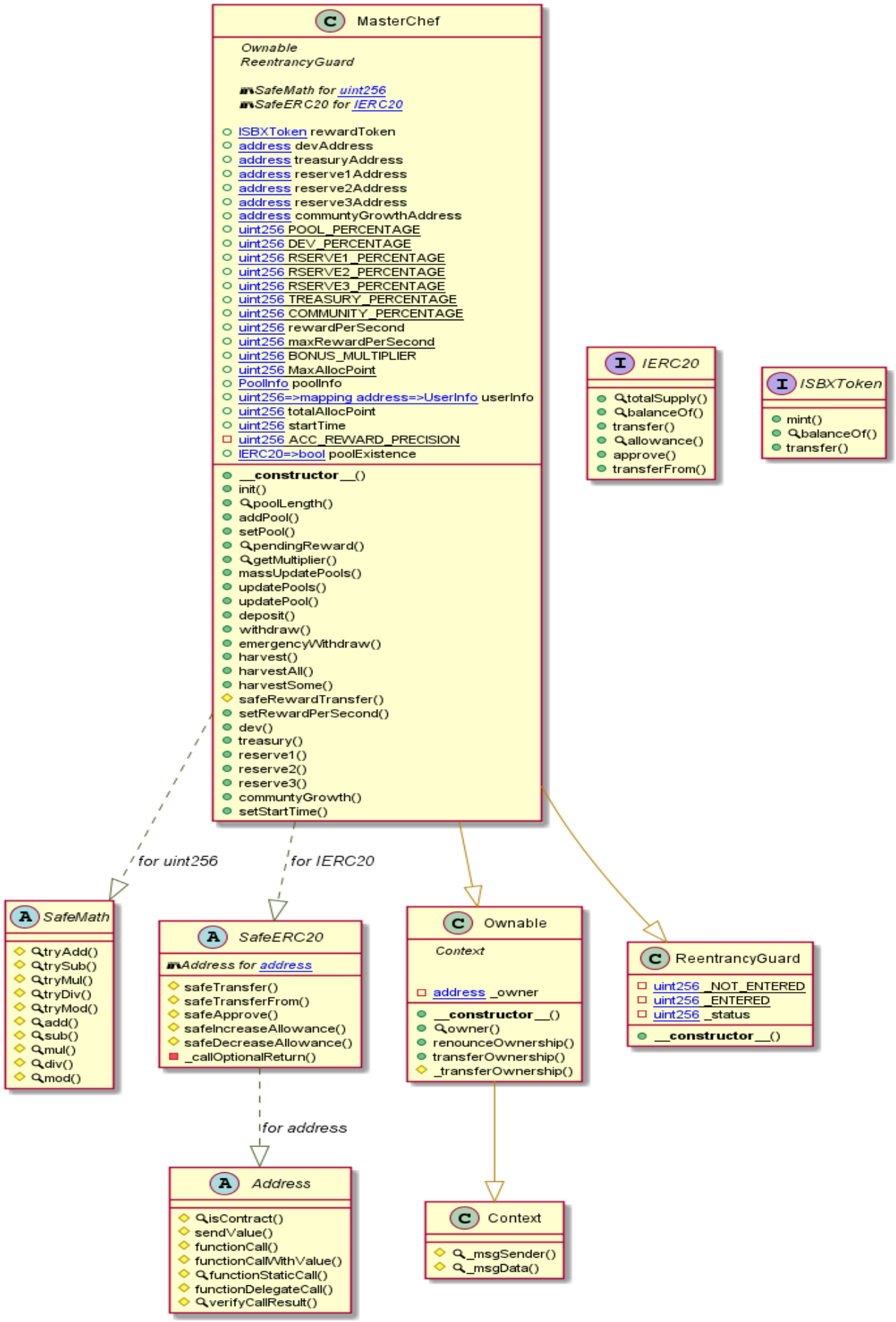
WeightedPool Diagram



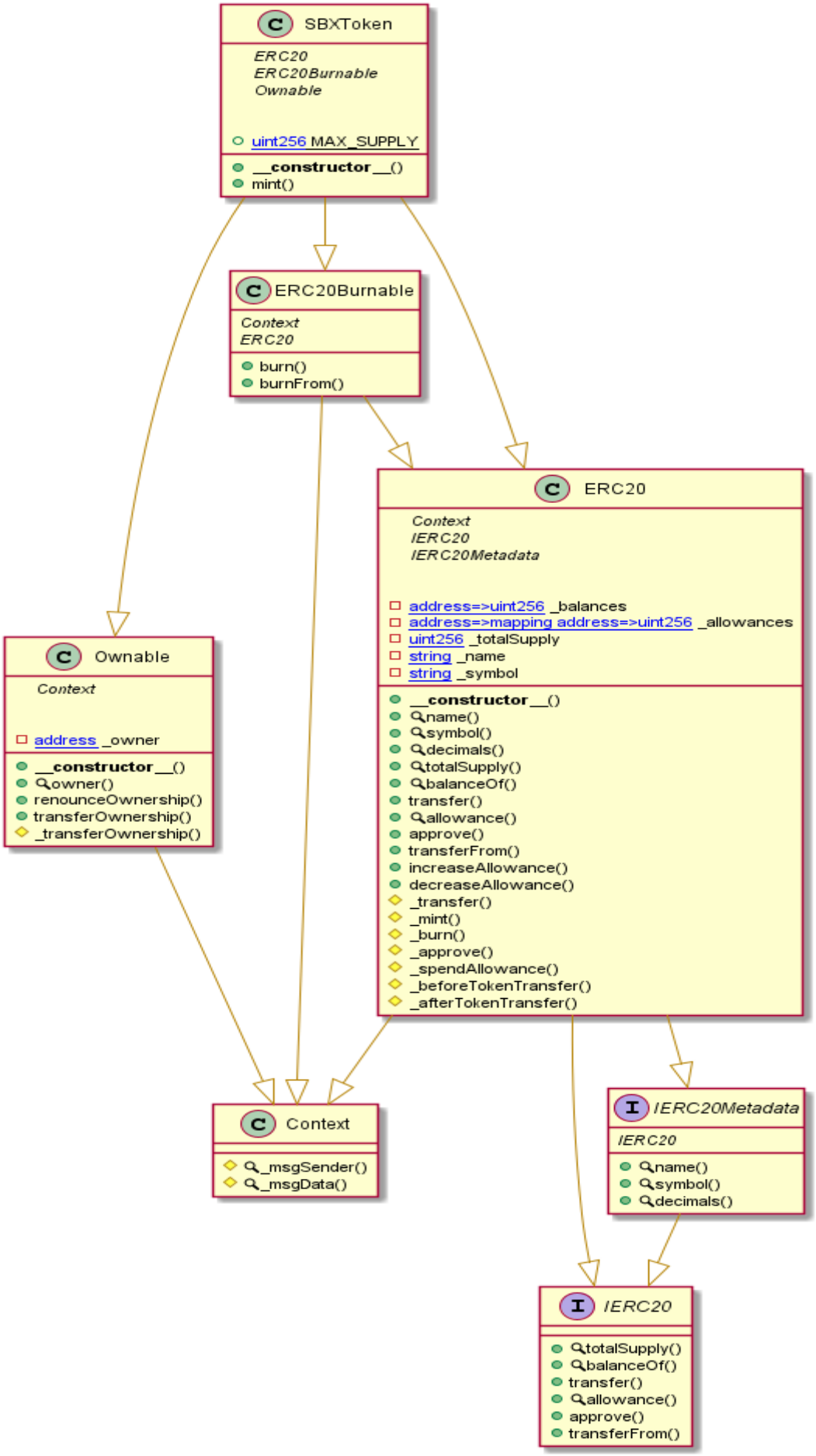
WeightedPool2TokensFactory Diagram



MasterChef Diagram

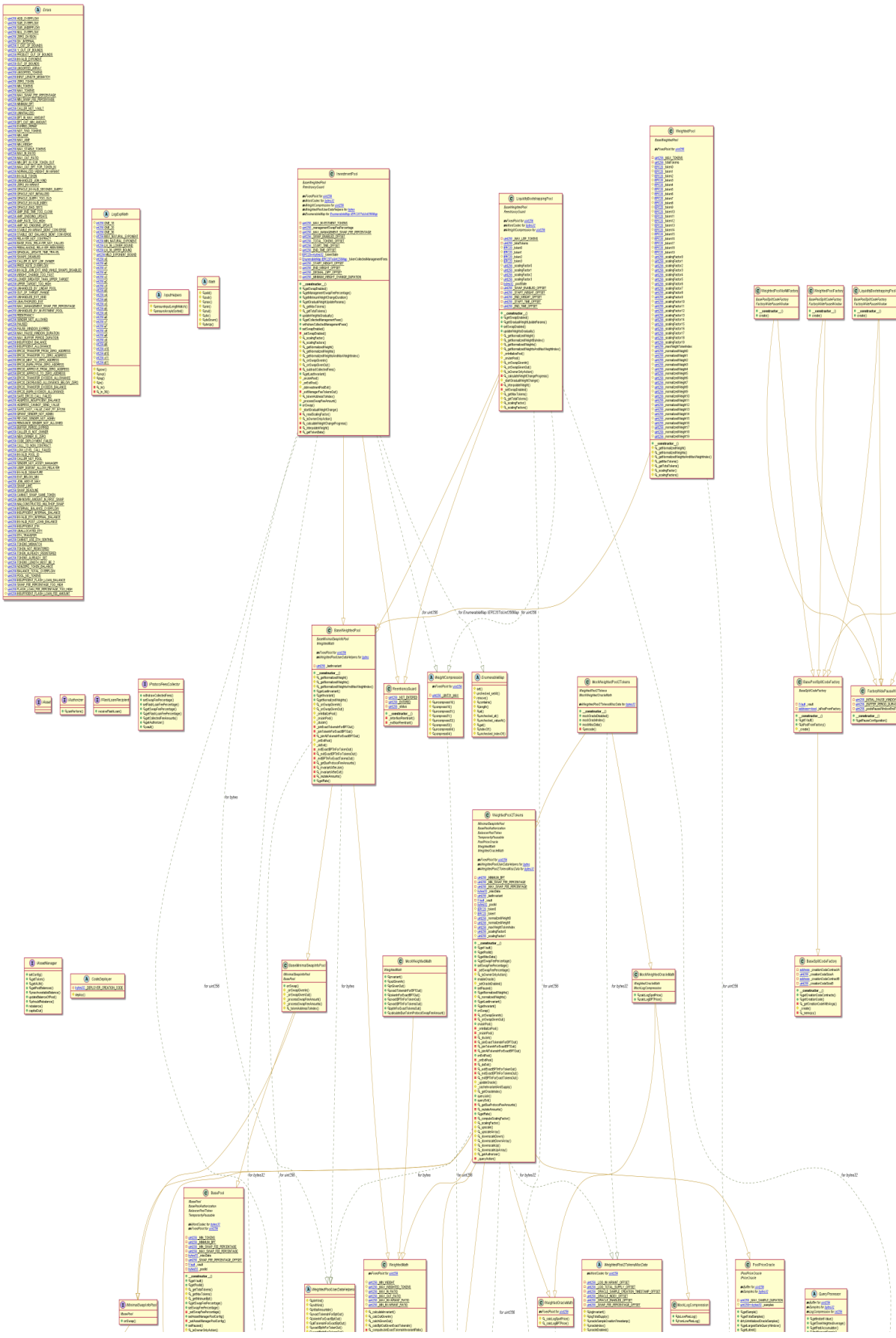


SBXToken Diagram



Email: audit@EtherAuthority.io

InvestmentPoolFactory Diagram



Slither Results Log

Slither log >> Authorizer.sol

```
INFO:Detectors:
Reentrancy in Authorizer.execute(uint256) (Authorizer_flat.sol#520-535):
        External calls:
        - result = scheduledAction.where.functionCall(scheduledAction.data) (Authorizer_flat.sol#533)
        Event emitted after the call(s):
        - ActionExecuted(id) (Authorizer_flat.sol#534)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Authorizer.execute(uint256) (Authorizer_flat.sol#520-535) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(id < scheduledActions.length,ACTION_DOES_NOT_EXIST) (Authorizer_flat.sol#521)
        - require(bool,string)(block.timestamp >= scheduledAction.executableAt,ACTION_NOT_EXECUTABLE) (Authorizer_flat.sol#527)
Authorizer.cancel(uint256) (Authorizer_flat.sol#540-552) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(id < scheduledActions.length,ACTION_DOES_NOT_EXIST) (Authorizer_flat.sol#541)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (Authorizer_flat.sol#165-176) uses assembly
        - INLINE ASM (Authorizer_flat.sol#172-174)
Address.verifyCallResult(bool,bytes) (Authorizer_flat.sol#233-249) uses assembly
        - INLINE ASM (Authorizer_flat.sol#241-244)
InputHelpers.ensureArrayIsSorted(IERC20[]) (Authorizer_flat.sol#340-347) uses assembly
        - INLINE ASM (Authorizer_flat.sol#343-345)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.isContract(address) (Authorizer_flat.sol#165-176) is never used and should be removed
Address.sendValue(address,uint256) (Authorizer_flat.sol#194-200) is never used and should be removed
InputHelpers.ensureArrayIsSorted(IERC20[]) (Authorizer_flat.sol#340-347) is never used and should be removed
InputHelpers.ensureArrayIsSorted(address[]) (Authorizer_flat.sol#349-360) is never used and should be removed
InputHelpers.ensureInputLengthMatch(uint256,uint256,uint256) (Authorizer_flat.sol#332-338) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Authorizer_flat.sol#194-200):
        - (success) = recipient.call{value: amount}() (Authorizer_flat.sol#198)
Low level call in Address.functionCall(address,bytes) (Authorizer_flat.sol#220-225):
        - (success,returndata) = target.call(data) (Authorizer_flat.sol#223)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable Errors.MAX_SWAP_FEE_PERCENTAGE (Authorizer_flat.sol#29) is too similar to Errors.MIN_SWAP_FEE_PERCENTAGE (Authorizer_flat.sol#
)
Variable Errors.X_OUT_OF_BOUNDS (Authorizer_flat.sol#14) is too similar to Errors.Y_OUT_OF_BOUNDS (Authorizer_flat.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
canPerform(bytes32,address,address) should be declared external:
        - Authorizer.canPerform(bytes32,address,address) (Authorizer_flat.sol#466-472)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Authorizer_flat.sol analyzed (7 contracts with 75 detectors), 16 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Multicall2.sol

```
INFO:Detectors:
Multicall2.aggregate(Multicall2.Call[]) (Multicall2_flat.sol#17-25) has external calls inside a loop: (success,ret) = calls
all(calls[i].callData) (Multicall2_flat.sol#21)
Multicall2.tryAggregate(bool,Multicall2.Call[]) (Multicall2_flat.sol#53-64) has external calls inside a loop: (success,ret)
target.call(calls[i].callData) (Multicall2_flat.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Pragma version>=0.5.0 (Multicall2_flat.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Multicall2.aggregate(Multicall2.Call[]) (Multicall2_flat.sol#17-25):
        - (success,ret) = calls[i].target.call(calls[i].callData) (Multicall2_flat.sol#21)
Low level call in Multicall2.tryAggregate(bool,Multicall2.Call[]) (Multicall2_flat.sol#53-64):
        - (success,ret) = calls[i].target.call(calls[i].callData) (Multicall2_flat.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
aggregate(Multicall2.Call[]) should be declared external:
        - Multicall2.aggregate(Multicall2.Call[]) (Multicall2_flat.sol#17-25)
blockAndAggregate(Multicall2.Call[]) should be declared external:
        - Multicall2.blockAndAggregate(Multicall2.Call[]) (Multicall2_flat.sol#26-28)
getBlockHash(uint256) should be declared external:
        - Multicall2.getBlockHash(uint256) (Multicall2_flat.sol#29-31)
getBlockNumber() should be declared external:
        - Multicall2.getBlockNumber() (Multicall2_flat.sol#32-34)
getCurrentBlockCoinbase() should be declared external:
        - Multicall2.getCurrentBlockCoinbase() (Multicall2_flat.sol#35-37)
getCurrentBlockDifficulty() should be declared external:
        - Multicall2.getCurrentBlockDifficulty() (Multicall2_flat.sol#38-40)
getCurrentBlockGasLimit() should be declared external:
        - Multicall2.getCurrentBlockGasLimit() (Multicall2_flat.sol#41-43)
getCurrentBlockTimestamp() should be declared external:
        - Multicall2.getCurrentBlockTimestamp() (Multicall2_flat.sol#44-46)
getEthBalance(address) should be declared external:
        - Multicall2.getEthBalance(address) (Multicall2_flat.sol#47-49)
getLastBlockHash() should be declared external:
        - Multicall2.getLastBlockHash() (Multicall2_flat.sol#50-52)
```

```
getCurrentBlockTimestamp() should be declared external:
        - Multicall2.getCurrentBlockTimestamp() (Multicall2_flat.sol#44-46)
getEthBalance(address) should be declared external:
        - Multicall2.getEthBalance(address) (Multicall2_flat.sol#47-49)
getLastBlockHash() should be declared external:
        - Multicall2.getLastBlockHash() (Multicall2_flat.sol#50-52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Multicall2_flat.sol analyzed (1 contracts with 75 detectors), 15 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> ProtocolFeesCollector.sol

```
INFO:Detectors:
ProtocolFeesCollector.getCollectedFeeAmounts(IERC20[]) (ProtocolFeesCollector_flat.sol#623-633) has external calls inside a
 loop: feeAmounts[i] = tokens[i].balanceOf(address(this)) (ProtocolFeesCollector_flat.sol#631)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
InputHelpers.ensureArrayIsSorted(IERC20[]) (ProtocolFeesCollector_flat.sol#183-189) uses assembly
        - INLINE ASM (ProtocolFeesCollector_flat.sol#185-187)
SafeERC20._callOptionalReturn(address,bytes) (ProtocolFeesCollector_flat.sol#280-291) uses assembly
        - INLINE ASM (ProtocolFeesCollector_flat.sol#283-288)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Authentication._canPerform(bytes32,address) (ProtocolFeesCollector_flat.sol#231) is never used and should be removed
InputHelpers.ensureArrayIsSorted(IERC20[]) (ProtocolFeesCollector_flat.sol#183-189) is never used and should be removed
InputHelpers.ensureArrayIsSorted(address[]) (ProtocolFeesCollector_flat.sol#191-202) is never used and should be removed
InputHelpers.ensureInputLengthMatch(uint256,uint256,uint256) (ProtocolFeesCollector_flat.sol#175-181) is never used and sho
uld be removed
ProtocolFeesCollector._canPerform(bytes32,address) (ProtocolFeesCollector_flat.sol#639-641) is never used and should be rem
oved
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (ProtocolFeesCollector_flat.sol#271-278) is never used and shoul
d be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in SafeERC20._callOptionalReturn(address,bytes) (ProtocolFeesCollector_flat.sol#280-291):
        - (success,returndata) = token.call(data) (ProtocolFeesCollector_flat.sol#281)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IVault.WETH() (ProtocolFeesCollector_flat.sol#542) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable Errors.MAX_SWAP_FEE_PERCENTAGE (ProtocolFeesCollector_flat.sol#56) is too similar to Errors.MIN_SWAP_FEE_PERCENTAG
E (ProtocolFeesCollector_flat.sol#57)
Variable Errors.X_OUT_OF_BOUNDS (ProtocolFeesCollector_flat.sol#43) is too similar to Errors.Y_OUT_OF_BOUNDS (ProtocolFeesC
ollector_flat.sol#44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

```
INFO:Detectors:
ProtocolFeesCollector._MAX_PROTOCOL_SWAP_FEE_PERCENTAGE (ProtocolFeesCollector_flat.sol#573) is never used in ProtocolFeesC
ollector (ProtocolFeesCollector_flat.sol#570-646)
ProtocolFeesCollector._MAX_PROTOCOL_FLASH_LOAN_FEE_PERCENTAGE (ProtocolFeesCollector_flat.sol#574) is never used in Protoco
lFeesCollector (ProtocolFeesCollector_flat.sol#570-646)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Slither:ProtocolFeesCollector_flat.sol analyzed (16 contracts with 75 detectors), 15 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MasterChef.sol

```
INFO:Detectors:
Pragma version0.8.4 (MasterChef.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MasterChef.sol#267-272):
        - (success) = recipient.call{value: amount}() (MasterChef.sol#270)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#335-346):
        - (success,returndata) = target.call{value: value}(data) (MasterChef.sol#344)
Low level call in Address.functionStaticCall(address,bytes,string) (MasterChef.sol#364-373):
        - (success,returndata) = target.staticcall(data) (MasterChef.sol#371)
Low level call in Address.functionDelegateCall(address,bytes,string) (MasterChef.sol#391-400):
        - (success,returndata) = target.delegatecall(data) (MasterChef.sol#398)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter MasterChef.init(address,uint256,uint256)._rewardToken (MasterChef.sol#792) is not in mixedCase
Parameter MasterChef.init(address,uint256,uint256)._startTime (MasterChef.sol#793) is not in mixedCase
Parameter MasterChef.init(address,uint256,uint256)._rewardPerSecond (MasterChef.sol#794) is not in mixedCase
```

```
Variable MasterChef.reserve1Address (MasterChef.sol#727) is too similar to MasterChef.reserve3Address (MasterChef.sol#731)
Variable MasterChef.reserve2Address (MasterChef.sol#729) is too similar to MasterChef.reserve3Address (MasterChef.sol#731)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
MasterChef.BONUS_MULTIPLIER (MasterChef.sol#759) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MasterChef.sol#637-639)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MasterChef.sol#645-648)
addPool(uint256,IERC20,bool) should be declared external:
        - MasterChef.addPool(uint256,IERC20,bool) (MasterChef.sol#820-847)
setPool(uint256,uint256,bool) should be declared external:
        - MasterChef.setPool(uint256,uint256,bool) (MasterChef.sol#850-868)
deposit(uint256,uint256) should be declared external:
        - MasterChef.deposit(uint256,uint256) (MasterChef.sol#950-978)
```

```
emergencyWithdraw(uint256) should be declared external:
        - MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1012-1024)
dev(address) should be declared external:
        - MasterChef.dev(address) (MasterChef.sol#1088-1090)
treasury(address) should be declared external:
        - MasterChef.treasury(address) (MasterChef.sol#1092-1094)
reserve1(address) should be declared external:
        - MasterChef.reserve1(address) (MasterChef.sol#1096-1098)
reserve2(address) should be declared external:
        - MasterChef.reserve2(address) (MasterChef.sol#1100-1102)
reserve3(address) should be declared external:
        - MasterChef.reserve3(address) (MasterChef.sol#1104-1106)
communtyGrowth(address) should be declared external:
        - MasterChef.communtyGrowth(address) (MasterChef.sol#1108-1110)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MasterChef.sol analyzed (9 contracts with 75 detectors), 129 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> SBXToken.sol

```
INFO:Detectors:
SBXToken.constructor() (SBXToken.sol#545-550) uses literals with too many digits:
        - _mint(msg.sender,12500000 * 10 ** decimals()) (SBXToken.sol#549)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SBXToken.sol#141-143)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SBXToken.sol#149-152)
name() should be declared external:
        - ERC20.name() (SBXToken.sol#192-194)
symbol() should be declared external:
        - ERC20.symbol() (SBXToken.sol#200-202)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (SBXToken.sol#231-233)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (SBXToken.sol#243-247)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (SBXToken.sol#266-270)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (SBXToken.sol#288-297)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (SBXToken.sol#311-315)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (SBXToken.sol#331-340)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (SBXToken.sol#521-523)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (SBXToken.sol#536-539)
mint(address,uint256) should be declared external:
        - SBXToken.mint(address,uint256) (SBXToken.sol#552-558)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SBXToken.sol analyzed (7 contracts with 75 detectors), 19 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> InvestmentPoolFactory.sol

```
INFO:Detectors:
ERC20._totalSupply (InvestmentPoolFactory_flat.sol#264) should be constant
ERC20Permit._PERMIT_TYPEHASH (InvestmentPoolFactory_flat.sol#646-647) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (InvestmentPoolFactory_flat.sol#273-275)
symbol() should be declared external:
        - ERC20.symbol() (InvestmentPoolFactory_flat.sol#276-278)
decimals() should be declared external:
        - ERC20.decimals() (InvestmentPoolFactory_flat.sol#279-281)
totalSupply() should be declared external:
        - ERC20.totalSupply() (InvestmentPoolFactory_flat.sol#282-284)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (InvestmentPoolFactory_flat.sol#285-287)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (InvestmentPoolFactory_flat.sol#288-291)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (InvestmentPoolFactory_flat.sol#295-298)
transferFrom(address,address,uint256) should be declared external:
        - BalancerPoolToken.transferFrom(address,address,uint256) (InvestmentPoolFactory_flat.sol#681-693)
        - ERC20.transferFrom(address,address,uint256) (InvestmentPoolFactory_flat.sol#299-307)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (InvestmentPoolFactory_flat.sol#649-666
)
nonces(address) should be declared external:
        - ERC20Permit.nonces(address) (InvestmentPoolFactory_flat.sol#667-669)
getActionId(bytes4) should be declared external:
        - Authentication.getActionId(bytes4) (InvestmentPoolFactory_flat.sol#717-719)
getPauseConfiguration() should be declared external:
        - FactoryWidePauseWindow.getPauseConfiguration() (InvestmentPoolFactory_flat.sol#730-739)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:InvestmentPoolFactory_flat.sol analyzed (24 contracts with 75 detectors), 73 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MetaStablePoolFactory.sol

```
INFO:Detectors:
FactoryWidePauseWindow.getPauseConfiguration() (MetaStablePoolFactory_flat.sol#67-78) uses timestamp for comparisons
        Dangerous comparisons:
        - currentTime < _poolsPauseWindowEndTime (MetaStablePoolFactory_flat.sol#69)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
getPauseConfiguration() should be declared external:
        - FactoryWidePauseWindow.getPauseConfiguration() (MetaStablePoolFactory_flat.sol#67-78)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MetaStablePoolFactory_flat.sol analyzed (6 contracts with 75 detectors), 2 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> NoProtocolFeeLiquidityBootstrappingPoolFactory.sol

```
INFO:Detectors:
FactoryWidePauseWindow.getPauseConfiguration() (NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol#82-93) uses timestamp for comparisons
        Dangerous comparisons:
        - currentTime < _poolsPauseWindowEndTime (NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Authentication._canPerform(bytes32,address) (NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol#68) is never used and should be removed
BalancerPoolToken._burnPoolTokens(address,uint256) (NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol#37-39) is never used and should be removed
BalancerPoolToken._mintPoolTokens(address,uint256) (NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol#33-35) is never used and should be removed
NoProtocolFeeLiquidityBootstrappingPoolFactory._canPerform(bytes32,address) (NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol#118-120) is never used and should be removed
SafeMath.add(uint256,uint256) (NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol#8-13) is never used and should be removed
SafeMath.sub(uint256,uint256) (NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol#15-17) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
getActionId(bytes4) should be declared external:
        - Authentication.getActionId(bytes4) (NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol#64-66)
getPauseConfiguration() should be declared external:
        - FactoryWidePauseWindow.getPauseConfiguration() (NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol#82-93)
isDisabled() should be declared external:
        - NoProtocolFeeLiquidityBootstrappingPoolFactory.isDisabled() (NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol#107-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:NoProtocolFeeLiquidityBootstrappingPoolFactory_flat.sol analyzed (6 contracts with 75 detectors), 10 result(s) found
```

Slither log >> StablePhantomPoolFactory.sol

```
INFO:Detectors:
FactoryWidePauseWindow.getPauseConfiguration() (StablePhantomPoolFactory_flat.sol#311-320) uses timestamp for comparisons
        Dangerous comparisons:
        - currentTime < _poolsPauseWindowEndTime (StablePhantomPoolFactory_flat.sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
CodeDeployer.deploy(bytes) (StablePhantomPoolFactory_flat.sol#181-190) uses assembly
        - INLINE ASM (StablePhantomPoolFactory_flat.sol#183-188)
BaseSplitCodeFactory.constructor(bytes) (StablePhantomPoolFactory_flat.sol#198-222) uses assembly
        - INLINE ASM (StablePhantomPoolFactory_flat.sol#205-208)
        - INLINE ASM (StablePhantomPoolFactory_flat.sol#212-216)
        - INLINE ASM (StablePhantomPoolFactory_flat.sol#218-221)
BaseSplitCodeFactory._getCreationCodeWithArgs(bytes) (StablePhantomPoolFactory_flat.sol#229-252) uses assembly
        - INLINE ASM (StablePhantomPoolFactory_flat.sol#237-244)
        - INLINE ASM (StablePhantomPoolFactory_flat.sol#247-250)
BaseSplitCodeFactory._create(bytes) (StablePhantomPoolFactory_flat.sol#253-266) uses assembly
        - INLINE ASM (StablePhantomPoolFactory_flat.sol#256-258)
        - INLINE ASM (StablePhantomPoolFactory_flat.sol#260-263)
BaseSplitCodeFactory._memcpy(uint256,uint256,uint256) (StablePhantomPoolFactory_flat.sol#267-285) uses assembly
        - INLINE ASM (StablePhantomPoolFactory_flat.sol#273-275)
        - INLINE ASM (StablePhantomPoolFactory_flat.sol#280-284)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Authentication._authenticateCaller() (StablePhantomPoolFactory_flat.sol#148-151) is never used and should be removed
Authentication._canPerform(bytes32,address) (StablePhantomPoolFactory_flat.sol#155) is never used and should be removed
BasePoolFactory._register(address) (StablePhantomPoolFactory_flat.sol#172-175) is never used and should be removed
BasePoolSplitCodeFactory._create(bytes) (StablePhantomPoolFactory_flat.sol#296-301) is never used and should be removed
BaseSplitCodeFactory._create(bytes) (StablePhantomPoolFactory_flat.sol#253-266) is never used and should be removed
ERC20._burn(address,uint256) (StablePhantomPoolFactory_flat.sol#109-115) is never used and should be removed
ERC20._mint(address,uint256) (StablePhantomPoolFactory_flat.sol#103-108) is never used and should be removed
ERC20._setupDecimals(uint8) (StablePhantomPoolFactory_flat.sol#124-126) is never used and should be removed
SafeMath.sub(uint256,uint256) (StablePhantomPoolFactory_flat.sol#26-28) is never used and should be removed
SafeMath.sub(uint256,uint256,uint256) (StablePhantomPoolFactory_flat.sol#29-33) is never used and should be removed
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Variable BaseSplitCodeFactory._creationCodeContractA (StablePhantomPoolFactory_flat.sol#194) is too similar to BaseSplitCodeFactory._creationCodeContractB (StablePhantomPoolFactory_flat.sol#196)
Variable BaseSplitCodeFactory._creationCodeSizeA (StablePhantomPoolFactory_flat.sol#195) is too similar to BaseSplitCodeFactory._creationCodeSizeB (StablePhantomPoolFactory_flat.sol#197)
Variable BaseSplitCodeFactory.constructor(bytes).creationCodeA (StablePhantomPoolFactory_flat.sol#204) is too similar to BaseSplitCodeFactory.constructor(bytes).creationCodeB (StablePhantomPoolFactory_flat.sol#210)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
Authentication (StablePhantomPoolFactory_flat.sol#139-156) does not implement functions:
        - Authentication._canPerform(bytes32,address) (StablePhantomPoolFactory_flat.sol#155)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (StablePhantomPoolFactory_flat.sol#49-51)
symbol() should be declared external:
        - ERC20.symbol() (StablePhantomPoolFactory_flat.sol#52-54)
decimals() should be declared external:
        - ERC20.decimals() (StablePhantomPoolFactory_flat.sol#55-57)
totalSupply() should be declared external:
        - ERC20.totalSupply() (StablePhantomPoolFactory_flat.sol#58-60)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (StablePhantomPoolFactory_flat.sol#61-63)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (StablePhantomPoolFactory_flat.sol#64-67)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (StablePhantomPoolFactory_flat.sol#68-70)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (StablePhantomPoolFactory_flat.sol#71-74)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (StablePhantomPoolFactory_flat.sol#75-82)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (StablePhantomPoolFactory_flat.sol#83-86)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (StablePhantomPoolFactory_flat.sol#87-90)
getCreationCodeContracts() should be declared external:
        - BaseSplitCodeFactory.getCreationCodeContracts() (StablePhantomPoolFactory_flat.sol#223-225)
getCreationCode() should be declared external:
        - BaseSplitCodeFactory.getCreationCode() (StablePhantomPoolFactory_flat.sol#226-228)
getPauseConfiguration() should be declared external:
        - FactoryWidePauseWindow.getPauseConfiguration() (StablePhantomPoolFactory_flat.sol#311-320)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StablePhantomPoolFactory_flat.sol analyzed (11 contracts with 75 detectors), 39 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> StablePoolFactory.sol

```
INFO:Detectors:
FactoryWidePauseWindow.getPauseConfiguration() (StablePoolFactory_flat.sol#154-163) uses timestamp for comparisons
        Dangerous comparisons:
        - currentTime < _poolsPauseWindowEndTime (StablePoolFactory_flat.sol#156)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
BasePoolFactory._register(address) (StablePoolFactory_flat.sol#142-145) is never used and should be removed
ERC20._beforeTokenTransfer(address,address,uint256) (StablePoolFactory_flat.sol#87-91) is never used and should be removed
ERC20._burn(address,uint256) (StablePoolFactory_flat.sol#69-75) is never used and should be removed
ERC20._mint(address,uint256) (StablePoolFactory_flat.sol#63-68) is never used and should be removed
ERC20._setupDecimals(uint8) (StablePoolFactory_flat.sol#84-86) is never used and should be removed
ERC20._transfer(address,address,uint256) (StablePoolFactory_flat.sol#51-62) is never used and should be removed
SafeMath.sub(uint256,uint256) (StablePoolFactory_flat.sol#11-13) is never used and should be removed
SafeMath.sub(uint256,uint256,uint256) (StablePoolFactory_flat.sol#14-18) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (StablePoolFactory_flat.sol#33-35)
symbol() should be declared external:
        - ERC20.symbol() (StablePoolFactory_flat.sol#36-38)
decimals() should be declared external:
        - ERC20.decimals() (StablePoolFactory_flat.sol#39-41)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (StablePoolFactory_flat.sol#44-47)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (StablePoolFactory_flat.sol#48-50)
getPauseConfiguration() should be declared external:
        - FactoryWidePauseWindow.getPauseConfiguration() (StablePoolFactory_flat.sol#154-163)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:StablePoolFactory_flat.sol analyzed (11 contracts with 75 detectors), 15 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Vault.sol

```
INFO:Detectors:
ProtocolFeesCollector.getCollectedFeeAmounts(IERC20[]) (Vault_flat.sol#388-398) has external calls inside a loop: feeAmount
s[i] = tokens[i].balanceOf(address(this)) (Vault_flat.sol#396)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
BalanceAllocation._decodeBalanceA(bytes32) (Vault_flat.sol#216-219) is never used and should be removed
BalanceAllocation._decodeBalanceB(bytes32) (Vault_flat.sol#220-223) is never used and should be removed
BalanceAllocation._pack(uint256,uint256,uint256) (Vault_flat.sol#237-243) is never used and should be removed
BalanceAllocation.cash(bytes32) (Vault_flat.sol#140-143) is never used and should be removed
BalanceAllocation.cashToManaged(bytes32,uint256) (Vault_flat.sol#199-204) is never used and should be removed
BalanceAllocation.decreaseCash(bytes32,uint256) (Vault_flat.sol#193-198) is never used and should be removed
BalanceAllocation.fromSharedToBalanceA(bytes32,bytes32) (Vault_flat.sol#224-226) is never used and should be removed
BalanceAllocation.fromSharedToBalanceB(bytes32,bytes32) (Vault_flat.sol#227-229) is never used and should be removed
BalanceAllocation.increaseCash(bytes32,uint256) (Vault_flat.sol#187-192) is never used and should be removed
BalanceAllocation.isNotZero(bytes32) (Vault_flat.sol#175-177) is never used and should be removed
BalanceAllocation.isZero(bytes32) (Vault_flat.sol#171-174) is never used and should be removed
BalanceAllocation.lastChangeBlock(bytes32) (Vault_flat.sol#148-151) is never used and should be removed
BalanceAllocation.managed(bytes32) (Vault_flat.sol#144-147) is never used and should be removed
BalanceAllocation.managedDelta(bytes32,bytes32) (Vault_flat.sol#152-154) is never used and should be removed
```

```
Variable BalanceAllocation.toSharedCash(bytes32,bytes32).tokenABalance (Vault_flat.sol#230) is too similar to BalanceAlloca
tion.toSharedCash(bytes32,bytes32).tokenBBalance (Vault_flat.sol#230)
Variable BalanceAllocation.toSharedCash(bytes32,bytes32).tokenABalance (Vault_flat.sol#230) is too similar to BalanceAlloca
tion.toSharedManaged(bytes32,bytes32).tokenBBalance (Vault_flat.sol#234)
Variable BalanceAllocation.toSharedManaged(bytes32,bytes32).tokenABalance (Vault_flat.sol#234) is too similar to BalanceAll
ocation.toSharedManaged(bytes32,bytes32).tokenBBalance (Vault_flat.sol#234)
Variable BalanceAllocation.toSharedManaged(bytes32,bytes32).tokenABalance (Vault_flat.sol#234) is too similar to BalanceAll
ocation.toSharedCash(bytes32,bytes32).tokenBBalance (Vault_flat.sol#230)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
VaultAuthorization._JOIN_TYPE_HASH (Vault_flat.sol#361) is never used in VaultAuthorization (Vault_flat.sol#359-378)
VaultAuthorization._EXIT_TYPE_HASH (Vault_flat.sol#362) is never used in VaultAuthorization (Vault_flat.sol#359-378)
VaultAuthorization._SWAP_TYPE_HASH (Vault_flat.sol#363) is never used in VaultAuthorization (Vault_flat.sol#359-378)
VaultAuthorization._BATCH_SWAP_TYPE_HASH (Vault_flat.sol#364) is never used in VaultAuthorization (Vault_flat.sol#359-378)
VaultAuthorization._SET_RELAYER_TYPE_HASH (Vault_flat.sol#365-366) is never used in VaultAuthorization (Vault_flat.sol#359-
378)
VaultAuthorization._approvedRelayers (Vault_flat.sol#368) is never used in VaultAuthorization (Vault_flat.sol#359-378)
ProtocolFeesCollector._MAX_PROTOCOL_SWAP_FEE_PERCENTAGE (Vault_flat.sol#381) is never used in ProtocolFeesCollector (Vault_
flat.sol#379-408)
ProtocolFeesCollector._MAX_PROTOCOL_FLASH_LOAN_FEE_PERCENTAGE (Vault_flat.sol#382) is never used in ProtocolFeesCollector (
Vault_flat.sol#379-408)
ProtocolFeesCollector._swapFeePercentage (Vault_flat.sol#384) is never used in ProtocolFeesCollector (Vault_flat.sol#379-40
8)
ProtocolFeesCollector._flashLoanFeePercentage (Vault_flat.sol#385) is never used in ProtocolFeesCollector (Vault_flat.sol#3
79-408)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
ProtocolFeesCollector._flashLoanFeePercentage (Vault_flat.sol#385) should be constant
ProtocolFeesCollector._swapFeePercentage (Vault_flat.sol#384) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
getProtocolFeesCollector() should be declared external:
        - Fees.getProtocolFeesCollector() (Vault_flat.sol#415-417)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Vault_flat.sol analyzed (14 contracts with 75 detectors), 60 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> WeightedPool.sol

```
INFO:Detectors:
FixedPoint.add(uint256,uint256) (WeightedPool_flat.sol#39-43) is never used and should be removed
FixedPoint.complement(uint256) (WeightedPool_flat.sol#93-95) is never used and should be removed
FixedPoint.divDown(uint256,uint256) (WeightedPool_flat.sol#63-72) is never used and should be removed
FixedPoint.divUp(uint256,uint256) (WeightedPool_flat.sol#73-82) is never used and should be removed
FixedPoint.mulDown(uint256,uint256) (WeightedPool_flat.sol#49-53) is never used and should be removed
FixedPoint.mulUp(uint256,uint256) (WeightedPool_flat.sol#54-62) is never used and should be removed
FixedPoint.powDown(uint256,uint256) (WeightedPool_flat.sol#83-87) is never used and should be removed
FixedPoint.powUp(uint256,uint256) (WeightedPool_flat.sol#88-92) is never used and should be removed
FixedPoint.sub(uint256,uint256) (WeightedPool_flat.sol#44-48) is never used and should be removed
SafeMath.add(uint256,uint256) (WeightedPool_flat.sol#20-24) is never used and should be removed
SafeMath.sub(uint256,uint256) (WeightedPool_flat.sol#25-27) is never used and should be removed
SafeMath.sub(uint256,uint256,uint256) (WeightedPool_flat.sol#28-32) is never used and should be removed
WeightedMath._calcBptInGivenExactTokensOut(uint256[],uint256[],uint256[],uint256,uint256) (WeightedPool_flat.sol#193-222) is
never used and should be removed
WeightedMath._calcBptOutGivenExactTokensIn(uint256[],uint256[],uint256[],uint256,uint256) (WeightedPool_flat.sol#145-176) is
never used and should be removed
WeightedMath._calcDueTokenProtocolSwapFeeAmount(uint256,uint256,uint256,uint256,uint256) (WeightedPool_flat.sol#251-267) is
ever used and should be removed
WeightedMath._calcInGivenOut(uint256,uint256,uint256,uint256,uint256) (WeightedPool_flat.sol#131-144) is never used and shou
d be removed
WeightedMath._calcOutGivenIn(uint256,uint256,uint256,uint256,uint256) (WeightedPool_flat.sol#117-130) is never used and shou
d be removed
WeightedMath._calcTokenInGivenExactBptOut(uint256,uint256,uint256,uint256,uint256) (WeightedPool_flat.sol#177-192) is never
sed and should be removed
WeightedMath._calcTokenOutGivenExactBptIn(uint256,uint256,uint256,uint256,uint256) (WeightedPool_flat.sol#223-238) is never
sed and should be removed
WeightedMath._calcTokensOutGivenExactBptIn(uint256[],uint256,uint256) (WeightedPool_flat.sol#239-250) is never used and shou
d be removed
WeightedMath._calculateInvariant(uint256[],uint256[]) (WeightedPool_flat.sol#106-116) is never used and should be removed
WeightedPool._doExit(uint256[],uint256[],bytes) (WeightedPool_flat.sol#436-442) is never used and should be removed
WeightedPool._doJoin(uint256[],uint256[],bytes) (WeightedPool_flat.sol#394-400) is never used and should be removed
WeightedPool._exitBPTInForExactTokensOut(uint256[],uint256[],bytes) (WeightedPool_flat.sol#457-463) is never used and should
be removed
WeightedPool._exitExactBPTInForTokenOut(uint256[],uint256[],bytes) (WeightedPool_flat.sol#443-449) is never used and should
```

```
WeightedPool._mutateAmounts(uint256[],uint256[],function(uint256,uint256) returns(uint256)) (WeightedPool_flat.sol#489-495)
s never used and should be removed
WeightedPool._normalizedWeight(IERC20) (WeightedPool_flat.sol#323-325) is never used and should be removed
WeightedPool._onExitPool(bytes32,address,address,uint256[],uint256,uint256,bytes) (WeightedPool_flat.sol#416-435) is never u
ed and should be removed
WeightedPool._onInitializePool(bytes32,address,address,bytes) (WeightedPool_flat.sol#353-360) is never used and should be re
oved
WeightedPool._onJoinPool(bytes32,address,address,uint256[],uint256,uint256,bytes) (WeightedPool_flat.sol#361-393) is never u
ed and should be removed
WeightedPool._onSwapGivenIn(uint256,uint256) (WeightedPool_flat.sol#339-345) is never used and should be removed
WeightedPool._onSwapGivenOut(uint256,uint256) (WeightedPool_flat.sol#346-352) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Variable WeightedPool._normalizedWeight0 (WeightedPool_flat.sol#274) is too similar to WeightedPool._normalizedWeight1 (Weig
tedPool_flat.sol#275)
Variable WeightedPool._normalizedWeight0 (WeightedPool_flat.sol#274) is too similar to WeightedPool._normalizedWeight2 (Weig
tedPool_flat.sol#276)
Variable WeightedPool._normalizedWeight0 (WeightedPool_flat.sol#274) is too similar to WeightedPool._normalizedWeight3 (Weig
tedPool_flat.sol#277)
Variable WeightedPool._normalizedWeight0 (WeightedPool_flat.sol#274) is too similar to WeightedPool._normalizedWeight4 (Weig
tedPool_flat.sol#278)
Variable WeightedPool._normalizedWeight1 (WeightedPool_flat.sol#275) is too similar to WeightedPool._normalizedWeight2 (Weig
tedPool_flat.sol#276)
Variable WeightedPool._normalizedWeight1 (WeightedPool_flat.sol#275) is too similar to WeightedPool._normalizedWeight3 (Weig
tedPool_flat.sol#277)
Variable WeightedPool._normalizedWeight1 (WeightedPool_flat.sol#275) is too similar to WeightedPool._normalizedWeight4 (Weig
tedPool_flat.sol#278)
Variable WeightedPool._normalizedWeight2 (WeightedPool_flat.sol#276) is too similar to WeightedPool._normalizedWeight3 (Weig
tedPool_flat.sol#277)
```

```
INFO:Detectors:
FixedPoint.MAX_POW_RELATIVE_ERROR (WeightedPool_flat.sol#37) is never used in FixedPoint (WeightedPool_flat.sol#35-96)
FixedPoint.MIN_POW_BASE_FREE_EXPONENT (WeightedPool_flat.sol#38) is never used in FixedPoint (WeightedPool_flat.sol#35-96)
WeightedMath._MIN_WEIGHT (WeightedPool_flat.sol#100) is never used in WeightedPool (WeightedPool_flat.sol#270-499)
WeightedMath._MAX_WEIGHTED_TOKENS (WeightedPool_flat.sol#101) is never used in WeightedPool (WeightedPool_flat.sol#270-499)
WeightedMath._MAX_IN_RATIO (WeightedPool_flat.sol#102) is never used in WeightedPool (WeightedPool_flat.sol#270-499)
WeightedMath._MAX_OUT_RATIO (WeightedPool_flat.sol#103) is never used in WeightedPool (WeightedPool_flat.sol#270-499)
WeightedMath._MAX_INVARIANT_RATIO (WeightedPool_flat.sol#104) is never used in WeightedPool (WeightedPool_flat.sol#270-499)
WeightedMath._MIN_INVARIANT_RATIO (WeightedPool_flat.sol#105) is never used in WeightedPool (WeightedPool_flat.sol#270-499)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
getInvariant() should be declared external:
        - WeightedPool.getInvariant() (WeightedPool_flat.sol#334-335)
getRate() should be declared external:
        - WeightedPool.getRate() (WeightedPool_flat.sol#496-498)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:WeightedPool_flat.sol analyzed (5 contracts with 75 detectors), 76 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> WeightedPool2TokensFactory.sol

```
INFO:Detectors:
ERC20._burn(address,uint256) (WeightedPool2TokensFactory_flat.sol#95-101) is never used and should be removed
ERC20._mint(address,uint256) (WeightedPool2TokensFactory_flat.sol#88-94) is never used and should be removed
ERC20._setupDecimals(uint8) (WeightedPool2TokensFactory_flat.sol#112-114) is never used and should be removed
Math.add(int256,int256) (WeightedPool2TokensFactory_flat.sol#127-131) is never used and should be removed
Math.add(uint256,uint256) (WeightedPool2TokensFactory_flat.sol#122-126) is never used and should be removed
Math.divDown(uint256,uint256) (WeightedPool2TokensFactory_flat.sol#153-156) is never used and should be removed
Math.divUp(uint256,uint256) (WeightedPool2TokensFactory_flat.sol#157-164) is never used and should be removed
Math.max(uint256,uint256) (WeightedPool2TokensFactory_flat.sol#142-144) is never used and should be removed
Math.min(uint256,uint256) (WeightedPool2TokensFactory_flat.sol#145-147) is never used and should be removed
Math.mul(uint256,uint256) (WeightedPool2TokensFactory_flat.sol#148-152) is never used and should be removed
Math.sub(int256,int256) (WeightedPool2TokensFactory_flat.sol#137-141) is never used and should be removed
Math.sub(uint256,uint256) (WeightedPool2TokensFactory_flat.sol#132-136) is never used and should be removed
SafeMath.sub(uint256,uint256) (WeightedPool2TokensFactory_flat.sol#11-13) is never used and should be removed
SafeMath.sub(uint256,uint256,uint256) (WeightedPool2TokensFactory_flat.sol#14-18) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (WeightedPool2TokensFactory_flat.sol#33-35)
symbol() should be declared external:
        - ERC20.symbol() (WeightedPool2TokensFactory_flat.sol#36-38)
decimals() should be declared external:
        - ERC20.decimals() (WeightedPool2TokensFactory_flat.sol#39-41)
totalSupply() should be declared external:
        - ERC20.totalSupply() (WeightedPool2TokensFactory_flat.sol#42-44)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (WeightedPool2TokensFactory_flat.sol#45-47)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (WeightedPool2TokensFactory_flat.sol#48-51)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (WeightedPool2TokensFactory_flat.sol#52-54)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (WeightedPool2TokensFactory_flat.sol#55-58)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (WeightedPool2TokensFactory_flat.sol#59-67)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (WeightedPool2TokensFactory_flat.sol#68-71)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (WeightedPool2TokensFactory_flat.sol#72-75)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:WeightedPool2TokensFactory_flat.sol analyzed (4 contracts with 75 detectors), 25 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

Authorizer.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Authorizer.cancel(uint256): Could potentially lead to re-entrancy vulnerability.

more

Pos: 540:4:

Block timestamp:

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

more

Pos: 629:31:

Gas & Economy

Gas costs:

Gas requirement of function Authorizer.hasPermission 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: 453:4:

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: 496:51:

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: 563:8:

Miscellaneous

Constant/View/Pure functions:

Authorizer._executeActionId(uint256) : Is constant but potentially should not be.

more

Pos: 643:4:

Similar variable names:

Authorizer.schedule(address,bytes,address[]) : Variables have very similar names "delay" and "delays".
Pos: 512:24:

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: 545:8:

InvestmentPoolFactory.sol

Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')
Pos: not available

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: 3688:55:

Gas & Economy

Gas costs:

INTERNAL ERROR in module Gas costs: Cannot read properties of undefined (reading 'type')
Pos: not available

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')
Pos: not available

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: 2938:5:

MetaStablePoolFactory.sol

Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')
Pos: not available

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 480:8:

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: 4239:64:

Gas & Economy

Gas costs:

INTERNAL ERROR in module Gas costs: Cannot read properties of undefined (reading 'type')
Pos: not available

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: 2251:61:

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')
Pos: not available

Similar variable names:

BaseSplitCodeFactory.(bytes) : Variables have very similar names "_creationCodeContractA" and "_creationCodeContractB". Note: Modifiers are currently not considered by this static analysis.
Pos: 516:8:

Multicall2.sol

Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')
Pos: not available

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: 45:20:

Gas & Economy

Gas costs:

Gas requirement of function Multicall2.aggregate is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 17:4:

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: 55:8:

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')
Pos: not available

No return:

INTERNAL ERROR in module No return: Cannot read properties of undefined (reading 'name')
Pos: not available

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: 59:16:

NoProtocolFeeLiquidityBootstrappingPoolFactory.sol

Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')

Pos: not available

Inline assembly:

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

more

Pos: 612:12:

Block timestamp:

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

more

Pos: 5337:83:

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: 5174:55:

Gas & Economy

Gas costs:

Gas requirement of function BalancerPoolToken.name 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: 902:28:

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')

Pos: not available

Similar variable names:

LogExpMath.pow(uint256,uint256) : Variables have very similar names "ONE_18" and "ONE_20". Note: Modifiers are currently not considered by this static analysis.
Pos: 184:1:

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: 4956:78:

ProtocolFeesCollector.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20._callOptionalReturn(address,bytes): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 280:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 185:8:

Gas & Economy

Gas costs:

Gas requirement of function ProtocolFeesCollector.getActionId 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: 227:4:

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: 630:8:

Miscellaneous

Constant/View/Pure functions:

ProtocolFeesCollector.getCollectedFeeAmounts(contract IERC20[]) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 623:4:

Similar variable names:

ProtocolFeesCollector.withdrawCollectedFees(contract IERC20[],uint256[],address) : Variables have very similar names "token" and "tokens". Note: Modifiers are currently not considered by this static analysis.

Pos: 599:12:

StablePhantomPoolFactory.sol

Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')

Pos: not available

Inline assembly:

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

more

Pos: 33:8:

Gas & Economy

Gas costs:

Gas requirement of function StablePool.getPausedState 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: 225:4:

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')

Pos: not available

Similar variable names:

WordCodec.insertUint10(bytes32,uint256,uint256) : Variables have very similar names "_MASK_10" and "_MASK_31". Note: Modifiers are currently not considered by this static analysis.

Pos: 96:56:

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: 2469:43:

StablePoolFactory.sol

Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')
Pos: not available

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 875:12:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 2520:6:

Gas & Economy

Gas costs:

Gas requirement of function MockStableMath.invariant 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: 771:28:

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')
Pos: not available

Similar variable names:

LogExpMath.pow(uint256,uint256) : Variables have very similar names "ONE_18" and "ONE_20".
Note: Modifiers are currently not considered by this static analysis.
Pos: 210:3:

No return:

INTERNAL ERROR in module No return: Cannot read properties of undefined (reading 'name')
Pos: not available

Vault.sol

Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')
Pos: not available

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 469:8:

Gas & Economy

Gas costs:

Gas requirement of function ProtocolFeesCollector.getActionId 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: 379:4:

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')
Pos: not available

Similar variable names:

BalanceAllocation.totalsAndLastChangeBlock(bytes32[]) : Variables have very similar names "balance" and "balances". Note: Modifiers are currently not considered by this static analysis.
Pos: 163:32:

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: 912:46:

WeightedPool.sol

Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')
Pos: not available

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 344:8:

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: 1056:39:

Gas & Economy

Gas costs:

Gas requirement of function BalancerPoolToken.name 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: 48:4:

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')
Pos: not available

Similar variable names:

ERC20.(string,string) : Variables have very similar names "_name" and "name_". Note: Modifiers are currently not considered by this static analysis.
Pos: 44:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1108:10:

WeightedPool2TokensFactory.sol

Security

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot read properties of undefined (reading 'name')
Pos: not available

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 156:8:

Gas & Economy

Gas costs:

Gas requirement of function ProtocolFeesCollector.getActionId is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 196:4:

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: 3185:32:

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot read properties of undefined (reading 'name')
Pos: not available

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: 3173:64:

MasterChef.sol

Security

Check-effects-interaction:

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

more

Pos: 335:4:

Block timestamp:

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

more

Pos: 836:63:

Gas & Economy

Gas costs:

Gas requirement of function MasterChef.init 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: 791:4:

Miscellaneous

Constant/View/Pure functions:

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

more

Pos: 354:4:

Similar variable names:

MasterChef.updatePool(uint256) : Variables have very similar names "reserve2Address" and "reserve3Address". Note: Modifiers are currently not considered by this static analysis.

Pos: 939:25:

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: 1007:8:

Data truncated:

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

Pos: 943:40:

Email: audit@EtherAuthority.io

SBXToken.sol

Gas & Economy

Gas costs:

Gas requirement of function ERC20.name 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: 192:4:

Gas costs:

Gas requirement of function SBXToken.mint is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 552:4:

Miscellaneous

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 24:4:

Similar variable names:

ERC20Burnable.burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 538:23:

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: 553:8:

Solhint Linter

Authorizer.sol

```
Authorizer.sol:3:1: Error: Compiler version ^0.6.12 does not satisfy
the r semver requirement
Authorizer.sol:198:10: Error: Variable "success" is unused
Authorizer.sol:223:51: Error: Avoid using low level calls.
Authorizer.sol:245:20: Error: Code contains empty blocks
Authorizer.sol:328:73: Error: Code contains empty blocks
Authorizer.sol:336:21: Error: Code contains empty blocks
Authorizer.sol:547:9: Error: Variable "action" is unused
Authorizer.sol:639:73: Error: Code contains empty blocks
```

InvestmentPoolFactory.sol

```
InvestmentPoolFactory.sol:3:1: Error: Compiler version ^0.6.12 does
not satisfy the r semver requirement
InvestmentPoolFactory.sol:139:5: Error: Explicitly mark visibility of
state
InvestmentPoolFactory.sol:140:5: Error: Explicitly mark visibility of
state
```

MetaStablePoolFactory.sol

```
MetaStablePoolFactory.sol:3820:11: Error: Visibility modifier must be
first in list of modifiers
MetaStablePoolFactory.sol:4068:70: Error: Avoid to make time-based
decisions in your business logic
MetaStablePoolFactory.sol:4126:16: Error: Code contains empty blocks
MetaStablePoolFactory.sol:4136:16: Error: Code contains empty blocks
MetaStablePoolFactory.sol:4136:16: Error: Code contains empty blocks
MetaStablePoolFactory.sol:4159:17: Error: Avoid to make time-based
decisions in your business logic39:41: Error: Avoid to make
time-based decisions in your business logic
```

Multicall2.sol

```
Multicall2.sol:3:1: Error: Compiler version >=0.5.0 does not satisfy
the r semver requirement
Multicall2.sol:21:48: Error: Avoid using low level calls.
Multicall2.sol:45:21: Error: Avoid to make time-based decisions in
your business logic
Multicall2.sol:56:48: Error: Avoid using low level calls.
```

NoProtocolFeeLiquidityBootstrappingPoolFactory.sol

```
NoProtocolFeeLiquidityBootstrappingPoolFactory.sol:4:1: Error:
Compiler version ^0.6.12 does not satisfy the r semver requirement
NoProtocolFeeLiquidityBootstrappingPoolFactory.sol:140:5: Error:
Explicitly mark visibility of state
NoProtocolFeeLiquidityBootstrappingPoolFactory.sol:142:5: Error:
Explicitly mark visibility of state
NoProtocolFeeLiquidityBootstrappingPoolFactory.sol:143:5: Error:
Explicitly mark visibility of state
NoProtocolFeeLiquidityBootstrappingPoolFactory.sol:145:5: Error:
Explicitly mark visibility of state
NoProtocolFeeLiquidityBootstrappingPoolFactory.sol:146:5: Error:
Explicitly mark visibility of state
NoProtocolFeeLiquidityBootstrappingPoolFactory.sol:148:5: Error:
Explicitly mark visibility of state
NoProtocolFeeLiquidityBootstrappingPoolFactory.sol:153:21: Error:
Constant name must be in capitalized SNAKE_CASE
```

ProtocolFeesCollector.sol

```
ProtocolFeesCollector.sol:3:1: Error: Compiler version ^0.6.12 does
not satisfy the r semver requirement
ProtocolFeesCollector.sol:171:73: Error: Code contains empty blocks
ProtocolFeesCollector.sol:179:21: Error: Code contains empty blocks
ProtocolFeesCollector.sol:185:9: Error: Avoid using inline assembly.
It is acceptable only in rare cases
ProtocolFeesCollector.sol:223:9: Error: Variable "actionId" is unused
ProtocolFeesCollector.sol:281:51: Error: Avoid using low level calls.
ProtocolFeesCollector.sol:283:9: Error: Avoid using inline assembly.
It is acceptable only in rare cases
ProtocolFeesCollector.sol:281:24: Error: Variable "returndata" is
unused
ProtocolFeesCollector.sol:319:1: Error: Code contains empty blocks
ProtocolFeesCollector.sol:542:5: Error: Function name must be in
mixedCase
```

StablePhantomPoolFactory.sol

```
StablePhantomPoolFactory.sol:3:1: Error: Compiler version ^0.6.12
does not satisfy the r semver requirement
StablePhantomPoolFactory.sol:21:73: Error: Code contains empty blocks
StablePhantomPoolFactory.sol:28:21: Error: Code contains empty blocks
StablePhantomPoolFactory.sol:33:9: Error: Avoid using inline
assembly. It is acceptable only in rare cases
StablePhantomPoolFactory.sol:217:38: Error: Avoid to make time-based
decisions in your business logic
StablePhantomPoolFactory.sol:240:21: Error: Code contains empty
blocks
StablePhantomPoolFactory.sol:242:16: Error: Code contains empty
```

```
blocks
StablePhantomPoolFactory.sol:248:47: Error: Code contains empty
blocks
StablePhantomPoolFactory.sol:251:44: Error: Code contains empty
blocks
StablePhantomPoolFactory.sol:256:16: Error: Avoid to make time-based
decisions in your business logic
StablePhantomPoolFactory.sol:275:40: Error: Variable "errorCode" is
unused
```

StablePoolFactory.sol

```
StablePoolFactory.sol:3:1: Error: Compiler version ^0.6.12 does not
satisfy the r semver requirement
StablePoolFactory.sol:174:5: Error: Explicitly mark visibility of
state
StablePoolFactory.sol:175:5: Error: Explicitly mark visibility of
state
StablePoolFactory.sol:176:5: Error: Explicitly mark visibility of
state
StablePoolFactory.sol:177:5: Error: Explicitly mark visibility of
state
StablePoolFactory.sol:178:5: Error: Explicitly mark visibility of
state
StablePoolFactory.sol:179:5: Error: Explicitly mark visibility of
state
StablePoolFactory.sol:180:5: Error: Explicitly mark visibility of
state
StablePoolFactory.sol:181:5: Error: Explicitly mark visibility of
state
StablePoolFactory.sol:182:5: Error: Explicitly mark visibility of
state
StablePoolFactory.sol:182:21: Error: Constant name must be in
capitalized SNAKE_CASE
StablePoolFactory.sol:183:5: Error: Explicitly mark visibility of
state
```

Vault.sol

```
Vault.sol:454:31: Error: Variable name must be in mixedCase
Vault.sol:455:31: Error: Variable name must be in mixedCase
Vault.sol:469:9: Error: Avoid using inline assembly. It is acceptable
only in rare cases
Vault.sol:477:55: Error: Visibility modifier must be first in list of
modifiers
Vault.sol:477:62: Error: Code contains empty blocks
Vault.sol:485:47: Error: Variable "errorCode" is unused
```

WeightedPoolsol

```
WeightedPool.sol:2:1: Error: Compiler version ^0.6.12 does not
satisfy the r semver requirement
WeightedPool.sol:28:40: Error: Variable "errorCode" is unused
WeightedPool.sol:142:24: Error: Code contains empty blocks
WeightedPool.sol:150:1: Error: Code contains empty blocks
WeightedPool.sol:172:5: Error: Function name must be in mixedCase
WeightedPool.sol:296:9: Error: Variable "actionId" is unused
WeightedPool.sol:343:51: Error: Avoid using low level calls.
WeightedPool.sol:344:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
WeightedPool.sol:343:24: Error: Variable "returndata" is unused
WeightedPool.sol:587:5: Error: Function name must be in mixedCase
WeightedPool.sol:649:36: Error: Avoid to make time-based decisions in
your business logic
```

WeightedPool2TokensFactory.sol

```
WeightedPool2TokensFactory.sol:3:1: Error: Compiler version 0.6.12
does not satisfy the r semver requirement
WeightedPool2TokensFactory.sol:23:1: Error: Code contains empty
blocks
WeightedPool2TokensFactory.sol:144:73: Error: Code contains empty
blocks
WeightedPool2TokensFactory.sol:151:21: Error: Code contains empty
blocks
WeightedPool2TokensFactory.sol:156:9: Error: Avoid using inline
assembly. It is acceptable only in rare cases
WeightedPool2TokensFactory.sol:163:9: Error: Avoid using inline
assembly. It is acceptable only in rare cases
```

MasterChef.sol

```
MasterChef.sol:11:18: Error: Parse error: missing ';' at '{'
MasterChef.sol:24:18: Error: Parse error: missing ';' at '{'
MasterChef.sol:36:18: Error: Parse error: missing ';' at '{'
MasterChef.sol:53:18: Error: Parse error: missing ';' at '{'
MasterChef.sol:65:18: Error: Parse error: missing ';' at '{'
MasterChef.sol:161:18: Error: Parse error: missing ';' at '{'
MasterChef.sol:184:18: Error: Parse error: missing ';' at '{'
MasterChef.sol:210:18: Error: Parse error: missing ';' at '{'
MasterChef.sol:561:18: Error: Parse error: missing ';' at '{'
```

SBXToken.sol

```
SBXToken.sol:335:18: Error: Parse error: missing ';' at '{'
SBXToken.sol:368:18: Error: Parse error: missing ';' at '{'
SBXToken.sol:417:18: Error: Parse error: missing ';' at '{'
SBXToken.sol:468:22: Error: Parse error: missing ';' at '{'
```

Software analysis result:

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

Email: audit@EtherAuthority.io

Email: audit@EtherAuthority.io