RD AUDITORS

Stabull, Smart Contract, Code Review and Security Analysis Report

Customer: Stabull
Prepared on: 24th Aug, 2023
Platform: Polygon
Language: Solidity

rdauditors.com

Table of Contents

Disclaimer

This document may contain confidential information about its systems and intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the customer or it can be disclosed publicly after all vulnerabilities are fixed - upon the decision of the customer.

Documentation

Name	Smart Contract Code Review and Security Analysis Report of Stabull
Platform	Polygon/ Solidity
File 1	AssimilatorV2.sol
MD5 hash	49ca62f22ce9b64bc7fe1cec533f72ec
SHA256 hash	56ce396b589f7586be1893ceaed2266460712977f694ce78a81aba0f8ca6480b
File 2	AssimilatorFactory.sol
MD5 hash	3356185f38e1baedc7dde24e7341d26a
SHA256 hash	e54559053a213d61dfd19dc81f9cd39fbc09c248fcf92842ae739f8069958c48
File 3	Assimilators.sol
MD5 hash	165dc2c69847610251bbf6190a81b7f2
SHA256 hash	763320949ff5a55653d5b07cbbebad8241fefe20cbe7e87c8075538499d27e38
File 4	Config.sol

MD5 hash	b2b8a245dea704fb89d4c02523010467
SHA256 hash	8f5280262d159149a81455735863831e0e369417495954956d7ff91f9b9a0f22
File 5	Curve.sol
MD5 hash	914f033648eff5be0850b7cc11feadf5
SHA256 hash	8feee9b4cdda018e13cb4120c5a841532d419ad58d758995026ff5a6e8c677c5
File 6	CurveFactory.sol
MD5 hash	ba7a399fc9b3c70e8dabbaba0a9f1bb5
SHA256 hash	4a1f5419341689db9b9147487295e15e4ee533b5f5c259f9c1f504d6e9e983ae
File 7	CurveFactoryV2.sol
MD5 hash	9fdd033ca6b01d97c443f00300d606e3
SHA256 hash	0ec84571117c5bb15f8828bb573cf504c8d19547305fae6a235c4aabd5b2ca78
File 8	CurveMath.sol
MD5 hash	8d7e1d8727275891fa21c5f46ab8da52

SHA256 hash	4ad6b82a8ffabc1b9c61312f46b8ddafc3c72728831d5dce6a78067e6b49ee9c
File 9	Orchestrator.sol
MD5 hash	da34aebb8a55026b1ba3c8b5a5f8f790
SHA256 hash	4c61ca20eca2f6e8054f98c28df553fbcd5f8ae330a343dcfb6b11386fc401a1
File 10	ProportionalLiquidity.sol
MD5 hash	2aa360dba401f3d3d8d906f0f0cdc080
SHA256 hash	ffeded1eda0dcf3c4b560c694958c9f4b1e00961e3bf54f7ba09aba904e9348e
File 11	Router.sol
MD5 hash	9538e1a571964825d4a210f5b9b75a72
SHA256 hash	732da6cb37b7a198dabb983d3ca3e1c28d383742a9f4732a37f2fbaa5a7c4894
File 12	Storage.sol
MD5 hash	7fc053cf9ce662fbb91f916776ecabe8

SHA256 hash	43ec52295561748afe7b6f6f0234d16d6f81200d9b05a9539f6661d8b4c9cb09
File 13	Structs.sol
MD5 hash	3e512b829b3ee7f36edce04524f55cc0
SHA256 hash	b9f3258d2aafcf4482504bd897a275d87eef1314c33c72ec762ef17b65426e0b
File 14	Swaps.sol
MD5 hash	d46e8ec7cda373700c25b126e8cf1047
SHA256 hash	408f59ed32f7b2e710d2c365d35fd67f7ff9f3fb57d7ec37ded061071da68764
File 15	ViewLiquidity.sol
MD5 hash	13028ac7547957abaec2ec72105cc999
SHA256 hash	20bffb687782f6dc7da648590b3b4ec50b5f4f37a61148f0fe8e2a4bc7246ca9
File 16	Zap.sol
MD5 hash	8d3b20defe5582f6bd31bc09909a1db2

SHA256 hash	94d12e23f0ce1b6e95748f7cc3fef99503e970b2116dca4f956cc2ec7e4a03a3
Date	24/08/2023

Introduction

RD Auditors (Consultant) were contracted by Stabull (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report represents the findings of the security assessment of the customer`s smart contract and its code review conducted between 11th- 24th Aug, 2023.

This contract consists of sixteen files.

Project Scope

The scope of the project is a smart contract. We have scanned this smart contract for commonly known and more specific vulnerabilities, below are those considered (the full list includes but is not limited to):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, the customer's solidity smart contract is now **Well-Secured.**

Automated checks are with smartDec, Mythril, Slither and remix IDE. All issues were performed by our team, which included the analysis of code functionality, the manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the AS-IS section and all issues found are located in the audit overview section.

We found the following;

Total Issues	0
Critical	0
High	0
Medium	0
Low	0
Very Low	0

Code Quality

The libraries within this smart contract are part of a 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 to a specific address and its properties/methods can be reused many times by other contracts.

The Stabull team has not provided scenario and unit test scripts, which would help to determine the integrity of the code in an automated way.

Documentation

We were given the Stabull smart contract code as an url link:

https://gitlab.rapidinnovation.tech/root/blockchain-contracts-stabull/-/tree/STB-Polygon/src

The hash of that code is mentioned above in the table. As mentioned above, It's recommended to write comments in the smart contract code, so anyone can quickly understand the programming flow as well as complex code logic.

Comments are very helpful in understanding the overall architecture of the protocol. It also provides a clear overview of the system components, including helpful details, like the lifetime of the background script.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure. Those were based on well known industry standard open source projects and even core code blocks that are written well and systematically.

AS-IS Overview

Stabull

File And Function Level Report

File: AssimilatorV2.sol

Contract: AssimilatorV2

Inherit: IAssimilator, ReentrancyGuard

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	quoteAddress	internal	Passed	All Passed	No Issue	Passed
2	getRate	public	Passed	All Passed	No Issue	Passed
3	intakeRawAndGetBalance	external	Passed	All Passed	No Issue	Passed
4	intakeRaw	external	Passed	All Passed	No Issue	Passed
5	intakeNumeraire	external	Passed	All Passed	No Issue	Passed
6	intakeNumeraireLPRatio	external	Passed	All Passed	No Issue	Passed
7	outputRawAndGetBalance	external	Passed	All Passed	No Issue	Passed
8	outputRaw	external	Passed	All Passed	No Issue	Passed
9	outputNumeraire	external	Passed	All Passed	No Issue	Passed
10	viewRawAmount	external	Passed	All Passed	No Issue	Passed

11	viewRawAmountLPRatio	external	Passed	All Passed	No Issue	Passed
12	viewNumeraireAmount	external	Passed	All Passed	No Issue	Passed
13	viewNumeraireBalance	external	Passed	All Passed	No Issue	Passed
14	viewNumeraireAmountAndBalance	external	Passed	All Passed	No Issue	Passed
15	viewNumeraireBalanceLPRatio	external	Passed	All Passed	No Issue	Passed
16	transferFee	external	Passed	All Passed	No Issue	Passed

File: AssimilatorFactory.sol

Contract: AssimilatorFactory

Inherit: IAssimilatorFactory, Ownable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setCurveFactory	external	Passed	All Passed	No Issue	Passed
2	getAssimilator	external	Passed	All Passed	No Issue	Passed
3	newAssimilator	external	Passed	All Passed	No Issue	Passed
4	revokeAssimilator	external	Passed	All Passed	No Issue	Passed

File: Assimilators.sol

Contract: Assimilators

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	delegate	internal	Passed	All Passed	No Issue	Passed
2	getRate	internal	Passed	All Passed	No Issue	Passed
3	viewRawAmount	internal	Passed	All Passed	No Issue	Passed
4	viewRawAmountLPRatio	internal	Passed	All Passed	No Issue	Passed
5	viewNumeraireAmount	internal	Passed	All Passed	No Issue	Passed
6	viewNumeraireAmountAndBalance	internal	Passed	All Passed	No Issue	Passed
7	viewNumeraireBalance	internal	Passed	All Passed	No Issue	Passed
8	viewNumeraireBalanceLPRatio	internal	Passed	All Passed	No Issue	Passed
9	intakeRaw	internal	Passed	All Passed	No Issue	Passed
10	intakeRawAndGetBalance	internal	Passed	All Passed	No Issue	Passed
11	intakeNumeraire	internal	Passed	All Passed	No Issue	Passed
12	intakeNumeraireLPRatio	internal	Passed	All Passed	No Issue	Passed
13	outputRaw	internal	Passed	All Passed	No Issue	Passed

14	outputRawAndGetBalance	internal	Passed	All Passed	No Issue	Passed
15	outputNumeraire	internal	Passed	All Passed	No Issue	Passed
16	transferFee	internal	Passed	All Passed	No Issue	Passed

File: Config.sol

Contract: Config

Inherit: Ownable, IConfig, ReentrancyGuard

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	getGlobalFrozenState	external	Passed	All Passed	No Issue	Passed
2	setGlobalFrozen	external	Passed	All Passed	No Issue	Passed
3	toggleGlobalGuarded	external	Passed	All Passed	No Issue	Passed
4	setPoolGuarded	external	Passed	All Passed	No Issue	Passed
5	setGlobalGuardAmount	external	Passed	All Passed	No Issue	Passed
6	setPoolCap	external	Passed	All Passed	No Issue	Passed
7	setPoolGuardAmount	external	Passed	All Passed	No Issue	Passed
8	isPoolGuarded	external	Passed	All Passed	No Issue	Passed

9	getPoolGuard Amount	external	Passed	All Passed	No Issue	Passed
10	setFlashable	external	Passed	All Passed	No Issue	Passed
11	updateProtoc olTreasury	external	Passed	All Passed	No Issue	Passed
12	updateProtoc olFee	external	Passed	All Passed	No Issue	Passed

File: Curve.sol

Contract: Curve

Inherit: Storage, NoDelegateCall

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setParams	external	Passed	All Passed	No Issue	Passed
2	setAssimilator	external	Passed	All Passed	No Issue	Passed
3	excludeDeriva tive	external	Passed	All Passed	No Issue	Passed
4	viewCurve	external	Passed	All Passed	No Issue	Passed
5	setEmergenc y	external	Passed	All Passed	No Issue	Passed
6	setFrozen	external	Passed	All Passed	No Issue	Passed
7	transferOwner ship	external	Passed	All Passed	No Issue	Passed
8	originSwap	external	Passed	All Passed	No Issue	Passed
9	viewOriginSw ap	external	Passed	All Passed	No Issue	Passed

10	targetSwap	external	Passed	All Passed	No Issue	Passed
11	viewTargetSwap	external	Passed	All Passed	No Issue	Passed
12	deposit	external	Passed	All Passed	No Issue	Passed
13	viewDeposit	external	Passed	All Passed	No Issue	Passed
14	emergencyWithdraw	external	Passed	All Passed	No Issue	Passed
15	withdraw	external	Passed	All Passed	No Issue	Passed
16	viewWithdraw	external	Passed	All Passed	No Issue	Passed
17	transfer	public	Passed	All Passed	No Issue	Passed
18	transferFrom	public	Passed	All Passed	No Issue	Passed
19	approve	public	Passed	All Passed	No Issue	Passed
20	flash	external	Passed	All Passed	No Issue	Passed

—-----

File: CurveFactory.sol

Contract: CurveFactory

Inherit: Ownable, ReentrancyGuard

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	getCurve	external	Passed	All Passed	No Issue	Passed
2	newCurve	external	Passed	All Passed	No Issue	Passed

File: CurveFactoryV2.sol

Contract: CurveFactoryV2

Inherit: ICurveFactory, Ownable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	getGlobalFrozenState	external	Passed	All Passed	No Issue	Passed
2	getFlashableState	external	Passed	All Passed	No Issue	Passed
3	getProtocolFee	external	Passed	All Passed	No Issue	Passed
4	getProtocolTreasury	external	Passed	All Passed	No Issue	Passed
5	isPoolGuarded	external	Passed	All Passed	No Issue	Passed
6	getPoolGuardAmount	external	Passed	All Passed	No Issue	Passed
7	getPoolCap	external	Passed	All Passed	No Issue	Passed
8	getCurve	external	Passed	All Passed	No Issue	Passed
9	newCurve	public	Passed	All Passed	No Issue	Passed
10	quoteAddress	internal	Passed	All Passed	No Issue	Passed

File: Router.sol

Contract: Router

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	viewOriginSwap	external	Passed	All Passed	No Issue	Passed
2	originSwap	public	Passed	All Passed	No Issue	Passed
3	viewTargetSwap	public	Passed	All Passed	No Issue	Passed

File: Zap.sol

Contract: Zap

Observation: Passed

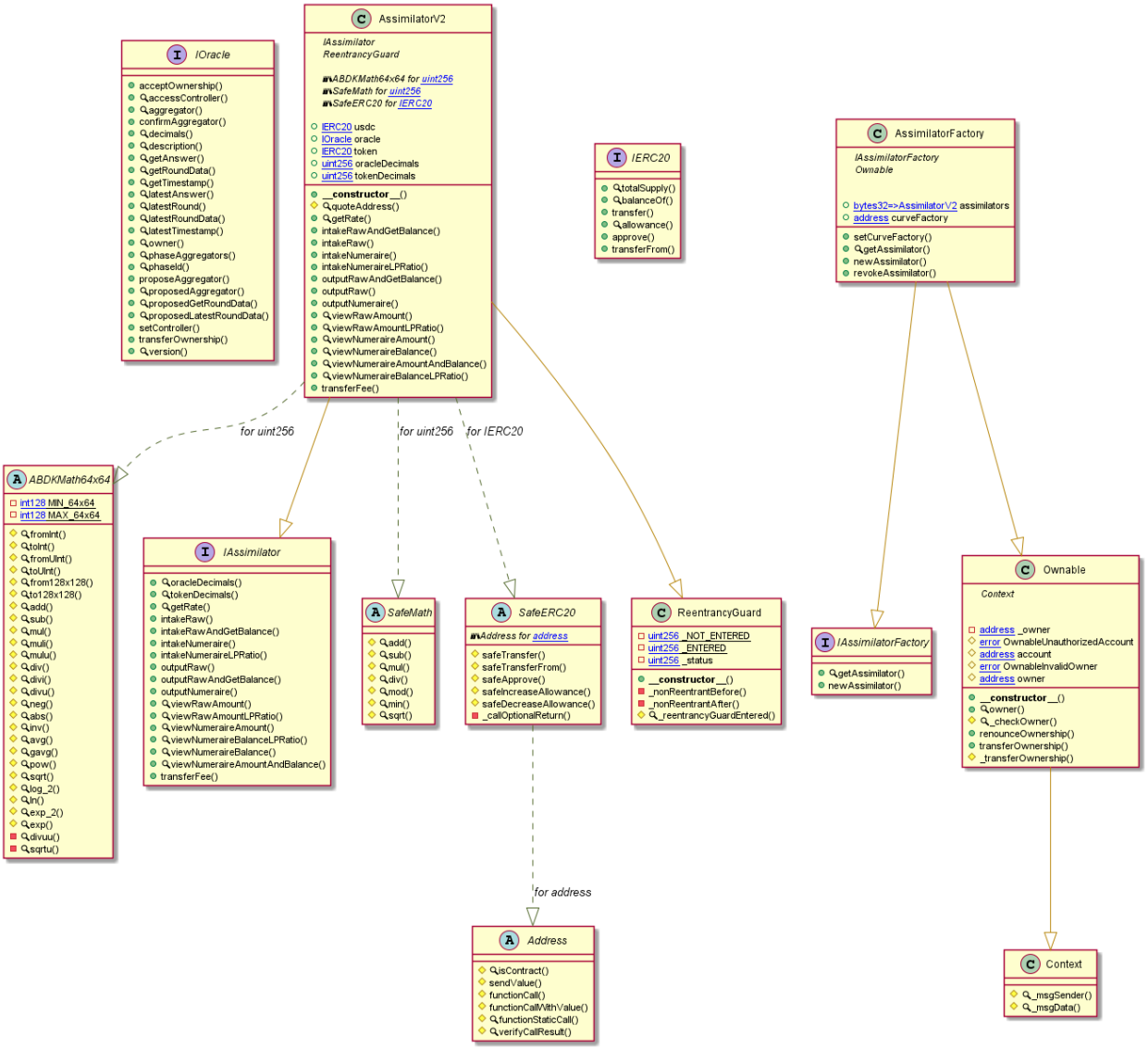
Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	quoteAddress	internal	Passed	All Passed	No Issue	Passed
2	zapFromBase	public	Passed	All Passed	No Issue	Passed
3	zapFromQuote	public	Passed	All Passed	No Issue	Passed
4	upzapFromBase	public	Passed	All Passed	No Issue	Passed
5	upzapFromQuote	public	Passed	All Passed	No Issue	Passed
6	unzap	public	Passed	All Passed	No Issue	Passed
7	zap	public	Passed	All Passed	No Issue	Passed
8	calcSwapAmountForZapFromBase	public	Passed	All Passed	No Issue	Passed

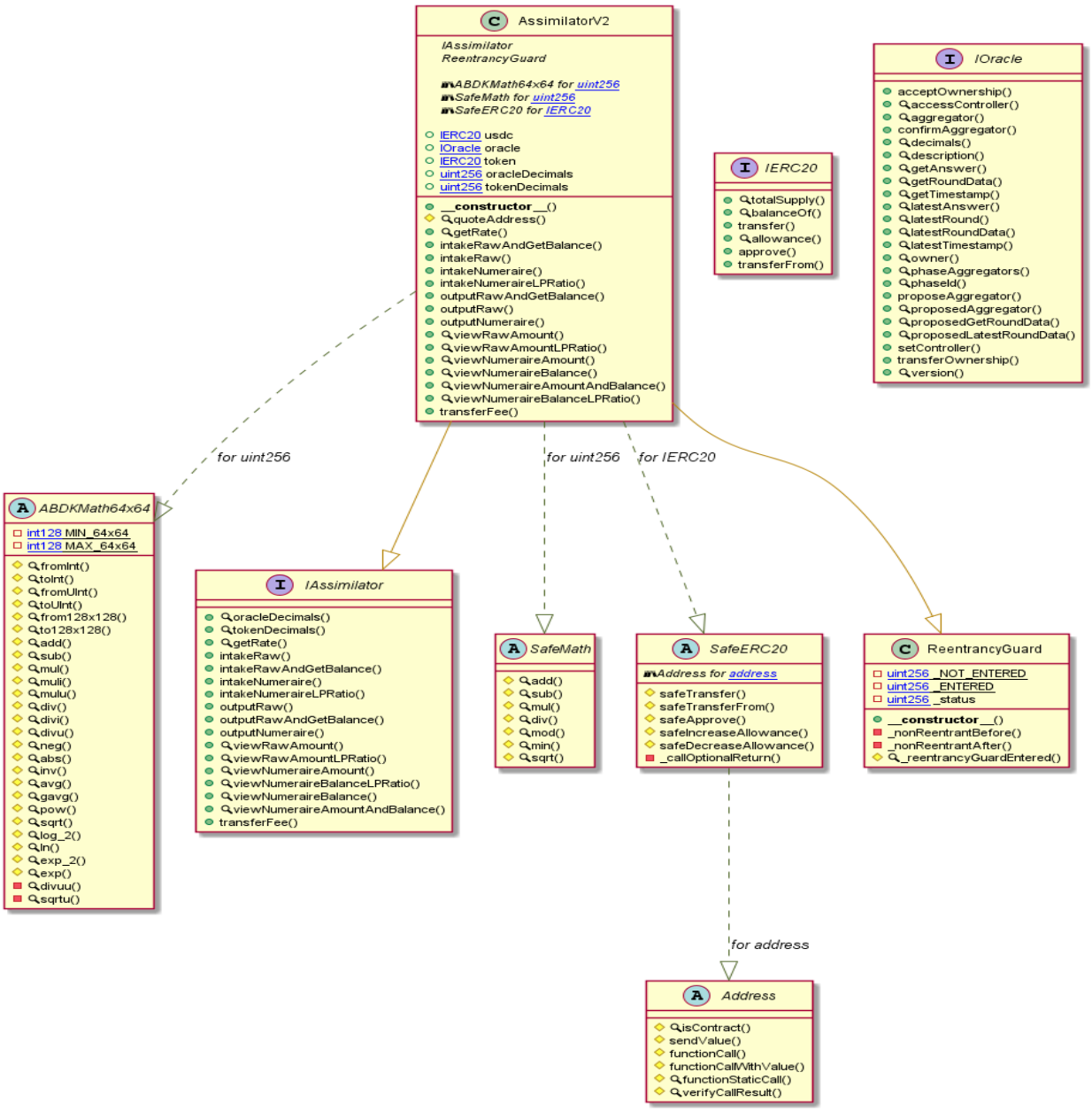
9	calcSwapAmountForZapFromQuote	public	Passed	All Passed	No Issue	Passed
10	calcSwapAmountForZap	public	Passed	All Passed	No Issue	Passed
11	calcMaxDepositAmountGivenQuote	public	Passed	All Passed	No Issue	Passed
12	calcMaxDepositAmountGivenBase	public	Passed	All Passed	No Issue	Passed
13	calcMaxBaseForDeposit	public	Passed	All Passed	No Issue	Passed
14	calcMaxQuoteForDeposit	public	Passed	All Passed	No Issue	Passed
15	_calcQuoteSwapAmount	internal	Passed	All Passed	No Issue	Passed
16	_calcBaseSwapAmount	internal	Passed	All Passed	No Issue	Passed
17	_calcDepositAmount	internal	Passed	All Passed	No Issue	Passed

Code Flow Diagram

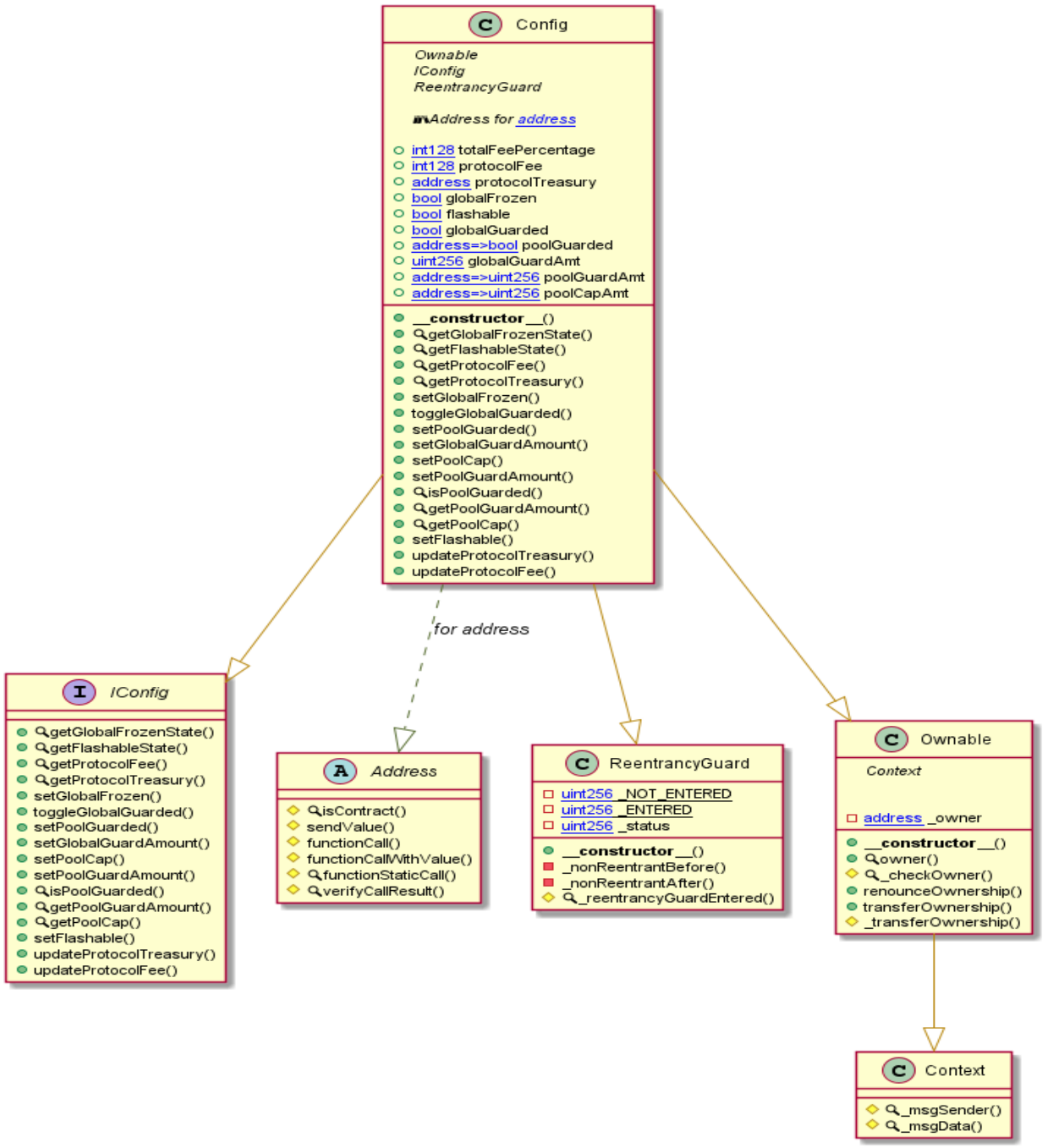
AssimilatorFactory.sol



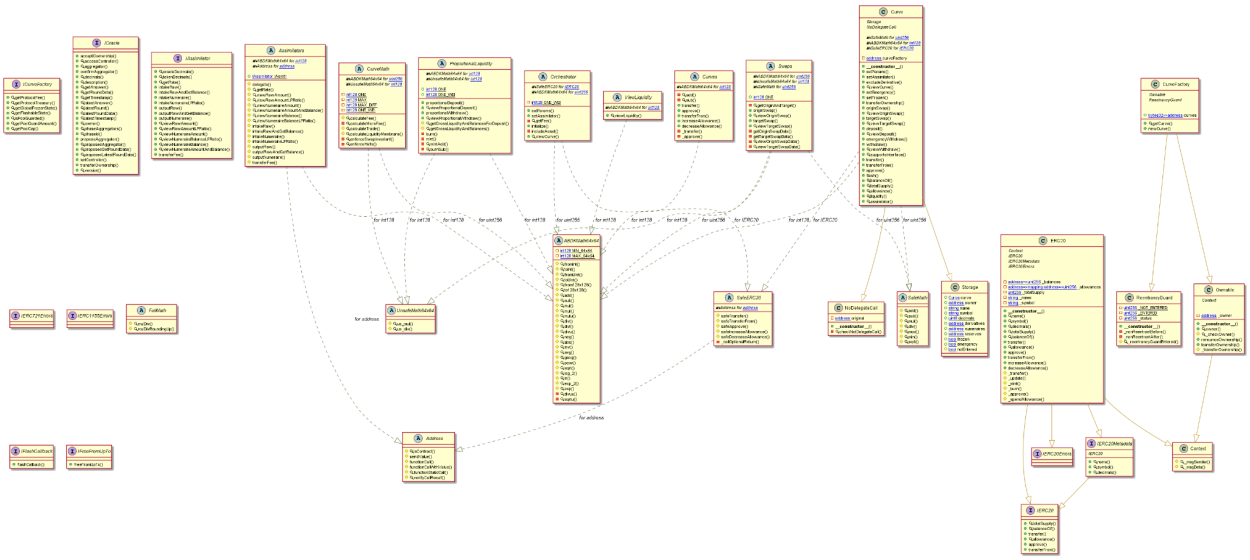
AssimilatorV2.sol



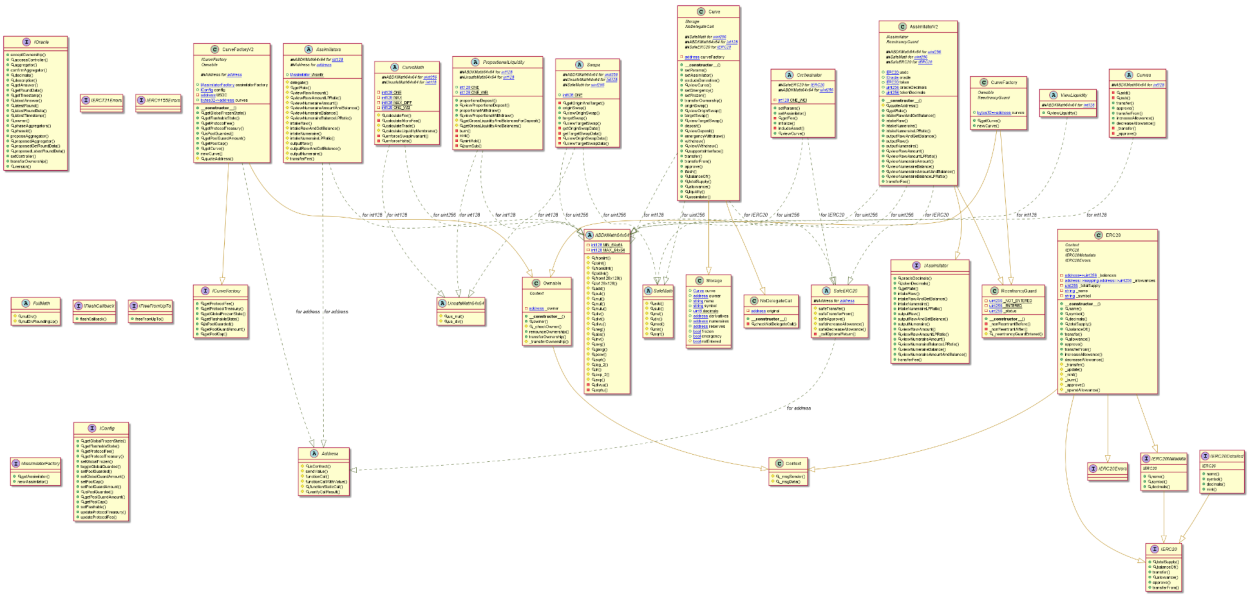
Config.sol


Config
Ownable
IConfig
ReentrancyGuard
Address for address
int128 totalFeePercentage
int128 protocolFee
address protocolTreasury
bool globalFrozen
bool flashable
bool globalGuarded
address=>bool poolGuarded
uint256 globalGuardAmt
address=>uint256 poolGuardAmt
address=>uint256 poolCapAmt
__constructor__()
getGlobalFrozenState()
getFlashableState()
getProtocolFee()
getProtocolTreasury()
setGlobalFrozen()
toggleGlobalGuarded()
setPoolGuarded()
setGlobalGuardAmount()
setPoolCap()
setPoolGuardAmount()
isPoolGuarded()
getPoolGuardAmount()
getPoolCap()
setFlashable()
updateProtocolTreasury()
updateProtocolFee()
for address
IConfig
getGlobalFrozenState()
getFlashableState()
getProtocolFee()
getProtocolTreasury()
setGlobalFrozen()
toggleGlobalGuarded()
setPoolGuarded()
setGlobalGuardAmount()
setPoolCap()
setPoolGuardAmount()
isPoolGuarded()
getPoolGuardAmount()
getPoolCap()
setFlashable()
updateProtocolTreasury()
updateProtocolFee()
Address
isContract()
sendValue()
functionCall()
functionCallWithValue()
functionStaticCall()
verifyCallResult()
ReentrancyGuard
uint256 _NOT_ENTERED
uint256 _ENTERED
uint256 _status
__constructor__()
_nonReentrantBefore()
_nonReentrantAfter()
_reentrancyGuardEntered()
Ownable
Context
address _owner
__constructor__()
owner()
_checkOwner()
renounceOwnership()
transferOwnership()
_transferOwnership()
Context
_msgSender()
_msgData()

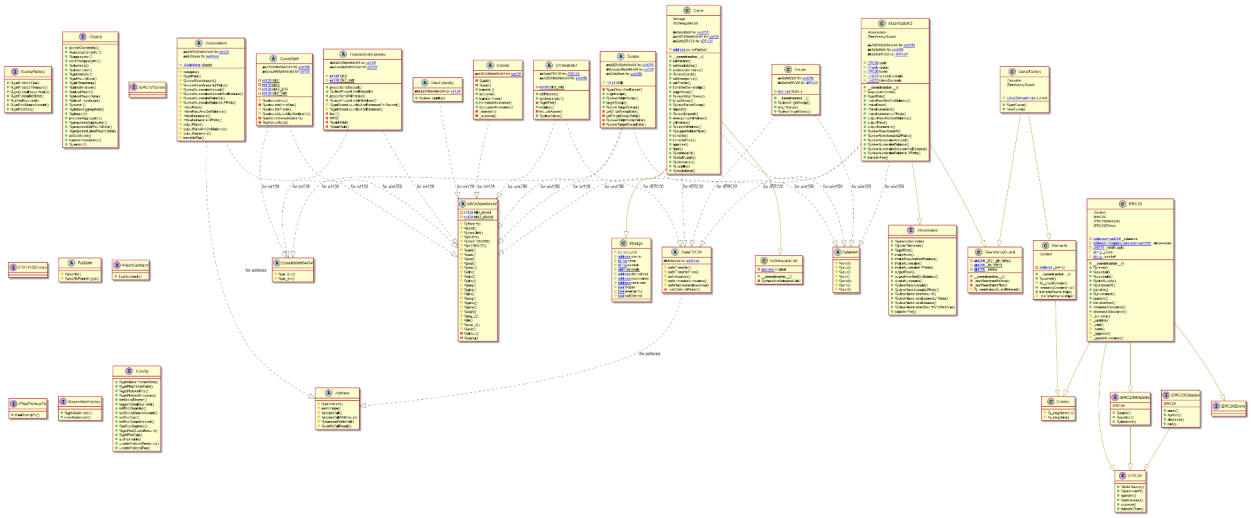
CurveFactory.sol



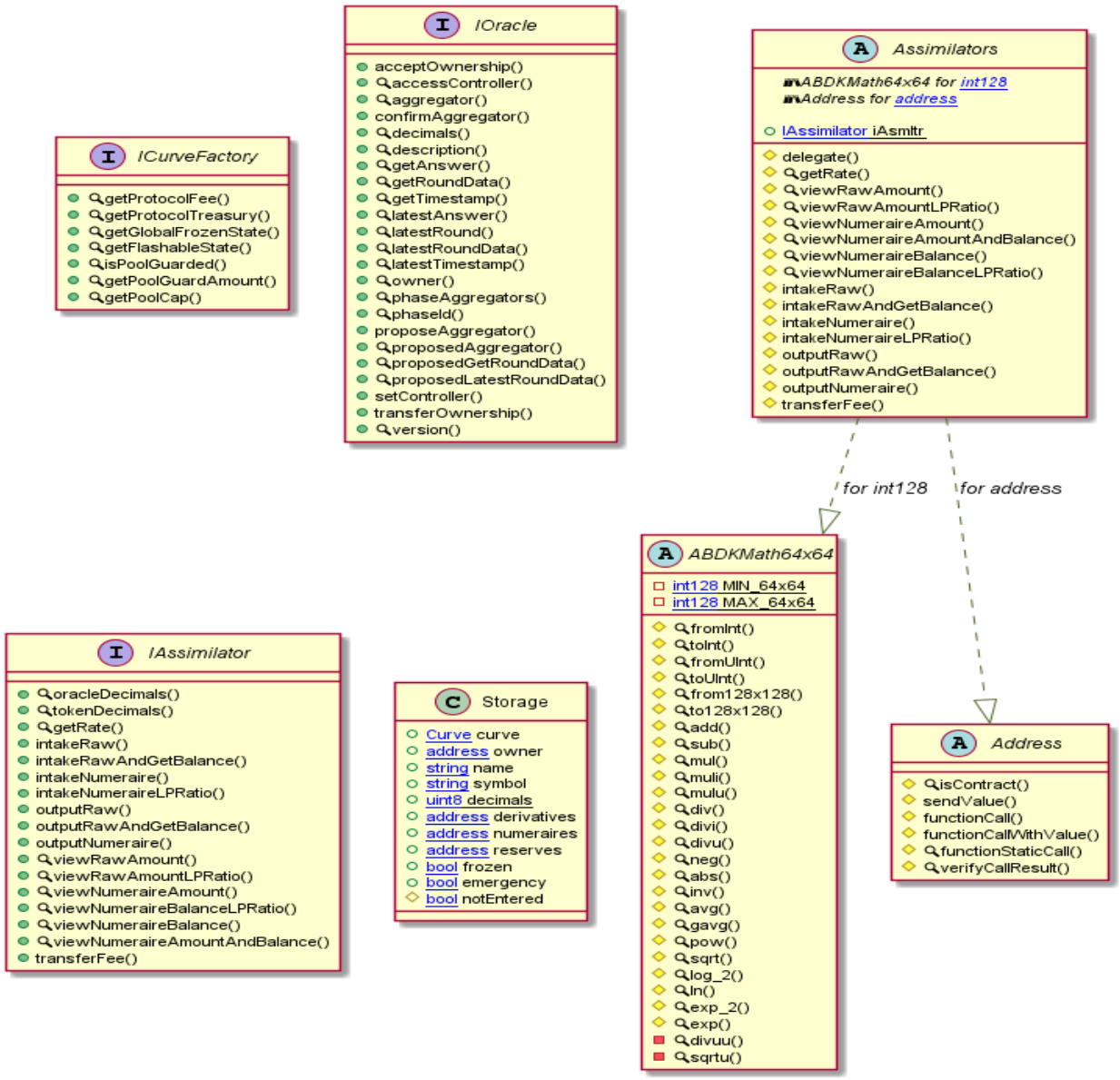
CurveFactoryV2.sol



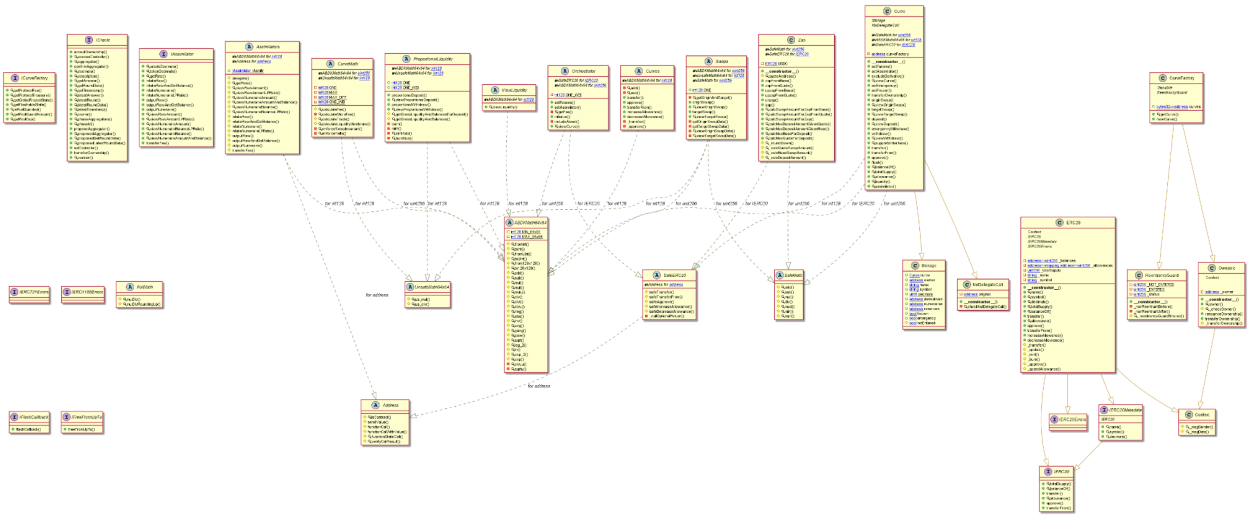
Router.sol



Storage.sol


IOracle
acceptOwnership()
accessController()
aggregator()
confirmAggregator()
decimals()
description()
getAnswer()
getRoundData()
getTimestamp()
latestAnswer()
latestRound()
latestRoundData()
latestTimestamp()
owner()
phaseAggregators()
phaseId()
proposeAggregator()
proposedAggregator()
proposedGetRoundData()
proposedLatestRoundData()
setController()
transferOwnership()
version()
ICurveFactory
getProtocolFee()
getProtocolTreasury()
getGlobalFrozenState()
getFlashableState()
isPoolGuarded()
getPoolGuardAmount()
getPoolCap()
Assimilators
ABDKMath64x64 for int128
Address for address
IAssimilator iAsmltr
delegate()
getRate()
viewRawAmount()
viewRawAmountLPRatio()
viewNumeraireAmount()
viewNumeraireAmountAndBalance()
viewNumeraireBalance()
viewNumeraireBalanceLPRatio()
intakeRaw()
intakeRawAndGetBalance()
intakeNumeraire()
intakeNumeraireLPRatio()
outputRaw()
outputRawAndGetBalance()
outputNumeraire()
transferFee()
for int128
for address
ABDKMath64x64
int128 MIN_64x64
int128 MAX_64x64
fromInt()
toInt()
fromUInt()
toUInt()
from128x128()
to128x128()
add()
sub()
mul()
muli()
mulu()
div()
divi()
divu()
neg()
abs()
inv()
avg()
gavg()
pow()
sqrt()
log_2()
ln()
exp_2()
exp()
divuu()
sqrtu()
IAssimilator
oracleDecimals()
tokenDecimals()
getRate()
intakeRaw()
intakeRawAndGetBalance()
intakeNumeraire()
intakeNumeraireLPRatio()
outputRaw()
outputRawAndGetBalance()
outputNumeraire()
viewRawAmount()
viewRawAmountLPRatio()
viewNumeraireAmount()
viewNumeraireBalanceLPRatio()
viewNumeraireBalance()
viewNumeraireAmountAndBalance()
transferFee()
Storage
Curve curve
address owner
string name
string symbol
uint8 decimals
address derivatives
address numeraires
address reserves
bool frozen
bool emergency
bool notEntered
Address
isContract()
sendValue()
functionCall()
functionCallWithValue()
functionStaticCall()
verifyCallResult()

Zap.sol



Code Flow Diagram - Slither Results Log

AssimilatorFactory.sol

```
Address.verifyCallResult(bool,bytes,string) (AssimilatorFactory.sol#1145-1165) uses assembly
        - INLINE ASM (AssimilatorFactory.sol#1157-1160)
AssimilatorV2.quoteAddress() (AssimilatorFactory.sol#1343-1359) uses assembly
        - INLINE ASM (AssimilatorFactory.sol#1345-1347)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ABDKMath64x64.abs(int128) (AssimilatorFactory.sol#273-278) is never used and should be removed
ABDKMath64x64.add(int128,int128) (AssimilatorFactory.sol#142-148) is never used and should be removed
ABDKMath64x64.avg(int128,int128) (AssimilatorFactory.sol#289-293) is never used and should be removed
ABDKMath64x64.divi(int256,int256) (AssimilatorFactory.sol#233-255) is never used and should be removed
ABDKMath64x64.exp(int128) (AssimilatorFactory.sol#610-623) is never used and should be removed
ABDKMath64x64.exp_2(int128) (AssimilatorFactory.sol#466-608) is never used and should be removed
ABDKMath64x64.from128x128(int256) (AssimilatorFactory.sol#128-134) is never used and should be removed
ABDKMath64x64.fromInt(int256) (AssimilatorFactory.sol#101-106) is never used and should be removed
ABDKMath64x64.gavg(int128,int128) (AssimilatorFactory.sol#295-305) is never used and should be removed
ABDKMath64x64.inv(int128) (AssimilatorFactory.sol#280-287) is never used and should be removed
ABDKMath64x64.ln(int128) (AssimilatorFactory.sol#452-464) is never used and should be removed
ABDKMath64x64.log_2(int128) (AssimilatorFactory.sol#407-450) is never used and should be removed
ABDKMath64x64.muli(int128,int256) (AssimilatorFactory.sol#166-200) is never used and should be removed
ABDKMath64x64.neg(int128) (AssimilatorFactory.sol#266-271) is never used and should be removed
ABDKMath64x64.pow(int128,uint256) (AssimilatorFactory.sol#307-398) is never used and should be removed
ABDKMath64x64.sqrt(int128) (AssimilatorFactory.sol#400-405) is never used and should be removed
ABDKMath64x64.sqrtu(uint256) (AssimilatorFactory.sol#683-727) is never used and should be removed
ABDKMath64x64.sub(int128,int128) (AssimilatorFactory.sol#150-156) is never used and should be removed
ABDKMath64x64.to128x128(int128) (AssimilatorFactory.sol#136-140) is never used and should be removed
ABDKMath64x64.toInt(int128) (AssimilatorFactory.sol#108-112) is never used and should be removed
ABDKMath64x64.toUInt(int128) (AssimilatorFactory.sol#121-126) is never used and should be removed
Address.functionCall(address,bytes) (AssimilatorFactory.sol#1029-1031) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (AssimilatorFactory.sol#1058-1064) is never used and should be removed
Address.functionStaticCall(address,bytes) (AssimilatorFactory.sol#1091-1093) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (AssimilatorFactory.sol#1101-1110) is never used and should be removed
Address.sendValue(address,uint256) (AssimilatorFactory.sol#1004-1009) is never used and should be removed
Context._msgData() (AssimilatorFactory.sol#1628-1630) is never used and should be removed
ReentrancyGuard._nonReentrantAfter() (AssimilatorFactory.sol#1300-1304) is never used and should be removed
ReentrancyGuard._nonReentrantBefore() (AssimilatorFactory.sol#1290-1298) is never used and should be removed
ReentrancyGuard._reentrancyGuardEntered() (AssimilatorFactory.sol#1310-1312) is never used and should be removed
```

```
Address.functionStaticCall(address,bytes,string) (AssimilatorFactory.sol#1101-1110) is never used and should be removed
Address.sendValue(address,uint256) (AssimilatorFactory.sol#1004-1009) is never used and should be removed
Context._msgData() (AssimilatorFactory.sol#1628-1630) is never used and should be removed
ReentrancyGuard._nonReentrantAfter() (AssimilatorFactory.sol#1300-1304) is never used and should be removed
ReentrancyGuard._nonReentrantBefore() (AssimilatorFactory.sol#1290-1298) is never used and should be removed
ReentrancyGuard._reentrancyGuardEntered() (AssimilatorFactory.sol#1310-1312) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (AssimilatorFactory.sol#1196-1209) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (AssimilatorFactory.sol#1220-1231) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (AssimilatorFactory.sol#1211-1218) is never used and should be removed
SafeMath.add(uint256,uint256) (AssimilatorFactory.sol#798-803) is never used and should be removed
SafeMath.div(uint256,uint256) (AssimilatorFactory.sol#831-833) is never used and should be removed
SafeMath.div(uint256,uint256,string) (AssimilatorFactory.sol#835-844) is never used and should be removed
SafeMath.min(uint256,uint256) (AssimilatorFactory.sol#859-861) is never used and should be removed
SafeMath.mod(uint256,uint256) (AssimilatorFactory.sol#846-848) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (AssimilatorFactory.sol#850-857) is never used and should be removed
SafeMath.mul(uint256,uint256) (AssimilatorFactory.sol#820-829) is never used and should be removed
SafeMath.sqrt(uint256) (AssimilatorFactory.sol#863-874) is never used and should be removed
SafeMath.sub(uint256,uint256) (AssimilatorFactory.sol#805-807) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (AssimilatorFactory.sol#809-818) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.19 (AssimilatorFactory.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.1
2/0.7.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (AssimilatorFactory.sol#1004-1009):
        - (success) = recipient.call{value: amount}() (AssimilatorFactory.sol#1007)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (AssimilatorFactory.sol#1072-1083):
        - (success,returndata) = target.call{value: value}(data) (AssimilatorFactory.sol#1081)
Low level call in Address.functionStaticCall(address,bytes,string) (AssimilatorFactory.sol#1101-1110):
        - (success,returndata) = target.staticcall(data) (AssimilatorFactory.sol#1108)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Function ABDKMath64x64.log_2(int128) (AssimilatorFactory.sol#407-450) is not in mixedCase
Function ABDKMath64x64.exp_2(int128) (AssimilatorFactory.sol#466-608) is not in mixedCase
Constant ABDKMath64x64.MIN_64x64 (AssimilatorFactory.sol#97) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ABDKMath64x64.MAX_64x64 (AssimilatorFactory.sol#99) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter AssimilatorV2.intakeRawAndGetBalance(uint256)._amount (AssimilatorFactory.sol#1368) is not in mixedCase
Parameter AssimilatorV2.intakeRaw(uint256)._amount (AssimilatorFactory.sol#1387) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraire(int128)._amount (AssimilatorFactory.sol#1400) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,uint256,address,int128)._baseWeight (As
similatorFactory.sol#1415) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,uint256,address,int128)._minBaseAmount
(AssimilatorFactory.sol#1416) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,uint256,address,int128)._maxBaseAmount
(AssimilatorFactory.sol#1417) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,uint256,address,int128)._quoteWeight (A
ssimilatorFactory.sol#1418) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,uint256,address,int128)._minQuoteAmount
 (AssimilatorFactory.sol#1419) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,uint256,address,int128)._maxQuoteAmount
 (AssimilatorFactory.sol#1420) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,uint256,address,int128)._addr (Assimila
torFactory.sol#1421) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,uint256,address,int128)._amount (Assimi
latorFactory.sol#1422) is not in mixedCase
Parameter AssimilatorV2.outputRawAndGetBalance(address,uint256)._dst (AssimilatorFactory.sol#1455) is not in mixedCase
Parameter AssimilatorV2.outputRawAndGetBalance(address,uint256)._amount (AssimilatorFactory.sol#1456) is not in mixedCase
Parameter AssimilatorV2.outputRaw(address,uint256)._dst (AssimilatorFactory.sol#1475) is not in mixedCase
Parameter AssimilatorV2.outputRaw(address,uint256)._amount (AssimilatorFactory.sol#1476) is not in mixedCase
Parameter AssimilatorV2.outputNumeraire(address,int128)._dst (AssimilatorFactory.sol#1489) is not in mixedCase
Parameter AssimilatorV2.outputNumeraire(address,int128)._amount (AssimilatorFactory.sol#1490) is not in mixedCase
Parameter AssimilatorV2.viewRawAmount(int128)._amount (AssimilatorFactory.sol#1503) is not in mixedCase
Parameter AssimilatorV2.viewRawAmountLPRatio(uint256,uint256,address,int128)._baseWeight (AssimilatorFactory.sol#1513) is not
in mixedCase
Parameter AssimilatorV2.viewRawAmountLPRatio(uint256,uint256,address,int128)._quoteWeight (AssimilatorFactory.sol#1514) is not
 in mixedCase
Parameter AssimilatorV2.viewRawAmountLPRatio(uint256,uint256,address,int128)._addr (AssimilatorFactory.sol#1515) is not in mix
edCase
```

```
Parameter AssimilatorV2.viewNumeraireBalance(address)._addr (AssimilatorFactory.sol#1547) is not in mixedCase
Parameter AssimilatorV2.viewNumeraireAmountAndBalance(address,uint256)._addr (AssimilatorFactory.sol#1562) is not in mixedCase
Parameter AssimilatorV2.viewNumeraireAmountAndBalance(address,uint256)._amount (AssimilatorFactory.sol#1563) is not in mixedCa
se
Parameter AssimilatorV2.viewNumeraireBalanceLPRatio(uint256,uint256,address)._baseWeight (AssimilatorFactory.sol#1582) is not
in mixedCase
Parameter AssimilatorV2.viewNumeraireBalanceLPRatio(uint256,uint256,address)._quoteWeight (AssimilatorFactory.sol#1583) is not
 in mixedCase
Parameter AssimilatorV2.viewNumeraireBalanceLPRatio(uint256,uint256,address)._addr (AssimilatorFactory.sol#1584) is not in mix
edCase
Parameter AssimilatorV2.transferFee(int128,address)._amount (AssimilatorFactory.sol#1600) is not in mixedCase
Parameter AssimilatorV2.transferFee(int128,address)._treasury (AssimilatorFactory.sol#1600) is not in mixedCase
Parameter AssimilatorFactory.setCurveFactory(address)._curveFactory (AssimilatorFactory.sol#1740) is not in mixedCase
Parameter AssimilatorFactory.getAssimilator(address)._token (AssimilatorFactory.sol#1750) is not in mixedCase
Parameter AssimilatorFactory.newAssimilator(IOracle,address,uint256)._oracle (AssimilatorFactory.sol#1757) is not in mixedCase
Parameter AssimilatorFactory.newAssimilator(IOracle,address,uint256)._token (AssimilatorFactory.sol#1758) is not in mixedCase
Parameter AssimilatorFactory.newAssimilator(IOracle,address,uint256)._tokenDecimals (AssimilatorFactory.sol#1759) is not in mi
xedCase
Parameter AssimilatorFactory.revokeAssimilator(address)._token (AssimilatorFactory.sol#1782) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

ABDKMath64x64.fromInt(int256) (AssimilatorFactory.sol#101-106) uses literals with too many digits:
        - require(bool)(x >= - 0x8000000000000000 && x <= 0x7FFFFFFFFFFFFFFF) (AssimilatorFactory.sol#103)
ABDKMath64x64.muli(int128,int256) (AssimilatorFactory.sol#166-200) uses literals with too many digits:
        - require(bool)(y >= - 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF && y <= 0x1000000000000000000000000000000000
000000000000000) (AssimilatorFactory.sol#169-172)
ABDKMath64x64.muli(int128,int256) (AssimilatorFactory.sol#166-200) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x8000000000000000000000000000000000000000000000000000000000000000) (AssimilatorFact
ory.sol#186-189)
ABDKMath64x64.divi(int256,int256) (AssimilatorFactory.sol#233-255) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x8000000000000000000000000000000) (AssimilatorFactory.sol#248)
ABDKMath64x64.inv(int128) (AssimilatorFactory.sol#280-287) uses literals with too many digits:
        - result = int256(0x100000000000000000000000000000000) / x (AssimilatorFactory.sol#283)
ABDKMath64x64.gavg(int128,int128) (AssimilatorFactory.sol#295-305) uses literals with too many digits:
        - require(bool)(m < 0x4000000000000000000000000000000000000000000000000000000000000000) (AssimilatorFactory.sol#299-30
2)
ABDKMath64x64.pow(int128,uint256) (AssimilatorFactory.sol#307-398) uses literals with too many digits:
```

```
ABDKMath64x64.exp_2(int128) (AssimilatorFactory.sol#466-608) uses literals with too many digits:
        - result = (result * 0x100000000000000B17217F7D1CF79E949) >> 128 (AssimilatorFactory.sol#585)
ABDKMath64x64.exp_2(int128) (AssimilatorFactory.sol#466-608) uses literals with too many digits:
        - result = (result * 0x10000000000000058B90BFBE8E7BCE544) >> 128 (AssimilatorFactory.sol#587)
ABDKMath64x64.exp_2(int128) (AssimilatorFactory.sol#466-608) uses literals with too many digits:
        - result = (result * 0x1000000000000002C5C85FDF473DE6ECA) >> 128 (AssimilatorFactory.sol#589)
ABDKMath64x64.exp_2(int128) (AssimilatorFactory.sol#466-608) uses literals with too many digits:
        - result = (result * 0x100000000000000162E42FEFA39EF366F) >> 128 (AssimilatorFactory.sol#591)
ABDKMath64x64.exp_2(int128) (AssimilatorFactory.sol#466-608) uses literals with too many digits:
        - result = (result * 0x100000000000000B17217F7D1CF79AFA) >> 128 (AssimilatorFactory.sol#593)
ABDKMath64x64.exp_2(int128) (AssimilatorFactory.sol#466-608) uses literals with too many digits:
        - result = (result * 0x10000000000000058B90BFBE8E7BCD6D) >> 128 (AssimilatorFactory.sol#595)
ABDKMath64x64.exp_2(int128) (AssimilatorFactory.sol#466-608) uses literals with too many digits:
        - result = (result * 0x1000000000000002C5C85FDF473DE6B2) >> 128 (AssimilatorFactory.sol#597)
ABDKMath64x64.exp_2(int128) (AssimilatorFactory.sol#466-608) uses literals with too many digits:
        - result = (result * 0x100000000000000162E42FEFA39EF358) >> 128 (AssimilatorFactory.sol#599)
ABDKMath64x64.exp_2(int128) (AssimilatorFactory.sol#466-608) uses literals with too many digits:
        - result = (result * 0x100000000000000B17217F7D1CF79AB) >> 128 (AssimilatorFactory.sol#601)
ABDKMath64x64.exp(int128) (AssimilatorFactory.sol#610-623) uses literals with too many digits:
        - require(bool)(x < 0x400000000000000000) (AssimilatorFactory.sol#612)
ABDKMath64x64.exp(int128) (AssimilatorFactory.sol#610-623) uses literals with too many digits:
        - x < - 0x400000000000000000 (AssimilatorFactory.sol#614)
ABDKMath64x64.divuu(uint256,uint256) (AssimilatorFactory.sol#625-681) uses literals with too many digits:
        - xc >= 0x100000000 (AssimilatorFactory.sol#636)
ABDKMath64x64.sqrtu(uint256) (AssimilatorFactory.sol#683-727) uses literals with too many digits:
        - xx >= 0x100000000000000000000000000000000 (AssimilatorFactory.sol#689)
ABDKMath64x64.sqrtu(uint256) (AssimilatorFactory.sol#683-727) uses literals with too many digits:
        - xx >= 0x10000000000000000 (AssimilatorFactory.sol#693)
ABDKMath64x64.sqrtu(uint256) (AssimilatorFactory.sol#683-727) uses literals with too many digits:
        - xx >= 0x100000000 (AssimilatorFactory.sol#697)
ABDKMath64x64.slitherConstructorConstantVariables() (AssimilatorFactory.sol#96-728) uses literals with too many digits:
        - MIN_64x64 = - 0x8000000000000000000000000000000000 (AssimilatorFactory.sol#97)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
AssimilatorFactory.sol analyzed (13 contracts with 84 detectors), 215 result(s) found
```

AssimilatorV2.sol

```
Address.verifyCallResult(bool,bytes,string) (AssimilatorV2.sol#875-893) uses assembly
        - INLINE ASM (AssimilatorV2.sol#885-888)
AssimilatorV2.quoteAddress() (AssimilatorV2.sol#1113-1129) uses assembly
        - INLINE ASM (AssimilatorV2.sol#1115-1117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ABDKMath64x64.abs(int128) (AssimilatorV2.sol#184-189) is never used and should be removed
ABDKMath64x64.add(int128,int128) (AssimilatorV2.sol#53-59) is never used and should be removed
ABDKMath64x64.avg(int128,int128) (AssimilatorV2.sol#200-204) is never used and should be removed
ABDKMath64x64.divi(int256,int256) (AssimilatorV2.sol#144-166) is never used and should be removed
ABDKMath64x64.exp(int128) (AssimilatorV2.sol#521-534) is never used and should be removed
ABDKMath64x64.exp_2(int128) (AssimilatorV2.sol#377-519) is never used and should be removed
ABDKMath64x64.from128x128(int256) (AssimilatorV2.sol#39-45) is never used and should be removed
```

```
SafeERC20.safeApprove(IERC20,address,uint256) (AssimilatorV2.sol#917-927) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (AssimilatorV2.sol#938-949) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (AssimilatorV2.sol#929-936) is never used and should be removed
SafeMath.add(uint256,uint256) (AssimilatorV2.sol#709-714) is never used and should be removed
SafeMath.div(uint256,uint256) (AssimilatorV2.sol#742-744) is never used and should be removed
SafeMath.div(uint256,uint256,string) (AssimilatorV2.sol#746-755) is never used and should be removed
SafeMath.min(uint256,uint256) (AssimilatorV2.sol#770-772) is never used and should be removed
SafeMath.mod(uint256,uint256) (AssimilatorV2.sol#757-759) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (AssimilatorV2.sol#761-768) is never used and should be removed
SafeMath.mul(uint256,uint256) (AssimilatorV2.sol#731-740) is never used and should be removed
SafeMath.sqrt(uint256) (AssimilatorV2.sol#774-785) is never used and should be removed
SafeMath.sub(uint256,uint256) (AssimilatorV2.sol#716-718) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (AssimilatorV2.sol#720-729) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.19 (AssimilatorV2.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (AssimilatorV2.sol#817-822):
        - (success) = recipient.call{value: amount}() (AssimilatorV2.sol#820)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (AssimilatorV2.sol#844-855):
        - (success,returndata) = target.call{value: value}(data) (AssimilatorV2.sol#853)
Low level call in Address.functionStaticCall(address,bytes,string) (AssimilatorV2.sol#861-870):
        - (success,returndata) = target.staticcall(data) (AssimilatorV2.sol#868)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function ABDKMath64x64.log_2(int128) (AssimilatorV2.sol#318-361) is not in mixedCase
Function ABDKMath64x64.exp_2(int128) (AssimilatorV2.sol#377-519) is not in mixedCase
Constant ABDKMath64x64.MIN_64x64 (AssimilatorV2.sol#8) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ABDKMath64x64.MAX_64x64 (AssimilatorV2.sol#10) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter AssimilatorV2.intakeRawAndGetBalance(uint256)._amount (AssimilatorV2.sol#1137) is not in mixedCase
```

```
Parameter AssimilatorV2.viewRawAmountLPRatio(uint256,uint256,address,int128)._amount (AssimilatorV2.sol#1277) is not in mixedC
ase
Parameter AssimilatorV2.viewNumeraireAmount(uint256)._amount (AssimilatorV2.sol#1293) is not in mixedCase
Parameter AssimilatorV2.viewNumeraireBalance(address)._addr (AssimilatorV2.sol#1303) is not in mixedCase
Parameter AssimilatorV2.viewNumeraireAmountAndBalance(address,uint256)._addr (AssimilatorV2.sol#1317) is not in mixedCase
Parameter AssimilatorV2.viewNumeraireAmountAndBalance(address,uint256)._amount (AssimilatorV2.sol#1318) is not in mixedCase
Parameter AssimilatorV2.viewNumeraireBalanceLPRatio(uint256,uint256,address)._baseWeight (AssimilatorV2.sol#1334) is not in mi
xedCase
Parameter AssimilatorV2.viewNumeraireBalanceLPRatio(uint256,uint256,address)._quoteWeight (AssimilatorV2.sol#1335) is not in m
ixedCase
Parameter AssimilatorV2.viewNumeraireBalanceLPRatio(uint256,uint256,address)._addr (AssimilatorV2.sol#1336) is not in mixedCas
e
Parameter AssimilatorV2.transferFee(int128,address)._amount (AssimilatorV2.sol#1351) is not in mixedCase
Parameter AssimilatorV2.transferFee(int128,address)._treasury (AssimilatorV2.sol#1351) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

ABDKMath64x64.fromInt(int256) (AssimilatorV2.sol#12-17) uses literals with too many digits:
        - require(bool)(x >= - 0x8000000000000000 && x <= 0x7FFFFFFFFFFFFFFF) (AssimilatorV2.sol#14)
ABDKMath64x64.muli(int128,int256) (AssimilatorV2.sol#77-111) uses literals with too many digits:
        - require(bool)(y >= - 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF && y <= 0x1000000000000000000000000000000000
000000000000000) (AssimilatorV2.sol#80-83)
ABDKMath64x64.muli(int128,int256) (AssimilatorV2.sol#77-111) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x8000000000000000000000000000000000000000000000000000000000000000) (AssimilatorV2.s
ol#97-100)
ABDKMath64x64.divi(int256,int256) (AssimilatorV2.sol#144-166) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x80000000000000000000000000000000) (AssimilatorV2.sol#159)
ABDKMath64x64.inv(int128) (AssimilatorV2.sol#191-198) uses literals with too many digits:
        - result = int256(0x100000000000000000000000000000000) / x (AssimilatorV2.sol#194)
```

```
ABDKMath64x64.exp_2(int128) (AssimilatorV2.sol#377-519) uses literals with too many digits:
        - result = (result * 0x10000000000000058B90BFBE8E7BCE544) >> 128 (AssimilatorV2.sol#498)
ABDKMath64x64.exp_2(int128) (AssimilatorV2.sol#377-519) uses literals with too many digits:
        - result = (result * 0x1000000000000002C5C85FDF473DE6ECA) >> 128 (AssimilatorV2.sol#500)
ABDKMath64x64.exp_2(int128) (AssimilatorV2.sol#377-519) uses literals with too many digits:
        - result = (result * 0x10000000000000162E42FEFA39EF366F) >> 128 (AssimilatorV2.sol#502)
ABDKMath64x64.exp_2(int128) (AssimilatorV2.sol#377-519) uses literals with too many digits:
        - result = (result * 0x1000000000000000B17217F7D1CF79AFA) >> 128 (AssimilatorV2.sol#504)
ABDKMath64x64.exp_2(int128) (AssimilatorV2.sol#377-519) uses literals with too many digits:
        - result = (result * 0x10000000000000058B90BFBE8E7BCD6D) >> 128 (AssimilatorV2.sol#506)
ABDKMath64x64.exp_2(int128) (AssimilatorV2.sol#377-519) uses literals with too many digits:
        - result = (result * 0x10000000000000002C5C85FDF473DE6B2) >> 128 (AssimilatorV2.sol#508)
ABDKMath64x64.exp_2(int128) (AssimilatorV2.sol#377-519) uses literals with too many digits:
        - result = (result * 0x10000000000000162E42FEFA39EF358) >> 128 (AssimilatorV2.sol#510)
ABDKMath64x64.exp_2(int128) (AssimilatorV2.sol#377-519) uses literals with too many digits:
        - result = (result * 0x10000000000000000B17217F7D1CF79AB) >> 128 (AssimilatorV2.sol#512)
ABDKMath64x64.exp(int128) (AssimilatorV2.sol#521-534) uses literals with too many digits:
        - require(bool)(x < 0x400000000000000000) (AssimilatorV2.sol#523)
ABDKMath64x64.exp(int128) (AssimilatorV2.sol#521-534) uses literals with too many digits:
        - x < - 0x400000000000000000 (AssimilatorV2.sol#525)
ABDKMath64x64.divuu(uint256,uint256) (AssimilatorV2.sol#536-592) uses literals with too many digits:
        - xc >= 0x100000000 (AssimilatorV2.sol#547)
ABDKMath64x64.sqrtu(uint256) (AssimilatorV2.sol#594-638) uses literals with too many digits:
        - xx >= 0x100000000000000000000000000000000 (AssimilatorV2.sol#600)
ABDKMath64x64.sqrtu(uint256) (AssimilatorV2.sol#594-638) uses literals with too many digits:
        - xx >= 0x10000000000000000 (AssimilatorV2.sol#604)
ABDKMath64x64.sqrtu(uint256) (AssimilatorV2.sol#594-638) uses literals with too many digits:
        - xx >= 0x100000000 (AssimilatorV2.sol#608)
ABDKMath64x64.slitherConstructorConstantVariables() (AssimilatorV2.sol#7-639) uses literals with too many digits:
        - MIN_64x64 = - 0x80000000000000000000000000000000 (AssimilatorV2.sol#8)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
AssimilatorV2.sol analyzed (9 contracts with 84 detectors), 208 result(s) found
```

Config.sol

```
Address.verifyCallResult(bool,bytes,string) (Config.sol#235-255) uses assembly
        - INLINE ASM (Config.sol#247-250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.functionCall(address,bytes) (Config.sol#119-121) is never used and should be removed
Address.functionCall(address,bytes,string) (Config.sol#129-135) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Config.sol#148-154) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Config.sol#162-173) is never used and should be removed
Address.functionStaticCall(address,bytes) (Config.sol#181-183) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (Config.sol#191-200) is never used and should be removed
Address.isContract(address) (Config.sol#70-76) is never used and should be removed
Address.sendValue(address,uint256) (Config.sol#94-99) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (Config.sol#235-255) is never used and should be removed
Context._msgData() (Config.sol#323-325) is never used and should be removed
ReentrancyGuard._reentrancyGuardEntered() (Config.sol#314-316) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.19 (Config.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.
16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Config.sol#94-99):
        - (success) = recipient.call{value: amount}() (Config.sol#97)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Config.sol#162-173):
        - (success,returndata) = target.call{value: value}(data) (Config.sol#171)
Low level call in Address.functionStaticCall(address,bytes,string) (Config.sol#191-200):
        - (success,returndata) = target.staticcall(data) (Config.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Parameter Config.setGlobalFrozen(bool)._toFreezeOrNotToFreeze (Config.sol#466) is not in mixedCase
Parameter Config.setFlashable(bool)._toFlashOrNotToFlash (Config.sol#540) is not in mixedCase
Parameter Config.updateProtocolTreasury(address)._newTreasury (Config.sol#548) is not in mixedCase
Parameter Config.updateProtocolFee(int128)._newFee (Config.sol#560) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Config.slitherConstructorVariables() (Config.sol#398-570) uses literals with too many digits:
        - totalFeePercentage = 100000 (Config.sol#402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Config.totalFeePercentage (Config.sol#402) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
Config.sol analyzed (6 contracts with 84 detectors), 23 result(s) found
```

CurveFactory.sol

```
Address.functionCallWithValue(address,bytes,uint256,string) (CurveFactory.sol#1177-1188) has external calls inside a loop: (su
ccess,returndata) = target.call{value: value}(data) (CurveFactory.sol#1186)
SafeERC20.safeApprove(IERC20,address,uint256) (CurveFactory.sol#1859-1872) has external calls inside a loop: require(bool,stri
ng)((value == 0) || (token.allowance(address(this),spender) == 0),SafeERC20: approve from non-zero to non-zero allowance) (Cur
veFactory.sol#1867-1870)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in Orchestrator.includeAsset(Storage.Curve,address,address,address,address,address,uint256) (CurveFactory.sol#2553-
2626):
        External calls:
        - IERC20(_numeraire).safeApprove(_reserveApproveTo,type()(uint256).max) (CurveFactory.sol#2585)
        Event emitted after the call(s):
        - AssetIncluded(_numeraire,_reserve,_weight) (CurveFactory.sol#2609)
        - AssimilatorIncluded(_numeraire,_numeraire,_reserve,_numeraireAssim) (CurveFactory.sol#2611-2616)
        - AssimilatorIncluded(_reserve,_numeraire,_reserve,_reserveAssim) (CurveFactory.sol#2619-2624)
Reentrancy in Swaps.originSwap(Storage.Curve,OriginSwapData) (CurveFactory.sol#3238-3313):
        External calls:
        - (_amt,_oGLiq,_nGLiq,_oBals,_nBals) = getOriginSwapData(curve,_o.ix,_t.ix,_o.addr,_swapData._originAmount) (CurveFact
ory.sol#3256-3268)
                - (_success,returnData_) = _callee.delegatecall(_data) (CurveFactory.sol#1242)
                - (amt_,_bal) = Assimilators.intakeRawAndGetBalance(_assim,_amt) (CurveFactory.sol#3504-3507)
        - Assimilators.transferFee(_t.addr,_swapInfo.amountToTreasury,_swapInfo.treasury) (CurveFactory.sol#3294-3298)
        - tAmt_ = Assimilators.outputNumeraire(_t.addr,_swapData._recipient,_swapInfo.amountToUser) (CurveFactory.sol#3299-330
3)
        Event emitted after the call(s):
        - Trade(msg.sender,_swapData._origin,_swapData._target,_swapData._originAmount,tAmt_,_swapInfo.amountToTreasury) (Curv
eFactory.sol#3305-3312)
Reentrancy in ProportionalLiquidity.proportionalDeposit(Storage.Curve,DepositData) (CurveFactory.sol#1440-1508):
        External calls:
        - deposits_[i] = Assimilators.intakeNumeraire(curve.assets[i].addr,_d.add(ONE_WEI)) (CurveFactory.sol#1463-1466)
        - deposits_[i_scope_0] = Assimilators.intakeNumeraireLPRatio(curve.assets[i_scope_0].addr,info) (CurveFactory.sol#1485
-1488)
```

```
Reentrancy in ProportionalLiquidity.proportionalWithdraw(Storage.Curve,uint256) (CurveFactory.sol#1566-1592):
        External calls:
        - withdrawals_[i] = Assimilators.outputNumeraire(curve.assets[i].addr,msg.sender,_oBals[i].mul(_multiplier)) (CurveFac
tory.sol#1582-1586)
        Event emitted after the call(s):
        - Transfer(msg.sender,address(0),amount) (CurveFactory.sol#1669)
                - burn(curve,msg.sender,_withdrawal) (CurveFactory.sol#1589)
Reentrancy in Swaps.targetSwap(Storage.Curve,TargetSwapData) (CurveFactory.sol#3356-3432):
        External calls:
        - (_amt,_oGLiq,_nGLiq,_oBals,_nBals) = getTargetSwapData(curve,_t.ix,_o.ix,_t.addr,_swapData._recipient,_swapData._tar
getAmount) (CurveFactory.sol#3376-3389)
                - (_success,returnData_) = _callee.delegatecall(_data) (CurveFactory.sol#1242)
                - (amt_,_bal) = Assimilators.outputRawAndGetBalance(_assim,_recipient,_amt) (CurveFactory.sol#3553-3557)
        - Assimilators.transferFee(_o.addr,_swapInfo.amountToTreasury,_swapInfo.treasury) (CurveFactory.sol#3416-3420)
        - oAmt_ = Assimilators.intakeNumeraire(_o.addr,_swapInfo.amountToUser) (CurveFactory.sol#3422)
        Event emitted after the call(s):
        - Trade(msg.sender,_swapData._origin,_swapData._target,oAmt_,_swapData._targetAmount,_swapInfo.amountToTreasury) (Curv
eFactory.sol#3424-3431)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.verifyCallResult(bool,bytes,string) (CurveFactory.sol#1208-1226) uses assembly
        - INLINE ASM (CurveFactory.sol#1218-1221)
Assimilators.delegate(address,bytes) (CurveFactory.sol#1236-1251) uses assembly
        - INLINE ASM (CurveFactory.sol#1244-1248)
FullMath.mulDiv(uint256,uint256,uint256) (CurveFactory.sol#2672-2728) uses assembly
        - INLINE ASM (CurveFactory.sol#2679-2683)
        - INLINE ASM (CurveFactory.sol#2687-2689)
        - INLINE ASM (CurveFactory.sol#2697-2699)
        - INLINE ASM (CurveFactory.sol#2700-2703)
        - INLINE ASM (CurveFactory.sol#2706-2708)
        - INLINE ASM (CurveFactory.sol#2710-2712)
        - INLINE ASM (CurveFactory.sol#2713-2715)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
SafeMath.div(uint256,uint256,string) (CurveFactory.sol#1981-1990) is never used and should be removed
SafeMath.min(uint256,uint256) (CurveFactory.sol#2005-2007) is never used and should be removed
SafeMath.mod(uint256,uint256) (CurveFactory.sol#1992-1994) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (CurveFactory.sol#1996-2003) is never used and should be removed
SafeMath.mul(uint256,uint256) (CurveFactory.sol#1966-1975) is never used and should be removed
SafeMath.sqrt(uint256) (CurveFactory.sol#2009-2020) is never used and should be removed
SafeMath.sub(uint256,uint256) (CurveFactory.sol#1951-1953) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (CurveFactory.sol#1955-1964) is never used and should be removed
UnsafeMath64x64.us_div(int128,int128) (CurveFactory.sol#847-850) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.19 (CurveFactory.sol#16) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (CurveFactory.sol#1150-1155):
        - (success) = recipient.call{value: amount}() (CurveFactory.sol#1153)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (CurveFactory.sol#1177-1188):
        - (success,returndata) = target.call{value: value}(data) (CurveFactory.sol#1186)
Low level call in Address.functionStaticCall(address,bytes,string) (CurveFactory.sol#1194-1203):
        - (success,returndata) = target.staticcall(data) (CurveFactory.sol#1201)
Low level call in Assimilators.delegate(address,bytes) (CurveFactory.sol#1236-1251):
        - (_success,returnData_) = _callee.delegatecall(_data) (CurveFactory.sol#1242)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Curve (CurveFactory.sol#3669-4335) should inherit from IERC20 (CurveFactory.sol#1756-1829)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
```

```
Variable Curve.viewTargetSwap(address,address,uint256)._targetAmount (CurveFactory.sol#4041) is too similar to Curve.originSwa
p(address,address,uint256,uint256,uint256).targetAmount_ (CurveFactory.sol#3954)
Variable Curve.targetSwap(address,address,uint256,uint256,uint256)._targetAmount (CurveFactory.sol#4006) is too similar to Cur
ve.originSwap(address,address,uint256,uint256,uint256).targetAmount_ (CurveFactory.sol#3954)
Variable Curve.targetSwap(address,address,uint256,uint256,uint256)._targetAmount (CurveFactory.sol#4006) is too similar to Cur
ve.viewOriginSwap(address,address,uint256).targetAmount_ (CurveFactory.sol#3985)
Variable Curve.viewTargetSwap(address,address,uint256)._targetAmount (CurveFactory.sol#4041) is too similar to Curve.viewOrigi
nSwap(address,address,uint256).targetAmount_ (CurveFactory.sol#3985)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

ABDKMath64x64.fromInt(int256) (CurveFactory.sol#193-198) uses literals with too many digits:
        - require(bool)(x >= - 0x8000000000000000 && x <= 0x7FFFFFFFFFFFFFFF) (CurveFactory.sol#195)
ABDKMath64x64.muli(int128,int256) (CurveFactory.sol#258-292) uses literals with too many digits:
        - require(bool)(y >= - 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF && y <= 0x1000000000000000000000000000000000
000000000000000) (CurveFactory.sol#261-264)
ABDKMath64x64.muli(int128,int256) (CurveFactory.sol#258-292) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x8000000000000000000000000000000000000000000000000000000000000000) (CurveFactory.so
l#278-281)
ABDKMath64x64.divi(int256,int256) (CurveFactory.sol#325-347) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x8000000000000000000000000000000) (CurveFactory.sol#340)
ABDKMath64x64.inv(int128) (CurveFactory.sol#372-379) uses literals with too many digits:
        - result = int256(0x100000000000000000000000000000000) / x (CurveFactory.sol#375)
ABDKMath64x64.gavg(int128,int128) (CurveFactory.sol#387-397) uses literals with too many digits:
        - require(bool)(m < 0x4000000000000000000000000000000000000000000000000000000000000000) (CurveFactory.sol#391-394)
ABDKMath64x64.pow(int128,uint256) (CurveFactory.sol#399-490) uses literals with too many digits:
        - absResult = 0x100000000000000000000000000000000 (CurveFactory.sol#405)
ABDKMath64x64.pow(int128,uint256) (CurveFactory.sol#399-490) uses literals with too many digits:
        - absX <= 0x10000000000000000 (CurveFactory.sol#407)
ABDKMath64x64.pow(int128,uint256) (CurveFactory.sol#399-490) uses literals with too many digits:
        - absX < 0x1000000000000000000000000 (CurveFactory.sol#436)
ABDKMath64x64.pow(int128,uint256) (CurveFactory.sol#399-490) uses literals with too many digits:
        - absX < 0x100000000000000000000000000000 (CurveFactory.sol#440)
ABDKMath64x64.pow(int128,uint256) (CurveFactory.sol#399-490) uses literals with too many digits:
        - absX < 0x100000000000000000000000000000000 (CurveFactory.sol#444)
ABDKMath64x64.pow(int128,uint256) (CurveFactory.sol#399-490) uses literals with too many digits:
        - absX < 0x1000000000000000000000000000000000 (CurveFactory.sol#448)
```

```
ABDKMath64x64.sqrtu(uint256) (CurveFactory.sol#775-819) uses literals with too many digits:
        - xx >= 0x100000000 (CurveFactory.sol#789)
ABDKMath64x64.slitherConstructorConstantVariables() (CurveFactory.sol#188-820) uses literals with too many digits:
        - MIN_64x64 = - 0x8000000000000000000000000000000000 (CurveFactory.sol#189)
CurveMath.slitherConstructorConstantVariables() (CurveFactory.sol#854-1077) uses literals with too many digits:
        - ONE = 0x10000000000000000 (CurveFactory.sol#855)
CurveMath.slitherConstructorConstantVariables() (CurveFactory.sol#854-1077) uses literals with too many digits:
        - MAX = 0x4000000000000000 (CurveFactory.sol#856)
ProportionalLiquidity.slitherConstructorConstantVariables() (CurveFactory.sol#1430-1712) uses literals with too many digits:
        - ONE = 0x10000000000000000 (CurveFactory.sol#1437)
Swaps.originSwap(Storage.Curve,OriginSwapData) (CurveFactory.sol#3238-3313) uses literals with too many digits:
        - _swapInfo.amountToTreasury = _swapInfo.totalFee.muli(_swapInfo.protocolFeePercentage).divi(100000) (CurveFactory.sol
#3290-3293)
Swaps.targetSwap(Storage.Curve,TargetSwapData) (CurveFactory.sol#3356-3432) uses literals with too many digits:
        - _swapInfo.amountToTreasury = _swapInfo.totalFee.muli(_swapInfo.protocolFeePercentage).divi(100000) (CurveFactory.sol
#3411-3414)
Swaps.slitherConstructorConstantVariables() (CurveFactory.sol#3202-3667) uses literals with too many digits:
        - ONE = 0x10000000000000000 (CurveFactory.sol#3218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

CurveMath.ONE_WEI (CurveFactory.sol#858) is never used in CurveMath (CurveFactory.sol#854-1077)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

Curve.curveFactory (CurveFactory.sol#3674) should be immutable
ERC20._name (CurveFactory.sol#2057) should be immutable
ERC20._symbol (CurveFactory.sol#2058) should be immutable
Storage.name (CurveFactory.sol#1740) should be immutable
Storage.symbol (CurveFactory.sol#1741) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
CurveFactory.sol analyzed (31 contracts with 84 detectors), 408 result(s) found
root@server:/chetan/gaza/mycontracts/Aki/ve/p2# slither CurveFactoryV2.sol
```

CurveFactoryV2.sol

```
Address.functionCallWithValue(address,bytes,uint256,string) (CurveFactoryV2.sol#1179-1190) has external calls inside a loop: (
success,returndata) = target.call{value: value}(data) (CurveFactoryV2.sol#1188)
SafeERC20.safeApprove(IERC20,address,uint256) (CurveFactoryV2.sol#1861-1874) has external calls inside a loop: require(bool,st
ring)((value == 0) || (token.allowance(address(this),spender) == 0),SafeERC20: approve from non-zero to non-zero allowance) (C
urveFactoryV2.sol#1869-1872)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in Orchestrator.includeAsset(Storage.Curve,address,address,address,address,address,uint256) (CurveFactoryV2.sol#255
7-2630):
        External calls:
        - IERC20(_numeraire).safeApprove(_reserveApproveTo,type()(uint256).max) (CurveFactoryV2.sol#2589)
        Event emitted after the call(s):
        - AssetIncluded(_numeraire,_reserve,_weight) (CurveFactoryV2.sol#2613)
        - AssimilatorIncluded(_numeraire,_numeraire,_reserve,_numeraireAssim) (CurveFactoryV2.sol#2615-2620)
        - AssimilatorIncluded(_reserve,_numeraire,_reserve,_reserveAssim) (CurveFactoryV2.sol#2623-2628)
Reentrancy in CurveFactoryV2.newCurve(CurveInfo) (CurveFactoryV2.sol#4775-4860):
        External calls:
        - quoteDec = IERC20Detailed(_info._quoteCurrency).decimals() (CurveFactoryV2.sol#4790)
        - baseDec = IERC20Detailed(_info._baseCurrency).decimals() (CurveFactoryV2.sol#4791)
        - _baseAssim = (assimilatorFactory.newAssimilator(_info._baseOracle,_info._baseCurrency,baseDec)) (CurveFactoryV2.sol#
4800-4806)
        - _quoteAssim = (assimilatorFactory.newAssimilator(_info._quoteOracle,_info._quoteCurrency,quoteDec)) (CurveFactoryV2.
sol#4810-4816)
        - curve = new Curve(_info._name,_info._symbol,_assets,_assetWeights,address(this)) (CurveFactoryV2.sol#4840-4846)
        - curve.setParams(_info._alpha,_info._beta,_info._feeAtHalt,_info._epsilon,_info._lambda) (CurveFactoryV2.sol#4847-485
3)
        - curve.transferOwnership(getProtocolTreasury()) (CurveFactoryV2.sol#4854)
        Event emitted after the call(s):
        - NewCurve(msg.sender,curveId,address(curve)) (CurveFactoryV2.sol#4857)
Reentrancy in Swaps.originSwap(Storage.Curve,OriginSwapData) (CurveFactoryV2.sol#3239-3314):
        External calls:
        - (_amt,_oGLiq,_nGLiq,_oBals,_nBals) = getOriginSwapData(curve,_o.ix,_t.ix,_o.addr,_swapData._originAmount) (CurveFact
oryV2.sol#3257-3269)
                - (_success,returnData_) = _callee.delegatecall(_data) (CurveFactoryV2.sol#1244)
                - (amt_,_bal) = Assimilators.intakeRawAndGetBalance(_assim,_amt) (CurveFactoryV2.sol#3505-3508)
```

```
Reentrancy in ProportionalLiquidity.proportionalWithdraw(Storage.Curve,uint256) (CurveFactoryV2.sol#1569-1595):
        External calls:
        - withdrawals_[i] = Assimilators.outputNumeraire(curve.assets[i].addr,msg.sender,_oBals[i].mul(_multiplier)) (CurveFac
toryV2.sol#1585-1589)
        Event emitted after the call(s):
        - Transfer(msg.sender,address(0),amount) (CurveFactoryV2.sol#1672)
                - burn(curve,msg.sender,_withdrawal) (CurveFactoryV2.sol#1592)
Reentrancy in Swaps.targetSwap(Storage.Curve,TargetSwapData) (CurveFactoryV2.sol#3357-3433):
        External calls:
        - (_amt,_oGLiq,_nGLiq,_oBals,_nBals) = getTargetSwapData(curve,_t.ix,_o.ix,_t.addr,_swapData._recipient,_swapData._tar
getAmount) (CurveFactoryV2.sol#3377-3390)
                - (_success,returnData_) = _callee.delegatecall(_data) (CurveFactoryV2.sol#1244)
                - (amt_,_bal) = Assimilators.outputRawAndGetBalance(_assim,_recipient,_amt) (CurveFactoryV2.sol#3554-3558)
        - Assimilators.transferFee(_o.addr,_swapInfo.amountToTreasury,_swapInfo.treasury) (CurveFactoryV2.sol#3417-3421)
        - oAmt_ = Assimilators.intakeNumeraire(_o.addr,_swapInfo.amountToUser) (CurveFactoryV2.sol#3423)
        Event emitted after the call(s):
        - Trade(msg.sender,_swapData._origin,_swapData._target,oAmt_,_swapData._targetAmount,_swapInfo.amountToTreasury) (Curv
eFactoryV2.sol#3425-3432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.verifyCallResult(bool,bytes,string) (CurveFactoryV2.sol#1210-1228) uses assembly
        - INLINE ASM (CurveFactoryV2.sol#1220-1223)
Assimilators.delegate(address,bytes) (CurveFactoryV2.sol#1238-1253) uses assembly
        - INLINE ASM (CurveFactoryV2.sol#1246-1250)
FullMath.mulDiv(uint256,uint256,uint256) (CurveFactoryV2.sol#2676-2732) uses assembly
        - INLINE ASM (CurveFactoryV2.sol#2683-2687)
        - INLINE ASM (CurveFactoryV2.sol#2691-2693)
        - INLINE ASM (CurveFactoryV2.sol#2701-2703)
        - INLINE ASM (CurveFactoryV2.sol#2704-2707)
        - INLINE ASM (CurveFactoryV2.sol#2710-2712)
        - INLINE ASM (CurveFactoryV2.sol#2714-2716)
        - INLINE ASM (CurveFactoryV2.sol#2717-2719)
AssimilatorV2.quoteAddress() (CurveFactoryV2.sol#4367-4383) uses assembly
        - INLINE ASM (CurveFactoryV2.sol#4369-4371)
CurveFactoryV2.quoteAddress() (CurveFactoryV2.sol#4862-4880) uses assembly
        - INLINE ASM (CurveFactoryV2.sol#4865-4867)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version0.8.19 (CurveFactoryV2.sol#16) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0
.7.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (CurveFactoryV2.sol#1152-1157):
        - (success) = recipient.call{value: amount}() (CurveFactoryV2.sol#1155)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (CurveFactoryV2.sol#1179-1190):
        - (success,returndata) = target.call{value: value}(data) (CurveFactoryV2.sol#1188)
Low level call in Address.functionStaticCall(address,bytes,string) (CurveFactoryV2.sol#1196-1205):
        - (success,returndata) = target.staticcall(data) (CurveFactoryV2.sol#1203)
Low level call in Assimilators.delegate(address,bytes) (CurveFactoryV2.sol#1238-1253):
        - (_success,returnData_) = _callee.delegatecall(_data) (CurveFactoryV2.sol#1244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Curve (CurveFactoryV2.sol#3670-4336) should inherit from IERC20 (CurveFactoryV2.sol#1758-1831)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Function ABDKMath64x64.log_2(int128) (CurveFactoryV2.sol#499-542) is not in mixedCase
Function ABDKMath64x64.exp_2(int128) (CurveFactoryV2.sol#558-700) is not in mixedCase
Constant ABDKMath64x64.MIN_64x64 (CurveFactoryV2.sol#189) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ABDKMath64x64.MAX_64x64 (CurveFactoryV2.sol#191) is not in UPPER_CASE_WITH_UNDERSCORES
Function UnsafeMath64x64.us_mul(int128,int128) (CurveFactoryV2.sol#833-836) is not in mixedCase
Function UnsafeMath64x64.us_div(int128,int128) (CurveFactoryV2.sol#847-850) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],Storage.Curve,int128[])._gLiq (CurveFactoryV2.sol#866) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],Storage.Curve,int128[])._bals (CurveFactoryV2.sol#867) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],Storage.Curve,int128[])._weights (CurveFactoryV2.sol#869) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._gLiq (CurveFactoryV2.sol#878) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._bals (CurveFactoryV2.sol#879) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._beta (CurveFactoryV2.sol#880) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._delta (CurveFactoryV2.sol#881) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._weights (CurveFactoryV2.sol#882) is not in mixedCase
Parameter CurveMath.calculateMicroFee(int128,int128,int128,int128)._bal (CurveFactoryV2.sol#893) is not in mixedCase
Parameter CurveMath.calculateMicroFee(int128,int128,int128,int128)._ideal (CurveFactoryV2.sol#894) is not in mixedCase
Parameter CurveMath.calculateMicroFee(int128,int128,int128,int128)._beta (CurveFactoryV2.sol#895) is not in mixedCase
```

```
Parameter Curve.deposit(uint256,uint256,uint256,uint256,uint256,uint256)._minQuoteAmount (CurveFactoryV2.sol#4065) is not in m
ixedCase
Parameter Curve.deposit(uint256,uint256,uint256,uint256,uint256,uint256)._minBaseAmount (CurveFactoryV2.sol#4066) is not in mi
xedCase
Parameter Curve.deposit(uint256,uint256,uint256,uint256,uint256,uint256)._maxQuoteAmount (CurveFactoryV2.sol#4067) is not in m
ixedCase
Parameter Curve.deposit(uint256,uint256,uint256,uint256,uint256,uint256)._maxBaseAmount (CurveFactoryV2.sol#4068) is not in mi
xedCase
Parameter Curve.deposit(uint256,uint256,uint256,uint256,uint256,uint256)._deadline (CurveFactoryV2.sol#4069) is not in mixedCa
se
Parameter Curve.viewDeposit(uint256)._deposit (CurveFactoryV2.sol#4103) is not in mixedCase
Parameter Curve.emergencyWithdraw(uint256,uint256)._curvesToBurn (CurveFactoryV2.sol#4121) is not in mixedCase
Parameter Curve.emergencyWithdraw(uint256,uint256)._deadline (CurveFactoryV2.sol#4122) is not in mixedCase
Parameter Curve.withdraw(uint256,uint256)._curvesToBurn (CurveFactoryV2.sol#4139) is not in mixedCase
Parameter Curve.withdraw(uint256,uint256)._deadline (CurveFactoryV2.sol#4140) is not in mixedCase
Parameter Curve.viewWithdraw(uint256)._curvesToBurn (CurveFactoryV2.sol#4157) is not in mixedCase
Parameter Curve.supportsInterface(bytes4)._interface (CurveFactoryV2.sol#4173) is not in mixedCase
Parameter Curve.transfer(address,uint256)._recipient (CurveFactoryV2.sol#4186) is not in mixedCase
Parameter Curve.transfer(address,uint256)._amount (CurveFactoryV2.sol#4187) is not in mixedCase
Parameter Curve.transferFrom(address,address,uint256)._sender (CurveFactoryV2.sol#4204) is not in mixedCase
Parameter Curve.transferFrom(address,address,uint256)._recipient (CurveFactoryV2.sol#4205) is not in mixedCase
Parameter Curve.transferFrom(address,address,uint256)._amount (CurveFactoryV2.sol#4206) is not in mixedCase
Parameter Curve.approve(address,uint256)._spender (CurveFactoryV2.sol#4222) is not in mixedCase
Parameter Curve.approve(address,uint256)._amount (CurveFactoryV2.sol#4223) is not in mixedCase
Parameter Curve.balanceOf(address)._account (CurveFactoryV2.sol#4296) is not in mixedCase
Parameter Curve.allowance(address,address)._owner (CurveFactoryV2.sol#4312) is not in mixedCase
Parameter Curve.allowance(address,address)._spender (CurveFactoryV2.sol#4313) is not in mixedCase
Parameter Curve.assimilator(address)._derivative (CurveFactoryV2.sol#4332) is not in mixedCase
Parameter AssimilatorV2.intakeRawAndGetBalance(uint256)._amount (CurveFactoryV2.sol#4392) is not in mixedCase
Parameter AssimilatorV2.intakeRaw(uint256)._amount (CurveFactoryV2.sol#4411) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraire(int128)._amount (CurveFactoryV2.sol#4424) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,uint256,address,int128)._baseWeight (Cu
rveFactoryV2.sol#4439) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,uint256,address,int128)._minBaseAmount
(CurveFactoryV2.sol#4440) is not in mixedCase
Parameter AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,uint256,address,int128)._maxBaseAmount
(CurveFactoryV2.sol#4441) is not in mixedCase
```

```
Parameter AssimilatorV2.viewNumeraireBalanceLPRatio(uint256,uint256,address)._baseWeight (CurveFactoryV2.sol#4606) is not in m
ixedCase
Parameter AssimilatorV2.viewNumeraireBalanceLPRatio(uint256,uint256,address)._quoteWeight (CurveFactoryV2.sol#4607) is not in
mixedCase
Parameter AssimilatorV2.viewNumeraireBalanceLPRatio(uint256,uint256,address)._addr (CurveFactoryV2.sol#4608) is not in mixedCa
se
Parameter AssimilatorV2.transferFee(int128,address)._amount (CurveFactoryV2.sol#4624) is not in mixedCase
Parameter AssimilatorV2.transferFee(int128,address)._treasury (CurveFactoryV2.sol#4624) is not in mixedCase
Parameter CurveFactoryV2.getCurve(address,address)._baseCurrency (CurveFactoryV2.sol#4768) is not in mixedCase
Parameter CurveFactoryV2.getCurve(address,address)._quoteCurrency (CurveFactoryV2.sol#4769) is not in mixedCase
Parameter CurveFactoryV2.newCurve(CurveInfo)._info (CurveFactoryV2.sol#4775) is not in mixedCase
Variable CurveFactoryV2.USDC (CurveFactoryV2.sol#4705) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (CurveFactoryV2.sol#2024)" inContext (CurveFactoryV2.sol#2018-2027)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable Curve.assimilator(address)._derivative (CurveFactoryV2.sol#4332) is too similar to Storage.derivatives (CurveFactoryV
2.sol#1748)
Variable Curve.originSwap(address,address,uint256,uint256,uint256)._originAmount (CurveFactoryV2.sol#3944) is too similar to C
urve.viewTargetSwap(address,address,uint256).originAmount_ (CurveFactoryV2.sol#4048)
Variable Curve.viewOriginSwap(address,address,uint256)._originAmount (CurveFactoryV2.sol#3980) is too similar to Curve.targetS
wap(address,address,uint256,uint256,uint256).originAmount_ (CurveFactoryV2.sol#4017)
Variable Curve.originSwap(address,address,uint256,uint256,uint256)._originAmount (CurveFactoryV2.sol#3944) is too similar to C
urve.targetSwap(address,address,uint256,uint256,uint256).originAmount_ (CurveFactoryV2.sol#4017)
Variable Curve.viewOriginSwap(address,address,uint256)._originAmount (CurveFactoryV2.sol#3980) is too similar to Curve.viewTar
getSwap(address,address,uint256).originAmount_ (CurveFactoryV2.sol#4048)
Variable Curve.flash(address,uint256,uint256,bytes).balance0After (CurveFactoryV2.sol#4270) is too similar to Curve.flash(addr
ess,uint256,uint256,bytes).balance1After (CurveFactoryV2.sol#4271)
Variable Curve.flash(address,uint256,uint256,bytes).balance0Before (CurveFactoryV2.sol#4256-4258) is too similar to Curve.flas
h(address,uint256,uint256,bytes).balance1Before (CurveFactoryV2.sol#4259-4261)
Variable Curve.excludeDerivative(address)._derivative (CurveFactoryV2.sol#3879) is too similar to Storage.derivatives (CurveFa
ctoryV2.sol#1748)
Variable Curve.targetSwap(address,address,uint256,uint256,uint256)._targetAmount (CurveFactoryV2.sol#4007) is too similar to C
urve.viewOriginSwap(address,address,uint256).targetAmount_ (CurveFactoryV2.sol#3986)
Variable Curve.targetSwap(address,address,uint256,uint256,uint256)._targetAmount (CurveFactoryV2.sol#4007) is too similar to C
urve.originSwap(address,address,uint256,uint256,uint256).targetAmount_ (CurveFactoryV2.sol#3955)
```

```
Variable Curve.viewTargetSwap(address,address,uint256)._targetAmount (CurveFactoryV2.sol#4042) is too similar to Curve.viewOri
ginSwap(address,address,uint256).targetAmount_ (CurveFactoryV2.sol#3986)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

ABDKMath64x64.fromInt(int256) (CurveFactoryV2.sol#193-198) uses literals with too many digits:
        - require(bool)(x >= - 0x8000000000000000 && x <= 0x7FFFFFFFFFFFFFFF) (CurveFactoryV2.sol#195)
ABDKMath64x64.muli(int128,int256) (CurveFactoryV2.sol#258-292) uses literals with too many digits:
        - require(bool)(y >= - 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF && y <= 0x1000000000000000000000000000000000
000000000000000) (CurveFactoryV2.sol#261-264)
ABDKMath64x64.muli(int128,int256) (CurveFactoryV2.sol#258-292) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x8000000000000000000000000000000000000000000000000000000000000000) (CurveFactoryV2.
sol#278-281)
ABDKMath64x64.divi(int256,int256) (CurveFactoryV2.sol#325-347) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x80000000000000000000000000000000) (CurveFactoryV2.sol#340)
ABDKMath64x64.inv(int128) (CurveFactoryV2.sol#372-379) uses literals with too many digits:
        - result = int256(0x100000000000000000000000000000000) / x (CurveFactoryV2.sol#375)
ABDKMath64x64.gavg(int128,int128) (CurveFactoryV2.sol#387-397) uses literals with too many digits:
        - require(bool)(m < 0x4000000000000000000000000000000000000000000000000000000000000000) (CurveFactoryV2.sol#391-394)
```

```
ABDKMath64x64.slitherConstructorConstantVariables() (CurveFactoryV2.sol#188-820) uses literals with too many digits:
        - MIN_64x64 = - 0x80000000000000000000000000000000 (CurveFactoryV2.sol#189)
CurveMath.slitherConstructorConstantVariables() (CurveFactoryV2.sol#854-1077) uses literals with too many digits:
        - ONE = 0x10000000000000000 (CurveFactoryV2.sol#855)
CurveMath.slitherConstructorConstantVariables() (CurveFactoryV2.sol#854-1077) uses literals with too many digits:
        - MAX = 0x4000000000000000 (CurveFactoryV2.sol#856)
ProportionalLiquidity.slitherConstructorConstantVariables() (CurveFactoryV2.sol#1433-1715) uses literals with too many digits:
        - ONE = 0x10000000000000000 (CurveFactoryV2.sol#1440)
Swaps.originSwap(Storage.Curve,OriginSwapData) (CurveFactoryV2.sol#3239-3314) uses literals with too many digits:
        - _swapInfo.amountToTreasury = _swapInfo.totalFee.muli(_swapInfo.protocolFeePercentage).divi(100000) (CurveFactoryV2.s
ol#3291-3294)
Swaps.targetSwap(Storage.Curve,TargetSwapData) (CurveFactoryV2.sol#3357-3433) uses literals with too many digits:
        - _swapInfo.amountToTreasury = _swapInfo.totalFee.muli(_swapInfo.protocolFeePercentage).divi(100000) (CurveFactoryV2.s
ol#3412-3415)
Swaps.slitherConstructorConstantVariables() (CurveFactoryV2.sol#3203-3668) uses literals with too many digits:
        - ONE = 0x10000000000000000 (CurveFactoryV2.sol#3219)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

CurveMath.ONE_WEI (CurveFactoryV2.sol#858) is never used in CurveMath (CurveFactoryV2.sol#854-1077)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

CurveFactoryV2.USDC (CurveFactoryV2.sol#4705) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

Curve.curveFactory (CurveFactoryV2.sol#3675) should be immutable
CurveFactoryV2.config (CurveFactoryV2.sol#4703) should be immutable
ERC20._name (CurveFactoryV2.sol#2053) should be immutable
ERC20._symbol (CurveFactoryV2.sol#2054) should be immutable
Storage.name (CurveFactoryV2.sol#1744) should be immutable
Storage.symbol (CurveFactoryV2.sol#1745) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
CurveFactoryV2.sol analyzed (36 contracts with 84 detectors), 453 result(s) found
```

Router.sol

```
SafeERC20.safeApprove(IERC20,address,uint256) (Router.sol#1757-1767) has external calls inside a loop: require(bool,string)((v
alue == 0) || (token.allowance(address(this),spender) == 0),SafeERC20: approve from non-zero to non-zero allowance) (Router.so
l#1762-1765)
Address.functionCallWithValue(address,bytes,uint256,string) (Router.sol#1150-1161) has external calls inside a loop: (success,
returndata) = target.call{value: value}(data) (Router.sol#1159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in Orchestrator.includeAsset(Storage.Curve,address,address,address,address,address,uint256) (Router.sol#2245-2318):
        External calls:
        - IERC20(_numeraire).safeApprove(_reserveApproveTo,type()(uint256).max) (Router.sol#2277)
        Event emitted after the call(s):
        - AssetIncluded(_numeraire,_reserve,_weight) (Router.sol#2301)
        - AssimilatorIncluded(_numeraire,_numeraire,_reserve,_numeraireAssim) (Router.sol#2303-2308)
        - AssimilatorIncluded(_reserve,_numeraire,_reserve,_reserveAssim) (Router.sol#2311-2316)
Reentrancy in Swaps.originSwap(Storage.Curve,OriginSwapData) (Router.sol#2789-2863):
        External calls:
        - (_amt,_oGLiq,_nGLiq,_oBals,_nBals) = getOriginSwapData(curve,_o.ix,_t.ix,_o.addr,_swapData._originAmount) (Router.so
l#2807-2819)
                - (_success,returnData_) = _callee.delegatecall(_data) (Router.sol#1215)
                - (amt_,_bal) = Assimilators.intakeRawAndGetBalance(_assim,_amt) (Router.sol#3054-3057)
        - Assimilators.transferFee(_t.addr,_swapInfo.amountToTreasury,_swapInfo.treasury) (Router.sol#2844-2848)
        - tAmt_ = Assimilators.outputNumeraire(_t.addr,_swapData._recipient,_swapInfo.amountToUser) (Router.sol#2849-2853)
        Event emitted after the call(s):
        - Trade(msg.sender,_swapData._origin,_swapData._target,_swapData._originAmount,tAmt_,_swapInfo.amountToTreasury) (Rout
er.sol#2855-2862)
Reentrancy in ProportionalLiquidity.proportionalDeposit(Storage.Curve,DepositData) (Router.sol#1413-1475):
        External calls:
        - deposits_[i] = Assimilators.intakeNumeraire(curve.assets[i].addr,_d.add(ONE_WEI)) (Router.sol#1432-1435)
        - deposits_[i_scope_0] = Assimilators.intakeNumeraireLPRatio(curve.assets[i_scope_0].addr,info) (Router.sol#1452-1455)
        Event emitted after the call(s):
        - Transfer(address(0),msg.sender,toMintAmt) (Router.sol#1652)
                - mint(curve,msg.sender,curves_ = _newShells.mulu(1e18)) (Router.sol#1472)
        - Transfer(address(this),address(0),minLock) (Router.sol#1658)
                - mint(curve,msg.sender,curves_ = _newShells.mulu(1e18)) (Router.sol#1472)
        - Transfer(address(0),msg.sender,amount) (Router.sol#1662)
                - mint(curve,msg.sender,curves_ = _newShells.mulu(1e18)) (Router.sol#1472)
Reentrancy in ProportionalLiquidity.proportionalWithdraw(Storage.Curve,uint256) (Router.sol#1529-1555):
```

```
Reentrancy in ProportionalLiquidity.proportionalWithdraw(Storage.Curve,uint256) (Router.sol#1529-1555):
        External calls:
        - withdrawals_[i] = Assimilators.outputNumeraire(curve.assets[i].addr,msg.sender,_oBals[i].mul(_multiplier)) (Router.s
ol#1545-1549)
        Event emitted after the call(s):
        - Transfer(msg.sender,address(0),amount) (Router.sol#1632)
                - burn(curve,msg.sender,_withdrawal) (Router.sol#1552)
Reentrancy in Swaps.targetSwap(Storage.Curve,TargetSwapData) (Router.sol#2906-2982):
        External calls:
        - (_amt,_oGLiq,_nGLiq,_oBals,_nBals) = getTargetSwapData(curve,_t.ix,_o.ix,_t.addr,_swapData._recipient,_swapData._tar
getAmount) (Router.sol#2926-2939)
                - (_success,returnData_) = _callee.delegatecall(_data) (Router.sol#1215)
                - (amt_,_bal) = Assimilators.outputRawAndGetBalance(_assim,_recipient,_amt) (Router.sol#3103-3107)
        - Assimilators.transferFee(_o.addr,_swapInfo.amountToTreasury,_swapInfo.treasury) (Router.sol#2966-2970)
        - oAmt_ = Assimilators.intakeNumeraire(_o.addr,_swapInfo.amountToUser) (Router.sol#2972)
        Event emitted after the call(s):
        - Trade(msg.sender,_swapData._origin,_swapData._target,oAmt_,_swapData._targetAmount,_swapInfo.amountToTreasury) (Rout
er.sol#2974-2981)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.verifyCallResult(bool,bytes,string) (Router.sol#1181-1199) uses assembly
        - INLINE ASM (Router.sol#1191-1194)
Assimilators.delegate(address,bytes) (Router.sol#1209-1224) uses assembly
        - INLINE ASM (Router.sol#1217-1221)
FullMath.mulDiv(uint256,uint256,uint256) (Router.sol#2364-2420) uses assembly
        - INLINE ASM (Router.sol#2371-2375)
        - INLINE ASM (Router.sol#2379-2381)
        - INLINE ASM (Router.sol#2389-2391)
        - INLINE ASM (Router.sol#2392-2395)
        - INLINE ASM (Router.sol#2398-2400)
        - INLINE ASM (Router.sol#2402-2404)
        - INLINE ASM (Router.sol#2405-2407)
AssimilatorV2.quoteAddress() (Router.sol#3821-3837) uses assembly
        - INLINE ASM (Router.sol#3823-3825)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version0.8.19 (Router.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.
16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Router.sol#1123-1128):
        - (success) = recipient.call{value: amount}() (Router.sol#1126)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Router.sol#1150-1161):
        - (success,returndata) = target.call{value: value}(data) (Router.sol#1159)
Low level call in Address.functionStaticCall(address,bytes,string) (Router.sol#1167-1176):
        - (success,returndata) = target.staticcall(data) (Router.sol#1174)
Low level call in Assimilators.delegate(address,bytes) (Router.sol#1209-1224):
        - (_success,returnData_) = _callee.delegatecall(_data) (Router.sol#1215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Curve (Router.sol#3219-3791) should inherit from IERC20 (Router.sol#1714-1734)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Function ABDKMath64x64.log_2(int128) (Router.sol#488-531) is not in mixedCase
Function ABDKMath64x64.exp_2(int128) (Router.sol#547-689) is not in mixedCase
Constant ABDKMath64x64.MIN_64x64 (Router.sol#178) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ABDKMath64x64.MAX_64x64 (Router.sol#180) is not in UPPER_CASE_WITH_UNDERSCORES
Function UnsafeMath64x64.us_mul(int128,int128) (Router.sol#815-818) is not in mixedCase
Function UnsafeMath64x64.us_div(int128,int128) (Router.sol#821-824) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],Storage.Curve,int128[])._gLiq (Router.sol#839) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],Storage.Curve,int128[])._bals (Router.sol#840) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],Storage.Curve,int128[])._weights (Router.sol#842) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._gLiq (Router.sol#851) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._bals (Router.sol#852) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._beta (Router.sol#853) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._delta (Router.sol#854) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._weights (Router.sol#855) is not in mixedCase
Parameter CurveMath.calculateMicroFee(int128,int128,int128,int128)._bal (Router.sol#866) is not in mixedCase
Parameter CurveMath.calculateMicroFee(int128,int128,int128,int128)._ideal (Router.sol#867) is not in mixedCase
Parameter CurveMath.calculateMicroFee(int128,int128,int128,int128)._beta (Router.sol#868) is not in mixedCase
```

```
ABDKMath64x64.sqrtu(uint256) (Router.sol#764-808) uses literals with too many digits:
        - xx >= 0x100000000000000000000000000000000 (Router.sol#770)
ABDKMath64x64.sqrtu(uint256) (Router.sol#764-808) uses literals with too many digits:
        - xx >= 0x10000000000000000 (Router.sol#774)
ABDKMath64x64.sqrtu(uint256) (Router.sol#764-808) uses literals with too many digits:
        - xx >= 0x100000000 (Router.sol#778)
ABDKMath64x64.slitherConstructorConstantVariables() (Router.sol#177-809) uses literals with too many digits:
        - MIN_64x64 = - 0x80000000000000000000000000000000 (Router.sol#178)
CurveMath.slitherConstructorConstantVariables() (Router.sol#828-1049) uses literals with too many digits:
        - ONE = 0x10000000000000000 (Router.sol#829)
CurveMath.slitherConstructorConstantVariables() (Router.sol#828-1049) uses literals with too many digits:
        - MAX = 0x4000000000000000 (Router.sol#830)
ProportionalLiquidity.slitherConstructorConstantVariables() (Router.sol#1403-1673) uses literals with too many digits:
        - ONE = 0x10000000000000000 (Router.sol#1410)
Swaps.originSwap(Storage.Curve,OriginSwapData) (Router.sol#2789-2863) uses literals with too many digits:
        - _swapInfo.amountToTreasury = _swapInfo.totalFee.muli(_swapInfo.protocolFeePercentage).divi(100000) (Router.sol#2840-
2843)
Swaps.targetSwap(Storage.Curve,TargetSwapData) (Router.sol#2906-2982) uses literals with too many digits:
        - _swapInfo.amountToTreasury = _swapInfo.totalFee.muli(_swapInfo.protocolFeePercentage).divi(100000) (Router.sol#2961-
2964)
Swaps.slitherConstructorConstantVariables() (Router.sol#2753-3217) uses literals with too many digits:
        - ONE = 0x10000000000000000 (Router.sol#2769)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

CurveMath.ONE_WEI (Router.sol#832) is never used in CurveMath (Router.sol#828-1049)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

Curve.curveFactory (Router.sol#3224) should be immutable
ERC20._name (Router.sol#1917) should be immutable
ERC20._symbol (Router.sol#1918) should be immutable
Router.factory (Router.sol#4135) should be immutable
Storage.name (Router.sol#1701) should be immutable
Storage.symbol (Router.sol#1702) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Router.sol analyzed (36 contracts with 84 detectors), 450 result(s) found
```

Storage.sol

```
Address.verifyCallResult(bool,bytes,string) (Storage.sol#948-966) uses assembly
        - INLINE ASM (Storage.sol#958-961)
Assimilators.delegate(address,bytes) (Storage.sol#976-991) uses assembly
        - INLINE ASM (Storage.sol#984-988)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ABDKMath64x64.abs(int128) (Storage.sol#362-367) is never used and should be removed
ABDKMath64x64.add(int128,int128) (Storage.sol#231-237) is never used and should be removed
ABDKMath64x64.avg(int128,int128) (Storage.sol#378-382) is never used and should be removed
ABDKMath64x64.div(int128,int128) (Storage.sol#313-320) is never used and should be removed
ABDKMath64x64.divi(int256,int256) (Storage.sol#322-344) is never used and should be removed
ABDKMath64x64.divu(uint256,uint256) (Storage.sol#346-353) is never used and should be removed
ABDKMath64x64.divuu(uint256,uint256) (Storage.sol#714-770) is never used and should be removed
```

```
Assimilators.viewNumeraireBalanceLPRatio(uint256,uint256,address) (Storage.sol#1041-1051) is never used and should be removed
Assimilators.viewRawAmount(address,int128) (Storage.sol#997-1002) is never used and should be removed
Assimilators.viewRawAmountLPRatio(address,uint256,uint256,int128) (Storage.sol#1004-1016) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.19 (Storage.sol#16) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.
8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Storage.sol#890-895):
        - (success) = recipient.call{value: amount}() (Storage.sol#893)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Storage.sol#917-928):
        - (success,returndata) = target.call{value: value}(data) (Storage.sol#926)
Low level call in Address.functionStaticCall(address,bytes,string) (Storage.sol#934-943):
        - (success,returndata) = target.staticcall(data) (Storage.sol#941)
Low level call in Assimilators.delegate(address,bytes) (Storage.sol#976-991):
        - (_success,returnData_) = _callee.delegatecall(_data) (Storage.sol#982)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function ABDKMath64x64.log_2(int128) (Storage.sol#496-539) is not in mixedCase
Function ABDKMath64x64.exp_2(int128) (Storage.sol#555-697) is not in mixedCase
Constant ABDKMath64x64.MIN_64x64 (Storage.sol#186) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ABDKMath64x64.MAX_64x64 (Storage.sol#188) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Assimilators.delegate(address,bytes)._callee (Storage.sol#977) is not in mixedCase
Parameter Assimilators.delegate(address,bytes)._data (Storage.sol#978) is not in mixedCase
Parameter Assimilators.getRate(address)._assim (Storage.sol#993) is not in mixedCase
Parameter Assimilators.viewRawAmount(address,int128)._assim (Storage.sol#998) is not in mixedCase
Parameter Assimilators.viewRawAmount(address,int128)._amt (Storage.sol#999) is not in mixedCase
Parameter Assimilators.viewRawAmountLPRatio(address,uint256,uint256,int128)._assim (Storage.sol#1005) is not in mixedCase
Parameter Assimilators.viewRawAmountLPRatio(address,uint256,uint256,int128)._baseWeight (Storage.sol#1006) is not in mixedCase
Parameter Assimilators.viewRawAmountLPRatio(address,uint256,uint256,int128)._quoteWeight (Storage.sol#1007) is not in mixedCas
```

```
Parameter Assimilators.intakeNumeraireLPRatio(address,IntakeNumLpRatioInfo)._assim (Storage.sol#1090) is not in mixedCase
Parameter Assimilators.outputRaw(address,address,uint256)._assim (Storage.sol#1109) is not in mixedCase
Parameter Assimilators.outputRaw(address,address,uint256)._dst (Storage.sol#1110) is not in mixedCase
Parameter Assimilators.outputRaw(address,address,uint256)._amt (Storage.sol#1111) is not in mixedCase
Parameter Assimilators.outputRawAndGetBalance(address,address,uint256)._assim (Storage.sol#1125) is not in mixedCase
Parameter Assimilators.outputRawAndGetBalance(address,address,uint256)._dst (Storage.sol#1126) is not in mixedCase
Parameter Assimilators.outputRawAndGetBalance(address,address,uint256)._amt (Storage.sol#1127) is not in mixedCase
Parameter Assimilators.outputNumeraire(address,address,int128)._assim (Storage.sol#1141) is not in mixedCase
Parameter Assimilators.outputNumeraire(address,address,int128)._dst (Storage.sol#1142) is not in mixedCase
Parameter Assimilators.outputNumeraire(address,address,int128)._amt (Storage.sol#1143) is not in mixedCase
Parameter Assimilators.transferFee(address,int128,address)._assim (Storage.sol#1155) is not in mixedCase
Parameter Assimilators.transferFee(address,int128,address)._amt (Storage.sol#1156) is not in mixedCase
Parameter Assimilators.transferFee(address,int128,address)._treasury (Storage.sol#1157) is not in mixedCase
Constant Assimilators.iAsmltr (Storage.sol#974) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

ABDKMath64x64.fromInt(int256) (Storage.sol#190-195) uses literals with too many digits:
        - require(bool)(x >= - 0x8000000000000000 && x <= 0x7FFFFFFFFFFFFFFF) (Storage.sol#192)
ABDKMath64x64.muli(int128,int256) (Storage.sol#255-289) uses literals with too many digits:
        - require(bool)(y >= - 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF && y <= 0x1000000000000000000000000000000000
000000000000000) (Storage.sol#258-261)
ABDKMath64x64.muli(int128,int256) (Storage.sol#255-289) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x8000000000000000000000000000000000000000000000000000000000000000) (Storage.sol#275
-278)
ABDKMath64x64.divi(int256,int256) (Storage.sol#322-344) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x8000000000000000000000000000000) (Storage.sol#337)
```

```
ABDKMath64x64.exp_2(int128) (Storage.sol#555-697) uses literals with too many digits:
        - result = (result * 0x1000000000000000B17217F7D1CF79AFA) >> 128 (Storage.sol#682)
ABDKMath64x64.exp_2(int128) (Storage.sol#555-697) uses literals with too many digits:
        - result = (result * 0x10000000000000058B90BFBE8E7BCD6D) >> 128 (Storage.sol#684)
ABDKMath64x64.exp_2(int128) (Storage.sol#555-697) uses literals with too many digits:
        - result = (result * 0x10000000000000002C5C85FDF473DE6B2) >> 128 (Storage.sol#686)
ABDKMath64x64.exp_2(int128) (Storage.sol#555-697) uses literals with too many digits:
        - result = (result * 0x10000000000000000162E42FEFA39EF358) >> 128 (Storage.sol#688)
ABDKMath64x64.exp_2(int128) (Storage.sol#555-697) uses literals with too many digits:
        - result = (result * 0x1000000000000000B17217F7D1CF79AB) >> 128 (Storage.sol#690)
ABDKMath64x64.exp(int128) (Storage.sol#699-712) uses literals with too many digits:
        - require(bool)(x < 0x400000000000000000) (Storage.sol#701)
ABDKMath64x64.exp(int128) (Storage.sol#699-712) uses literals with too many digits:
        - x < - 0x400000000000000000 (Storage.sol#703)
ABDKMath64x64.divuu(uint256,uint256) (Storage.sol#714-770) uses literals with too many digits:
        - xc >= 0x100000000 (Storage.sol#725)
ABDKMath64x64.sqrtu(uint256) (Storage.sol#772-816) uses literals with too many digits:
        - xx >= 0x100000000000000000000000000000000 (Storage.sol#778)
ABDKMath64x64.sqrtu(uint256) (Storage.sol#772-816) uses literals with too many digits:
        - xx >= 0x10000000000000000 (Storage.sol#782)
ABDKMath64x64.sqrtu(uint256) (Storage.sol#772-816) uses literals with too many digits:
        - xx >= 0x100000000 (Storage.sol#786)
ABDKMath64x64.slitherConstructorConstantVariables() (Storage.sol#185-817) uses literals with too many digits:
        - MIN_64x64 = - 0x8000000000000000000000000000000 (Storage.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Storage.emergency (Storage.sol#1205) should be constant
Storage.frozen (Storage.sol#1204) should be constant
Storage.name (Storage.sol#1196) should be constant
Storage.notEntered (Storage.sol#1206) should be constant
Storage.owner (Storage.sol#1194) should be constant
Storage.symbol (Storage.sol#1197) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
Storage.sol analyzed (7 contracts with 84 detectors), 227 result(s) found
```

Zap.sol

```
Address.functionCallWithValue(address,bytes,uint256,string) (Zap.sol#758-769) has external calls inside a loop: (success,retur
ndata) = target.call{value: value}(data) (Zap.sol#767)
SafeERC20.safeApprove(IERC20,address,uint256) (Zap.sol#1310-1320) has external calls inside a loop: require(bool,string)((valu
e == 0) || (token.allowance(address(this),spender) == 0),SafeERC20: approve from non-zero to non-zero allowance) (Zap.sol#1315
-1318)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in Orchestrator.includeAsset(Storage.Curve,address,address,address,address,address,uint256) (Zap.sol#1793-1866):
        External calls:
        - IERC20(_numeraire).safeApprove(_reserveApproveTo,type()(uint256).max) (Zap.sol#1825)
        Event emitted after the call(s):
        - AssetIncluded(_numeraire,_reserve,_weight) (Zap.sol#1849)
        - AssimilatorIncluded(_numeraire,_numeraire,_reserve,_numeraireAssim) (Zap.sol#1851-1856)
        - AssimilatorIncluded(_reserve,_numeraire,_reserve,_reserveAssim) (Zap.sol#1859-1864)
Reentrancy in ProportionalLiquidity.proportionalWithdraw(Storage.Curve,uint256) (Zap.sol#1081-1107):
        External calls:
        - withdrawals_[i] = Assimilators.outputNumeraire(curve.assets[i].addr,msg.sender,_oBals[i].mul(_multiplier)) (Zap.sol#
1097-1101)
        Event emitted after the call(s):
        - Transfer(msg.sender,address(0),amount) (Zap.sol#1184)
                - burn(curve,msg.sender,_withdrawal) (Zap.sol#1104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.verifyCallResult(bool,bytes,string) (Zap.sol#789-807) uses assembly
        - INLINE ASM (Zap.sol#799-802)
Assimilators.delegate(address,bytes) (Zap.sol#817-832) uses assembly
        - INLINE ASM (Zap.sol#825-829)
FullMath.mulDiv(uint256,uint256,uint256) (Zap.sol#1912-1968) uses assembly
        - INLINE ASM (Zap.sol#1919-1923)
        - INLINE ASM (Zap.sol#1927-1929)
        - INLINE ASM (Zap.sol#1937-1939)
        - INLINE ASM (Zap.sol#1940-1943)
        - INLINE ASM (Zap.sol#1946-1948)
        - INLINE ASM (Zap.sol#1950-1952)
        - INLINE ASM (Zap.sol#1953-1955)
Zap.quoteAddress() (Zap.sol#3229-3245) uses assembly
```

```
Zap.quoteAddress() (Zap.sol#3229-3245) uses assembly
        - INLINE ASM (Zap.sol#3231-3233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ABDKMath64x64.avg(int128,int128) (Zap.sol#356-360) is never used and should be removed
ABDKMath64x64.divi(int256,int256) (Zap.sol#310-332) is never used and should be removed
ABDKMath64x64.divuu(uint256,uint256) (Zap.sol#411-413) is never used and should be removed
ABDKMath64x64.exp(int128) (Zap.sol#407-409) is never used and should be removed
ABDKMath64x64.exp_2(int128) (Zap.sol#403-405) is never used and should be removed
ABDKMath64x64.from128x128(int256) (Zap.sol#209-215) is never used and should be removed
ABDKMath64x64.fromInt(int256) (Zap.sol#182-187) is never used and should be removed
ABDKMath64x64.gavg(int128,int128) (Zap.sol#362-372) is never used and should be removed
```

```
Assimilators.outputRawAndGetBalance(address,address,uint256) (Zap.sol#965-979) is never used and should be removed
Assimilators.transferFee(address,int128,address) (Zap.sol#995-1006) is never used and should be removed
Context._msgData() (Zap.sol#1451-1454) is never used and should be removed
CurveMath.calculateLiquidityMembrane(Storage.Curve,int128,int128,int128[],int128[]) (Zap.sol#556-596) is never used and should
 be removed
ERC20._burn(address,uint256) (Zap.sol#1581-1585) is never used and should be removed
ERC20._mint(address,uint256) (Zap.sol#1575-1579) is never used and should be removed
ProportionalLiquidity.mint(Storage.Curve,address,uint256) (Zap.sol#1187-1216) is never used and should be removed
ProportionalLiquidity.mintAdd(uint256,uint256) (Zap.sol#1218-1220) is never used and should be removed
ReentrancyGuard._reentrancyGuardEntered() (Zap.sol#2189-2191) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (Zap.sol#1331-1342) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (Zap.sol#1322-1329) is never used and should be removed
SafeMath.min(uint256,uint256) (Zap.sol#1429-1431) is never used and should be removed
SafeMath.mod(uint256,uint256) (Zap.sol#1416-1418) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Zap.sol#1420-1427) is never used and should be removed
SafeMath.sqrt(uint256) (Zap.sol#1433-1444) is never used and should be removed
Swaps.getOriginSwapData(Storage.Curve,uint256,uint256,address,uint256) (Zap.sol#2434-2480) is never used and should be removed
Swaps.getTargetSwapData(Storage.Curve,uint256,uint256,address,address,uint256) (Zap.sol#2482-2530) is never used and should be
 removed
UnsafeMath64x64.us_div(int128,int128) (Zap.sol#430-433) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.19 (Zap.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Low level call in Address.sendValue(address,uint256) (Zap.sol#731-736):
        - (success) = recipient.call{value: amount}() (Zap.sol#734)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Zap.sol#758-769):
        - (success,returndata) = target.call{value: value}(data) (Zap.sol#767)
Low level call in Address.functionStaticCall(address,bytes,string) (Zap.sol#775-784):
        - (success,returndata) = target.staticcall(data) (Zap.sol#782)
Low level call in Assimilators.delegate(address,bytes) (Zap.sol#817-832):
        - (_success,returnData_) = _callee.delegatecall(_data) (Zap.sol#823)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Curve (Zap.sol#2628-3200) should inherit from IERC20 (Zap.sol#1267-1287)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Function ABDKMath64x64.log_2(int128) (Zap.sol#385-387) is not in mixedCase
Function ABDKMath64x64.exp_2(int128) (Zap.sol#403-405) is not in mixedCase
Constant ABDKMath64x64.MIN_64x64 (Zap.sol#178) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ABDKMath64x64.MAX_64x64 (Zap.sol#180) is not in UPPER_CASE_WITH_UNDERSCORES
Function UnsafeMath64x64.us_mul(int128,int128) (Zap.sol#424-427) is not in mixedCase
Function UnsafeMath64x64.us_div(int128,int128) (Zap.sol#430-433) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],Storage.Curve,int128[])._gLiq (Zap.sol#448) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],Storage.Curve,int128[])._bals (Zap.sol#449) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],Storage.Curve,int128[])._weights (Zap.sol#451) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._gLiq (Zap.sol#460) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._bals (Zap.sol#461) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._beta (Zap.sol#462) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._delta (Zap.sol#463) is not in mixedCase
Parameter CurveMath.calculateFee(int128,int128[],int128,int128,int128[])._weights (Zap.sol#464) is not in mixedCase
Parameter CurveMath.calculateMicroFee(int128,int128,int128,int128)._bal (Zap.sol#475) is not in mixedCase
```

```
Parameter Zap.calcSwapAmountForZap(address,uint256,bool)._curve (Zap.sol#3437) is not in mixedCase
Parameter Zap.calcSwapAmountForZap(address,uint256,bool)._zapAmount (Zap.sol#3438) is not in mixedCase
Parameter Zap.calcMaxDepositAmountGivenQuote(address,uint256)._curve (Zap.sol#3484) is not in mixedCase
Parameter Zap.calcMaxDepositAmountGivenQuote(address,uint256)._quoteAmount (Zap.sol#3485) is not in mixedCase
Parameter Zap.calcMaxDepositAmountGivenBase(address,uint256)._curve (Zap.sol#3504) is not in mixedCase
Parameter Zap.calcMaxDepositAmountGivenBase(address,uint256)._baseAmount (Zap.sol#3505) is not in mixedCase
Parameter Zap.calcMaxBaseForDeposit(address,uint256)._curve (Zap.sol#3524) is not in mixedCase
Parameter Zap.calcMaxBaseForDeposit(address,uint256)._quoteAmount (Zap.sol#3525) is not in mixedCase
Parameter Zap.calcMaxQuoteForDeposit(address,uint256)._curve (Zap.sol#3534) is not in mixedCase
Parameter Zap.calcMaxQuoteForDeposit(address,uint256)._baseAmount (Zap.sol#3535) is not in mixedCase
Variable Zap.USDC (Zap.sol#3207) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Zap.sol#1452)" inContext (Zap.sol#1447-1455)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable Curve.assimilator(address)._derivative (Zap.sol#3196) is too similar to Storage.derivatives (Zap.sol#1257)
Variable Curve.originSwap(address,address,uint256,uint256,uint256)._originAmount (Zap.sol#2880) is too similar to Curve.viewTa
rgetSwap(address,address,uint256).originAmount_ (Zap.sol#2965)
Variable Curve.viewOriginSwap(address,address,uint256)._originAmount (Zap.sol#2910) is too similar to Curve.viewTargetSwap(add
ress,address,uint256).originAmount_ (Zap.sol#2965)
Variable Curve.viewOriginSwap(address,address,uint256)._originAmount (Zap.sol#2910) is too similar to Curve.targetSwap(address
,address,uint256,uint256,uint256).originAmount_ (Zap.sol#2940)
Variable Curve.originSwap(address,address,uint256,uint256,uint256)._originAmount (Zap.sol#2880) is too similar to Curve.target
Swap(address,address,uint256,uint256,uint256).originAmount_ (Zap.sol#2940)
```

```
Variable Curve.targetSwap(address,address,uint256,uint256,uint256)._targetAmount (Zap.sol#2930) is too similar to Curve.viewOr
iginSwap(address,address,uint256).targetAmount_ (Zap.sol#2916)
Variable Curve.targetSwap(address,address,uint256,uint256,uint256)._targetAmount (Zap.sol#2930) is too similar to Curve.origin
Swap(address,address,uint256,uint256,uint256).targetAmount_ (Zap.sol#2891)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

ABDKMath64x64.fromInt(int256) (Zap.sol#182-187) uses literals with too many digits:
        - require(bool)(x >= - 0x8000000000000000 && x <= 0x7FFFFFFFFFFFFFFF) (Zap.sol#184)
ABDKMath64x64.muli(int128,int256) (Zap.sol#247-277) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x8000000000000000000000000000000000000000000000000000000000000000) (Zap.sol#263-266
)
ABDKMath64x64.divi(int256,int256) (Zap.sol#310-332) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x80000000000000000000000000000000) (Zap.sol#325)
ABDKMath64x64.gavg(int128,int128) (Zap.sol#362-372) uses literals with too many digits:
        - require(bool)(m < 0x4000000000000000000000000000000000000000000000000000000000000000) (Zap.sol#366-369)
```

```
ABDKMath64x64.muli(int128,int256) (Zap.sol#247-277) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x8000000000000000000000000000000000000000000000000000000000000000) (Zap.sol#263-266
)
ABDKMath64x64.divi(int256,int256) (Zap.sol#310-332) uses literals with too many digits:
        - require(bool)(absoluteResult <= 0x80000000000000000000000000000000) (Zap.sol#325)
ABDKMath64x64.gavg(int128,int128) (Zap.sol#362-372) uses literals with too many digits:
        - require(bool)(m < 0x4000000000000000000000000000000000000000000000000000000000000000) (Zap.sol#366-369)
ABDKMath64x64.slitherConstructorConstantVariables() (Zap.sol#177-418) uses literals with too many digits:
        - MIN_64x64 = - 0x8000000000000000000000000000000 (Zap.sol#178)
CurveMath.slitherConstructorConstantVariables() (Zap.sol#437-658) uses literals with too many digits:
        - ONE = 0x10000000000000000 (Zap.sol#438)
CurveMath.slitherConstructorConstantVariables() (Zap.sol#437-658) uses literals with too many digits:
        - MAX = 0x4000000000000000 (Zap.sol#439)
ProportionalLiquidity.slitherConstructorConstantVariables() (Zap.sol#1012-1225) uses literals with too many digits:
        - ONE = 0x10000000000000000 (Zap.sol#1019)
Swaps.slitherConstructorConstantVariables() (Zap.sol#2302-2626) uses literals with too many digits:
        - ONE = 0x10000000000000000 (Zap.sol#2318)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

CurveMath.ONE_WEI (Zap.sol#441) is never used in CurveMath (Zap.sol#437-658)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

Curve.curveFactory (Zap.sol#2633) should be immutable
ERC20._name (Zap.sol#1472) should be immutable
ERC20._symbol (Zap.sol#1473) should be immutable
Storage.name (Zap.sol#1253) should be immutable
Storage.symbol (Zap.sol#1254) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Zap.sol analyzed (32 contracts with 84 detectors), 338 result(s) found
```

Solidity Static Analysis

AssimilatorFactory.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in AssimilatorFactory.newAssimilator(contract IOracle,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 55:1:

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: 57:2:

Gas costs:

Gas requirement of function AssimilatorV2.oracle 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: 36:1:

Constant/View/Pure functions:

AssimilatorFactory.getAssimilator(address) : Is constant but potentially should not be.
Note: Modifiers are currently not considered by this static analysis.
more
Pos: 48:1:

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: 78:4:

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: 315:19:

AssimilatorV2.sol

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: 57:2:

Gas costs:

Gas requirement of function AssimilatorV2.usdc 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: 34:1:

Constant/View/Pure functions:

AssimilatorV2.intakeRaw(uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 98:1:

Similar variable names:

AssimilatorV2.intakeNumeraireLPRatio(uint256,uint256,uint256,uint256,uint256,ui
: Variables have very similar names "_amount" and "amount_". Note: Modifiers are currently not considered by this static analysis.
Pos: 156:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 443:6:

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: 176:13:

Config.sol

Gas costs:

Gas requirement of function Config.setPoolGuarded 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: 97:1:

Gas costs:

Gas requirement of function Config.setFlashable 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: 152:1:

Constant/View/Pure functions:

IConfig.updateProtocolTreasury(address) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 34:1:

No return:

IConfig.isPoolGuarded(address): Defines a return type but never explicitly returns a value.
Pos: 26:1:

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: 45:2:

CurveFactory.sol

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: 78:4:

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: 364:10:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 36:46:

Gas costs:

Gas requirement of function CurveFactory.newCurve 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: 43:1:

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: 767:3:

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: 471:2:

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: 64:4:

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: 29:18:

CurveFactoryV2.sol

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: 200:3:

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: 364:10:

Gas costs:

Gas requirement of function CurveFactory.newCurve 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: 43:1:

Gas costs:

Gas requirement of function CurveFactoryV2.newCurve 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: 110:1:

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: 767:3:

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: 471:2:

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: 233:2:

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: 29:18:

Router.sol

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: 200:3:

Gas costs:

Gas requirement of function Router.originSwap 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: 103:1:

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: 146:2:

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: 65:2:

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: 29:18:

Storage.sol

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: 39:2:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 36:46:

Gas costs:

Gas requirement of function Storage.curve 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: 49:1:

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: 364:10:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 36:46:

Zap.sol

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: 39:2:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.

more

Pos: 36:46:

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: 364:10:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 36:46:

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: 767:3:

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: 471:2:

Similar variable names:

Zap.zap(address,uint256,uint256,uint256,bool) : Variables have very similar names "swapAmount" and "_zapAmount". Note: Modifiers are currently not considered by this static analysis.
Pos: 183:3:

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: 250:2:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 478:2:

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to lost tokens etc.
High	High level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g. public access to crucial functions.
Medium	Medium level vulnerabilities are important to fix; however, they cannot lead to lost tokens.
Low	Low level vulnerabilities are most related to outdated, unused etc. These code snippets cannot have a significant impact on execution.
Lowest Code Style/ Best Practice	Lowest level vulnerabilities, code style violations and information statements cannot affect smart contract execution and can be ignored.

Audit Findings

Critical:

No critical severity vulnerabilities were found.

High:

No high severity vulnerabilities were found.

Medium:

No medium severity vulnerabilities were found.

Low:

No low severity vulnerabilities were found.

Very Low:

No very low severity vulnerabilities were found.

Conclusion

We were given a contract code in the form of a link and have used all possible tests based on the given object. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

The security state of the reviewed contract is "**Well-secured**".

Note For Contract Users

Technical auditing does not guarantee the project's ethical side.

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 analyse 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 scrutinised 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

RD Auditors Disclaimer

The smart contracts given for audit have been analysed 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).

Because the total number of test cases are unlimited, the audit makes no statements or warranties on the 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 recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain. 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.

RD
AUDITORS

Email: info@rdauditors.com

Website: www.rdauditors.com