SmartDec

Kyber Network Smart Contracts Security Analysis

This report is private.

Published: December 2, 2019.

Abstract 2
Disclaimer 2
Summary 2
General recommendations 2
Checklist 3
Procedure 4
Checked vulnerabilities 5
Project overview 6
Project description 6
The latest version of the code 6
Project architecture 6
Scope of work 6
Automated analysis 7
Manual analysis 9
Critical issues 9
Medium severity issues 9
Integer overflow (fixed) 9
Low severity issues 10
Redundant code 10
Prefer external to public visibility level 11
Prefer internal to public visibility level 11
Missing input validation 12
Code logic 12
Notes 13
Outdated compiler version 13
Appendix 14
Compilation output 14
Tests output 17
Solhint output 19
Ethlint output 22

Abstract

In this report, we consider the security of the Kyber Network project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, security audit is not an investment advice.

Summary

In this report, we considered the security of Kyber Network smart contracts. We performed our audit according to the procedure described below.

The initial audit showed no critical issues. However, one medium severity and a number of low severity issues were found. They do not endanger project security. Nevertheless, we highly recommend addressing them.

The medium severity issue was fixed in the latest version of the code as well as some of the low severity issues.

General recommendations

The contracts code is of good code quality. The audit did not reveal any issues that endanger project security.

However, if the developers decide to improve the code, we recommend removing Redundant code. This issue is minor and does not affect code operation, though.

Checklist

Security

One of the audit goals is to check the implementation of the pricing algorithm presented in the specification. We have researched the test coverage of this functionality and also checked the mathematics. ✓

The audit showed no issues or vulnerabilities that could affect the security from the market maker side. We did not find any attack scenario that could deplete the market maker's inventory.

Compliance with the documentation

The audit showed no discrepancies between the code and the provided documentation. The contracts implement the algorithm presented in the documentation. ✓

Tests

All the tests passed successfully. ✓

The text below is for technical use; it details the statements made in Summary and General recommendations.

Procedure

In our audit, we consider the following crucial features of the smart contract code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices in efficient use of gas, code readability, etc.

We perform our audit according to the following procedure:

- automated analysis
  - we scan project's smart contracts with our own Solidity static code analyzer SmartCheck
  - we scan project's smart contracts with several publicly available automated Solidity analysis tools such as Remix, Ethlint, and Solhint
  - we manually verify (reject or confirm) all the issues found by tools
- manual audit
  - we manually analyze smart contracts for security vulnerabilities
  - we check smart contracts logic and compare it with the one described in the documentation
  - we run tests
- report
  - we reflect all the gathered information in the report

Checked vulnerabilities

We have scanned Kyber Network smart contracts for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered (the full list includes them but is not limited to them):

- Reentrancy
- DoS with (unexpected) revert
- DoS with block gas limit
- Gas limit and loops
- Locked money
- Integer overflow/underflow
- Unchecked external call
- ERC20 Standard violation
- Authentication with tx.origin
- Unsafe use of timestamp
- Using blockhash for randomness
- Balance equality
- Unsafe transfer of ether
- Fallback abuse
- Using inline assembly
- Short address attack
- Private modifier
- Compiler version not fixed
- Style guide violation
- Unsafe type deduction
- Implicit visibility level
- Use delete for arrays
- Byte array
- Incorrect use of assert/require
- Using deprecated constructions

Project overview

Project description

In our analysis we consider Kyber Network specification (”price discovery.pdf”, sha1sum: 71dcb39e4303e368fa59553aa19764ae8db29edf) and smart contracts' code (version on commit 76fadb44f679bcd3ff0449a3be8beab11357d0f9).

The latest version of the code

After the initial audit, some fixes were applied and the code was updated to the latest version (commit a64af37dd6f679511d6ba1864e57dd057e371722).

Project architecture

For the audit, we were provided with the truffle project. The project is an npm package and includes tests.

- The project successfully compiles with `truffle compile` command (with some warnings, see Compilation output in Appendix)
- The project successfully passes all the tests, however, code coverage was not generated

The total LOC of audited Solidity sources is 337.

Scope of work

The following files were audited:

- **LiquidityFormula.sol**
- **LiquidityConversionRates.sol**

Automated analysis

We used several publicly available automated Solidity analysis tools. Here are the combined results of SmartCheck, Solhint, and Remix scanning. All the issues found by tools were manually checked (rejected or confirmed).

True positives are constructions that were discovered by the tools as vulnerabilities and can actually be exploited by attackers or lead to incorrect contracts operation.

False positives are constructions that were discovered by the tools as vulnerabilities but do not consist a security threat.

Cases when these issues lead to actual bugs or vulnerabilities are described in the next section.

Tool	Rule	True positives	False positives
Solhint	Compiler version 0.4.18 does not satisfy the 0.5.10 semver requirement	2	
Total Solhint		2	0
Ethlint	Constructor declaration style is deprecated		1
	Provide an error message for require().		35
	Use emit statements for triggering events.		3
Total Ethlint		0	39
SmartCheck	Prefer external to public visibility level	5	3
	Costly loop		1
Total SmartCheck		5	4
Remix	Potential Violation of Checks-Effects-Interaction pattern		2
Total Remix		0	2

Total Overall	**7**	**45**

Manual analysis

The contracts were completely manually analyzed, their logic was checked and compared with the one described in the documentation. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger smart contracts security. We highly recommend fixing them.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence smart contracts operation in current implementation. We highly recommend addressing them.

Integer overflow (fixed)

In the code there are some places where an overflow issue can occur:

- **LiquidityFormula.sol**, line 124

  ```
  uint expRE = exp(r*e, precision*precision, precision);
  ```

  The result of `r*e` is not checked for overflow.

- **LiquidityFormula.sol**, line 132

  ```
  uint erdeltaE = exp(r*deltaE, precision*precision,
  precision);
  ```

  The result of `r*deltaE` is not checked for overflow.

- **LiquidityFormula.sol**, line 150

  ```
  uint lnPart = ln(precision*precision + rpe*deltaT/precision,
  precision*precision, numPrecisionBits);
  ```

  The result of `precision*precision + rpe*deltaT/precision` is not checked for overflow.

- **LiquidityConversionRates.sol**, line 71

  ```
  require(_numFpBits < 256);
  ```

  LiquidityFormula.sol, line 74

  ```
  require(numPrecisionBits < 125);
  ```

  If `_numFpBits < 125` then `formulaPrecision < uint(1)<<125`. However, `formulaPrecision` can be equal to 2^{124}. This will lead to issues due to line 211 in **LiquidityConversionRates.sol**

  ```
  uint deltaTInFp = deltaTFunc(rInFp, pMinInFp, eInFp,
  deltaEInFp, formulaPrecision);
  ```

  `deltaTFunc()` contains the following line

  ```
  return (erdeltaE - precision) * precision * precision *
  precision / (rpe*erdeltaE);
  ```

  If `precision` is greater than $2^{256/3}$, then

  ```
  (erdeltaE - precision) * precision * precision * precision
  ```

  will be greater than 2^{256}. Hence, we recommend making check at line 71 in **LiquidityConversionRates.sol** more strict.

We highly recommend adding extra checks for the overflow for these operations.

The issues have been fixed and are not present in the latest version of the code.

Low severity issues

Low severity issues can influence smart contracts operation in future versions of code. We recommend taking them into account.

Redundant code

The following lines contain redundant code:

- **LiquidityConversionRates.sol**, line 24

```
uint public collectedFeesInTwei = 0;
```

Variables of `uint` type defaults to zero value. Hence, explicit assignment is redundant.

- **LiquidityConversionRates.sol**, lines 113–115

```
conversionToken;
rateUpdateBlock;
currentBlock;
```

These variable names should be omitted in order to fix compiler warnings.

We highly recommend removing redundant code in order to improve code readability and transparency and decrease cost of deployment and execution.

Prefer external to public visibility level

In **LiquidityConversionRates.sol** the following functions has `public` visibility level but are not called internally:

- `setReserveAddress()` function
- `setLiquidityParams()` function
- `recordImbalance()` function
- `resetCollectedFees()` function
- `getRate()` function

We recommend changing visibility level of such functions to `external` in order to improve code readability. Moreover, in many cases functions with `external` visibility modifier require less gas comparing to functions with `public` visibility modifier.

Prefer internal to public visibility level

In the code there are functions with the `public` visibility level that are not intended to be called by users or other contracts.

We recommend changing visibility level of such functions to `internal` in order to improve code readability and decrease gas costs of the deployment.

Missing input validation

In the code there are some functions that require additional checks on input variables:

- **LiquidityConversionRates.sol**, `setLiquidityParams()` function. The following variables should be nonzero: `_rInFp`, `_pMinInFp`, `_numFpBits`.

  The issue has been fixed and is not present in the latest version of the code.

- **LiquidityFormula.sol**, `deltaEFunc()` function. `r` should be checked to be nonzero.

  The issue has been fixed and is not present in the latest version of the code.

- **LiquidityFormula.sol**, `countLeadingZeros()` function. `q` should be checked to be nonzero.

  The issue has been fixed and is not present in the latest version of the code.

- In **LiquidityFormula.sol**, there are functions with `public` visibility that accept `q` as a parameter. In these functions some values are divided by `q`. Hence, `q` must be nonzero. However, it is not checked. We recommend either changing these functions visibility level to `internal` or adding additional checks.

Code logic

There are some code logic issues in the code:

- **LiquidityConversionRates.sol**, line 72

  ```
  require(formulaPrecision <= MAX_QTY);
  ```

  When considered function is called first time, `formulaPrecision` is equal to zero, hence this check always passes. However, when this function is called again, the previous `formulaPrecision` value is checked, not the one that will be set. There is no sence in such check.

  We recommend checking new `formulaPrecision` value instead of the current one.

  The issue has been fixed and is not present in the latest version of the code.

- In the code, `precision` is equal to 2^{bits}. There is inconsistency with how `precision` is obtained in different functions. To some functions it is passed as an argument (for example, to `deltaEFunc()` function). In other functions `precision` is calculated from the `bits` value (for example, in `logBase2()` function).

  We recommend either obtaining `precision` in the same way in all of the functions or clarifying the logic in the documentation.

Notes

Outdated compiler version

The project uses Solidity compiler version 0.4.18. We recommend using the latest compiler version since it contains gas optimizations as well as bug fixes.

Comment from the developers: ”Solidity version will most likely change in later time. But we will consider doing it now.”

This analysis was performed by SmartDec.

Sergei Pavlin, Chief Operating Officer
Igor Sobolev, Analyst
Pavel Kondratenkov, Analyst

December 2, 2019

Appendix

Compilation output

```
npx truffle compile
Compiling ./contracts/ConversionRates.sol...
Compiling ./contracts/ConversionRatesInterface.sol...
Compiling ./contracts/ERC20Interface.sol...
Compiling ./contracts/ExpectedRate.sol...
Compiling ./contracts/ExpectedRateInterface.sol...
Compiling ./contracts/FeeBurner.sol...
Compiling ./contracts/FeeBurnerInterface.sol...
Compiling ./contracts/KyberNetwork.sol...
Compiling ./contracts/KyberNetworkInterface.sol...
Compiling ./contracts/KyberNetworkProxy.sol...
Compiling ./contracts/KyberNetworkProxyInterface.sol...
Compiling ./contracts/KyberReserve.sol...
Compiling ./contracts/KyberReserveInterface.sol...
Compiling ./contracts/LiquidityConversionRates.sol...
Compiling ./contracts/LiquidityFormula.sol...
Compiling ./contracts/Migrations.sol...
Compiling ./contracts/PermissionGroups.sol...
Compiling ./contracts/SanityRates.sol...
Compiling ./contracts/SanityRatesInterface.sol...
Compiling ./contracts/SimpleNetworkInterface.sol...
Compiling ./contracts/Utils.sol...
Compiling ./contracts/Utils2.sol...
Compiling ./contracts/VolumeImbalanceRecorder.sol...
Compiling ./contracts/WhiteList.sol...
Compiling ./contracts/WhiteListInterface.sol...
Compiling ./contracts/Withdrawable.sol...
Compiling ./contracts/dutchX/KyberDutchXReserve.sol...
Compiling ./contracts/dutchX/mock/MockDutchX.sol...
Compiling ./contracts/dutchX/mock/WETH9.sol...
Compiling ./contracts/mockContracts/GenerousKyberNetwork.sol
...
Compiling ./contracts/mockContracts/KyberNetworkNoMaxDest.so
l...
Compiling ./contracts/mockContracts/MaliciousKyberNetwork.so
l...
Compiling ./contracts/mockContracts/MaliciousKyberNetwork2.s
ol...
Compiling ./contracts/mockContracts/MaliciousReserve.sol...
Compiling ./contracts/mockContracts/MockCentralBank.sol...
Compiling ./contracts/mockContracts/MockConversionRate.sol..
.
Compiling ./contracts/mockContracts/MockDepositAddress.sol..
.
Compiling ./contracts/mockContracts/MockDepositAddressEther.
sol...
Compiling ./contracts/mockContracts/MockDepositAddressToken.
sol...
Compiling ./contracts/mockContracts/MockERC20.sol...
Compiling ./contracts/mockContracts/MockExchange.sol...
Compiling ./contracts/mockContracts/MockImbalanceRecorder.so
l...
Compiling ./contracts/mockContracts/MockKyberNetwork.sol...
Compiling ./contracts/mockContracts/MockNetworkFailsListing.
sol...
Compiling ./contracts/mockContracts/MockPermission.sol...
Compiling ./contracts/mockContracts/MockUtils.sol...
Compiling ./contracts/mockContracts/MockUtils2.sol...
Compiling ./contracts/mockContracts/MockWithdrawable.sol...
Compiling ./contracts/mockContracts/NetworkFailingGetRate.so
l...
Compiling ./contracts/mockContracts/SetStepFunctionWrapper.s
ol...
Compiling ./contracts/mockContracts/TestToken.sol...
Compiling ./contracts/mockContracts/TestTokenFailing.sol...
Compiling ./contracts/mockContracts/TestTokenTransferFailing
.sol...
Compiling ./contracts/mockContracts/TokenNoDecimals.sol...
Compiling ./contracts/mockContracts/TokenReverseSend.sol...
Compiling ./contracts/mockContracts/Wrapper.sol...
Compiling ./contracts/oasisContracts/KyberOasisReserve.sol..
.
Compiling ./contracts/oasisContracts/mockContracts/MockOtc.s
ol...
Compiling ./contracts/oasisContracts/mockContracts/WethToken
.sol...
Compiling ./contracts/permissionless/OrderIdManager.sol...
Compiling ./contracts/permissionless/OrderList.sol...
Compiling ./contracts/permissionless/OrderListFactory.sol...
Compiling ./contracts/permissionless/OrderListFactoryInterfa
ce.sol...
Compiling ./contracts/permissionless/OrderListInterface.sol.
..
```

```
Compiling ./contracts/permissionless/OrderbookReserve.sol...
Compiling ./contracts/permissionless/OrderbookReserveInterfa
ce.sol...
Compiling ./contracts/permissionless/PermissionlessOrderbook
ReserveLister.sol...
Compiling ./contracts/permissionless/mock/MockMedianizer.sol
...
Compiling ./contracts/permissionless/mock/MockOrderIdManager
.sol...
Compiling ./contracts/permissionless/mock/MockOrderbookReser
ve.sol...
Compiling ./contracts/permissionless/mock/TestBytes32.sol...
Compiling ./contracts/previousContracts/KyberReserveV1.sol..
.
Compiling ./contracts/uniswap/KyberUniswapReserve.sol...
Compiling ./contracts/uniswap/forTesting/KyberTestingUniswap
Reserve.sol...
Compiling ./contracts/uniswap/forTesting/MockUniswapFactory.
sol...
Compiling ./contracts/wethContracts/KyberWethReserve.sol...
Compiling ./contracts/wethContracts/mockContracts/MockWeth.s
ol...
Compiling ./contracts/wrapperContracts/FeeBurnerWrapperProxy
.sol...
Compiling ./contracts/wrapperContracts/KyberRegisterWallet.s
ol...
Compiling ./contracts/wrapperContracts/WrapConversionRate.so
l...
Compiling ./contracts/wrapperContracts/WrapFeeBurner.sol...
Compiling ./contracts/wrapperContracts/WrapReadTokenData.sol
...
Compiling ./contracts/wrapperContracts/WrapperBase.sol...

Compilation warnings encountered:

./contracts/ExpectedRate.sol:148:24: Warning: The "staticcal
l" instruction is only available after the Metropolis hard f
ork. Before that it acts as an invalid instruction.
success := staticcall(
^---------^

Writing artifacts to ./build/contracts
```

Tests output

```
  Contract: LiquidityConversionRates
 should init globals
 should init LiquidityConversionRates Inst and setting of re
serve address (254ms)
 should test abs. (107ms)
 should set liquidity params (81ms)
 should test calculation of collected fee for buy case. (293
ms)
 should test calculation of collected fee for sell case. (10
6ms)
 should test reducing fees from amount. (46ms)
 should test converting from wei to formula formulaPrecision
.
 should test converting from token wei to formula formulaPre
cision.
 should test calculation of buy rate for zero quantity. (136
ms)
 should test calculation of sell rate for zero quantity. (66
ms)
 should test calculation of deltaT. (80ms)
 should test calculation of buy rate for non zero quantity.
(76ms)
 should test calculation of deltaE. (173ms)
 should test calculation of sell rate for non zero quantity.
(165ms)
 should test recording of imbalance. (104ms)
 should test resetting of imbalance not by admin. (95ms)
 should test resetting of imbalance. (81ms)
 should test getrate for buy=true and qtyInSrcWei = non_0. (
85ms)
 should test getrate for buy=true and qtyInSrcWei = 0. (65ms
)
 should test getrate for buy=true and qtyInSrcWei very small
. (74ms)
 should test getrate for buy=false and qtyInSrcWei = non_0.
(145ms)
 should test getrate for buy=false and qtyInSrcWei = 0. (54m
s)
 should test getrate for buy=false and qtyInSrcWei very smal
l. (78ms)
 should test set liquidity params not as admin. (83ms)
 should test setting formula precisioin bits != 40 . (86ms)
 should test can't set r = 0 . (78ms)
 should test can't set minSellRateInPrecision = 0 . (72ms)
 should test can't set pMin = 0 . (863ms)
 should test set liquidity params with illegal fee in BPS co
nfiguration. (953ms)
 should test get rate with invalid token. (416ms)
 should test max sell rate smaller then expected rate and mi
n sell rate larger then expected rate . (3972ms)
 should test exceeding max cap buy (100ms)
 should test exceeding max cap sell (174ms)
 should test get rate with E (81ms)
 should test recording of imbalance from non reserve address
.

Contract: kyberReserve for Liquidity
 should init globals. init ConversionRates Inst, token, set
liquidity params . (913ms)
 should init reserve and set all reserve data including bala
nces (608ms)
 should test getConversionRate of buy rate for zero quantity
. (77ms)
 should test getConversionRate of sell rate for zero quantit
y. (70ms)
 should test getConversionRate of buy rate for non zero quan
tity. (91ms)
 should test getConversionRate of sell rate for non zero qua
ntity. (150ms)
 should perform a series of buys and check: correct balances
change, rates and fees as expected. (8115ms)
 should perform a series of sells and check: correct balance
s change, rates and fees as expected. (57273ms)
 should check setting liquidity params again with new p0 and
adjusting pmin and pmax to existing balances. (112ms)
 Should test buy tokens and then sell back (9883ms)
 Should test sell tokens and then buy back (8376ms)
 Should allow buy/sell all inventory, check rate never fall
above max rate or below min rate (17627ms)
 should check getting prices for random values. (55920ms)


49 passing (3m)
Contract: FeeBurner
 deploy liquidity contract (145ms)
 check checkMultOverflow (64ms)
 check exp with fixed input (177ms)
 check ln with fixed input (123ms)
 check P(E) with fixed input (44ms)
190325.48172459751367568969
190325.16392826076984129424
 check deltaT with fixed input (58ms)
4.9375623632222414016 7
4.9375706991437713185 5
 check deltaE with fixed input (155ms)


7 passing (1s)
```

Solhint output

```
./contracts/LiquidityFormula.sol
1:17  error    Compiler version 0.4.18 does not satisfy the
^0.5.8 semver requirement  compiler-version
73:17  warning  Provide an error message for require
eason-string
74:17  warning  Provide an error message for require
eason-string
96:17  warning  Provide an error message for require
eason-string
97:17  warning  Provide an error message for require
eason-string
98:17  warning  Provide an error message for require
eason-string
103:17  warning  Provide an error message for require
eason-string
104:17  warning  Provide an error message for require
eason-string
115:17  warning  Provide an error message for require
eason-string
125:17  warning  Provide an error message for require
eason-string
134:17  warning  Provide an error message for require
eason-string
135:17  warning  Provide an error message for require
eason-string
136:17  warning  Provide an error message for require
eason-string
137:17  warning  Provide an error message for require
```

```
eason-string
138:17  warning  Provide an error message for require
eason-string
139:17  warning  Provide an error message for require
eason-string
152:17  warning  Provide an error message for require
eason-string
153:17  warning  Provide an error message for require
eason-string
154:17  warning  Provide an error message for require
eason-string
155:17  warning  Provide an error message for require
eason-string

 20 problems (1 error, 19 warnings)

./contracts/LiquidityConversionRates.sol
1:17  error    Compiler version 0.4.18 does not satisfy the
^0.5.8 semver requirement  compiler-version
35:17  warning  Provide an error message for require
eason-string
35:39  warning  Explicitly mark all external contracts as tr
usted or untrusted          mark-callable-contracts
71:17  warning  Provide an error message for require
eason-string
72:17  warning  Provide an error message for require
eason-string
72:37  warning  Explicitly mark all external contracts as tr
usted or untrusted          mark-callable-contracts
73:17  warning  Provide an error message for require
eason-string
74:17  warning  Provide an error message for require
eason-string
79:34  warning  Explicitly mark all external contracts as tr
usted or untrusted          mark-callable-contracts
84:45  warning  Explicitly mark all external contracts as tr
usted or untrusted          mark-callable-contracts
84:33  warning  Explicitly mark all external contracts as tr
usted or untrusted          mark-callable-contracts
85:45  warning  Explicitly mark all external contracts as tr
usted or untrusted          mark-callable-contracts
85:33  warning  Explicitly mark all external contracts as tr
usted or untrusted          mark-callable-contracts
117:17  warning  Provide an error message for require
```

```
eason-string
145:17  warning  Provide an error message for require
eason-string
145:32  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
148:17  warning  Provide an error message for require
eason-string
148:36  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
158:17  warning  Provide an error message for require
eason-string
158:32  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
159:17  warning  Provide an error message for require
eason-string
204:38  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
213:17  warning  Provide an error message for require
eason-string
215:29  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
219:63  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
229:17  warning  Provide an error message for require
eason-string
230:40  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
234:91  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
239:17  warning  Provide an error message for require
eason-string
239:30  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
244:17  warning  Provide an error message for require
eason-string
244:29  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
245:53  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
249:17  warning  Provide an error message for require
eason-string
249:24  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts
254:17  warning  Provide an error message for require
```

```
eason-string
254:24  warning  Explicitly mark all external contracts as t
rusted or untrusted          mark-callable-contracts

 37 problems (1 error, 36 warnings)
```

Ethlint output

```
./contracts/LiquidityFormula.sol
8:20      error      Consequent should exist exactly on the
line after condition.                     lbrace
13:39     error      Consequent should exist exactly on the
line after condition.                     lbrace
18:4      warning    FunctionDeclaration must be succeeded b
y 1 blank line                            blank-lines
28:56     error      Consequent should exist exactly on the
line after condition.                     lbrace
29:52     error      Consequent should exist exactly on the
line after condition.                     lbrace
33:32     error      Consequent should exist exactly on the
line after condition.                     lbrace
38:48     error      Consequent should exist exactly on the
line after condition.                     lbrace
39:48     error      Consequent should exist exactly on the
line after condition.                     lbrace
40:45     error      Consequent should exist exactly on the
line after condition.                     lbrace
59:37     error      Consequent should exist exactly on the
line after condition.                     lbrace
67:4      error      "log2ForSmallNumber": Avoid assigning t
o function parameters.                    security/no-assign
-params
67:4      error      "log2ForSmallNumber": Avoid assigning t
o function parameters.                    security/no-assign
-params
73:8      warning    Provide an error message for require().
error-reason
74:8      warning    Provide an error message for require().
error-reason
96:8      warning    Provide an error message for require().
error-reason
97:8      warning    Provide an error message for require().
```

```
error-reason
98:8      warning    Provide an error message for require().
error-reason
103:8     warning    Provide an error message for require().
error-reason
104:8     warning    Provide an error message for require().
error-reason
110:8     error      Assignment operator must have exactly s
ingle space on both sides of it.          operator-whitespac
e
115:8     warning    Provide an error message for require().
error-reason
123:4     warning    In case of more than 3 parameters, drop
each into its own line.                  arg-overflow
125:8     warning    Provide an error message for require().
error-reason
129:4     warning    In case of more than 3 parameters, drop
each into its own line.                  arg-overflow
130:18    warning    Function "pE": in case of more than 3 a
rguments, drop each into its own line.    arg-overflow
134:8     warning    Provide an error message for require().
error-reason
135:8     warning    Provide an error message for require().
error-reason
136:8     warning    Provide an error message for require().
error-reason
137:8     warning    Provide an error message for require().
error-reason
138:8     warning    Provide an error message for require().
error-reason
139:8     warning    Provide an error message for require().
error-reason
144:4     warning    In case of more than 3 parameters, drop
each into its own line.                  arg-overflow
148:18    warning    Function "pE": in case of more than 3 a
rguments, drop each into its own line.    arg-overflow
152:8     warning    Provide an error message for require().
error-reason
153:8     warning    Provide an error message for require().
error-reason
154:8     warning    Provide an error message for require().
error-reason
155:8     warning    Provide an error message for require().
error-reason
```

```
 12 errors, 25 warnings found.

./contracts/LiquidityConversionRates.sol
31:4      warning    Constructor declaration style is deprec
ated                                                construc
tor
35:8      warning    Provide an error message for require().
error-reason
42:8      warning    Use emit statements for triggering even
ts.                                                emit
69:23     error      Opening brace must be on the line after
last modifier.                                    lbrace
71:8      warning    Provide an error message for require().
error-reason
72:8      warning    Provide an error message for require().
error-reason
73:8      warning    Provide an error message for require().
error-reason
74:8      warning    Provide an error message for require().
error-reason
89:8      warning    Use emit statements for triggering even
ts.                                                emit
117:8     warning    Provide an error message for require().
error-reason
133:8     warning    Use emit statements for triggering even
ts.                                                emit
137:12    error      Only use indent of 8 spaces.
ndentation
138:12    error      Only use indent of 8 spaces.
ndentation
139:12    error      Only use indent of 8 spaces.
ndentation
140:12    error      Only use indent of 8 spaces.
ndentation
141:32    error      Opening brace must be on the line after
returns declaration.                              lbrace
145:8     warning    Provide an error message for require().
error-reason
147:31    warning    Function "getRateWithE": in case of mor
e than 3 arguments, drop each into its own line.    arg-over
flow
148:8     warning    Provide an error message for require().
error-reason
152:4     warning    In case of more than 3 parameters, drop
```

```
each into its own line.                            arg-overf
low
158:8     warning    Provide an error message for require().
error-reason
159:8     warning    Provide an error message for require().
error-reason
160:38    error      Consequent should exist exactly on the
line after condition.                              lbrace
166:47    error      Consequent should exist exactly on the
line after condition.                              lbrace
183:48    error      Consequent should exist exactly on the
line after condition.                              lbrace
212:26    warning    Function "deltaTFunc": in case of more
than 3 arguments, drop each into its own line.      arg-over
flow
213:8     warning    Provide an error message for require().
error-reason
219:74    warning    Function "pE": in case of more than 3 a
rguments, drop each into its own line.              arg-over
flow
227:65    error      Opening brace must be on the line after
returns declaration.                               lbrace
228:21    warning    Function "deltaEFunc": in case of more
than 3 arguments, drop each into its own line.      arg-over
flow
229:8     warning    Provide an error message for require().
error-reason
234:43    warning    Function "pE": in case of more than 3 a
rguments, drop each into its own line.              arg-over
flow
239:8     warning    Provide an error message for require().
error-reason
244:8     warning    Provide an error message for require().
error-reason
249:8     warning    Provide an error message for require().
error-reason
254:8     warning    Provide an error message for require().
error-reason

 10 errors, 26 warnings found.
```