SmartDec

Kryll Smart Contracts Security Analysis

This report is public.

Published: April 25, 2018

Abstract

In this report, we consider the security of the Kryll 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 can not 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 have considered the security of Kryll smart contracts. We performed our audit according to the procedure described below.

The audit has shown one medium issue and many lower issues. They do not endanger project security. However, we recommend to fix all detected issues.

General recommendations

The contracts code is of good code quality and does not contain issues that endanger project security. However we recommend taking into account ERC20 approve issue.

Also, if the developer decides to improve the code, we recommend avoiding Overpowered controller, fixing Discrepancies with the whitepaper, using constructor for Initialization, connecting the OpenZeppelin library via npm, following best practices for Pragmas version, removing Redundant code and Deprecated constructions, and following Solidity Style Guide (see Solhint output in Appendix) to increase overall code quality.

However, mentioned above are minor issues. They do not influence code operation.

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, Oyente, 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 whitepaper
  - we check ERC20 compliance
  - we run tests
- report
  - we reflect all the gathered information in the report

Checked vulnerabilities

We have scanned Kryll 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
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DOS with (Unexpected) revert
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level
- Address hardcoded
- Using delete for arrays
- Integer overflow/underflow
- Locked money
- Private modifier
- Revert/require functions
- Using var
- Visibility
- Using blockhash
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Project Overview

Project description

In our analysis we consider Kryll whitepaper (WhitePaper.pdf, sha1sum 1fcf3aabf82951b717ebfd2b7e7213f1a276b962, version on 1.7) and smart contracts code (kryll.zip, sha1sum 54dcc0c63bcace70e254585246f0c24c5520bdcd).

Project architecture

For the audit, we have been provided with the truffle project.

- The project successfully compiles with `truffle compile` command (see Compilation output in Appendix).
- The project successfully passes all the tests (`truffle test` command, see Tests output in Appendix).

The project includes the following files:

- **KryllToken.sol**
- **KryllVesting.sol**
- **TransferableToken.sol**

These files contain contracts of the same name:

- KryllToken (inherits TransferableToken contract)
- KryllVesting (inherits Ownable contract from OpenZeppelin library)
- TransferableToken (inherits Ownable and StandardToken contract from OpenZeppelin library)

Total volume of audited files is 392 lines of Solidity code.

Code logic

TransferableToken is StandardToken from OpenZeppelin library.

Besides, some additional functionality was implemented:

1. The contract is ownable, i.e. the contract has an owner (firstly, it is the address the contract was deployed from, later ownership can be delegated using `transferOwnership` function).
2. The contract includes a whitelist. Whitelisted addresses can call `transfer`, `transferFrom`, `approve`, `increaseApproval`, `decreaseApproval` functions without any restrictions. Initially, only the address the contract was deployed from is whitelisted.
3. For non-whitelisted addresses, the owner of the contract can pause and unpause token transfers using `restrictTransfert` and `allowTransfert` functions respectively.

KryllToken is ERC20 compatible (compatibility has been checked during the audit) contract with the following parameters:

- token name: “Kryll.io Token”
- token symbol: “KRL”
- token decimals: 18

It is actually TransferableToken. `distribute` function initializes pre-distribution of tokens. Addresses of pre-distribution can be changed using `reset` function. In the deploy script only the address of the crowdsale contract is whitelisted.

KryllVesting contract is also ownable. The contract implements the KryllToken vesting logic for one address. `token` and `beneficiary` addresses are set in `setup` function. The vesting start time is time of calling `start` function. Cliff period (during this period tokens can not be vested) is 3 months since the vesting start time.

- `changeBeneficiary` function, which can be called only by the owner, changes the vesting beneficiary.
- Using `releasableAmount` function user can check how many tokens are available for him/her at the moment.
- Using `vestedAmount` function user can check how many tokens are available for him/her at the moment plus tokens that have already been released.
- User cannot take available tokens during the cliff period. After the cliff period tokens become available proportionally to the time since the beginning of the vesting period. The whole vesting period is 1 year. User can receive all the currently available tokens using `release` function. This function transfers tokens from vesting contract address to the user’s address.

Automated Analysis

We used several publicly available automated Solidity analysis tools. Here are the combined results of SmartCheck, Solhint, and Remix. Oyente has found no issues.

All the issues found by tools were manually checked (rejected or confirmed).

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

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.

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

Tool	Rule	False positives	True positives
Remix	Gas requirement of function high	20	
	Potential Violation of Checks-Effects-Interaction pattern	1	
	Potentially should be constant but is not	5	
	use of "now"	4	
	Variables have very similar names	2	
Total Remix		32	
SmartCheck	Address Hardcoded	7	
	Constant Functions		2
	Erc20 Approve		1
	Erc20 Transfer Should Throw	2	
	No Payable Fallback	3	

	Pragmas Version		3
	Private Modifier	1	
	Reentrancy External Call	2	
	Timestamp Dependence	1	
	Unchecked Math	8	
Total SmartCheck		24	6
Solhint	Avoid to make time-based decisions in your business logic	4	
	Compiler version must be fixed		3
	Possible reentrancy vulnerabilities. Avoid state changes after transfer	1	
Total Solhint		5	3
Total Overall		**61**	**9**

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 confirmed issues are described below.

Critical issues

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

The audit has shown no critical issues.

Medium severity issues

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

ERC20 approve issue

There is ERC20 approve issue (TransferableToken.sol, line142). We recommend instructing users not to use `approve` directly and to use `increaseApproval/decreaseApproval` functions instead. Or to change the approved amount to 0, wait for the transaction to be mined, and then to change the approved amount to the desired value — link.

Changing the approved amount from a nonzero value to another nonzero value allows a double spending with a front-running attack.

Low severity issues

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

Overpowered controller

Initially, only owner can transfer tokens. This means that, for instance, tokens will be available for vesting only after vesting contract is manually whitelisted by the owner. We recommend adding vesting contract's address to the whitelist in the deployment script.

Discrepancies with the whitepaper

There are several discrepancies between the code and the whitepaper:

1. According to the whitepaper, crowdsale volume is 40000000 KRL, however it is actually 40320000 KRL in the contract code.
2. Vesting procedure is not mentioned in the whitepaper, but it is implemented in the code.

We highly recommend explicitly describing all the functionality and all the token parameters' values in the documentation.

Pragmas version

Solidity source files indicate the versions of the compiler they can be compiled with. Example:

```
pragma solidity ^0.4.8; // bad: compiles w 0.4.8 and above
pragma solidity 0.4.8; // good: compiles w 0.4.8 only
```

We recommend following the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee. Besides, we recommend using the latest compiler version – 0.4.21 at the moment of the report.

Constant functions

We recommend using `view` instead of `constant`, which will be deprecated for functions. If a function is not supposed to modify the state or read from state, consider declaring it as `pure`.

```
function isInitialDistributionDone() public constant returns
(bool)
```

No constructor

We did not found the constructor in the KryllToken contract code, but some critical parameters are assigned by calling setup function:

- KryllToken.sol, line 63: `function reset()`

We recommend using constructor and deployment script for assigning all critical parameters.

Initialization

We discovered a strange rewriting of the addresses associated with the contract: `sale_address, team_address, advisors_address, security_address, press_address, user_acq_address, bounty_address`. Firstly, these addresses are explicitly initialized in the contract code, but then in the deploy script they are overwritten with other values. We highly recommend initializing the parameters of the token once and transparently.

Redundant code

There is redundant code in the following places of project:

1. KryllToken.sol, line 102
   The `initialDistributionDone`'s visiblity modifier can be changed into `public`. Thus, getter function will be generated automatically and there will be no need in the `isInitialDistributionDone` function.
2. TransferableToken.sol, line 127
   Since `transferable` state variable has public modifier, getter function is generated automatically and there is no need in `isTransferable` function.
3. KryllToken.sol
   We draw your attention to the fact, that using SafeMath library is excessive since there is no chance of under-/overflow in this contract.

We recommend removing this code. This will improve the contract logic and code readability.

Outdated OpenZeppelin library

The version of OpenZeppelin library used in the project is outdated and added to the repo. Best practice is connecting the library via npm, and using the latest version of OpenZeppelin library. However, we recommend taking into account possible compatibility problems.

Misleading comment

We found a contradictory comment in the code (KryllToken.sol, line 48):

```
// User Acquisition 6%
```

According to the comment, user acquisition is 6%, however it equals 14% in the code. We recommend fixing this comment for better code readability.

This analysis was performed by SmartDec.

Evgeniy Marchenko, Lead Developer
Alexander Seleznev, Chief Business Development Officer
Ivan Ivanitskiy, Chief Analytics Officer
Artyom Orlov, Project Manager
Igor Sobolev, Analyst
Alexander Drygin, Analyst

Sergey Pavlin, Chief Operating Officer 

April 25, 2018

Appendix

Compilation output

```
$ truffle compile
Compiling .\contracts\KryllToken.sol...
Compiling .\contracts\KryllVesting.sol...
Compiling .\contracts\Migrations.sol...
Compiling .\contracts\TransferableToken.sol...
Compiling zeppelin/contracts/math/SafeMath.sol...
Compiling zeppelin/contracts/ownership/Ownable.sol...
Compiling zeppelin/contracts/token/BasicToken.sol...
Compiling zeppelin/contracts/token/ERC20.sol...
Compiling zeppelin/contracts/token/ERC20Basic.sol...
Compiling zeppelin/contracts/token/StandardToken.sol...

Compilation warnings encountered:

zeppelin/contracts/ownership/Ownable.sol:20:3: Warning: Defining
constructors as functions with the same name as the contract is
deprecated. Use "constructor(...) { ... }" instead.
  function Ownable() public {
  ^ (Relevant source part starts here and spans across multiple
lines).
,/contracts/TransferableToken.sol:54:5: Warning: Defining
constructors as functions with the same name as the contract is
deprecated. Use "constructor(...) { ... }" instead.
    function TransferableToken()
    ^ (Relevant source part starts here and spans across multiple
lines).
,/contracts/Migrations.sol:11:3: Warning: Defining constructors as
functions with the same name as the contract is deprecated. Use
"constructor(...) { ... }" instead.
  function Migrations() public {
  ^ (Relevant source part starts here and spans across multiple
lines).
,zeppelin/contracts/token/BasicToken.sol:28:5: Warning: Invoking
events without "emit" prefix is deprecated.
    Transfer(msg.sender, _to, _value);
    ^-----------------------------^
,zeppelin/contracts/token/StandardToken.sol:37:5: Warning: Invoking
events without "emit" prefix is deprecated.
    Transfer(_from, _to, _value);
```

```
    ^-------------------------^
,zeppelin/contracts/token/StandardToken.sol:53:5: Warning: Invoking
events without "emit" prefix is deprecated.
    Approval(msg.sender, _spender, _value);
    ^-------------------------------------^
,zeppelin/contracts/token/StandardToken.sol:76:5: Warning: Invoking
events without "emit" prefix is deprecated.
    Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    ^-----------------------------------------------------------^
,zeppelin/contracts/token/StandardToken.sol:88:5: Warning: Invoking
events without "emit" prefix is deprecated.
    Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    ^-----------------------------------------------------------^
,zeppelin/contracts/ownership/Ownable.sol:40:5: Warning: Invoking
events without "emit" prefix is deprecated.
    OwnershipTransferred(owner, newOwner);
    ^-----------------------------------^
,/contracts/TransferableToken.sol:101:9: Warning: Invoking events
without "emit" prefix is deprecated.
        Transferable();
        ^------------^
,/contracts/TransferableToken.sol:109:9: Warning: Invoking events
without "emit" prefix is deprecated.
        UnTransferable();
        ^--------------^
,/contracts/KryllToken.sol:42:50: Warning: Initial value for
constant variable has to be compile-time constant. This will fail to
compile with the next breaking version change.
    uint256 constant internal DECIMAL_CASES    = (10 **
uint256(decimals));
                                                 ^------------------
-----^
,/contracts/KryllToken.sol:87:9: Warning: Invoking events without
"emit" prefix is deprecated.
        Transfer(0x0, owner, totalSupply);
        ^-------------------------------^
,/contracts/KryllVesting.sol:108:9: Warning: Invoking events without
"emit" prefix is deprecated.
        Released(unreleased);
        ^------------------^
,zeppelin/contracts/math/SafeMath.sol:9:3: Warning: Function state
mutability can be restricted to pure
  function mul(uint256 a, uint256 b) internal constant returns
(uint256) {
  ^ (Relevant source part starts here and spans across multiple
lines).
,zeppelin/contracts/math/SafeMath.sol:15:3: Warning: Function state
```

```
mutability can be restricted to pure
  function div(uint256 a, uint256 b) internal constant returns
(uint256) {
  ^ (Relevant source part starts here and spans across multiple
lines).
,zeppelin/contracts/math/SafeMath.sol:22:3: Warning: Function state
mutability can be restricted to pure
  function sub(uint256 a, uint256 b) internal constant returns
(uint256) {
  ^ (Relevant source part starts here and spans across multiple
lines).
,zeppelin/contracts/math/SafeMath.sol:27:3: Warning: Function state
mutability can be restricted to pure
  function add(uint256 a, uint256 b) internal constant returns
(uint256) {
  ^ (Relevant source part starts here and spans across multiple
lines).

Writing artifacts to .\build\contracts
```

Tests output

```
$ truffle test
Using network 'test'.



  Contract: Kryll Token
    Checking token initialization
      √ Owner initializes correctly
      √ Decimals initializes correctly
      √ Initial Distribution status initializes correctly
      √ Transferable status initializes correctly
    Checking token distribution
      √ Distribution config correctly (175ms)
      √ Distributes correctly (358ms)
      √ Fails to change distribution config if it has occurred
(202ms)
      √ Fails to redistribute if distribution it has occurred
(174ms)
    Transfer mechanism
      √ Should throw, transfer is disabled
      √ Should transfer correctly when transfer is enabled (136ms)
      √ Should throw when trying to transfer to 0x0 (41ms)
```

```
      √ Should throw when trying to transfer to negative amount
(57ms)
      √ Should show the transfer event
                Transfer
      √ Should throw when transfering more than owned (122ms)
      √ Should throw when transfer is called and transfers are re-
disabled (74ms)
      √ Owner can transfers even if transfers are disabled (64ms)
      √ Owner can whitelist addresses to bypass the transfer lock
(57ms)
      √ Other users cannot whitelist addresses
      √ Whitelisted address can bypass the transfer lock (57ms)
      √ Should throw when approve is called and transfers are
disabled (41ms)
      √ Allowance should be 0 after a transferFrom call when
transfers are disabled
      √ Should approve correctly when transfers are enabled (116ms)
      √ Should throw when transferFrom is called and transfers are
disabled (68ms)
      √ Should transferFrom correctly when transfers are enabled
(103ms)
      √ Should throw when transferFrom call is transfering more than
allowed/pending
      √ Can Approve, TransferFrom correctly if whitelisted and
transfers are disabled (213ms)
    Ownership mechanism
      √ Transfert Ownership  (50ms)
      √ Should throw when transfering ownership and not owner
anymore

  Contract: Kryll Vesting
    Checking vesting initialization
      √ Owner initializes correctly (75ms)
      √ Variables initializes correctly
      √ Initial started status initializes correctly
      √ Released amount initializes correctly
      √ Another account canont initializes the contract
      √ Owner can change the contract settings if not started (60ms)
      √ Owner cannot change the contract settings if vesting is
started (77ms)
      √ Only the Owner can change the beneficiary address when
vesting is started (81ms)
    Checking vesting allocation
      √ Contract token recieved
      √ No Token released
    Vesting mechanism
      √ Can't start if not setuped (73ms)
```

```
      √ Can't release if not started (53ms)
      √ Start vesting (42ms)
      √ Cannot be released before cliff (114ms)
        Releasable : 2154082.4657534244
      √ Can be released after cliff (53ms)
        Released : 2154082.4657534244
        Pending : 0
      √ Should transfer releasable tokens correctly (can be done by
anyone) (185ms)
      √ Beneficiary should receive the tokens correctly (49ms)
        Pending : 32.87671232876713
      √ Few tokens are releasable after a small period (42ms)
    Vesting flow
        Vested : 8640000
         D+0 -> released : 0
         D+15 -> released : 0
         D+30 -> released : 0
         D+45 -> released : 0
         D+60 -> released : 0
         D+75 -> released : 0
         D+90 -> released : 2130411.2328767125
         D+105 -> released : 2485479.7260273974
         D+120 -> released : 2840548.219178082
         D+135 -> released : 3195616.7123287674
         D+150 -> released : 3550685.479452055
         D+165 -> released : 3905753.9726027395
         D+180 -> released : 4260822.465753425
         D+195 -> released : 4615890.95890411
         D+210 -> released : 4970959.452054795
         D+225 -> released : 5326028.219178082
         D+240 -> released : 5681096.712328767
         D+255 -> released : 6036165.205479451
         D+270 -> released : 6391233.698630136
         D+285 -> released : 6746302.191780822
         D+300 -> released : 7101370.95890411
         D+315 -> released : 7456439.452054794
         D+330 -> released : 7811507.945205479
         D+345 -> released : 8166576.438356165
         D+360 -> released : 8521644.93150685
         D+375 -> released : 8640000
      √ should linearly release tokens during vesting period
(4089ms)
      √ should no longer own tokens
      √ All token have been recieved (39ms)
    Ownership mechanism
      √ Transfert Ownership  (41ms)
      √ Should throw when transfering ownership and not owner
```

```
anymore


  51 passing (11s)
```

Solhint output

```
KryllToken.sol
  28:17   warning  Compiler version must be
fixed                                          compiler-fixed
  30:8    error    Use double quotes for string
literals                                     quotes
  31:8    error    Use double quotes for string
literals                                     quotes
  51:20   error    Variable name must be in
mixedCase                                      var-name-mixedcase
  52:20   error    Variable name must be in
mixedCase                                      var-name-mixedcase
  53:20   error    Variable name must be in
mixedCase                                      var-name-mixedcase
  54:20   error    Variable name must be in
mixedCase                                      var-name-mixedcase
  55:20   error    Variable name must be in
mixedCase                                      var-name-mixedcase
  56:20   error    Variable name must be in
mixedCase                                      var-name-mixedcase
  57:20   error    Variable name must be in
mixedCase                                      var-name-mixedcase
  63:2    error    Line length must be no more than 120 but current
length is 196          max-line-length
  63:196  error    Open bracket must be indented by other
constructions by space            bracket-align
  96:9    warning  Possible reentrancy vulnerabilities. Avoid state
changes after transfer  reentrancy

KryllVesting.sol
   28:17  warning  Compiler version must be
fixed                                          compiler-fixed
   29:8   error    Use double quotes for string
literals                                   quotes
   30:8   error    Use double quotes for string
literals                                   quotes
   31:8   error    Use double quotes for string
literals                                   quotes
```

```
   38:1   error    Definition must be surrounded with two blank line
indent             two-lines-top-level-separator
   62:5   error    Definitions inside contract / library must be
separated by one line  separate-by-one-line-in-contract
   62:40  error    Comma must be separated from next element by
space                   space-after-comma
   62:73  error    Open bracket must be indented by other
constructions by space        bracket-align
   73:38  error    Open bracket must be indented by other
constructions by space        bracket-align
   76:21  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time
   91:5   error    Definitions inside contract / library must be
separated by one line  separate-by-one-line-in-contract
   91:70  error    Open bracket must be indented by other
constructions by space        bracket-align
   99:5   error    Definitions inside contract / library must be
separated by one line  separate-by-one-line-in-contract
  125:13  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time
  127:20  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time
  130:37  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time

Migrations.sol
   1:17  warning  Compiler version must be
fixed                           compiler-fixed
   3:1   error    Definition must be surrounded with two blank line
indent  two-lines-top-level-separator
   4:3   error    Expected indentation of 4 spaces but found
2              indent
   5:3   error    Expected indentation of 4 spaces but found
2              indent
   5:15  error    Variable name must be in
mixedCase                       var-name-mixedcase
   7:3   error    Expected indentation of 4 spaces but found
2              indent
   8:5   error    Expected indentation of 8 spaces but found
4              indent
   9:3   error    Expected indentation of 4 spaces but found
2              indent
  11:3   error    Expected indentation of 4 spaces but found
2              indent
  12:5   error    Expected indentation of 8 spaces but found
4              indent
  13:3   error    Expected indentation of 4 spaces but found
```

```
2               indent
  15:3   error    Expected indentation of 4 spaces but found
2               indent
  16:5   error    Expected indentation of 8 spaces but found
4               indent
  17:3   error    Expected indentation of 4 spaces but found
2               indent
  19:3   error    Expected indentation of 4 spaces but found
2               indent
  19:28  error    Function param name must be in
mixedCase                   func-param-name-mixedcase
  20:5   error    Expected indentation of 8 spaces but found
4               indent
  21:5   error    Expected indentation of 8 spaces but found
4               indent
  22:3   error    Expected indentation of 4 spaces but found
2               indent

TransferableToken.sol
   28:17  warning  Compiler version must be
fixed                                                          compi
ler-fixed
   29:8   error    Use double quotes for string
literals                                                      quotes
   30:8   error    Use double quotes for string
literals                                                      quotes
   38:44  error    Comma must be separated from next element by
space                                      space-after-comma
   54:5   error    Definitions inside contract / library must be
separated by one line                     separate-by-one-line-in-
contract
   55:9   error    Visibility modifier must be first in list of
modifiers                                   visibility-modifier-order
   69:5   error    Definitions inside contract / library must be
separated by one line                     separate-by-one-line-in-
contract
   86:9   error    Statement indentation is incorrect. Required
space after if                              statement-indent
   86:39  error    Open bracket must be on same line. It must be
indented by other constructions by space  bracket-align
   99:5   error    Definitions inside contract / library must be
separated by one line                     separate-by-one-line-in-
contract
   99:31  error    Visibility modifier must be first in list of
modifiers                                   visibility-modifier-order
  107:34  error    Visibility modifier must be first in list of
modifiers                                   visibility-modifier-order
```

```
  115:42  error    Visibility modifier must be first in list of
modifiers                                  visibility-modifier-order
  127:5   error    Definitions inside contract / library must be
separated by one line                      separate-by-one-line-in-
contract

✖ 62 problems (53 errors, 9 warnings)
```