Report

v. 3.0

Customer

RealT

Smart Contract Audit

RMM V3

26th February 2024

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	25.01.24	A. Zveryanskaya	Initial Draft
0.2	25.01.24	A. Zveryanskaya	Minor revision
1.0	25.01.24	A. Zveryanskaya	Release
1.1	26.02.24	A. Zveryanskaya	Project scope updated
1.2	26.02.24	A. Zveryanskaya	CVF-1,3 downgraded
1.3	26.02.24	A. Zveryanskaya	CVF-9 removed
2.0	26.02.24	A. Zveryanskaya	Release
2.1	26.02.24	A. Zveryanskaya	CVF-3 downgraded
3.0	26.02.24	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

RealT is a platform that offers simplified investments in real estate. The company mission is to democratize access to real estate investment opportunities, curated by a team of real estate professionals.

3 Project scope

We were asked to review the following files in the 83ddbe4 commit:

```
/
    RealTokenWrapper.sol    RealTokenShareholder Meeting.sol    RTW.sol
    ATokenRTW.sol
```

Additionally we have checked the difference between ATokenWithGovernance.sol and AToken.sol in the following commits:

- commit c38c627
- commit b82448a

Fixes were provided in the 7df56d3 commit.

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Minor issues** contain code style, best practices and other recommendations.

5 Our findings

We found 6 major, and a few less important issues. All identified Major issues have been fixed.

	Info	Fixed
Major	0	6
Moderate	2	3
Minor	71	29

Fixed 38 out of 112 issues

6 Major Issues

CVF-2. FIXED

- **Category** Suboptimal
- **Source** ATokenRTW.sol

Recommendation What happens here is effectively a transfer, so it would be better to emit a single transfer event from ”oldWallet” to ”newWallet” rather than a pair of burn and mint events.

Client Comment *Delete these 2 Transfer events since there are already Transfer event in _mintScaled and _burnScaled. If we add 2 more Transfer events in the function, this might make subgraphs update the event twice. Keep BurnAndMintByGovernance event, add liquidityIndex to event.*

```
336 emit Transfer(oldWallet, address(0), recoverAmount);
```

```
340 emit Transfer(address(0), newWallet, recoverAmount);
```

CVF-4. FIXED

- **Category** Unclear behavior
- **Source** RealTokenWrapper.sol

Description This function allows whitelisting assets that are already whitelisted.

Recommendation Consider forbidding this.

Client Comment *Does not affect functionality, should do to avoid redundant actions*

```
542 function whitelistAssets(address[] memory assets) external override
    ↪ onlyRole(DEFAULT_ADMIN_ROLE) {
```

CVF-5. FIXED

- **Category** Unclear behavior
- **Source** RealTokenWrapper.sol

Description This function allows unwhitelisting assets that aren't whitelisted.

Recommendation Consider forbidding this.

Client Comment *Does not affect functionality, should do to avoid redundant actions*

```
560 function unwhitelistAssets(
```

CVF-6. FIXED

- **Category** Unclear behavior
- **Source** RealTokenWrapper.sol

Description This loop should terminate once otherVote > yesVote - 1 ether.

Client Comment *Delete because RealTokenShareholderMeeting already checks this. There is no need to work correctly, should do for gas optimisation*

```
631 for (uint256 i = 1; i < meetingResults.length; ) {
```

CVF-7. FIXED

- **Category** Unclear behavior
- **Source** RealTokenShareholderMeeting.sol

Description This code should be executed only when block.timestamp > votingEnd and votingEnd > 0.

Client Comment *There is no need to work correctly, should do for gas optimisation*

```
343 uint256[] memory meetingResults = _resolutions[0].proposals;
    uint256 minimumVoteDifference = 1 ether;
    uint256 yesVote = meetingResults[0];
    uint256 otherVote;
    for (uint256 i = 1; i < meetingResults.length; ) {
      otherVote += meetingResults[i];
      unchecked {
350     ++i;
      }
    }
```

CVF-8. FIXED

- **Category** Unclear behavior
- **Source** RealTokenShareholderMeeting.sol

Description This loop should terminate in case otherVote >= yesVote - minimumVoteDifference.

Client Comment *There is no need to work correctly, should do for gas optimisation*

```
347 for (uint256 i = 1; i < meetingResults.length; ) {
```

7 Moderate Issues

CVF-1. INFO

- **Category** Unclear behavior
- **Source** ATokenRTW.sol

Description Here, the same error is reported for two different problems: i) an invalid signature and ii) a valid signature produced by a wrong owner.

Recommendation Consider reporting different errors in such two cases.

Client Comment *This is the original version of Aave code. As it does not affect the functionality of the code, we prefer not to change the original code except for security issues. This could be considered as minor issue since it does not pose a functional issue.*

```
228 require(owner == ecrecover(digest, v, r, s), Errors.
      ↪ INVALID_SIGNATURE);
```

CVF-10. INFO

- **Category** Unclear behavior
- **Source** ATokenRTW.sol

Description It is very suspicious that normal balance and scaled balance differ only for the Real token wrapper, but for all other accounts they are the same.

Recommendation Consider clearly explaining this logic in a documentation comment.

Client Comment *No need change, the balanceOf/scaledBalanceOf are correct. armmRTW fetch balance of user from RealTokenWrapper.getUserIndex. This function calculates the total value of RealTokens deposited into the wrapper by the user in RTW equivalent. When calling getUserIndex(wrapper) returns 0 because the wrapper does not deposit anything into the wrapper. However, the wrapper need intermediate balance for intermediary actions during a supply/withdraw/liquidationCall*

```
153 function balanceOf(
```

```
156   return (user == _realTokenWrapper) ? super.balanceOf(user) :
        ↪ getUserIndex(user);
```

```
160 function scaledBalanceOf(
```

```
163   return (user == _realTokenWrapper) ? super.scaledBalanceOf(user) :
        ↪  getUserIndex(user);
```

CVF-11. FIXED

- **Category** Unclear behavior
- **Source** RealTokenWrapper.sol

Description The uniqueness of the assets is not checked, which could lead to undesired behavior.

Recommendation Consider forbidding non-unique assets. A simple way to do so is to require the “collateralAssets” array to be sorted.

Client Comment *Require token array to be sorted in _validateRealTokenInputs() function.*

```
240 address[] memory collateralAssets,
```

CVF-12. FIXED

- **Category** Overflow/Underflow
- **Source** RealTokenWrapper.sol

Description Unsafe type conversion here could lead to underflow.

Recommendation Consider using safe conversion.

Client Comment *Check before conversion (chainlink price feed is always positive).*

```
501 _tokenPrice[assets[i]] = uint256(
```

CVF-13. FIXED

- **Category** Procedural
- **Source** RealTokenWrapper.sol

Description In the shareholder meeting code, in order for a vote to pass the number of yes votes should be strictly greater than the number of other votes plus 1 ether, while here is should be greater or equal.

Recommendation Consider making the logic consistent across contracts.

Client Comment *Delete because RealTokenShareholderMeeting already checks. Right, to modify.*

```
638 if (yesVote < otherVote + 1 ether)
```

8 Minor Issues

CVF-3. INFO

- **Category** Suboptimal
- **Source** RealTokenWrapper.sol

Description The storage slot of "_tokenBalanceOfUser[user][vars.realTokenAddress]" is calculated several times.

Recommendation Consider refactoring to calculate it only once.

Client Comment *There is no real need to change. It is not calculated serveral time. It was calculated differently in 2 cases (if/else).*

```
285 vars.realTokenBalanceOfUser = _tokenBalanceOfUser[user][vars.
      ↪ realTokenAddress];

300   _tokenBalanceOfUser[user][vars.realTokenAddress] =

305     _tokenBalanceOfUser[msg.sender][vars.realTokenAddress] += vars.
          ↪ realTokenAmount;

324   finalLiquidatorBalances[i] = _tokenBalanceOfUser[msg.sender][vars.
        ↪ realTokenAddress];

339   _tokenBalanceOfUser[user][vars.realTokenAddress] =

344     _tokenBalanceOfUser[msg.sender][vars.realTokenAddress] += vars.
          ↪ realTokenAmountFromRtw;

363   finalLiquidatorBalances[i] = _tokenBalanceOfUser[msg.sender][vars.
        ↪ realTokenAddress];
```

CVF-14. FIXED

- **Category** Procedural
- **Source** ATokenRTW.sol

Description Specifying a particular compiler version makes it harder migrating to newer versions.

Recommendation Consider specifying as "^0.8.0". Also relevant for: RTW.sol, RealTokenWrapper.sol, RealTokenShareholderMeeting.sol.

Client Comment *Use ^0.8.0 in .sol files and fix solc version in hardhat.*

```
2 pragma solidity 0.8.10;
```

CVF-15. INFO

- **Category** Procedural
- **Source** ATokenRTW.sol

Description We didn't review these files.

```
5  import {GPv2SafeERC20} from '../../dependencies/gnosis/contracts/
       ↪ GPv2SafeERC20.sol';

7  import {VersionedInitializable} from '../../protocol/libraries/aave-
       ↪ upgradeability/VersionedInitializable.sol';
   import {Errors} from '../../protocol/libraries/helpers/Errors.sol';
   import {WadRayMath} from '../../protocol/libraries/math/WadRayMath.
       ↪ sol';
10 import {IPool} from '../../interfaces/IPool.sol';
   import {IAToken} from '../../interfaces/IAToken.sol';
   import {IAaveIncentivesController} from '../../interfaces/
       ↪ IAaveIncentivesController.sol';
   import {IInitializableAToken} from '../../interfaces/
       ↪ IInitializableAToken.sol';
   import {ScaledBalanceTokenBaseRTW} from './base/
       ↪ ScaledBalanceTokenBaseRTW.sol';
   import {IncentivizedERC20RTW} from './base/IncentivizedERC20RTW.sol'
       ↪ ;
   import {EIP712Base} from '../../protocol/tokenization/base/
       ↪ EIP712Base.sol';
   import {IPriceOracleGetter} from '../../interfaces/
       ↪ IPriceOracleGetter.sol';
   import {IATokenRTW} from '../interfaces/IATokenRTW.sol';
   import {IRealTokenWrapper} from '../interfaces/IRealTokenWrapper.sol
       ↪ ';
20 import {IScaledBalanceToken} from '../../interfaces/
       ↪ IScaledBalanceToken.sol';
   import {TokenErrors} from '../libraries/helpers/TokenErrors.sol';
```

CVF-16. INFO

- **Category** Procedural
- **Source** ATokenRTW.sol

Description We didn't review these base contracts.

Client Comment *Needs to be verified carefully here. We used the same version of Aave ScaledBalanceTokenBase except omitting _approve() in transferFrom. This should be safe since ATokenRTW can only transferFrom users to Wrapper or Wrapper to users. The modifier onlyPoolOrWrapper is used in _transfer() function (ATokenRTW.sol in line 275) which is used in transfer() and transferFrom() function.*

```
29 VersionedInitializable,
30 ScaledBalanceTokenBaseRTW,
   EIP712Base,
   IAToken,
   ITokenRTW
```

CVF-17. INFO

- **Category** Suboptimal
- **Source** ATokenRTW.sol

Recommendation Making this constant public is redundant, as there is an explicit getter function for it.

Client Comment *Aave code, do not change except for security issues.*

```
42 uint256 public constant ATOKEN_REVISION = 0x2;
```

CVF-18. INFO

- **Category** Bad datatype
- **Source** ATokenRTW.sol

Recommendation The type for this variable should be "IERC20".

Client Comment *Aave code, do not change except for security issues.*

```
45 address internal _underlyingAsset;
```

CVF-19. INFO

- **Category** Bad datatype
- **Source** ATokenRTW.sol

Recommendation The type for this variable should be ”IRealTokenWrapper”.

```
46 address internal _realTokenWrapper;
```

CVF-20. INFO

- **Category** Suboptimal
- **Source** ATokenRTW.sol

Description The expression ”_msgSender()” is calculated several times.

Recommendation Consider calculating once and reusing.

Client Comment *Use msg.sender directly, no need to use _msgSender()*

```
49 if (_msgSender() != address(POOL) && _msgSender() !=
   ↪ _realTokenWrapper)
50   revert TokenErrors.CallerNotPoolOrWrapper(_msgSender());
```

CVF-21. INFO

- **Category** Suboptimal
- **Source** ATokenRTW.sol

Description The expression ”_msgSender()” is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *Use msg.sender directly, no need to use _msgSender()*

```
55 if (_msgSender() != _realTokenWrapper) revert TokenErrors.
   ↪ CallerNotWrapper(_msgSender());
```

CVF-22. INFO

- **Category** Bad datatype
- **Source** ATokenRTW.sol

Recommendation The type for this argument should be ”IERC20”.

Client Comment *Aave code, do not change except for security issues.*

```
78 address underlyingAsset,
```

CVF-23. INFO

- **Category** Procedural
- **Source** ATokenRTW.sol

Description We didn’t review the ”_calculateDomainSeparator” function.

Client Comment *Aave code, do not change except for security issues.*

```
94 _domainSeparator = _calculateDomainSeparator();
```

CVF-24. INFO

- **Category** Documentation
- **Source** ATokenRTW.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or documenting.

Client Comment *Aave code, do not change except for security issues.*

```
114 ) external virtual override onlyPool returns (bool) {
```

CVF-25. FIXED

- **Category** Procedural
- **Source** ATokenRTW.sol

Recommendation This commented out line should be removed.

Client Comment *Delete comment.*

```
157 // return super.balanceOf(user).rayMul(POOL.
    ↪ getReserveNormalizedIncome(_underlyingAsset));
```

CVF-26. INFO

- **Category** Bad datatype
- **Source** ATokenRTW.sol

Recommendation The return type should be ”IERC20”.

Client Comment *Aave code, do not change except for security issues.*

```
189 function UNDERLYING_ASSET_ADDRESS() external view override returns (
    ↪ address) {
```

CVF-27. INFO

- **Category** Bad datatype
- **Source** ATokenRTW.sol

Recommendation The type for the ”token” argument should be ”IERC20”.

Client Comment *Aave code, do not change except for security issues.*

```
294 function rescueTokens(address token, address to, uint256 amount)
    ↪ external override onlyPoolAdmin {
```

CVF-28. FIXED

- **Category** Bad datatype
- **Source** ATokenRTW.sol

Recommendation The value ”10000” should be a named constant.

```
320 (totalDebtETHOldWallet * 10000) /
```

CVF-29. INFO

- **Category** Overflow/Underflow
- **Source** ATokenRTW.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculation overflows.

Recommendation Consider using the ”mulDiv” function.

Client Comment *Not possible to overflow in our use case.*

```
320   (totalDebtETHOldWallet * 10000) /
      currentLiquidationThresholdOldWallet;
```

```
328 uint256 recoverAmount = (withdrawableAmountETHOldWallet * 10 ** 18)
      ↪ / tokenPrice;
```

CVF-30. FIXED

- **Category** Readability
- **Source** ATokenRTW.sol

Recommendation The value ”10 ** 18” could be rendered as ”1e18”.

```
328 uint256 recoverAmount = (withdrawableAmountETHOldWallet * 10 ** 18)
      ↪ / tokenPrice;
```

CVF-31. FIXED

- **Category** Readability
- **Source** ATokenRTW.sol

Recommendation The value ”10 ** 18” should be a named constant.

```
328 uint256 recoverAmount = (withdrawableAmountETHOldWallet * 10 ** 18)
      ↪ / tokenPrice;
```

CVF-32. INFO

- **Category** Bad datatype
- **Source** ATokenRTW.sol

Recommendation The return type should be ”IRealTokenWrapper”.

```
349 function getRealTokenWrapper() external view returns (address) {
```

CVF-33. FIXED

- **Category** Unclear behavior
- **Source** ATokenRTW.sol

Description This function always returns true.

Recommendation Consider returning nothing.

```
357 function updateUserState(address user, uint256 newBalance) external
      ↪ onlyWrapper returns (bool) {
```

```
369 function updateUsersStatesByAdmin(
```

CVF-34. INFO

- **Category** Suboptimal
- **Source** ATokenRTW.sol

Recommendation It would be more efficient to return a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

```
370 address[] memory users,
    uint128[] memory amounts
```

CVF-35. INFO

- **Category** Procedural
- **Source** RTW.sol

Description We didn't review these files.

```
4 import {IRTW} from '../interfaces/IRTW.sol';
```

```
9 import {IRealTokenWrapper} from '../interfaces/IRealTokenWrapper.sol
    ↪ ';
10 import {TokenErrors} from '../libraries/helpers/TokenErrors.sol';
```

CVF-36. INFO

- **Category** Bad datatype
- **Source** RTW.sol

Recommendation The type for this variable should be "IPool".

```
27 address private _pool;
```

CVF-37. INFO

- **Category** Bad datatype
- **Source** RTW.sol

Recommendation The type for this variable should be "IRealTokenWrapper".

```
28 address private _realTokenWrapper;
```

CVF-38. INFO

- **Category** Bad datatype
- **Source** RTW.sol

Recommendation The type for the "pool" argument should be "IPool".

```
46 function initialize(address admin, address pool, address
    ↪ realTokenWrapper) external initializer {
```

CVF-39. INFO

- **Category** Bad datatype
- **Source** RTW.sol

Recommendation The type for the “realTokenWrapper” should be “IReadTokenWrapper”.

```
46 function initialize(address admin, address pool, address
      ↪ realTokenWrapper) external initializer {
```

CVF-40. FIXED

- **Category** Procedural
- **Source** RTW.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
61 ) internal virtual override onlyRole(UPGRADER_ROLE) {}
```

CVF-41. INFO

- **Category** Procedural
- **Source** RealTokenWrapper.sol

Description We didn’t review these files.

```
5  import {GPv2SafeERC20} from '../dependencies/gnosis/contracts/
      ↪ GPv2SafeERC20.sol';
   import {IRTW} from './interfaces/IRTW.sol';
   import {IATokenRTW} from './interfaces/IATokenRTW.sol';
   import {IBridgeERC20} from './interfaces/IBridgeERC20.sol';
   import {IRealTokenPriceFeedMultiCurrency} from './interfaces/
      ↪ IRealTokenPriceFeedMultiCurrency.sol';
10 import {IPool} from '../interfaces/IPool.sol';
   import {IRealTokenWrapper} from './interfaces/IRealTokenWrapper.sol'
      ↪ ;
   import {WrapperErrors} from './libraries/helpers/WrapperErrors.sol';

15 import {IRealTokenShareholderMeeting} from './interfaces/
      ↪ IRealTokenShareholderMeeting.sol';
   import {IRealTokenDex} from './interfaces/IRealTokenDex.sol';
```

CVF-42. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for this variable should be “IRTW”.

```
32 address private _rtw;
```

CVF-43. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for his variable should be “IATokenRTW”.

```
33 address private _aRtw;
```

CVF-44. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for this variable should be “IRealTokenDex”.

```
34 address private _realTokenDex;
```

CVF-45. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for this variable should be “IERC20”.

```
35 address private _stablecoin;
```

CVF-46. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for this variable should be “IRealTokenShareholderMeeting”.

```
36 address private _shareholderMeetingAddress;
```

CVF-47. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for the second key of this mapping should be “IBridgeERC20”.

```
38 mapping(address => mapping(address => uint256)) private
      ↪ _tokenBalanceOfUser;
40 mapping(address => mapping(address => bool)) private
      ↪ _tokenIsSuppliedByUser;
```

CVF-48. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation It would be more efficient to merge these mappings into a single mapping whose keys are user accounts and tokens, and values are structs encapsulating the values of the original mappings.

Client Comment *Keep mapping separated, in case of upgrade in the future (more mapping) to avoid struct storage collision.*

```
38 mapping(address => mapping(address => uint256)) private
      ↪ _tokenBalanceOfUser;
40 mapping(address => mapping(address => bool)) private
      ↪ _tokenIsSuppliedByUser;
```

CVF-49. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The key type for this mapping should be “IBridgeERC20”.

```
42 mapping(address => uint256) private _tokenBalanceOfWrapper;
44 mapping(address => bool) private _isRealTokenWhitelisted;
46 mapping(address => address[]) private _tokenListOfUser;
48 mapping(address => uint256) private _tokenPrice;
50 mapping(address => bool) private _inRealTokenList;
```

CVF-50. INFO

- **Category** Suboptimal
- **Source** RealTokenWrapper.sol

Recommendation It would be more efficient to merge these mappings into a single mapping whose keys are tokens and values are structs encapsulating the values of the original mappings.

Client Comment *Keep mapping separated, in case of upgrade in the future (more mapping) to avoid struct storage collision.*

```
42 mapping(address => uint256) private _tokenBalanceOfWrapper;
44 mapping(address => bool) private _isRealTokenWhitelisted;
46 mapping(address => address[]) private _tokenListOfUser;
48 mapping(address => uint256) private _tokenPrice;
50 mapping(address => bool) private _inRealTokenList;
```

CVF-51. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for this variable should be “IERC20[]”.

```
52 address[] private _realTokenList;
```

CVF-52. FIXED

- **Category** Procedural
- **Source** RealTokenWrapper.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
80 function _authorizeUpgrade(address newImplementation) internal
   ↪ override onlyRole(UPGRADER_ROLE) {}
```

CVF-53. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for the “asset” arguments should be “IBridgeEC20”.

```
84   address asset,
```

```
160 function withdraw(address asset, uint256 amount, address to)
    ↪ external override returns (uint256) {
```

CVF-54. INFO

- **Category** Suboptimal
- **Source** RealTokenWrapper.sol

Description The storage position of "_tokenBalanceOfUser[onBehalfOf][asset]" is calculated twice.

Recommendation Consider calculating once and reusing. This would require changing the mapping value type to a struct, as Solidity doesn't support storage references to atomic types.

Client Comment *It is not calculated twice. It is 1 sload + 1 sstore.*

```
102 uint256 newRealTokenBalance = _tokenBalanceOfUser[onBehalfOf][asset]
       ↪  + amount;
    _tokenBalanceOfUser[onBehalfOf][asset] = newRealTokenBalance;
```

CVF-55. INFO

- **Category** Suboptimal
- **Source** RealTokenWrapper.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

```
117 address[] memory assets,
    uint256[] memory amounts,
```

```
191 address[] memory assets,
    uint256[] memory amounts,
```

```
240 address[] memory collateralAssets,
    uint256[] memory collateralAmounts,
```

CVF-56. FIXED

- **Category** Suboptimal
- **Source** RealTokenWrapper.sol

Description In case assets are not unique, these values could be odd.

Recommendation Consider reverting in case assets are non unique. A simple way to do so, is to require the “assets” array to be sorted.

Client Comment *Require token array to be sorted in _validateRealTokenInputs() function.*

143
```
newRealTokenBalances[i] = newRealTokenBalance;
```

217
```
newRealTokenBalances[i] = newRealTokenBalance;
```

CVF-57. INFO

- **Category** Documentation
- **Source** RealTokenWrapper.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or documenting.

Client Comment *Natspec inherits document from IRealTokenWrapper.*

160
```
function withdraw(address asset, uint256 amount, address to)
    ↪ external override returns (uint256) {
```

194
```
) external override returns (uint256) {
```

CVF-58. INFO

- **Category** Suboptimal
- **Source** RealTokenWrapper.sol

Description The storage position of "_tokenBalanceOfUser[msg.sender][asset]" is calculated twice.

Recommendation Consider calculating once and reusing. This would require changing the mapping value type to a struct, as Solidity doesn't support storage references to atomic types.

Client Comment *It is not calculated twice. It is 1 sload + 1 sstore.*

```
164 if (amount == 0) revert WrapperErrors.InvalidAmount(amount);
```

```
172 _tokenBalanceOfUser[msg.sender][asset] = newRealTokenBalance;
```

CVF-59. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for this argument should be "IERC20".

```
242 address debtAsset,
```

CVF-60. INFO

- **Category** Suboptimal
- **Source** RealTokenWrapper.sol

Recommendation It would be more efficient to allocate a single array of structs with three fields, rather than three parallel arrays.

```
257 uint256[] memory actualCollateralAmounts = new uint256[](length);
    uint256[] memory finalUserBalances = new uint256[](length);
    uint256[] memory finalLiquidatorBalances = new uint256[](length);
```

CVF-61. FIXED

- **Category** Readability
- **Source** RealTokenWrapper.sol

Recommendation The value “10 ** 8” could be rendered as “1e8”.

```
397 uint256 realTokenAmount = (amount * 10 ** 8) / realTokenPrice;
412 uint256 rtwAmount = (amount * realTokenPrice) / 10 ** 8;
468 return artwIndex / 10 ** 8;
484 return userIndex / 10 ** 8;
```

CVF-62. FIXED

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The value “10 ** 8” should be a named constant.

```
397 uint256 realTokenAmount = (amount * 10 ** 8) / realTokenPrice;
412 uint256 rtwAmount = (amount * realTokenPrice) / 10 ** 8;
468 return artwIndex / 10 ** 8;
484 return userIndex / 10 ** 8;
```

CVF-63. INFO

- **Category** Overflow/Underflow
- **Source** RealTokenWrapper.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculation overflows.

Recommendation Consider using the “mulDiv” function.

Client Comment *Not possible to overflow in our use case.*

```
397 uint256 realTokenAmount = (amount * 10 ** 8) / realTokenPrice;
```

```
412 uint256 rtwAmount = (amount * realTokenPrice) / 10 ** 8;
```

CVF-64. INFO

- **Category** Procedural
- **Source** RealTokenWrapper.sol

Recommendation Brackets are redundant.

Client Comment *Keep for readeability.*

```
397 uint256 realTokenAmount = (amount * 10 ** 8) / realTokenPrice;
```

```
412 uint256 rtwAmount = (amount * realTokenPrice) / 10 ** 8;
```

CVF-65. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for the “token” argument should be “IBridgeERC20”.

```
417 function getTokenBalanceOfUser(address user, address token) external
    ↪  view returns (uint256) {
```

CVF-66. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for the first returned value should be “IBridgeERC20[]”.

```
424 ) external view override returns (address[] memory, uint256[] memory
      ↪ ) {
```

```
442   returns (address[] memory, uint256[] memory)
```

CVF-67. INFO

- **Category** Suboptimal
- **Source** RealTokenWrapper.sol

Recommendation It would be more efficient to return a single array of structs with two fields, rather than two parallel arrays.

```
424 ) external view override returns (address[] memory, uint256[] memory
      ↪ ) {
```

```
442   returns (address[] memory, uint256[] memory)
```

CVF-68. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The argument type should be “IBridgeERC20[]”.

```
496 function updateTokenPrice(address[] memory assets) external onlyRole
      ↪ (UPDATER_ROLE) {
```

```
542 function whitelistAssets(address[] memory assets) external override
      ↪ onlyRole(DEFAULT_ADMIN_ROLE) {
```

```
561   address[] memory assets
```

```
574 function isRealTokenWhitelisted(address[] memory assets) external
      ↪ view returns (bool[] memory) {
```

CVF-69. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The argument type should be “IRTW”.

```
512 function setRtw(address rtw) external override onlyRole(
    ↪ DEFAULT_ADMIN_ROLE) {
```

CVF-70. FIXED

- **Category** Unclear behavior
- **Source** RealTokenWrapper.sol

Description These functions should emit some events.

```
512 function setRtw(address rtw) external override onlyRole(
    ↪ DEFAULT_ADMIN_ROLE) {
```

```
517 function setArtw(address artw) external override onlyRole(
    ↪ DEFAULT_ADMIN_ROLE) {
```

```
590 function activateLiquidationWithDex(
```

CVF-71. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The argument type should be “IATokenRTW”.

```
517 function setArtw(address artw) external override onlyRole(
    ↪ DEFAULT_ADMIN_ROLE) {
```

CVF-72. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The return type should be “IRTW”.

```
522 function getRtw() external view override returns (address) {
```

CVF-73. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The return type should be “IATokenRTW”.

```
527 function getArtw() external view override returns (address) {
```

CVF-74. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The return type should be “IPool”.

```
532 function getPool() external view override returns (address) {
```

CVF-75. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The argument type should be “IBridgeERC20”.

```
537 function getRealTokenPrice(address realToken) external view override
    ↪  returns (uint256) {
```

CVF-76. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type of the “token” argument should be “IERC20”.

```
586 function approveTokenToPool(address token, uint256 amount) external
      ↪ onlyRole(DEFAULT_ADMIN_ROLE) {
```

CVF-77. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for this argument should be “IRealTokenDex”.

```
591 address realTokenDex,
```

CVF-78. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The type for this argument should be “IERC20”.

```
592 address stablecoin
```

CVF-79. INFO

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The argument type should be “ IRealTokenShareholderMeeting”.

```
599 address shareholderMeetingAddress
```

CVF-80. FIXED

- **Category** Unclear behavior
- **Source** RealTokenWrapper.sol

Description This event is emitted even if nothing actually changed.

Client Comment *Fixed, revert when new address is the same.*

```
601 emit UpdateRealTokenShareholderMeeting(_shareholderMeetingAddress,
    ↪ shareholderMeetingAddress);
```

CVF-81. FIXED

- **Category** Bad datatype
- **Source** RealTokenWrapper.sol

Recommendation The value “1 days” should be a named constant.

```
624 if (block.timestamp <= votingEnd || block.timestamp >= votingEnd + 1
    ↪  days)
```

CVF-82. FIXED

- **Category** Unclear behavior
- **Source** RealTokenWrapper.sol

Description The function should revert immediately in case “yesVote” is less than “1 ether”.

```
629 uint256 yesVote = meetingResults[0];
```

CVF-83. INFO

- **Category** Procedural
- **Source** RealTokenShareholderMeeting.sol

Description We didn’t review these files.

```
41 import {IRealTokenShareholderMeeting} from '../interfaces/
    ↪ IRealTokenShareholderMeeting.sol';
import {RealTokenShareholderMeetingErrors} from '../libraries/
    ↪ helpers/RealTokenShareholderMeetingErrors.sol';
```

CVF-84. INFO

- **Category** Bad datatype
- **Source** RealTokenShareholderMeeting.sol

Recommendation The type of this variable should be more specific.

Client Comment *Not sure, we don't know which TOKEN will be voted.*

```
51 address public immutable TOKEN;
```

CVF-85. INFO

- **Category** Bad datatype
- **Source** RealTokenShareholderMeeting.sol

Recommendation The argument type should be more specific.

Client Comment *Not sure, we don't know which TOKEN will be voted.*

```
61 constructor(address token) {
```

CVF-86. INFO

- **Category** Suboptimal
- **Source** RealTokenShareholderMeeting.sol

Recommendation It would be more efficient to pass a single array of structs with three fields, rather than three parallel arrays. This would also make the length checks unnecessary.

```
86 bytes32[] calldata _names,
   bytes32[] calldata _urls,
   uint256[] calldata _proposalCounts
```

CVF-87. INFO

- **Category** Suboptimal
- **Source** RealTokenShareholderMeeting.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

```
130 address[] calldata voters,
    bytes32[] calldata voterIds
```

CVF-88. INFO

- **Category** Suboptimal
- **Source** RealTokenShareholderMeeting.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

```
150 bytes32[] calldata voterIds,
    uint256[] calldata weights
```

CVF-89. FIXED

- **Category** Suboptimal
- **Source** RealTokenShareholderMeeting.sol

Description The expression "_resolutions[resolutionId].votingEnd" is calculating twice.

Recommendation Consider calculating once and reusing.

```
200 _resolutions[resolutionId].votingEnd == 0 ||
    _resolutions[resolutionId].votingEnd <= block.timestamp
```

CVF-90. FIXED

- **Category** Suboptimal
- **Source** RealTokenShareholderMeeting.sol

Recommendation This could be simplified as: if (_voters[voterId].voted[resolutionId])

```
207 if (_voters[voterId].voted[resolutionId] != false)
```

CVF-91. FIXED

- **Category** Suboptimal
- **Source** RealTokenShareholderMeeting.sol

Description The expression "_voters[voterId]" is calculated several times.

Recommendation Consider calculating once and reusing.

```
207 if (_voters[voterId].voted[resolutionId] != false)
```

```
210 _voters[voterId].voted[resolutionId] = true;
```

```
213   _voters[voterId].weight;
emit Vote(voterId, resolutionId, proposalId, _voters[voterId].weight
    ↪ );
```

CVF-92. FIXED

- **Category** Suboptimal
- **Source** RealTokenShareholderMeeting.sol

Recommendation This could be simplified using the "+=" operator.

```
211 _resolutions[resolutionId].proposals[proposalId] =
    _resolutions[resolutionId].proposals[proposalId] +
```

CVF-93. FIXED

- **Category** Suboptimal
- **Source** RealTokenShareholderMeeting.sol

Description The expression "_voters[voterId]" is calculated several times.

Recommendation Consider calculating once and reusing.

```
220 if (voterId != 0x0 && _voters[voterId].voted[resolutionId] == false)
      ↪ {
    _voters[voterId].voted[resolutionId] = true;
```

```
224     _voters[voterId].weight;
    emit Vote(voterId, resolutionId, proposalId, _voters[voterId].
      ↪ weight);
```

CVF-94. FIXED

- **Category** Suboptimal
- **Source** RealTokenShareholderMeeting.sol

Recommendation This could be simplified using the "+=" operator.

```
222 _resolutions[resolutionId].proposals[proposalId] =
    _resolutions[resolutionId].proposals[proposalId] +
```

CVF-95. INFO

- **Category** Suboptimal
- **Source** RealTokenShareholderMeeting.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

```
309 address[] memory assets,
310 uint256[] memory amounts
```

CVF-96. FIXED

- **Category** Bad datatype
- **Source** RealTokenShareholderMeeting.sol

Recommendation This value should be a named constant.

```
344 uint256 minimumVoteDifference = 1 ether;
```

CVF-97. FIXED

- **Category** Unclear behavior
- **Source** RealTokenShareholderMeeting.sol

Description The function should immediately return false in case yesVote < minimumVote-Difference.

```
345 uint256 yesVote = meetingResults[0];
```

CVF-98. INFO

- **Category** Procedural
- **Source** ATokenWithGovernance.sol

Description We didn't review these files.

```
10 +import {VersionedInitializable} from '../../protocol/libraries/aave
      ↪ -upgradeability/VersionedInitializable.sol';
   +import {Errors} from '../../protocol/libraries/helpers/Errors.sol';
   +import {WadRayMath} from '../../protocol/libraries/math/WadRayMath.
      ↪ sol';

20 +import {ScaledBalanceTokenBase} from '../../protocol/tokenization/
      ↪ base/ScaledBalanceTokenBase.sol';
   +import {IncentivizedERC20} from '../../protocol/tokenization/base/
      ↪ IncentivizedERC20.sol';
   +import {EIP712Base} from '../../protocol/tokenization/base/
      ↪ EIP712Base.sol';
   +import {IPriceOracleGetter} from '../../interfaces/
      ↪ IPriceOracleGetter.sol';
   +import {IATokenWithGovernance} from '../interfaces/
      ↪ IATokenWithGovernance.sol';
   +import {IRealTokenShareholderMeeting} from '../interfaces/
      ↪ IRealTokenShareholderMeeting.sol';
   +import {TokenErrors} from '../libraries/helpers/TokenErrors.sol';
```

CVF-99. FIXED

- **Category** Procedural
- **Source** ATokenWithGovernance.sol

Recommendation Consider listing Aave as a co-author.

```
33 +* @author RealT
```

CVF-100. INFO

- **Category** Procedural
- **Source** ATokenWithGovernance.sol

Description This could collide with an AToken revision from Aave.

Recommendation Consider changing the constant name.

Client Comment *This is AToken revision constant from Aave.*

```
52 +uint256 public constant ATOKEN_REVISION = 0x2;
```

CVF-101. INFO

- **Category** Suboptimal
- **Source** ATokenWithGovernance.sol

Description Hardcoding mainnet addresses makes it harder to test contracts.

Recommendation Consider passing the initial governance admin address as a constructor argument and storing in an immutable variable.

Client Comment *This will need to change the initialize function and other Aave contract (Configurator), keep this for now.*

```
58 +address public constant INITIAL_GOVERNANCE_ADMIN = 0
      ↪ x5Fc96c182Bb7E0413c08e8e03e9d7EFc6cf0B099;
```

CVF-102. INFO

- **Category** Bad datatype
- **Source** ATokenWithGovernance.sol

Recommendation The type for this variable should be “IRealTokenShareholderMeeting”.

```
60 +address private _shareholderMeetingAddress;
```

CVF-103. FIXED

- **Category** Unclear behavior
- **Source** ATokenWithGovernance.sol

Description These events are emitted even if nothing actually changed.

```
326 +emit GovernanceAdminTransferred(_governanceAdmin,
    ↪ newGovernanceAdmin);
```

```
335 +emit ShareholderMeetingAddressUpdated(_shareholderMeetingAddress,
    ↪ shareholderMeetingAddress);
```

CVF-104. FIXED

- **Category** Documentation
- **Source** ATokenWithGovernance.sol

Description This comment is confusing.

Recommendation Consider rephrasing.

```
345 +// Check if the vote pass has passed (yes)
```

CVF-105. FIXED

- **Category** Bad datatype
- **Source** ATokenWithGovernance.sol

Recommendation The “1 days” value should be a named constant.

```
351 +if (block.timestamp > votingEnd + 1 days) revert TokenErrors.
    ↪ VotePassedMoreThan1Day();
```

CVF-106. FIXED

- **Category** Bad datatype
- **Source** ATokenWithGovernance.sol

Recommendation The value “1000” should be a named constant.

```
393 +(totalDebtETHOldWallet * 10000) /
```

CVF-107. INFO

- **Category** Overflow/Underflow
- **Source** ATokenWithGovernance.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculation overflows.

Recommendation Consider using the “mulDiv” function.

Client Comment *Not possible to overflow in our use case.*

```
393 +(totalDebtETHOldWallet * 10000) /
    +currentLiquidationThresholdOldWallet;
```

```
403 +? (amount * tokenPrice) / 10 ** (this.decimals())
```

CVF-108. INFO

- **Category** Procedural
- **Source** ATokenWithGovernance.sol

Recommendation Brackets are redundant.

Client Comment *Keep for readeability.*

```
393 +(totalDebtETHOldWallet * 10000) /
```

CVF-109. INFO

- **Category** Procedural
- **Source** ATokenWithGovernance.sol

Recommendation Brackets around “amount * tokenPrice” and around “this.decimals()” are redundant.

Client Comment *Keep for readeability.*

```
403 +? (amount * tokenPrice) / 10 ** (this.decimals())
```

CVF-110. FIXED

- **Category** Procedural
- **Source** ATokenWithGovernance.sol

Recommendation The value “10 ** this.decimals()” should be precomputed.

```
403 +? (amount * tokenPrice) / 10 ** (this.decimals())
```

CVF-111. INFO

- **Category** Suboptimal
- **Source** ATokenWithGovernance.sol

Description This is effectively a transfer.

Recommendation Consider emitting a single “Transfer” event from “oldWallet” to “newWallet” rather than a pair or burn and mint events.

Client Comment *Deleted those event since there are Transfer events in _mintScaled/_burnScaled functions already*

```
411 +emit Transfer(oldWallet, address(0), actualWithdrawAmount);
```

```
414 +emit Transfer(address(0), newWallet, actualWithdrawAmount);
```

CVF-112. INFO

- **Category** Bad datatype
- **Source** ATokenWithGovernance.sol

Recommendation The return type should be “IRealTokenShareholderMeeting”.

```
420 +function getRealTokenShareholderMeeting() external view returns (
      ↪ address) {
```

ABDK
Consulting

About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting