Report

v. 1.0

Customer

RealT

Smart Contract Audit

RealTokenWrapper

21th September 2024

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	21.09.24	A. Zveryanskaya	Initial Draft
0.2	21.09.24	A. Zveryanskaya	Minor revision
1.0	21.09.24	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 modifications in the RealTokenWrapper contract :

- Original Code
- Code with Fixes

Files:

```
wrapper/
        RealTokenWrapper.sol
```

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.
- **Recommendations** contain code style, best practices and other suggestions.

5 Our findings

We provided Client with a few recommendations.

Moderate	Info	Fixed
	2	7

Fixed 7 out of 9 issues

6 Moderate Issues

CVF-1. FIXED

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

Description These comments explain code that doesn't exist anymore. Using comments to track and document code changes is a bad practice, as it makes code cumbersome and harder to read.

Recommendation Remove these comments and let Git track code changes and their reasoning.

Client Comment *We agree with this remark. The issue is fixed.*

```
170 +// Delete check _isRealTokenWhitelisted here since users can
        ↪ withdraw after unwhitelisting
171 +// if (!_isRealTokenWhitelisted[asset]) revert WrapperErrors.
        ↪ TokenNotWhitelisted(asset);
```

```
174 +// Here if the old balance > 0, it means the token was whitelisted
```

CVF-2. FIXED

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

Description The expression "_tokenBalanceOfUser[user]" is calculated on every loop iteration.

Recommendation Calculate once before the loop.

Client Comment *We agree with this remark. The issue is fixed.*

```
484 490 userIndex += _tokenBalanceOfUser[user][realTokenAddress] *
        ↪ _tokenPrice[realTokenAddress];
```

CVF-3. 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 Use the "mulDiv" function or some other way to prevent phantom overflow.

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

```
686 +(vars.totalRepayUSDOfRepayWallet * percent) /
687 +(10 ** 18 * PERCENTAGE_FACTOR);
```

```
724 +    (vars.debtAmount * vars.totalRepayUSDOfRepayWallet) /
725 +    vars.totalDebtUSDOfRepayWallet;
```

```
791 +  (_tokenBalanceOfUser[oldWallet][vars.realTokenAddress] * percent)
    ↪  /
792 +  PERCENTAGE_FACTOR;
```

CVF-4. INFO

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

Description In ERC20 the "decimals" property is used by UI to render token amounts in a human readable way. Using this property in smart contracts is discouraged.

Recommendation Treat all token amounts as integers.

Client Comment *This decimals changes with the decimals of underlying assets so it is mandatory to adjust it for each token. Here, we think it is reasonable to use decimals in the calculation.*

```
702 +10 ** IERC20Detailed(vars.debtAssetAddress).decimals()) /
```

CVF-5. FIXED

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

Description Transferring tokens from the payer to this contract and then immediately transferring the same amount of the same tokens to the refund wallet is waste of gas.

Recommendation Transfer directly from the payer to the refund wallet.

Client Comment *We agree with this remark. The issue is fixed.*

```
707 +IERC20(vars.debtAssetAddress).safeTransferFrom(payer, address(this)
        ↪ , vars.debtAmount);
708 +IERC20(vars.debtAssetAddress).safeTransfer(refundWallet, vars.
        ↪ debtAmount);
```

CVF-6. FIXED

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

Description The value "_tokenBalanceOfUser[oldWallet]" is calculated on every loop iteration.

Recommendation Calculate once before the loop.

Client Comment *We agree with this remark. The issue is fixed.*

```
829 +_tokenBalanceOfUser[oldWallet][vars.realTokenAddress] -= vars.
        ↪ realTokenAmount;
```

CVF-7. FIXED

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

Description The value "_tokenBalanceOfUser[newWallet]" is calculated on every loop iteration.

Recommendation Calculate once before the loop.

Client Comment *We agree with this remark. The issue is fixed.*

```
839 +_tokenBalanceOfUser[newWallet][vars.realTokenAddress] += vars.
        ↪ realTokenAmount;
```

CVF-8. FIXED

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

Description The value "_tokenIsSuppliedByUser[newWallet]" is calculated on every loop iteration.

Recommendation Calculate once before the loop.

Client Comment *We agree with this remark. The issue is fixed.*

```
842 +if (!_tokenIsSuppliedByUser[newWallet][vars.realTokenAddress]) {
```

```
844 +  _tokenIsSuppliedByUser[newWallet][vars.realTokenAddress] = true;
```

CVF-9. FIXED

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

Description The value "_tokenListOfUser[newWallet]." is calculated on every loop iteration.

Recommendation Calculate once before the loop.

Client Comment *We agree with this remark. The issue is fixed.*

```
843 +_tokenListOfUser[newWallet].push(vars.realTokenAddress);
```

7 Minor Issues

CVF-10. INFO

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

Description We didn't review these files.

```
5   +import {IERC20Detailed} from '../dependencies/openzeppelin/
       ↪ contracts/IERC20Detailed.sol';
```

```
17  +import {IPriceOracleGetter} from '../interfaces/IPriceOracleGetter.
       ↪ sol';
```

CVF-11. INFO

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

Recommendation Once a list of token addresses is read from the storage all at once, it would be event more efficient to store the list in compact form, i.e. using 20 bytes per address, rather than 32 bytes. This would require a bit of assembly to deal with such storage format.

Client Comment *It is nice to have but we decided to use simple Solidity here.*

```
483 +// Cache user token list for gas optimization
484 +address[] memory userTokenList = _tokenListOfUser[user];
```

CVF-12. FIXED

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

Recommendation A better name for the error would be "ZeroLength" or "EmptyArray".

Client Comment *We agree with this remark. The issue is fixed.*

```
655 +if (vars.recoverLength == 0) revert WrapperErrors.InvalidLength(
       ↪ vars.recoverLength);
```

```
658 +if (vars.debtLength == 0) revert WrapperErrors.InvalidLength(vars.
       ↪ debtLength);
```

CVF-13. INFO

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

Description The error includes the value, which is invalid, but gives no clue to what argument or variable contains this very value.

Recommendation Use different errors for different arguments, and reflect the argument names in the names of corresponding errors.

Client Comment *If the transaction fails, then the revert message will give information about the "percent" value. This will allow us to know if the percent is zero or superior than PERCENTAGE_FACTOR. It should be enough to debug the failed transaction.*

```
660 +if (percent == 0 || percent > PERCENTAGE_FACTOR) revert
       ↪ WrapperErrors.InvalidValue(percent);
```

CVF-14. INFO

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

Description The error name is confusing, as one could think that there could be "valid zero addresses", which is not true.

Recommendation Rename the error.

Client Comment *In somecase, it is ok to have zero address as argument. In this case, the token address can not be zero so the name "InvalidZeroAddress" is reasonable.*

```
663 +revert WrapperErrors.InvalidZeroAddress();
```

```
673 +if (vars.realTokenAddress == address(0)) revert WrapperErrors.
       ↪ InvalidZeroAddress();
```

CVF-15. INFO

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

Recommendation Brackets around multiplication are redundant.

Client Comment *We think the brackets are good for readeability.*

```
686 +(vars.totalRepayUSDOfRepayWallet * percent) /
```

```
701 +    (vars.totalRepayUSDOfRepayWallet *
702 +      10 ** IERC20Detailed(vars.debtAssetAddress).decimals()) /
```

```
724 +    (vars.debtAmount * vars.totalRepayUSDOfRepayWallet) /
```

```
791 +  (_tokenBalanceOfUser[oldWallet][vars.realTokenAddress] * percent)
    ↪  /
```

CVF-16. FIXED

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

Description The error name is too generic and gives no clue regarding what exactly is wrong with the amounts.

Recommendation Rename the error.

Client Comment *We agree with this remark. The issue is fixed.*

```
816 +revert WrapperErrors.InvalidAmounts(vars.actualRtwRecoverAmount,
    ↪ vars.rtwRecoverAmount);
```

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