PUBLIC

Code Assessment

# of the Exchange Smart Contracts

Jan 24, 2025

Produced for

by CHAINSECURITY

Contents

1 Executive Summary 3

2 Assessment Overview 5

3 Limitations and use of report 7

4 Terminology 8

5 Findings 9

6 Informational 10

1 Executive Summary

Dear Decentralized USD team,

Thank you for trusting us to help Decentralized USD with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of Exchange according to Scope to support you in forming an opinion on their security risks.

Decentralized USD implements an USDD Exchange contract which facilitates the one way exchange from the old TRC-20 USDD token to the new USDD token with a 1:1 exchange rate.

The most critical subjects covered in our audit are asset solvency, functional correctness, and access control.

The general subjects covered are gas optimizations and specifications.

Security regarding all the aforementioned subjects is high.

In summary, we find that the codebase provides a good level of security.

It is important to note that security audits are time-boxed and cannot uncover all vulnerabilities. They complement but don't replace other vital measures to secure a project.

The following sections will give an overview of the system, our methodology, the issues uncovered, and how they have been addressed. We are happy to receive questions and feedback to improve our service.

Sincerely yours,

ChainSecurity

1.1 Overview of the Findings

Below we provide a brief numerical overview of the findings and how they have been addressed.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	0

2 Assessment Overview

In this section, we briefly describe the overall structure and scope of the engagement, including the code commit which is referenced throughout this report.

2.1 Scope

The assessment was performed on the source code files inside the Exchange repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	16 Dec 2024	beace0de00ea7b7f7c6ff352a1d2550bebb3c202	Initial Version

For the solidity smart contracts, the compiler version `0.8.20` was chosen.

The following files were in the scope of this review:

```
usdd-exchange/
    contracts/
        USDDExchange.sol
        utils/
            Address.sol
            SafeTRC20.sol
```

2.1.1 *Excluded from scope*

Any other file not explicitly mentioned in the scope section is considered out of scope. In particular, the tests, external dependencies and configurations files are not part of this audit. The tokens used especially the old USDD token and their mechanisms / tokenomics are not in the scope of this review.

2.2 System Overview

This system overview describes the initially received version (**Version 1**) of the contracts as defined in the Assessment Overview.

Furthermore, in the findings section, we have added a version icon to each of the findings to increase the readability of the report.

Decentralized USD offers an USDD Exchange contract which facilitates the one way exchange from the old TRC-20 USDD token to the new USDD token with a 1:1 exchange rate.

The existing TRC-10 USDD is governed by the Tron DAO Reserve (TRD) via multisigs which oversees its issuance and conversion into TRC-20 USDD, a wrapped version of TRC-10 USDD. The circulation supply (TRC-20 USDD) is backed by TRX and stablecoins (i.e. USDT, USDC).

The new TRC-20 USDD is a fork of Maker DAI multi-collateral stablecoin.

In the USDDExchange contract, `exchange()` is the main entrypoint for users to exchange old USDD to new USDD. Before being called by users, this contract should be filled with sufficient new USDD tokens, and users should grant sufficient allowance to this contract. If not paused, this function will transfer the old USDD from the `msg.sender` and send the new USDD to the `msg.sender`. Exchanging zero amount is prevented.

The following privileged functions are also implemented and can only be called by the owner:

1. `withdrawOldUSDD()`: withdraw old USDD tokens to a specified `_to` address.
2. `withdrawNewUSDD()`: withdraw new USDD tokens to a specified `_to` address.
3. `recoverToken()`: the owner can recover any TRC-20 tokens (except old and new USDD) to a specified address `_to`.
4. `recoverTRX()`: the native token TRX can be recovered to a specified `_to` address.
5. `pause()`: pause the system and prevent calls to `exchange()`.
6. `unpause()`: unpause the system so `exchange()` can be called.

In addition, the ownership transfer will be achieved in two steps:

1. `transferOwnership()`: it starts the ownership transfer to a non-zero new address by setting it to the pending owner.
2. `acceptOwnership()`: the pending owner should call this explicitly to accept the ownership transfer.
3. `renounceOwnership()`: it is forbidden to renounce the ownership.

2.2.1 Roles & Trust Model

The new USDD token is assumed to be the fork of Maker DAI multi-collateral stablecoin with 18 decimals (the same as the old USDD).

The Tron DAO Reserve (TDR) is fully trusted and never misbehave, otherwise unbacked TRC-20 USDD may be released into circulation to exchange for the new USDD.

The owner is fully trusted otherwise:

1. There may be insufficient new USDD for exchange.
2. The contract can be paused so no one can exchange.
3. The old and new USDD can be withdrawn to any designated address.

3 Limitations and use of report

Security assessments cannot uncover all existing vulnerabilities; even an assessment in which no vulnerabilities are found is not a guarantee of a secure system. However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it. Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code assessment aimed at determining all locations that need to be fixed. Within the customer-determined time frame, ChainSecurity has performed an assessment in order to discover as many vulnerabilities as possible.

The focus of our assessment was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself. Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.

4 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- *Likelihood* represents the likelihood of a finding to be triggered or exploited in practice
- *Impact* specifies the technical and business-related consequences of a finding
- *Severity* is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

Likelihood	Impact		
	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

5 Findings

In this section, we describe our findings. The findings are split into these different categories:

Below we provide a numerical overview of the identified findings, split up by their severity.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	0

6 Informational

We utilize this section to point out informational findings that are less severe than issues. These informational issues allow us to point out more theoretical findings. Their explanation hopefully improves the overall understanding of the project's security. Furthermore, we point out findings which are unrelated to security.

6.1 Gas Optimizations

Informational **Version 1** **Acknowledged**

CS-USDD-EXCH-001

In `exchange()`, the old USDD balance of `msg.sender` and new USDD balance of this contract are both checked to be greater than the amount required. These checks are redundant as both will be checked in the token `transfer()` / `transferFrom()` implementations.

Acknowledged:

Decentralized USD has acknowledged this optimization and decided not to change it.

6.2 Unchecked Token Decimals

Informational **Version 1** **Acknowledged**

CS-USDD-EXCH-002

The exchange assumes a 1:1 conversion rate between the old USDD and new USDD based on the assumption that they have the same decimals, however, this is never checked in the constructor.

Acknowledged:

Decentralized USD has acknowledged this issues and decided not to change the code with the following response:

```
When deploying the exchange contract, both the old USDD and the new token
decimals will be checked to ensure they are equal to 18.
```

6.3 Unused Functions

Informational **Version 1** **Acknowledged**

CS-USDD-EXCH-003

The following functions in the libraries are never used:

1. `Address.toPayable()`
2. `Address.sendValue()`
3. `SafeTRC20.safeApprove()`
4. `SafeTRC20.safeIncreaseAllowance()`
5. `SafeTRC20.safeDecreaseAllowance()`

Acknowledged:

Decentralized USD has acknowledged this issue and decided not to change it.