PUBLIC

Code Assessment

of the USDS Wrappers Smart Contracts

September 04, 2024

Produced for

by

Contents

1 Executive Summary

Dear all,

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

MakerDAO implements a permissionless wrapper to swap USDS using the PSM-lite.

The most critical subjects covered in our audit are functional correctness and security. The general subjects covered include usability, gas efficiency, and documentation.

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 USDS Wrappers 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	11 June 2024	e501197974840cff321d7ae7465dcdb97410d4b2	Initial Version
2	31 July 2024	4056ade8fa218f73f74fd2c8c4119e473f8ef279	Compatibility Getters
3	28 August 2024	e57d0c43c397d69596cbb6a1580cc450988f2d04	Renaming

For the solidity smart contracts, the compiler version `0.8.21` was chosen.

The following contract is in the scope of this review:

```
src/NstPsmWrapper.sol
```

As of version 3, the files have been renamed as a result of a rebranding. The files below are in scope:

```
src/UsdsPsmWrapper.sol
```

2.1.1 Excluded from scope

Any file not explicitly listed above including tests are out of 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.

MakerDAO offers a permissionless wrapper to swap NST using the PSM-lite.

The wrapper provides two functions for batching actions:

- `sellGem()`: transfers gem tokens from `msg.sender` in order to sell them to the PSM for DAI. Using the VAT and the join adapters this DAI amount is exchanged for NST and exited to the address specified.
- `buyGem()`: Calculates the amount of NST required taking into account the fee of the PSM. Transfers this amount of NST tokens from `msg.sender`. Using the VAT and the join adapters these NST tokens are exchanged for DAI, these DAI tokens are then used to buy gem for the specified address in the PSM.

2.2.1 Changes in Version 2

Getters have been added to provide partial backward compatibility with the Lite PSM. Notably, `dai()` returns the address of NST and `gemJoin()` the address of this wrapper. Note that `daiJoin()` is not available due unclearness of where it should point to.

2.2.2 Changes in Version 3

NST has been renamed to USDS.

2.2.3 Trust Model & Roles

This contract is a wrapper only and features no privileged roles.

The gem token is expected to be a standard ERC-20 token without special behaviour.

The lite-psm is assumed to be deployed with DAI.

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