PUBLIC

Code Assessment

of the NST Deployment Scripts

October 11, 2023

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 NST according to Scope to support you in forming an opinion on their security risks.

MakerDAO implements a stable token and an exchange contract for exchanging DAI against the new token in a 1:1 ratio. This audit report reviews the security and correctness of the corresponding deployment scripts.

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

Security regarding all the aforementioned subjects is high.

In summary, we find that the codebase provides a high 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 NST repository based on the documentation files.

The following deployment scripts are part of the scope of this review:

1. deploy/NstDeploy.sol
2. deploy/NstInit.sol
3. deploy/NstInstance.sol

The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	18 September 2023	8dea00f1545925dce63511e1710e2ec3c7fc59d2	Initial Version

For the solidity smart contracts, the compiler version `0.8.16` was chosen.

2.1.1 Excluded from scope

Any other file not explicitly mentioned in the scope section. In particular tests, scripts, external dependencies, and configuration files are not part of the audit scope.

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 an ERC-20 contract that is going to replace MakerDAO's DAI token. A contract for integration of the token in the MakerDAO ecosystem as well as a contract for permissionlessly exchanging DAI against the new token are supplied along with it.

2.2.1 Contracts

`Nst` is the new token contract.

`NstJoin` can mint and burn tokens on the `Nst` contract after users have given approval (`vat.hope()`) in the underlying MakerDAO core system.

`DaiNst` allows any user to exchange their DAI token to the new token or exchange the new token back to DAI.

2.2.2 Deployment overview

The systems are deployed in two steps:

1. Some EOA deploys the contracts and - if necessary - changes the owner of these contracts to the `PauseProxy`.
2. A governance `Spell` with quorum executes the initialization of the contracts through the `PauseProxy`.

Contract deployment

After deployment, the owner of the `Nst` contract is changed to the `PauseProxy`. `NstJoin` is deployed with the given `Nst` address and the address of the `vat`. `DaiNst` is then deployed with the address of the `DaiJoin` and newly deployed `NstJoin` contracts.

During the initialization of the contracts, the deployment parameters are checked for validity. The owner of the `Nst` contract is switched from the `PauseProxy` to the `NstJoin` contract so that it is allowed to call `mint()` and `burn()`.

Finally, the addresses are added to the chainlog.

2.2.3 Roles & Trust Model

The contracts are deployed without proxy.

`Nst`, `NstJoin` and `DaiNst` operate trustlessly as long as no extra `ward` is added to the `nst` contract.

The underlying system is subject to MakerDAO governance.

Deployers are supposed to deploy the contracts as specified by the reviewed script. Deployers are, however, EOAs that could perform unlawful actions besides simple deployment, such as changing the settings of the system or granting themselves special privileges. It is important that after deployment, concerned parties thoroughly check the state of the deployed contracts to ensure that no unexpected action has been taken on them during deployment.

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 Notes

We leverage this section to highlight further findings that are not necessarily issues. The mentioned topics serve to clarify or support the report, but do not require an immediate modification inside the project. Instead, they should raise awareness in order to improve the overall understanding.

6.1 Deployment Verification

Note Version 1

Since deployment of the contracts is not performed by the governance directly, special care has to be taken that all contracts have been deployed correctly. While some variables can be checked upon initialization through the `PauseProxy`, some things have to be checked beforehand.

We therefore assume that all mappings in the deployed contracts are checked for any unwanted entries (by verifying the bytecode of the contract and then looking at the emitted events). This is especially crucial for `wards` mappings.