PUBLIC

Code Assessment

of the NGT Smart Contracts

October 17, 2023

Produced for

by

Contents

1 Executive Summary 3
2 Assessment Overview 5
3 Limitations and use of report 7
4 Terminology 8
5 Findings 9
6 Notes 10

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

MakerDAO implements a rebranded version of the MKR token and an immutable converter with a fixed conversion rate. The converter functions by minting and burning tokens. The new governance token is ERC-20 compliant and the converter allows for permissionless conversion between MKR and NGT tokens.

The most critical subjects covered in our audit are security, functional correctness and seamless integration with the existing system. 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 NGT repository. No documentation was made available for the review.

The scope consists of the two solidity smart contracts:

1. ./src/MkrNgt.sol
2. ./src/Ngt.sol

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

V	Date	Commit Hash	Note
1	11 May 2023	f05f38322aa9910e2e708157c27aa251db0bd72d	Initial Version
2	16 October 2023	2d675f802ac00ed2fb436d5311da415fdb774b50	Updated 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.

MakerDAO implements a new governance token (NGT, a rebranded version of the MKR token) and an immutable converter which converts MKR to NGT at a fixed rate (1 MKR:1200 NGT) and vice versa.

2.2.1 NGT

NGT is an ERC-20 compliant token with 18 decimals. The contract is controlled by privileged roles `wards` initialized with `msg.sender` in the constructor. Any address in `wards` has privileged access to:

- Add a new ward by `rely()`.
- Remove a ward by `deny()`.
- Mint any amount of NGT tokens to an address by `mint()`.

Token transfers work the same way as a normal ERC-20 token but with a few restrictions. Specifically, transfers to the zero address (address(0)) or the contract itself are not allowed. A user can also burn its tokens by calling `burn()` with its own address. In case the address specified is different to the `msg.sender`, the user will burn on behalf of others if its allowance is sufficient.

NGT supports unlimited allowance by approving `max(uint256)`. In addition, `permit()` is provided for setting allowance with signatures either from an EOA or a contract (EIP-1271). A contract can give permission to a spender by implementing `isValidSignature()` with customized verification logic. If the signature length does not equal to 65 bytes, it is assumed the allowance owner is a contract, which will be queried for signature validation.

2.2.2 MkrNgt

MkrNgt is an immutable and permissionless converter for NGT and MKR tokens with a fixed conversion rate, which is expected to be 1 MKR to 1200 NGT (`rate=1200`). It only provides two functions:

- `mkrToNgt()`: burns `msg.sender`'s MKR tokens (`mkrAmt`) and mints NGT tokens(`mkrAmt*rate`) to the specified receiver.
- `ngtToMkr()`: burns `msg.sender`'s NGT tokens (`ngtAmt`) and mints MKR tokens (`ngtAmt/rate`) to the specified receiver. This rounds down if `ngtAmt` is not a multiple of `rate`.

To achieve the functionalities above, MkrNgt requires the following privileges:

- be authorized by the MKR contract. This is required for MKR minting and burning.
- be a member of wards of NGT contract for NGT minting.

Users must give allowance to the contract for the tokens to be burned successfully.

In addition, although MkrNgt is non-stoppable, functions above could still revert if the MKR contract is paused (stopped in the contract's terminology).

Changes in Version 2

- Functions `increaseAllowance` and `decreaseAllowance` have been removed. Only functions `approve` and `permit` remain to modify allowances.
- Permit functionality: Now, when validating a signature with a contract, if there is no code deployed at the given address, the transaction will revert with a clear error message. Previously the transaction would just revert.

2.2.3 Roles and Trust Model

Wards of NGT are the only privileged role. Besides managing the addition and removal of wards, their privileges also includes:

- Directly minting new NGT tokens.
- Indirectly diluting MKR by minting new NGT tokens.

It is expected that MkrNgt is the only ward of NGT or at minimum the sole ward minting NGT tokens. Otherwise, MKR tokens and MakerDAO can be manipulated if additional NGT tokens are minted and converted to MKR by malicious wards.

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 MkrNgt Converter Can Be Paused if MKR Is Paused

Note Version 1

The MkrNgt converter itself is permissionless, however, if MKR token is paused, the converter will be indirectly paused as `mint()` and `burn()` will revert on MKR.