Mangrove DAO - Token Audit

This document serves as a security review of Mangrove DAO's token implementation that inherits the semi-transferable token implementation which was built by a joint effort between Mangrove DAO and Morpho DAO and is located here. Omniscia has performed an audit of the token in the past which we cross-referenced when producing this report for Mangrove DAO.

Overview

The scope of the audit is constrained **to the Mangrove token implementation** and does not constitute an audit of other modules present in the Mangrove DAO codebase, such as their "Mangrove Chief" or "Vesting Vault" which are indicated by the contract's deployment scripts.

Over the course of the audit, we validated that the Mangrove token (`MGV`) has properly made use of the semi-transferable `Token` implementation co-implemented by Morpho DAO & Mangrove DAO via a local git-module that points to the latest commit of Morpho DAO's codebase, `d80c603` . While this commit differs from the one we covered in our separate audit (`51d657bcfa`), no changes were performed to the Solidity code retaining the security guarantees of our original audit.

In detail, the scope of the audit is as follows:

Files in Scope	Repository	Commit(s)
MangroveToken.sol (MTN)	mangrove-token	fa04a41830,efea503944

Verdict

The Mangrove DAO team evaluated the exhibits laid out in the report and opted to retain the current behaviour of the codebase, performing no code changes in the latest commit hash we covered. Given that the Mangrove DAO assessed the manual review finding we identified and has decided to retain the decreasing supply paradigm, we consider the exhibit as acknowledged with no other outstanding manual review issues present in the report.

Compilation

The project utilizes `foundry` as its development pipeline tool, containing an array of tests and scripts coded in Solidity.

To compile the project, the `build` command needs to be issued via the `forge` CLI tool:

```
forge build
```

The `forge` tool automatically selects version `0.8.17` based on the version specified in the `foundry.toml` configurational file.

The project contains discrepancies with regards to the Solidity version used as the `pragma` statement of the `MangroveToken` contract is open-ended (`^0.8.13`).

We advise it to be locked to `0.8.17` (`=0.8.17`), the same version utilized by our static analysis as well as optimizational review of the codebase.

During compilation with the `foundry` pipeline, no errors were identified that relate to the syntax or bytecode size of the contracts.

While version `0.8.17` is a relatively too recent version to deploy a production contract, multiple vulnerabilities have been identified and patched between versions `0.8.13` and `0.8.17` of Solidity.

As such, we consider the current compiler version adequate for the purposes of the Mangrove DAO's token.

Static Analysis

The static analysis toolkit we utilized identified two valid results for the codebase one of which pertains to the unchecked `_owner` argument of the `constructor`. The other static analysis finding is covered in the `Compilation` chapter and refers to the unlocked `pragma` version.

MTN-01S: Inexistent Sanitization of Input Address

Type	Severity	Location
Input Sanitization	Minor	MangroveToken.sol:L9

Description:

We inspected the code path of the `_owner` argument (`MangroveToken` @ Magrove DAO -> `Token` @ Morpho -> `RolesAuthority` @ Morpho -> `Auth` @ Morpho) and properly validated that no sanitization is present for the `address` argument.

Impact:

The presence of zero-value addresses, especially in `constructor` implementations, can cause the contract to be permanently inoperable. These checks are advised as zero-value inputs are a common side-effect of off-chain software related bugs.

Recommendation:

We advise the `_owner` argument of the `MangroveToken` constructor to be validated as non-zero, ensuring that misconfigurations of the contract cannot occur.

Alleviation:

The Mangrove DAO team evaluated this exhibit and will not provide an alleviation for it as they deem the current behaviour desirable.

Manual Review

The `MangroveToken` itself is simplistic in nature, defining configurational values for the `Token` constructor by Morpho DAO. These values are as follows:

- Name: "Mangrove Token"
- Symbol (Ticker): "MGV"
- Decimals: 18
- Owner: `_owner` Argument

The implementation of the token is mintable by authorized parties as well as burnable by any party.

MTN-01M: Potentially Undesirable Token Behaviour

Type	Severity	Location
Indeterminate Code	Unknown	MangroveToken.sol:L4

Description:

Given that the token's supply can be controlled in a downward action by any party due to the public availability of the `burn` function, the Mangrove DAO team should consider whether they wish this capability to be present in the code.

Impact:

Other ecosystem contracts of Mangrove DAO (such as their `Mangrove Chief` implementation) may rely on a perpetually increasing `totalSupply` which could be susceptible to an attack as the token's `totalSupply` can be reduced by anyone in custody of a non-zero Mangrove Token balance.

Recommendation:

If permitting `burn` operations to be performed by anyone for their own tokens is undesirable, we advise the Mangrove DAO team to adequately `override` the `burn` function limiting or preventing its execution.

Alleviation:

The Mangrove DAO team evaluated this exhibit and will not provide an alleviation for it as they deem the current behaviour desirable.

Code Style

The code is developed using a basic `constructor` structure with literal values passed in to the constructor of the `Token` implementation the contract inherits.

As a legibility enhancement, we would advise the `Token` constructor to be annotated as to what each index-based argument represents (i.e. `Token(string memory name, string memory symbol, uint8 decimals, address owner)`).

Conclusion

We advise the Mangrove DAO team to consider adding a zero-address check for the `constructor` of the `MangroveToken` as identified during the static analysis section of the report as well as to re-evaluate whether they wish to expose the `burn` function to arbitrary parties. All findings identified in the report do not represent an active threat and as such can be safely ignored as long as the `burn` function's exposure is considered the token's expected behaviour.