CANTINA

Maple Finance

Security Review

Cantina Managed review by:

0xleastwood, Lead Security Researcher

kankodu, Security Researcher

September 4, 2023

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Directly* exploitable security vulnerabilities that need to be fixed.
High	Security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All high issues should be addressed.
Medium	Objective in nature but are not security vulnerabilities. Should be addressed unless there is a clear reason not to.
Low	Subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. When determining the severity one first needs to determine whether the finding is subjective or objective. All subjective findings are considered of Minor severity.

Next it is determined whether the finding can be regarded as a security vulnerability. Some findings might be objective improvements that need to be fixed, but do not impact the project's security overall (Medium).

Finally, objective findings of security vulnerabilities are classified as either critical or major. Critical findings should be directly vulnerable and have a high likelihood of being exploited. Major findings on the other hand may require specific conditions that need to be met before the vulnerability becomes exploitable.

2 Security Review Summary

Maple Finance is an institutional crypto-capital network built on Ethereum and Solana. Maple provides the infrastructure for credit experts to efficiently manage and scale crypto lending businesses and connect capital from institutional and individual lenders to innovative, blue-chip companies. Built with both traditional financial institutions and decentralized finance leaders, Maple is transforming capital markets by combining industry-standard compliance and due diligence with the frictionless lending enabled by smart contracts and blockchain technology.

From Aug 14th to Aug 23rd the Cantina team conducted a review of mplv2 on commit tag v1.0.0-rc.1. The team identified a total of **5** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 0
- Gas Optimizations: 2
- Informational: 3

3 Findings

3.1 Gas Optimization

3.1.1 `onlyGovernor` and `onlyScheduled` Modifiers Can be Combined

Severity: Gas Optimization

Context: MapleToken.sol#L16-L30

Description: Both of these modifiers are solely used together so there is no need to separate the logic of the *two* when a single `GLOBALS_SLOT SLOAD` can saved.

Recommendation: Consider removing the `governor()` function altogether and in-lining its behaviour under a newly defined `onlyGovernorAndScheduled` modifier. A `GLOBALS_SLOT` address read is cached and used to retrieve the token's relevant governor address.

```
modifier onlyGovernorAndScheduled(bytes32 functionId_) {
    IGlobalsLike globals_         = IGlobalsLike(globals());
    require(msg.sender == globals_.governor(), "MT:NOT_GOVERNOR");
    bool         isScheduledCall_ = globals_.isValidScheduledCall(msg.sender, address(this), functionId_,
    ↪   msg.data);

    require(isScheduledCall_, "MT:NOT_SCHEDULED");

    globals_.unscheduleCall(msg.sender, functionId_, msg.data);

    _;
}
```

Maple: Acknowledged, Optimisation PR is linked with the fix.

Cantina: Verified fix in PR 60.

3.1.2 `setImplementation` Can be Modified to Avoid Unnecessary Check and `SLOAD`

Severity: Gas Optimization

Context: MapleTokenProxy.sol#L33

Description: The `setImplementation()` function can be optimised slightly by checking for revert conditions earlier on in the code.

In this instance, `msg.sender` can be checked against the governor address at the beginning of the code and the internal `_governor()` function can be in-lined and removed altogether.

Recommendation: Consider implementing the following changes:

```
function setImplementation(address newImplementation_) override external {
    IGlobalsLike globals_         = IGlobalsLike(_globals());
+   require(msg.sender == globals_.governor(), "MTP:SI:NOT_GOVERNOR");
    bool         isScheduledCall_ = globals_.isValidScheduledCall(msg.sender, address(this),
    ↪   "MTP:SET_IMPLEMENTATION", msg.data);

-   require(msg.sender == _governor(), "MTP:SI:NOT_GOVERNOR");
    require(isScheduledCall_,          "MTP:SI:NOT_SCHEDULED");

    globals_.unscheduleCall(msg.sender, "MTP:SET_IMPLEMENTATION", msg.data);

    _setAddress(IMPLEMENTATION_SLOT, newImplementation_);

    emit ImplementationSet(newImplementation_);
}


...


- function _governor() internal view returns (address admin_) {
-     admin_ = IGlobalsLike(_globals()).governor();
- }
```

Maple: Acknowledged, Optimisation PR is linked with the fix.

Cantina: Verified fix in PR 59.

3.2 Informational

3.2.1 Claim Schedule Can be Enforced Within `RecapitalizationModule`

Severity: Informational

Context: RecapitalizationModule.sol#L93-L108

Description: An `onlyClaimer` modifier is used to restrict who can claim unissued tokens according to the schedule of recapitalization. While it is noted that the Maple team wishes to hold this role in an effort to reduce token volatility, the claim schedule can also be enforced entirely within the contract. There is no concern of this role being lost as governance has complete control over the delegation of this role.

Recommendation: Ensure this claim schedule is well-documented to users. Otherwise, it may be worthwhile to enforce a time schedule. Although, it would likely involve added complexity as this should ideally be a governance controlled parameter.

Maple: Acknowledged, as mentioned above we will manage claiming via the claimer role and manage the 'claiming schedule' off-chain as opposed to codifying this logic.

Cantina: Acknowledged.

3.2.2 Proxy Initialization Pattern Via Contract Could be Potentially Dangerous

Severity: Informational

Context: MapleTokenInitializer.sol, MapleToken.sol

Description: While the current use of proxy initialization is safe, it is important to point out that the equivalent `MapleTokenInitializer` needs to have a storage layout which completely mirrors that of `MapleToken`, otherwise we may be accessing the wrong storage slots.

Recommendation: Ensure this is noted for future use of this pattern. It may be worthwhile to verify the correct token mint amounts in `MapleTokenProxy.constructor()`.

Maple: Acknowledged, we will keep this in mind for any future upgrades.

Cantina: Acknowledged.

3.2.3 Inconsistency in `DOMAIN_SEPARATOR` Version Calculation in MPLv2 Token

Severity: Informational

Context: MapleToken.sol#9

Description: In `ERC20Proxied`, `DOMAIN_SEPARATOR` is calculated with `version = keccak256(bytes("1"))`.

When USDC upgraded from FiatTokenV1 to FiatTokenV2, in FiatTokenV2 the version was 2, even though there was no signature verification in `FiatTokenV1` at all.

There is no possibility of signature replay for MPLv1 and MPLv2, as both have different names and addresses (verifying contract). However, users are aware that there are two versions and might expect the `DOMAIN_SEPARATOR` of MPLv2 token to be calculated with version = `keccak256(bytes("2"))`, as there is a precedent of widely used `USDC` doing this.

Recommendation: Use `version = keccak256(bytes("2"))` while calculating the `DOMAIN_SEPARATOR` for MPLv2 token.

Maple: Thanks for the suggestion as you mentioned as our token is being redeployed completely and MPLv1 is not being upgraded there isn't an issue of a signature replay as the address will be different. We'll choose not to update the version number in MPLv2 and doing so would result in slightly more messy code with the overriding needed for the `DOMAIN_SEPARATOR`. We will keep this in mind for any future change of implementation as a proxy is now being used!

Cantina: Acknowledged.