PUBLIC

Code Assessment

of the Controllers for TrueFi Carbon Smart Contracts

March 07, 2023

Produced for

archblock

truefi

by

CHAINSECURITY

Contents

1 Executive Summary

Dear Archblock Team,

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

Archblock implements some modified controllers to be used with the TrueFi Carbon protocol.

The most critical subjects covered in our audit are access control, security of the funds and ERC4626 compliance. Only minor issues were uncovered. All the issues are addressed in the second iteration of the codebase. The security of all aforementioned subjects is high.

The general subjects covered are code complexity, gas efficiency, documentation and testing. 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	1
• Code Corrected	1

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 Controllers for TrueFi Carbon repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

contracts-multi-withdrawal-controller

V	Date	Commit Hash	Note
1	16 February 2023	ca6da68c230e43a18cce8c1effc0d1dade4c2de5	Initial Version
2	02 March 2023	679c11e55ca7b3f0529608517ab579ff165560a3	Second Version

contracts-minimum-deposit-controller

V	Date	Commit Hash	Note
1	16 February 2023	6d2994bcaa9597a2dbbbe1c26073c45a2d01bdb5	Initial Version

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

The contracts in scope are:

- `contracts-minimum-deposit-controller/contracts/`:
  - `MinimumDepositController/MinimumDepositController.sol`
  - `MinimumDepositController/IMinimumDepositController.sol`
- `contracts-multi-withdrawal-controller/contracts/`:
  - `MultiWithdrawalController/MultiWithdrawalController.sol`
  - `MultiWithdrawalController/IMultiWithdrawalController.sol`

As these contracts are an extension of already audited controllers, only the extra features introduced are considered in scope.

2.1.1 *Excluded from scope*

All the contracts explicitly not mentioned in scope are excluded. Moreover, the controllers are used by TrueFi Carbon which has been covered by a previous audit. Hence, Carbon is considered out-of-scope. All the extra libraries used, such as the ones from OpenZeppelin, are also considered out-of-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.

Archblock implements a Deposit and a Withdrawal controller for the TrueFi Carbon system. The controllers are implementing some changes in the logic of the default Controllers of Carbon.

2.2.1 MinimumDepositController

It implements the same logic as the `DepositController` of Carbon but additionally enforces a minimum deposit limit for the users.

2.2.2 MultiWithdrawalController

The extra feature it introduces, compared to the Carbon `WithdrawController`, is the ability for one authorized user, namely the manager of the controller, to withdraw assets during a state of the portfolio where withdrawals are not normally allowed (`multiRedeem`). For this to happen, users must have approved the controller in order for it to withdraw their funds. Moreover, the `withdrawFeeRate` has been removed as there are no fees for withdrawals by users. A fee is only paid during `multiRedeem` and is determined by the manager.

2.2.3 Roles and Trust Model

The only role defined by the controllers is the manager. The manager is allowed to set all the parameters of the system. Moreover, in the case of the withdrawal controller, they are allowed to call `multiRedeem` and arbitrarily set the price of the tranche shares. The managers of the controllers are trusted roles by the system and, thus, they are expected to never act maliciously.

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 any open findings. Findings that have been resolved have been moved to the Resolved Findings section. The findings are split into these different categories:

- Design: Architectural shortcomings and design inefficiencies

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 Resolved Findings

Here, we list findings that have been resolved during the course of the engagement. Their categories are explained in the Findings section.

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	1

- EIP-4626 Non-Compliance Code Corrected

6.1 EIP-4626 Non-Compliance

Design Low Version 1 Code Corrected

The functions `MultiWithdrawalController.maxWithdraw` and `maxRedeem` return values greater than 0 when the withdrawals are not allowed in the current status of the protocol. This is in violation of the following rule:

> MUST return the maximum amount of assets that could be transferred from owner through withdraw and not cause a revert, which MUST NOT be higher than the actual maximum that would be accepted (it should underestimate if necessary).

Additionally, `MultiWithdrawalController._globalMaxWithdraw` sets the maximum amount of tokens that can be withdrawn from a tranche. This maximum is determined by the function `TrancheVault.totalAssets`. In `Live` state, this function returns the current waterfall value of the tranche which contains the `virtualTokenBalance` of the entire portfolio, as well as the value of active loans. The returned value can therefore be higher than the actual amount of assets that are available for withdrawal.

Code corrected:

`MultiWithdrawalController.maxWithdraw` and `maxRedeem` now return `0` if withdrawals are disallowed in the current status of the protocol.

7 Informational

We utilize this section to point out informational findings that are less severe than issues. These informational issues allow us to point out more theoretical findings. Their explanation hopefully improves the overall understanding of the project's security. Furthermore, we point out findings which are unrelated to security.

7.1 Redundant Event Emission

Informational **Version 1**

A manager can configure the `floor` and the `withdrawalAllowed` mapping by calling `configure`. In case the parameters are the same as the ones already set, the execution of the actual setter i.e., `setFloor` and `setWithdrawalAllowed`, is skipped. However, the manager can call `setFloor` and `setWithdrawalAllowed` directly, where there are no checks if the new values are different than the ones stored. In this case, redundant events will be emitted.

8 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.

8.1 No Asset Conversion

Note **Version 1**

`MultiWithdrawalController.onRedeem` does not check whether the given `assetAmount` of an exception matches `convertToAssets(sharesAmount)`. If the manager makes a mistake or a repayment is executed on the contract between the time the manager sends their `multiRedeem` transaction and the time the transaction actually executes, the values will be wrong, resulting in either a loss or a gain for the given lender. This behavior is well documented by the specification provided to us.

8.2 `Redeem` Event Emission

Note **Version 1**

`MultiWithdrawalController.onRedeem` can be called by any user (without reverting) using the following arguments:

- `sender`: The address of the controller contract.
- `shares`: 0.
- `owner`: `address(0)`.

This emits a `Redeem` event every time. The `assets` parameter can be completely arbitrary. Off-chain systems reading these events should be aware of this behavior.