PUBLIC

Code Assessment

of the yDiscount Smart Contracts

Aug 29, 2023

Produced for

by

Contents

1 Executive Summary

Dear Yearn Team,

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

Yearn implements a program allowing Yearn contributors to buy `YFI` at a discount each month, the discount is subject to the duration of their `veYFI` lock and the purchased `YFI` are immediately locked into `veYFI` according to the contributor's current lock.

During this assessment, we did not uncover any severe issues and 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	2
• Code Corrected	1
• Specification Changed	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 yDiscount repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	17 July 2023	d81212a46da5d8efeb16638c6cd889507f274baa	Initial Version
2	08 August 2023	9f5725acb2c8368cdfe178a67cfe2a99f14ed606	Second Version

For the Vyper smart contracts, the compiler version `0.3.7` was chosen.

The only file in scope is `Discount.vy`.

2.1.1 Excluded from scope

All test and mock files are excluded from the 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.

Yearn offers a possibility for contributors to the Yearn protocol to purchase `YFI` at a discount given the current market price. The discount is computed from the user's current `veYFI` lock in such a way that the longer the lock duration is, the greater the discount will be. The purchased `YFI` are immediately locked into `veYFI`, keeping the same duration.

The `Discount` contract is the entry point of the system and offers the following state modifying functions:

- `set_team_allowances`: can only be called by the `management` of the contract. This function can be used to distribute allowances to different teams that will later be able to distribute these allowances to their respective contributors using `set_contributor_allowances`. The management can specify if the allowances that are being given are for a new month, in which case, all previously unused allowances are invalidated and can no longer be used both by teams and contributors.
- `set_contributor_allowance` can be only called by addresses that have a team allowance greater than `0` and that is still valid (has not expired and has been given for the current month). Given a list of contributors and their respective allowances, the team allowance is distributed to the contributors.
- `buy`: Can be called by anyone having a non-null allowance that is still valid (has not expired and has been given for the current month). It essentially let a user buy `YFI` at a discount for some ETH, and immediately locks it into veYFI, keeping the same lock duration.

- If the contributor is buying `YFI` for himself and has a lock duration of at least 4 weeks, a discount between 10% and 60% relative to his lock duration is given.
- If the contributor is buying `YFI` for someone else (the specified account does not match the message sender), a special rule applies: a discount of 10% is given if the lock of the account has a remaining duration of at least 104 weeks (2 years), as opposed to the previous case, the discount does not scale with the lock duration.

Once the discount has been computed, the amount of `YFI` that can be bought with the ETH sent along the call is computed using the discount factor and the price of `YFI` in `ETH` obtained using Chainlink and/or Curve oracles. The ETH is sent directly to the `management` and the purchased `YFI` are locked int `veYFI`.

- `withdraw`: Withdraw any ERC20 token from the contract, can only be called by the `management`.

Important notes:

- A contributor's address can be part of multiple teams, in which case, the allowances given by each team are added to obtain the contributor's allowance.
- All allowances expire after 30 days or whenever a new allowance is set, whichever happens first.

2.3 Trust Model

The `management` is fully trusted and expected to behave honestly and correctly by correctly giving the allowances and funding the contract with enough `YFI` to cover the contributors' purchases.

The oracles are trusted in returning accurate rates for the pair YFI/ETH.

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	2

- Missing Indexing of Events Code Corrected
- Race Condition on Team Allowance Specification Changed

6.1 Missing Indexing of Events

Design Low Version 1 Code Corrected

CS-YRNDSCNT-004

The event `NewMonth` contains no indexed fields. The event's field `month` is a specific number. Yearn might consider indexing it if needed.

Code corrected

The field `month` was indexed in the event `NewMonth`.

6.2 Race Condition on Team Allowance

Design Low Version 1 Specification Changed

CS-YRNDSCNT-005

If the management calls `set_team_allowances` a second time during the same month while a team has some allowance left, similar to the well-documented issue with the ERC20 `approve` function, it is possible for a team to front-run the transaction to spend its remaining allowance before the management set its allowance to the new amount.

Specification changed

Yearn highlighted the trust assumption that the team is a fully trusted party. Misbehaving will lead to disqualification from participating in the program.

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 Event Reentrancy

Informational **Version 1**

CS-YRNDSCNT-001

In the function `buy`, the callback is done before logging the event, in the case that the call would reenter the contract, it would be possible to have events emitted out of order.

7.2 Gas Optimizations

Informational **Version 1**

CS-YRNDSCNT-002

In `set_contributor_allowances`, `self.expiration` is read from storage once before entering the loop and then once at each iteration of the loop, caching it in memory would avoid several SLOAD.

7.3 Inconsistency of the Interface `ChainlinkOracle`

Informational **Version 1**

CS-YRNDSCNT-003

While the Chainlink documentation specifies that the return type of `decimals` is an `uint8`, the interface `ChainlinkOracle` defines the function `decimals` as returning an `uint256`. Although a `uint8` will always fit in a `uint256`, it would be more consistent to use `uint8` as described in the documentation.