Security Audit Report

AshSwap farm
MultiversX smart contract

by ARDA

on December 15, 2023

Table of Contents

Disclaimer

The report makes no statements or warranties, either expressed or implied, regarding the security of the code, the information herein or its usage. It also cannot be considered as a sufficient assessment regarding the utility, safety and bugfree status of the code, or any other statements.

This report does not constitute legal or investment advice. It is for informational purposes only and is provided on an "as-is" basis. You acknowledge that any use of this report and the information contained herein is at your own risk. The authors of this report shall not be liable to you or any third parties for any acts or omissions undertaken by you or any third parties based on the information contained herein.

Terminology

Code: The code with which users interact.

Inherent risk: A risk for users that comes from a behavior inherent to the code's design.

Inherent risks only represent the risks inherent to the code's design, which are a subset of all the possible risks. **No inherent risk doesn’t mean no risk.** It only means that no risk inherent to the code's design has been identified. Other kind of risks could still be present. For example, the issues not fixed incur risks for the users, or the upgradability of the code might also incur risks for the users.

Issue: A behavior unexpected by the users or by the project, or a practice that increases the chances of unexpected behaviors to appear.

Critical issue: An issue intolerable for the users or the project, that must be addressed.

Major issue: An issue undesirable for the users or the project, that we strongly recommend to address.

Medium issue: An issue uncomfortable for the users or the project, that we recommend to address.

Minor issue: An issue imperceptible for the users or the project, that we advise to address for the overall project security.

Objective

Our objective is to share everything we have found that would help assessing and improving the safety of the code:

1. The **inherent risks** of the code, labelled R1, R2, etc.
2. The **issues** in the **code**, labelled C1, C2, etc.
3. The **issues** in the **testing** of the code, labelled T1, T2, etc.
4. The **issues** in the **other** parts related to the code, labelled O1, O2, etc.
5. The **recommendations** to address each issue.

Audit Summary

Initial scope

- **Repository:** https://github.com/ashswap/ash-exchange-sc
- **Commit:** `50757b865b7dfb41c9084b90682dac90e01c4b71`
- **MultiversX smart contract path:** ./contracts-0.39.8/dex/farm/

Final scope

- **Repository:** https://github.com/ashswap/ash-exchange-sc
- **Commit:** `e3ff87288b2fd26dea8edd7c7b02a07cd81d19e6`
- **MultiversX smart contract path:** ./contracts-0.39.8/dex/farm/

2 inherent risks in the final scope

2 issues in the final scope

16 issues reported in the initial scope and 2 remaining in the final scope:

Severity	Reported			Remaining		
	Code	Test	Other	Code	Test	Other
Critical	0	0	0	0	0	0
Major	4	0	0	0	0	0
Medium	3	0	0	1	0	0
Minor	9	0	0	1	0	0

Inherent Risks

R1: The more often users claim rewards, the less boosted rewards they earn.

This is because the boost of a farm position does not decrease until the user claims his rewards. It is even possible to maintain a boosted position beyond the veASH lock expiry. For example, if a user boosts his farm token with veASH locked for 1 year, and only claims 5 years later, then he will earn boosted rewards as if he had veASH locked for 1 year for the past 5 years.

R2: When users send a boosted farm token to other users, they may never be able to get back their boost used by that token.

This is especially problematic if a user who sent his boosted farm token intended to use this boost later for his other farm tokens. This is because as long as the recipient does not claim, the boost used by that farm token would not decrease but still count as being used by the user, thereby reducing the remaining boost available for his other farm tokens.

Code Issues & Recommendations

Since the code is not open-source, only the remaining issues are published.

C6: Can’t withdraw if farm router not registered in rewarder

Severity: Medium **Status:** Won't fix

Description

Users are unable to withdraw if the router is not registered in the rewarder. Indeed, when a user withdraws, the farm calls the rewarder method `mint_tokens`, which asks the farm router if the caller contract is a farm. So if the address of the farm router is not set in the rewarder, then the transaction will fail.

Recommendation

We recommend implementing the emergency withdraw solution from C8: generate_ash_rewards can reach gas or API limits and users can’t withdraw, as this solution incidentally resolves this issue as well.

Alternatively, before allowing users interactions in `resume`, the farm can check that the farm router is set in the rewarder, and that it matches the address of the farm’s owner.

C15: Duplicated logic to round timestamp to start of the week

Severity: Minor **Status:** Won't fix

Location

`contracts-0.39.8/dex/farm/src/lib.rs`

Description

The logic to compute the timestamp of the start of the week is duplicated multiple times across the contract. This is error-prone as if a future change is needed, the developer will have to apply the modification at several different locations.

Recommendation

We recommend implementing a helper method `timestamp_to_start_of_week` to compute the timestamp of the start of the week.