ALGEBRA FARMINGS SECURITY AUDIT REPORT

May 24, 2023

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals
- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

Algebra Finance is an AMM and concentrated liquidity protocol for decentralized exchanges running on adaptive fees.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Algebra
Project name	Farmings
Timeline	April 24 2023 - May 15 2023
Number of Auditors	4

Project Log

Date	Commit Hash	Note
24.04.2023	7290fad656bfa89db3743c52af631154f6a8a2d5	Commit for the audit
15.05.2023	714bb9a8bad638fade9fef7da371016d8b7d2fd3	Commit for the reaudit

Project Scope

The audit covered the following files:

File name	Link
FarmingCenter.sol	FarmingCenter.sol
IncentiveId.sol	IncentiveId.sol
NFTPositionInfo.sol	NFTPositionInfo.sol
VirtualTickManagement.sol	VirtualTickManagement.sol
AlgebraEternalFarming.sol	AlgebraEternalFarming.sol
EternalVirtualPool.sol	EternalVirtualPool.sol
IncentiveKey.sol	IncentiveKey.sol
VirtualTickStructure.sol	VirtualTickStructure.sol

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	5
Medium	4
Low	17

ID	Name	Severity	Status
H-1	Users' rewards can be lost if the incentive maker adds a new farming with the same key	High	Fixed
H-2	The reward distributor can send funds to an unattached pool	High	Fixed
H-3	The creation of the incentive with zero rewards blocks the pool's incentive system	High	Fixed
H-4	Unsynchronized positions in `NonFungiblePositionManager` and `AlgebraEternalFarming` can be exploited for receiving unfair rewards	High	Fixed
H-5	Virtual pools detached from the corresponding `AlgebraPool` result in an inconsistent state	High	Acknowledged
M-1	An incorrect sequence of calls can lead to a revert	Medium	Fixed
M-2	Rates can be updated for a deactivated farming	Medium	Fixed
M-3	Initialization of the `virtualPool` should be made earlier	Medium	Fixed
M-4	Admin can drain all rewards from all incentives	Medium	Acknowledged

L-1	Unused variables	Low	Fixed
L-2	Events missing	Low	Acknowledged
L-3	Unchecked calculations can be dangerous in some places	Low	Fixed
L-4	If a user transfers NFT, they can lose rewards	Low	Acknowledged
L-5	`EternalVirtualPool._distributeRewards()` has an unnecessary parameter	Low	Fixed
L-6	An unnecessary allowed caller in `EternalVirtualPool.crossTo()`	Low	Fixed
L-7	An unnecessary incentive storage change	Low	Fixed
L-8	`INCENTIVE_MAKER` can create an incentive for any IAlgebraPool	Low	Fixed
L-9	Anvone can call `AlgebraEternalFarming.addRewards()`	Low	Acknowledged
L-10	An incorrect (misspelled) word in the comment	Low	Fixed
L-11	An unnecessary type cast	Low	Fixed
L-12	The detached `Incentive` cannot be deactivated manually	Low	Fixed
L-13	A redundant call to `_updatePositionInVirtualPool` in the `_updatePosition` function	Low	Fixed
L-14	`getInnerRewardsGrowth` can be called for non-existed ticks	Low	Fixed
L-15	The check can be made earlier to save gas	Low	Fixed
L-16	Rebasable tokens cannot be used for rewards	Low	Acknowledged
L-17	Zero checks are missing	Low	Fixed

1.6 Conclusion

During the audit process 5 HIGH, 4 MEDIUM, and 17 LOW severity findings were spotted. After working on the reported findings, all of them were acknowledged or fixed by the client. The current architecture of the protocol allows pools to detach virtual pools if a user's position update reverts. These events can be monitored and we recommend that the Algebra team monitor such events.

Disclaimer

The client could provide the smart contracts for the deployment by a third party. To make sure that the deployed code hasn't been modified after the last audited commit, one should conduct their own investigation and deployment verification.

2.FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

H-1	Users' rewards can be lost if the incentive maker adds a new farming with the same key
Severity	High
Status	Fixed in 714bb9a8

Description

If the incentive maker adds a new farming after the previous one was unattached by the pool, then all users' rewards will be lost.
AlgebraEternalFarming.sol#L113
AlgebraEternalFarming.sol#L303

Recommendation

We recommend adding a check that the maker cannot create a farming with the same key.

Client's commentary

> Fixed nonce usage. Now it is not possible (the probability is negligible) to create same key.

H-2	The reward distributor can send funds to an unattached pool
Severity	High
Status	Fixed in 714bb9a8

Description

There is no check in the `addRewards()` function that would stop the reward distributor from sending funds to a virtual pool that was unattached from the Algebra pool due to a revert in the `crossTo()` function.
AlgebraEternalFarming.sol#L186-L195

Recommendation

We recommend adding the following check:

```
IAlgebraEternalVirtualPool virtualPool =
    IAlgebraEternalVirtualPool(_getCurrentVirtualPool(key.pool));
if (incentive.virtualPoolAddress !=
    address(virtualPool)) revert anotherFarmingIsActive();
```

Client's commentary

> Fixed

H-3	The creation of the incentive with zero rewards blocks the pool's incentive system
Severity	High
Status	Fixed in 714bb9a8

Description

It's possible to create an incentive with 0 rewards using `AlgebraEternalFarming.createEternalFarming()`: AlgebraEternalFarming.sol#L103.
Later `AlgebraEternalFarming._getIncentiveByKey()` will revert for such incentives: AlgebraEternalFarming.sol#L418.
So, it will be impossible to call `AlgebraEternalFarming.deactivateIncentive()`, `AlgebraEternalFarming.decreaseRewardsAmount()`, or `AlgebraEternalFarming.addRewards()`. So, the pool will be blocked from the creation of any incentive. If there are some bonus rewards for this incentive, they will be stuck forever.

Recommendation

We recommend adding a require clause for non-zero rewards in `AlgebraEternalFarming.createEternalFarming()`.

Client's commentary

> Fixed

H-4	Unsynchronized positions in `NonFungiblePositionManager` and `AlgebraEternalFarming` can be exploited for receiving unfair rewards
Severity	High
Status	Fixed in 714bb9a8

Description

If `FarmingCenter._updatePosition` reverts internally, `NonFungiblePositionManager` ignores that revert here: NonfungiblePositionManager.sol#L369 leading to unsynchronized positions between the actual position and the farming position. The `_updatePosition` function consists of sequential calls to the `exitFarming` and `enterFarming` functions within the `AlgebraEternalFarming` contract FarmingCenter.sol#L94. While the `exitFarming` function does not possess explicit revert statements, the `enterFarming` function can revert here: AlgebraEternalFarming.sol#L385. These reverts prevent the creation of farming positions in detached or deactivated farmings. Consequently, if a pool is detached, every call to `NonFungiblePositionManager.decreaseLiquidity` will revert within the `FarmingCenter._updatePosition` function. As a result, the actual liquidity of the position in the pool will decrease, but the liquidity in the farming position will remain unchanged, allowing the `tokenId` owner to continue collecting higher rewards than deserved.

Furthermore, this vulnerability can be exploited by users who can find a way to manually detach the pool from its incentive. Consider the following scenario:

- The exploiter obtains a flashloan from an external project
- The exploiter mints a position in the pool using loaned tokens and enters farming
- The exploiter detaches the pool
- The exploiter decreases liquidity of the position to the minimum possible causing desynchronization in `AlgebraEternalFarming` which retains the liquidity of the flahsloaned tokens.
- The exploiter repays the flashloan

In this scenario, the exploiter can collect rewards without maintaining an actual position.

Recommendation

We recommend implementing the forced exit from the farming position in cases when a revert occurs within the call of `NonFungiblePositionManager` here: NonfungiblePositionManager.sol#L369, if feasible, to prevent unsynchronized farming positions.

Client's commentary

> In general, we consider desynchronization as a consequence of an emergency that should not occur. In such a case, we prefer to minimize the amount of code that will be called by the peripheral contract (NonfungiblePositionManager). We also believe that the distribution of undeserved rewards is a less undesirable event than the potential blocking of liquidity withdrawal by users.
> We have added checks to prevent the described scenario. Now in this place there will be no attempt to re-enter the farming if it is detached.

H-5	Virtual pools detached from the corresponding `AlgebraPool` result in an inconsistent state
Severity	High
Status	Acknowledged

Description

In cases where `AlgebraPool` connected to `VirtualAlgebraPool` catches a revert during a call to the `crossTo` function, the pool will reset its `activeIncentive` parameter to the 0 address here: SwapCalculation.sol#L210.
Subsequently, the virtual pool enters a state where it is not `deactivated`, its `rewardRates` remain non-zero, and the detached pool no longer updates subsequent price changes. As a result, current liquidity providers continue to collect rewards based on the "frozen" price present in the pool at the moment of detachment.

Recommendation

We recommend updating `rewardRates` for the pool to 0 and setting the `deactivated` state to `true` during the detachment process to prevent the accumulation of rewards based on outdated prices.

Client's commentary

> The described sequence of events is correct. Such events are considered by us as an emergency and should never occur.
> We prefer to minimize the amount of code that is called by the liquidity pool, so monitoring tools will be used to identify such situations. In the event of such an out-of-sync, we do not expect a significant amount of rewards to be distributed based on outdated prices.
> All things considered, we prefer the current implementation to minimize costs and risks for the real pool.

2.3 Medium

M-1	An incorrect sequence of calls can lead to a revert
Severity	Medium
Status	Fixed in 714bb9a8

Description

Getting reserves before applying new rewards can lead to a revert on a call of the `decreaseRewardsAmount()` function.
AlgebraEternalFarming.sol#L161

Recommendation

We recommend calling `distributeRewards()` before getting reserves of the farming.

Client's commentary

> Fixed

M-2	Rates can be updated for a deactivated farming
Severity	Medium
Status	Fixed in 714bb9a8

Description

There is no check that the incentive maker can't set non-zero rates for a deactivated pool.
AlgebraEternalFarming.sol#L198-L202

Recommendation

We recommend adding a check that the farming was deactivated and this case allows setting only zero rates.

Client's commentary

> Fixed

M-3	Initialization of the `virtualPool` should be made earlier
Severity	Medium
Status	Fixed in 714bb9a8

Description

`virtualPool` is currently initialized after using it in the check.
AlgebraEternalFarming.sol#L398

Recommendation

We recommend initializing `virtualPool` at the beginning of the function.

Client's commentary

> Fixed during the audit.

M-4	Admin can drain all rewards from all incentives
Severity	Medium
Status	Acknowledged

Description

`AlgebraEternalFarming.decreaseRewardsAmount()` sends rewards to the address of admin: AlgebraEternalFarming.sol#L172.
It creates the danger of a rugpull and reduces the trust in the incentive system.

Recommendation

We recommend changing the logic of rewards refunds. An incentive for specific farming should be controlled by a registered address for this incentive.

Client's commentary

> At the moment, using this feature, we can withdraw tokens that have not yet been distributed as farming rewards. Among other things, this allows us to solve situations when tokens were sent by mistake or to the wrong farming or in too large amounts.
> The AlgebraFactory owner has the right to do this by default.
> Also, an address can be given a separate role that will allow it to withdraw rewards from reserves. We encourage the transfer of the AlgebraFactory owner role to the appropriate multisig or DAO contract. A separate administrator role can be used for a more granular division of responsibility.
> In our opinion, adding separate roles for different farming just creates an additional layer that will not stop in the case of rugpull by DEX team. The only way to remove such a risk is to completely get rid of this feature, which creates other, more frequent, threats (mentioned above).

2.4 Low

L-1	Unused variables
Severity	Low
Status	Fixed in 714bb9a8

Description

Several storage variables can be removed:
FarmingCenter.sol#L21
VirtualTickManagement.sol#L23
EternalVirtualPool.sol#L18.

Recommendation

We recommend removing these variables from storage.

Client's commentary

> 1. Now it's being used 2. Removed 3. Removed.

L-2	Events missing
Severity	Low
Status	Acknowledged

Description

Several functions should emit an event for better UX in possible integrations:
AlgebraEternalFarming.sol#L186-L195
AlgebraEternalFarming.sol#L198-L202.

Recommendation

We recommend emitting events after these function calls.

Client's commentary

> Events are called in private functions.

L-3	Unchecked calculations can be dangerous in some places
Severity	Low
Status	Fixed in 714bb9a8

Description

There are several places where it is better to not use unchecked calculations:
AlgebraEternalFarming.sol#L327-L332
AlgebraEternalFarming.sol#L160-L168.

Recommendation

We recommend not using unchecked calculations in these places.

Client's commentary

> Fixed

L-4	If a user transfers NFT, they can lose rewards
Severity	Low
Status	Acknowledged

Description

If the user transfer NFT before calling `collectRewards`, they will lose accrued rewards FarmingCenter.sol#L113-L115.

Recommendation

We recommend adding a check to the transfer function that rewards were collected before the transfer.

Client's commentary

> Logic works as intended.

L-5	`EternalVirtualPool._distributeRewards()` has an unnecessary parameter
Severity	Low
Status	Fixed in 714bb9a8

Description

Internal function `EternalVirtualPool._distributeRewards()` uses the `currentTimestamp` parameter:
EternalVirtualPool.sol#L208.

Recommendation

We recommend using `uint32(block.timestamp)` directly in the function.

Client's commentary

> Fixed

L-6	An unnecessary allowed caller in `EternalVirtualPool.crossTo()`
Severity	Low
Status	Fixed in 714bb9a8

Description

`EternalVirtualPool.crossTo()` is allowed to be called by `farmingCenterAddress` but it's never called from the farming center:
EternalVirtualPool.sol#L103.

Recommendation

We recommend removing `farmingCenterAddress` from allowed callers.

Client's commentary

> Fixed

L-7	An unnecessary incentive storage change
Severity	Low
Status	Fixed in 714bb9a8

Description

`AlgebraEternalFarming._enterFarming()` marks the incentive as deactivated in the storage and after that reverts in case of the virtual pool address desynchronization: AlgebraEternalFarming.sol#L386.
But the storage changes are not saved in case of a revert.

Recommendation

We recommend reverting without storage changes.

Client's commentary

> Fixed

L-8	`INCENTIVE_MAKER` can create an incentive for any IAlgebraPool
Severity	Low
Status	Fixed in 714bb9a8

Description

`INCENTIVE_MAKER` can create an incentive for any `IAlgebraPool`, not only for the ones created by the factory:
AlgebraEternalFarming.sol#L103.
It can be used in some complicated attacks by `INCENTIVE_MAKER`.

Recommendation

We recommend checking the pool address if it's created by the factory.

Client's commentary

> Fixed

L-9	Anyone can call `AlgebraEternalFarming.addRewards()`
Severity	Low
Status	Acknowledged

Description

`AlgebraEternalFarming.addRewards()` can be called by any address: AlgebraEternalFarming.sol#L186.
So, anyone can manipulate the size of the total rewards. It can be used in some complicated attacks.

Recommendation

We recommend implementing a whitelist for addresses that can add rewards.

Client's commentary

> Works as intended. We need a specific attack vector to decide on a change.

L-10	An incorrect (misspelled) word in the comment
Severity	Low
Status	Fixed in 714bb9a8

Description

At line VirtualTickManagement.sol#L134 a misspelled word is used.

Recommendation

We recommend changing `if` to `of`.

Client's commentary

> Fixed

L-11	An unnecessary type cast
Severity	Low
Status	Fixed in 714bb9a8

Description

At line AlgebraEternalFarming.sol#L99 variable `factory` is cast to the `IAlgebraFactory` type, but it already has the `IAlgebraFactory` type by definition - AlgebraEternalFarming.sol#L52.

Recommendation

We recommend removing unnecessary cast operations.

Client's commentary

> Fixed

L-12	The detached `Incentive` cannot be deactivated manually
Severity	Low
Status	Fixed in 714bb9a8

Description

If the virtual pool is detached from its corresponding algebra pool, then it is impossible for the incentive manager to manually change its status to the `deactivated` state due to this check: AlgebraEternalFarming.sol#L143.

Recommendation

We recommend modifying the specified expression to

```
if (incentive.virtualPoolAddress == address(0)) ...
```

Client's commentary

> Fixed

L-13	A redundant call to `_updatePositionInVirtualPool` in the `_updatePosition` function
Severity	Low
Status	Fixed in 714bb9a8

Description

The call to `_updatePositionInVirtualPool` here AlgebraEternalFarming.sol#L263 can be replaced with the `distributeRewards` call since the`liquidityDelta` argument is set to 0 at that point.

Recommendation

We recommend changing the `_updatePositionInVirtualPool` call in the specified line to `virtualPool.distributeRewards()`.

Client's commentary

> Fixed

L-14	`getInnerRewardsGrowth` can be called for non-existed ticks
Severity	Low
Status	Fixed in 714bb9a8

Description

There are no checks in the `getInnerRewardsGrowth` function that `bottomTick` and `topTick` were added to the tick tree.
EternalVirtualPool.sol#L64-L67

Recommendation

We recommend adding checks that ticks were added to restrict external calls with incorrect input parameters.

Client's commentary

> Fixed

L-15	The check can be made earlier to save gas
Severity	Low
Status	Fixed in 714bb9a8

Description

The liquidity of the position check can be made earlier to save some gas.
AlgebraEternalFarming.sol#L209

Recommendation

We recommend checking the liquidity of the position before calling `_enterFarming()`.

Client's commentary

> Fixed

L-16	Rebasable tokens cannot be used for rewards
Severity	Low
Status	Acknowledged

Description

The current farming design doesn't allow work with rebasable tokens.

Recommendation

We recommend restricting work with rebasable tokens. This information can be added to the protocol documentation.

Client's commentary

> Information will be added to the documentation.

L-17	Zero checks are missing
Severity	Low
Status	Fixed in 714bb9a8

Description

The protocol allows to transfer 0 amount of tokens to the 0 address.
AlgebraEternalFarming.sol#L410

Recommendation

We recommend adding a check that tokens cannot be transferred to the 0 address and farming doesn't try to send 0 tokens.

Client's commentary

> Fixed

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts


https://github.com/mixbytes/audits_public


https://mixbytes.io/


hello@mixbytes.io


https://twitter.com/mixbytes