ALGEBRA PLUGINS SECURITY AUDIT REPORT

September 6, 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 a concentrated liquidity protocol for decentralized exchanges that allows the usage of adaptive fees.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Algebra
Project name	Farming & Plugins
Timeline	August 22 2023 - September 05 2023
Number of Auditors	4

Project Log

Date	Commit Hash	Note
22.08.2023	6f57b3e218630106a4d41aedefd38f9e83b41e2b	Commit for the audit
05.09.2023	16e104c4ae072c071914fcfe680214c88a5cf9de	Commit for the re-audit

Project Scope

The audit covered the following files:

File name	Link

File name	Link
AlgebraBasePluginV1.sol	AlgebraBasePluginV1.sol
BasePluginV1Factory.sol	BasePluginV1Factory.sol
AlgebraFeeConfiguration.sol	AlgebraFeeConfiguration.sol
AdaptiveFee.sol	AdaptiveFee.sol
VolatilityOracle.sol	VolatilityOracle.sol
FarmingCenter.sol	FarmingCenter.sol
IncentiveKey.sol	IncentiveKey.sol
VirtualTickStructure.sol	VirtualTickStructure.sol
AlgebraEternalFarming.sol	AlgebraEternalFarming.sol
EternalVirtualPool.sol	EternalVirtualPool.sol
IncentiveId.sol	IncentiveId.sol
NFTPositionInfo.sol	NFTPositionInfo.sol

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	2
Medium	3
Low	6

ID	Name	Severity	Status
H-1	Farming can be deactivated by accident	High	Fixed
H-2	Overflow will entirely block the plugin	High	Fixed
M-1	Manipulation of volatility through zero-amount swaps	Medium	Acknowledged
M-2	Missing flags validation in new `pluginConfig`	Medium	Fixed
M-3	Plugin timestamps data can be overwritten by a call to the `prepayTimepointsStorageSlot` function	Medium	Fixed
L-1	An incorrect comment	Low	Fixed
L-2	Farming deactivation can be triggered on the farming entering	Low	Acknowledged
L-3	There is no check in `AlgebraEternalFarming._isIncentiveActive()` that the plugin is still connected to the pool	Low	Acknowledged
L-4	`AlgebraBasePluginV1.isIncentiveActive()` returns true for deactivated incentives	Low	Fixed
L-5	TODO comments should be resolved	Low	Fixed

L-6	The risk of storage overflow in volatility oracle	Low	Acknowledged

1.6 Conclusion

During the audit process 2 HIGH, 3 MEDIUM, and 6 LOW severity findings were spotted. After working on the reported findings, all of them were acknowledged or fixed by the client. No other issues which could affect the protocol's function were identified.
The test coverage of the protocol is sufficient. A lot of different edge-case scenarios are covered in the tests. Also, there are a lot of fuzzing tests written with Echidna that help check invariants in the code. The overall quality of the code is very high which is why there are not so many findings in the report. To increase the readability of the code, we recommend adding code style best practices to the process of code development (e.g., https://docs.soliditylang.org/en/latest/style-guide.html)

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	Farming can be deactivated by accident
Severity	High
Status	Fixed in 16e104c4

Description

The current check for farming deactivation can be triggered by accident if `tick` hasn't changed during the swap and `zeroToOne` is false EternalVirtualPool.sol#L131-L134. This action will disconnect farming from the pool and will affect incentive distribution for users.

Recommendation

We recommend using this check EternalVirtualPool.sol#L137-L141 before setting the `deactive` state to the virtual pool.

H-2	Overflow will entirely block the plugin
Severity	High
Status	Fixed in 16e104c4

Description

Overflow will occur here on the second time of the list initialization VolatilityOracle.sol#L72-L74. This will block plugin updates that will affect the pool and further recovery can be executed by admin only.

Recommendation

We recommend using the `unchecked` block here since overflow is desired.

2.3 Medium

M-1	Manipulation of volatility through zero-amount swaps
Severity	Medium
Status	Acknowledged

Description

Swaps executed with a zero amount are currently factored into the volatility oracle. This vulnerability allows an attacker to artificially manipulate the volatility index, which in turn affects the dynamic fee calculations to their advantage.

The issue has been categorized with a MEDIUM severity level. This is because the attacker can manipulate the fee only under specific conditions and within a predefined range.

Recommendation

To mitigate this issue, it is recommended to exclude zero-amount swaps from volatility calculations.

Client's commentary

> This is expected behavior.
> To influence the value of the fee, the price must be kept in the wrong state for a certain period of time. It is difficult to implement such a scenario in the presence of active liquidity. In the absence of liquidity, manipulation of the volatility value is possible and the fee value is not relevant.

M-2	Missing flags validation in new `pluginConfig`
Severity	Medium
Status	Fixed in 16e104c4

Description

There is no validation of flags in new `pluginConfig` set with `AlgebraPool.setPluginConfig()`. This can lead to the invocation of hooks that will constantly revert. E.g. `BEFORE_FLASH_FLAG`, `AFTER_FLASH_FLAG`, `AFTER_SWAP_FLAG` (without currently connected incentive) can be switched on. This issue has a MEDIUM severity since it can partially or fully stop the pool until `pluginConfig` is corrected.

Recommendation

We recommend moving all interaction with `pluginConfig` to the plugin contract. Any modification of `pluginConfig` should consider the current version and state of the plugin.

M-3	Plugin timestamps data can be overwritten by a call to the `prepayTimepointsStorageSlot` function
Severity	Medium
Status	Fixed in 16e104c4

Description

There is an issue at the line AlgebraBasePluginV1.sol#L67.
Plugin timestamps data can be overwritten by a call to the `prepayTimepointsStorageSlots` function in cases when the
plugin was configured from AlgebraPool by a call to `setPluginConfig` (and `setPlugin`) and the `initialize` function inside plugin wasn't called. An attacker can wait for some swaps and data in the timepoints array to overwrite the existing timestamps. But due to uninitialized ticks in the first timepoint, users won't be able to perform many swaps from AlgebraPool because the plugin will operate with null initial data.
This issue has been assigned a MEDIUM severity level because an attacker is able to overwrite data only for a few timepoints.

Recommendation

We recommend restricting the `prepayTimepointsStorageSlots` function to be called only by the plugin factory manager.

2.4 Low

L-1	An incorrect comment
Severity	Low
Status	Fixed in 16e104c4

Description

There is a comment that says that volume is used to calculate volatility, which is wrong AdaptiveFee.sol#L36.

Recommendation

We recommend updating the comment.

L-2	Farming deactivation can be triggered on the farming entering
Severity	Low
Status	Acknowledged

Description

Farming deactivation can be triggered on the farming entering, so the user who enters farming and deactivates it will need to exit farming immediately to avoid losing rewards.

Recommendation

We recommend updating the project architecture to account for this scenario.

Client's commentary

> Works as designed. The user will not lose any rewards. The scenario described here is an expected emergency case: farming will stop and the user will be able to exit farming.

L-3	There is no check in `AlgebraEternalFarming._isIncentiveActive()` that the plugin is still connected to the pool
Severity	Low
Status	Acknowledged

Description

AlgebraEternalFarming.sol#L349

A plugin with some incentive connected to that plugin can be replaced with another plugin by calling `Algebra.setPlugin()`. `AlgebraEternalFarming._isIncentiveActive()` will still return `true` for that incentive for some time even if there is already another active incentive in the new plugin. So, new users can still enter that old incentive and new rewards can be brought to it.

Recommendation

We recommend adding a check to `AlgebraEternalFarming._isIncentiveActive()` that `Incentive.pluginAddress` is a current pool's plugin.

Client's commentary

> Works as designed. But to simplify the perception, it was replaced with `isIncentiveDeactivated`

L-4	`AlgebraBasePluginV1.isIncentiveActive()` returns true for deactivated incentives
Severity	Low
Status	Fixed in 16e104c4

Description

The external view function `AlgebraBasePluginV1.isIncentiveActive()` doesn't check whether the current state of incentive and `EternalVirtualPool` is active or not. It just checks that some incentive is connected to the plugin. So, the result can be misinterpreted.

Recommendation

We recommend renaming this function.

L-5	TODO comments should be resolved
Severity	Low
Status	Fixed in 16e104c4

Description

There are multiple TODO comments - AlgebraEternalFarming.sol#L169, AlgebraEternalFarming.sol#L354, FarmingCenter.sol#L87 and FarmingCenter.sol#L115.

Recommendation

We recommend removing the above-mentioned comments.

L-6	The risk of storage overflow in volatility oracle
Severity	Low
Status	Acknowledged

Description

The volatility oracle's storage is constrained by a `UINT16_MODULO` capacity, allowing for approximately 45 values to be recorded per minute over a 24-hour period.

VolatilityOracle.sol#L14-L15

While this capacity is sufficient for the Ethereum mainnet, which generates about 5 blocks per minute, it may be insufficient for other EVM-compatible networks with more frequent block generation. An attacker could exploit this limitation to cause the loss of some volatility data by triggering an excessive number of volatility records.

This issue has been rated as LOW severity due to its limited impact and the complexity involved in reproducing the exploit.

Recommendation

To address this issue, it is recommended to adjust the frequency of volatility storage updates in accordance with both the storage capacity and the desired time window for data retention.

Client's commentary

> Works as designed. In practice, overflow during the day is almost impossible, but such a scenario is taken into account in the code. There are also related tests.

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