PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Algebra Protocol (Direction Based Fee Update V2)

05 April 2023

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Algebra Protocol's updated direction-based fee contracts on the Arbitrum network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Introduction & Scope

This audit covers the incremental changes of the Algebra protocol related to the directional fee logic within their second protocol iteration ("V2"). This V2 has undergone an audit with ABDK, while the fee logic has not. This logic was introduced for Camelot on Arbitrum to allow them to configure a different swap fee for purchases and sales. We assume they will want to tax sells more heavily than purchases of certain tokens and allow for a deflationary component to the governance tokens listed on their platform.

This section is meant to clearly define the scope of this audit. As of this point, this audit does not cover the base Algebra V2 protocol. Even though it was audited recently, we ourselves can, as of this moment, not make any warranties over the base protocol.

The goal of this audit is to ensure a high probability that the changes which were made for Camelot are safe and sound. This means that the goal of this audit is to ensure that all of the changes made do not introduce new bugs, side-effects, exploit vectors, ways that funds might get stuck and so forth. If the audit is successful, it can be used by third-parties as an extension to the audits which cover the original protocol.

To conclude, this audit aims to validate the soundness of the Direction Based Fee Update to the Algebra V2 protocol. It aims to ensure that this change does not introduce unintended side-effects of any kind. These side-effects might be identifiable in isolation, but Paladin has also conducted a thorough background study of the Algebra V2 protocol to also be able to identify more nuanced side-effects which result from interactions with the Algebra V2 protocol at large.

As will be readable throughout this audit, no major concerns with the directional fee based logic were found within our assessment. This means that, up to the best of our ability, we do not expect that this change worsens the security model for Algebra, as defined as the set of all security concerns present in the original Algebra V2 protocol.

The original Algebra V2 protocol is defined as https://github.com/cryptoalgebra/Algebra/tree/5ba86211a15c91caae27ce604c98e61544a31fee, which Paladin did not audit. It should be noted that a few fixes related to the ABDK audit came in after this commit, which were also not within scope of this feature audit. This audit focuses on the changes related to the directional fees.

1.2 Summary

Project Name	Algebra Protocol
URL	https://algebra.finance/
Platform	Arbitrum
Language	Solidity
Preliminary Contracts	https://github.com/cryptoalgebra/Algebra/compare/abdk-fixes-v2...direction-based-fee-v2

1.3 Contracts Assessed

Name	Contract	Live Code Match
`AlgebraPoolBase`		
`AlgebraFactory`		
`DataStorageOperator`		
`Positions`		
`Quoter`		
`SwapCalculation`		

1.4 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	0	-	-	-
Medium	0	-	-	-
Low	0	-	-	-
Informational	0	-	-	-
Total	**0**	**-**	**-**	**-**

Classification of Issues

Severity	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.4.1 AlgebraPoolBase

No issues found.

1.4.2 AlgebraFactory

No issues found.

1.4.3 DataStorageOperator

No issues found.

1.4.4 Positions

No issues found.

1.4.5 Quoter

No issues found.

1.4.6 SwapCalculation

No issues found.

2 Findings

2.1 AlgebraPoolBase

`AlgebraPoolBase` is the core contract underlying every Algebra pool. It contains various of the essential storage variables and functions needed for the functioning of a pool.

Several changes have been made to `AlgebraPoolBase`. First of all, the global state fee variable has been split up into two fees: zero to one and one to zero, representing each of the swap directions (whether you swap the one token for the other). Secondly, the `prevInitializedTick` variable has been moved outside of the global scope into a separate variable. Next, the flash loans now always require the payment of the base fee, regardless of which token is lent. Finally, various portions of the contract were reformatted.

No significant side-effects or issues were introduced with these changes.

2.2 AlgebraFactory

`AlgebraFactory` is used to deploy concentrated liquidity pool pairs and their respective data storages.

The primary function within the factory is `createPool`, which takes in two token addresses and deploys a concentrated liquidity pool for them.

The only change which has been introduced within this contract is the fee initialization logic. Instead of initializing a single global fee for the pool, the factory now initializes a directional buy fee and a sell fee.

The code section which was changed was at first:

```
dataStorage.changeFeeConfiguration(baseFeeConfiguration);
```

It has been changed to:

```
dataStorage.changeFeeConfiguration(true, defaultFeeConfiguration);
dataStorage.changeFeeConfiguration(false, defaultFeeConfiguration);
```

This change is simple and straightforward and therefore does not introduce any new issues.

No issues were found in this change.

2.3 DataStorageOperator

`DataStorageOperator` is a utility contract deployed for each Algebra pool which contains certain storage types and operations which can be executed on these types.

Most notable for this audit, this contract contains the present fee configuration for the pool and the function which allows the factory owner or any fee manager to re-configure this fee.

As part of the update, the single pool fee has been split up into two directional fees: A buy and sell fee.

The following line was replaced:

```
AdaptiveFee.Configuration public feeConfig;
```

By the following two lines:

```
AdaptiveFee.Configuration public feeConfigZto;
AdaptiveFee.Configuration public feeConfigOtz;
```

These two new fees are shorthand for:

- fee config for swaps from token zero to token one
- fee config for swaps from token one to token zero

Previously, the global fee was configured as follows:

```
function changeFeeConfiguration(AlgebraFeeConfiguration calldata _config)
external override {
```

In the updated code, it is configured using a boolean which identifies whether the new fee is a buy or sell fee:

```
function changeFeeConfiguration(bool zto, AlgebraFeeConfiguration
calldata _config) external override {
```

The boolean then causes either `feeConfigZto(true)` or `feeConfigOtz(false)` to be updated.

Finally, a minor change was made to the `getFee` function which previously returned the single pool fee. Now, it returns both of the directional fees.

No significant side-effects or issues were introduced with these changes.

2.4 Positions

`Positions` is a module for the Algebra pool used to calculate and update positions data.

The main change which was made was to change the fee variable in the `UpdatePositionCache` to two fee variables, based on the swap direction. Various functions of the contract were adjusted accordingly with no side-effects found.

No significant side-effects or issues were introduced with these changes.

2.5 Quoter

`Quoter` is used by the frontend to quote prices and the amounts one would receive from a swap. The only change which was made was to update the fee logic to utilize the correct fee depending on the swap direction.

No issues were found in the changes.

2.6 SwapCalculation

`SwapCalculation` is a module for the Algebra pool used to calculate and process swaps.

The main change was the addition of the ability to choose the fee between two fees depending on the swap direction. No side-effects were found from this introduction other than that the fee is stored more quickly into storage, which we did not find any problems with.

No significant side-effects or issues were introduced with these changes.


PALADIN
BLOCKCHAIN SECURITY