PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Algebra Protocol (Tick Spacing Upgrade)

14 June 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 full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Paladin is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Paladin may, at its discretion, claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Algebra Protocol's tick spacing upgrade 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 upgrade to the Algebra Protocol that introduced adjustable tick spacing. The tick spacing was previously fixed at 60 ticks (eg. users had a granularity of 60 nominal ticks for the actual usable ticks), but this can now be freely set by the team.

The goal of this audit is to check if there are any undesirable side-effects or bugs introduced with this adjustment.

1.2 Summary

Project Name	Algebra Protocol
URL	https://algebra.finance/
Platform	Arbitrum
Language	Solidity
Preliminary Contracts	https://github.com/cryptoalgebra/Algebra/compare/a3dc950f7b2ad6dfe628beda16fd0dd1c9cd3874..237524b05538944128463bdedf7a85e5e82110bd

1.3 Contracts Assessed

Name	Contract	Live Code Match
`AlgebraPool`		
`PoolState`		
`PoolImmutables`		
`Constants`		
`TickTable`		
`PoolTicksCounter (Periphery)`		
`AlgebraFarming`		

1.4 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	-	-	-	-
Medium	-	-	-	-
Low	3	3	-	-
Informational	1	-	-	1
Total	**4**	**3**	-	**1**

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 AlgebraPool

ID	Severity	Summary	Status
01	INFO	Adjustable tick spacing might cause unexpected issues with vaults	ACKNOWLEDGED

1.4.2 PoolState

No issues found.

1.4.3 PoolImmutables

No issues found.

1.4.4 Constants

ID	Severity	Summary	Status
02	LOW	`MAX_LIQUIDITY_PER_TICK` is set to the wrong value	✓ RESOLVED

1.4.5 TickTable

No issues found.

1.4.6 PoolTicksCounter (Periphery)

ID	Severity	Summary	Status
03	LOW	`tickBefore` or `tickAfter` may not be counted	✓ RESOLVED

1.4.7 AlgebraFarming

No issues found.

1.4.8 Documentation

ID	Severity	Summary	Status
04	LOW	The markdown files are outdated	✓ RESOLVED

2 Findings

2.1 AlgebraPool

`AlgebraPool` is the main contract that contains the logic of the pool to swap, add or withdraw liquidity positions.

The following changes have been made:

Constructor

```
tickSpacing = 60;
```

Set tick spacing to 60. Previously this value was set to the same value, but as a constant.

———

Minting

```
{
    int24 _tickSpacing = tickSpacing;
    require(bottomTick % _tickSpacing | topTick % _tickSpacing == 0, 'tick is
not spaced'); // ensure that the tick is spaced
}
```

Added a requirement on the tick spacing.

———

```
uint128 liquidityForRA1 = uint128(FullMath.mulDiv(uint256(liquidityDesired),
receivedAmount1, amount1));
```

This line initially `liquidityActual`; it has been changed to `liquidityDesired`.

```
/// @inheritdoc IAlgebraPoolPermissionedActions
function setTickSpacing(int24 newTickSpacing) external override lock
onlyFactoryOwner {
    require(newTickSpacing > 0 && newTickSpacing <= Constants.MAX_TICK_SPACING
&& tickSpacing != newTickSpacing, 'Invalid newTickSpacing');
    tickSpacing = newTickSpacing;
    emit TickSpacing(newTickSpacing);
}
```

A `setTickSpacing` function was added to adjust the tick spacing.

2.1.1 Issues & Recommendations

Issue #01	Adjustable tick spacing might cause unexpected issues with vaults
Severity	INFORMATIONAL
Description	The tick spacing upgrade added a requirement that only certain ticks can be used to add liquidity to. However, the factory owner can change these ticks at any time. This might cause unexpected issues if vaults do not expect a tick to become non-addable over time. Certain vault logic might assume that they can top up the ticks they added to in the past freely.
Recommendation	Consider documenting this change compared to the normal Algebra protocol carefully, as to avoid any issues with dependencies.
Resolution	ACKNOWLEDGED

2.2 PoolState

`PoolState` contains the storage values of the pool and also functions that allow access to the values in storage.

```
The following changes have been made:
int24 public override tickSpacing;
```

A `tickSpacing` variable has been added, replacing the immutable spacing of the original Algebra protocol.

2.2.1 Issues & Recommendations

No issues found.

2.3 PoolImmutables

`PoolImmutables` contains pure functions and immutable variables for the pools.

The following changes have been made:

```
function tickSpacing() external pure override returns (int24) {
    return Constants.TICK_SPACING;
 }
```

This function has been removed in favor of a `tickSpacing` variable within `PoolState`.

2.3.1 Issues & Recommendations

No issues found.

2.4 Constants

`Constants` contains various constants for the pool.

The following changes have been made:

```
int24 internal constant MAX_TICK_SPACING = 500;
```

The `TICK_SPACING` constant has been replaced with `MAX_TICK_SPACING`. The maximum liquidity per tick value has also been adjusted.

2.4.1 Issues & Recommendations

Issue #02	**`MAX_LIQUIDITY_PER_TICK` is set to the wrong value**
Severity	LOW SEVERITY
Description	The `MAX_LIQUIDITY_PER_TICK` should be set to `max(uint128) / (MAX_TICK - MIN_TICK)`, however it is set to `(2**128-1) // 8388607` by mistake. The issue is only rated as Low as the current variable is lower than the real value and merely prevents some liquidity from being added.
Recommendation	Consider setting it to the right variable: `191757638537527648490752896198554`
Resolution	RESOLVED

2.5 TickTable

The TickTable defines the logic to flip a tick, to account if a tick has liquidity or not.

The following section has been removed:

```
require(tick % Constants.TICK_SPACING == 0, 'tick is not spaced'); // ensure
that the tick is spaced
tick /= Constants.TICK_SPACING; // compress tick
```

The tick was previously compressed as the tick spacing was a constant. It is no longer compressed as the tick spacing might change overtime. The requirement that the tick is correctly spaced was moved to the functions that use the library instead of being in the library itself.

`uncompressAndBoundTick` was updated to `boundTick` as the ticks are not compressed anymore.

2.5.1 Issues & Recommendations

No issues found.

2.6 PoolTicksCounter (Periphery)

`PoolTicksCounter` is responsible for utilities like counting the number of initialized ticks.

2.6.1 Issues & Recommendations

Issue #03	**`tickBefore` or `tickAfter` may not be counted**
Severity	LOW SEVERITY
Description	If `tickSpacing` is updated, the check to ensure the ticks are spaced `(tickBefore % self.tickSpacing()) == 0` and `(tickAfter % self.tickSpacing()) == 0` might not be up to date as the tick spacing might have changed.
Recommendation	Consider remocing the tick spacing check to make sure the number of tick crossed is accurate.
Resolution	RESOLVED

2.7 AlgebraFarming

`AlgebraFarming` is a base contract for staking LPs. The constant `tickSpacing` logic has been removed as this is now handled inside the pool.

2.7.1 Issues & Recommendations

No issues found.

2.8 Documentation

Documentation of the different markdown files allows users and partners to better understand how to use and integrate Algebra protocol.

2.8.1 Issues & Recommendations

Issue #04	The markdown files are outdated
Severity	LOW SEVERITY
Description	Different markdown files still metion tickSpacing as an immutable variable — it should be moved to the documentation of state variables.
Recommendation	Consider updating the documentation.
Resolution	RESOLVED


PALADIN
BLOCKCHAIN SECURITY