GA GUARDIAN

Axis

Modular Auctions

Security Assessment

August 21st, 2024

Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Owen Thurm, Wafflemakr, Mark Jonathas, Osman Ozdemir, Kiki, Michael Lett

Client Firm Axis

Final Report Date August 21, 2024

Audit Summary

Axis engaged Guardian to review the security of its periphery contracts supporting auction creation and settlement. From the 22nd of July to the 29th of July a team of 7 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement 10 High/Critical issues were uncovered and promptly remediated by the Axis team. Several issues impacted the fundamental behavior of the protocol, following their remediation Guardian believes the protocol to uphold the functionality described for the lending protocol product.

Security Recommendation Given the number of High and Critical issues detected, Guardian supports an independent security review of the protocol at a finalized frozen commit. Furthermore, the Axis team should increase testing with token donations which may present opportunities to DoS the system.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Arbitrum, Blast, Base**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/axis-fuzzing

Table of Contents

Project Information

Project Overview 4

Audit Scope & Methodology 5

Smart Contract Risk Assessment

Invariants Assessed 6

Findings & Resolutions 11

Addendum

Disclaimer 42

About Guardian Audits 43

Project Overview

Project Summary

Project Name	Axis
Language	Solidity
Codebase	https://github.com/Axis-Fi/axis-periphery
Commit(s)	9da756b9662ea0c5c125ab0c33f2f2d8c7b1c42f

Audit Summary

Delivery Date	August 21, 2024
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	2	0	0	0	0	2
High	8	0	0	0	1	7
Medium	6	0	0	3	0	3
Low	12	0	1	2	0	9

Audit Scope & Methodology

Vulnerability Classifications

Vulnerability Level	Classification
🔴 Critical	Easily exploitable by anyone, causing loss/manipulation of assets or data.
🟠 High	Arduously exploitable by a subset of addresses, causing loss/manipulation of assets or data.
🟡 Medium	Inherent risk of future exploits that may or may not impact the smart contract execution.
🟢 Low	Minor deviation from best practices.

Methodology

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.
- Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Invariants Assessed

During Guardian's review of Axis, fuzz-testing with Echidna was performed on the protocol's main functionalities. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 10,000,000+ runs with a prepared Echidna fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
AX-01	UniswapV2Dtl_onCreate() should set DTL Config recipient	✅	✅	✅	10M+
AX-02	UniswapV2Dtl_onCreate() should set DTL Config lotCapacity	✅	✅	✅	10M+
AX-03	UniswapV2Dtl_onCreate() should set DTL Config lotCuratorPayout	✅	✅	✅	10M+
AX-04	UniswapV2Dtl_onCreate() should set DTL Config proceedsUtilisationPercent	✅	✅	✅	10M+
AX-05	UniswapV2Dtl_onCreate() should set DTL Config vestingStart	✅	✅	✅	10M+
AX-06	UniswapV2Dtl_onCreate() should set DTL Config vestingExpiry	✅	✅	✅	10M+
AX-07	UniswapV2Dtl_onCreate() should set DTL Config linearVestingModule	✅	✅	✅	10M+
AX-08	UniswapV2Dtl_onCreate() should set DTL Config active to true	✅	✅	✅	10M+
AX-09	DTL Callbacks should not change seller base token balance	✅	✅	✅	10M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
AX-10	DTL Callbacks should not change dtl base token balance	✅	✅	✅	10M+
AX-11	DTL_onCancel() should set DTL Config active to false	✅	✅	✅	10M+
AX-12	DTL_onCurate should set DTL Config lotCuratorPayout	✅	✅	✅	10M+
AX-13	When calling DTL_onCurate auction house base token balance should be equal to lot Capacity of each lotId	✅	✅	✅	10M+
AX-14	DTL_onSettle should should credit seller the expected LP token balance	✅	✅	✅	10M+
AX-15	DTL_onSettle should should credit linearVestingModule the expected LP token balance	✅	✅	✅	10M+
AX-16	DTL_onSettle should should credit seller the expected wrapped vesting token balance	✅	✅	✅	10M+
AX-17	After DTL_onSettle DTL Address quote token balance should equal 0	✅	✅	✅	10M+
AX-18	After DTL_onSettle DTL Address base token balance should equal 0	✅	✅	✅	10M+
AX-19	After UniswapV2DTL_onSettle DTL Address quote token allowance for the UniswapV2 Router should equal 0	✅	✅	✅	10M+
AX-20	After UniswapV2DTL_onSettle DTL Address base token allowance UniswapV2 Router should equal 0	✅	✅	✅	10M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
AX-21	UniswapV3Dtl_onCreate() should set DTL Config recipient	✅	✅	✅	10M+
AX-22	UniswapV3Dtl_onCreate() should set DTL Config lotCapacity	✅	✅	✅	10M+
AX-23	UniswapV3Dtl_onCreate() should set DTL Config lotCuratorPayout	✅	✅	✅	10M+
AX-24	UniswapV3Dtl_onCreate() should set DTL Config proceedsUtilisationPercent	✅	✅	✅	10M+
AX-25	UniswapV3Dtl_onCreate() should set DTL Config vestingStart	✅	✅	✅	10M+
AX-26	UniswapV3Dtl_onCreate() should set DTL Config vestingExpiry	✅	✅	✅	10M+
AX-27	UniswapV3Dtl_onCreate() should set DTL Config linearVestingModule	✅	✅	✅	10M+
AX-28	UniswapV3Dtl_onCreate() should set DTL Config active to true	✅	✅	✅	10M+
AX-29	On UniswapV3DTL_OnSettle() calculated sqrt price should equal pool sqrt price	✅	✅	✅	10M+
AX-30	After UniswapV3DTL_onSettle DTL Address base token allowance for the GUniPool should equal 0	✅	✅	✅	10M+
AX-31	After UniswapV3DTL_onSettle DTL Address base token allowance for the GUniPool should equal 0	✅	✅	✅	10M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
AX-32	When calling BaselineDTL_createLot auction house base token balance should be equal to lot Capacity lotId	✅	✅	✅	10M+
AX-33	After DTL_onSettle quote token balance of quote token should equal 0	✅	✅	✅	10M+
AX-34	BaselineDTL_onSettle should credit baseline pool with correct quote token proceeds	✅	✅	✅	10M+
AX-35	BaselineDTL_onSettle should credit seller quote token proceeds	✅	✅	✅	10M+
AX-36	Baseline token total supply after _onCancel should equal 0	✅	✅	✅	10M+
AX-37	BaselineDTL_onCancel should mark auction completed	✅	✅	✅	10M+
AX-38	When calling BaselineDTL_onCancel DTL base token balance should equal 0	✅	✅	✅	10M+
AX-39	When calling BaselineDTL_onCancel baseline contract base token balance should equal 0	✅	✅	✅	10M+
AX-40	BaselineDTL_onCurate should credit auction house correct base token fees	✅	✅	✅	10M+
AX-41	After BaselineDTL_onSettle baseline token base token balance should equal 0	✅	✅	✅	10M+
AX-42	After BaselineDTL_onSettle baseline pool base token balance should equal baseline pool supply	✅	✅	✅	10M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
AX-43	After BaselineDTL_onSettle seller baseline token balance should equal 0	✅	✅	✅	10M+
AX-44	circulating supply should equal lot capacity plus curatorFee minus refund	✅	✅	✅	10M+
AX-45	BaselineDTL_onSettle should mark auction complete	✅	✅	✅	10M+
AX-46	After BaselineDTL_onSettle floor reserves should equal floor proceeds	✅	✅	✅	10M+
AX-47	After BaselineDTL_onSettle anchor reserves should equal pool proceeds - floor proceeds	✅	✅	✅	10M+
AX-48	After BaselineDTL_onSettle discovery reserves should equal 0	✅	✅	✅	10M+
AX-49	After BaselineDTL_onSettle floor bAssets should equal 0	✅	✅	✅	10M+
AX-50	After BaselineDTL_onSettle anchor bAssets should be greater than 0	✅	✅	✅	10M+
AX-51	After BaselineDTL_onSettle discovery bAssets should be greater than 0	✅	✅	✅	10M+
AX-52	UniswapV2DTL_onSettle should not fail with 'UniswapV2Library: INSUFFICIENT_LIQUIDITY'	✅	❌	✅	10M+
AX-53	Profit should not be extractable due to UniswapV3Pool price manipulation	✅	❌	N/A	10M+

Findings & Resolutions

ID	Title	Category	Severity	Status
C-01	Gaming Uni V3 Initialization	DoS	Critical	Resolved
C-02	Uniswap V2 Auction Settlement DoS	Logical Error	Critical	Resolved
H-01	Curator Fees Will DoS Auction Settlement	Logical Error	High	Resolved
H-02	Pool Percentage Causes DoS	DoS	High	Resolved
H-03	Baseline Settlement DoS Via External Liquidity	DoS	High	Resolved
H-04	bAsset Price Set Below Floor	Gaming	High	Partially Resolved
H-05	Malicious Sellers Can Steal Auction Proceeds	Gaming	High	Resolved
H-06	Anchor Width Param Will DoS Auction Settlement	DoS	High	Resolved
H-07	Lot Creation DoS Through Uniswap Pair	DoS	High	Resolved
H-08	Baseline Launches Can Not Be Done	DoS	High	Resolved
M-01	discoveryTickWidth Configured Too Low	Logical Error	Medium	Resolved
M-02	Inconsistent Prices Between Pool and Auction	Validation	Medium	Resolved
M-03	Incorrect Liquidity Structure Deployed	Logical Error	Medium	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
M-04	Auction Settlement DoS'd By Creator	DoS	Medium	Acknowledged
M-05	Blacklisted Addresses Halt Auction Settlement	DoS	Medium	Acknowledged
M-06	Outdated Vesting Modules Used	Logical Error	Medium	Acknowledged
L-01	console.log Present in Code	Best Practices	Low	Declined
L-02	slide Inoperable Due To Floor Config	Validation	Low	Resolved
L-03	Early Unvesting Is Possible	Logical Error	Low	Resolved
L-04	Unused active Flag	Validation	Low	Resolved
L-05	Discovery Range Loose Validations	Validation	Low	Acknowledged
L-06	Unused Error	Best Practices	Low	Resolved
L-07	High floorReservesPercent Causes DoS	Validation	Low	Resolved
L-08	Arbitrary Byte Length Risk	Warning	Low	Resolved
L-09	Esoteric Token Pairs Are Not Supported	Warning	Low	Acknowledged
L-10	Launches Restrict Liquidity Structure	Validation	Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-11	Misleading Comment	Best Practices	Low	Resolved
L-12	Seller May Receive All Proceeds	Logical Error	Low	Resolved

C-01 | Gaming Uni V3 Initialization

Category	Severity	Location	Status
DoS	● Critical	UniswapV3DTL.sol: 193, 205	Resolved

Description PoC

Through initializing the Uniswap V3 pool, a malicious user can manipulate the price of the tokens from an auction and sell their newly purchased tokens for a profit. This attack is possible because any person can call `settle()` and set the `maxSlippage` to an unreasonably high number. In order to execute this attack, a malicious user would need to:

1. Buy tokens in an auction.
2. Initialize a Uniswap V3 Pool with a high price.
3. Call `settle()` with `maxSlippage` set to `100e2`.
4. Sell their tokens in the Uniswap V3 Pool.

It is also worth noting that if `settle()` is first called by the seller with a reasonable slippage tolerance, the transaction will revert, and the attacker will be able to follow up and execute the transaction with the proper slippage.

Recommendation

There are two possible solutions for this issue. The first is to make `settle()` only callable by the seller, so that they can specify the slippage tolerance. Then wrap the callback in a `try/catch` block. Inside the `catch`, transfer the tokens back to the seller if the transaction reverts.

Alternatively if you do not wish to make changes to the core system, add `minSlippageAmtOutToken0` and `minSlippageAmtOutToken1` to `DTLConfiguration`. This way the owner can be certain that slippage is not set to an intolerable amount, regardless of who calls `settle()`.

Resolution

Axis Team: The issue was resolved in commit 04e73e9.

C-02 | Uniswap V2 Auction Settlement DoS

Category	Severity	Location	Status
Logical Error	● Critical	UniswapV2DTL.sol: 119	Resolved

Description PoC

When a UniV2 pool is used as apart of the token launch the settle function will attempt to add liquidity to the pair. However this will only be successful if the pool has either a non-zero balance of both assets or a zero balance of both assets.

If the pair has a non-zero reserve for one asset and a zero reserve for the other it will revert when attempting to add liquidity. This is due to the following check when the `addLiquidity` function calls the `quote` function: `require(reserveA > 0 && reserveB > 0, 'UniswapV2Library: INSUFFICIENT_LIQUIDITY');`

Typically it is not possible to get a pair into this state as the first time adding liquidity to a pair, both assets must be added. However an attacker can bypass this by donating a dust amount of one of the assets to the pair and then calling `sync`. By doing this the reserves of that pair will update and to having one reserve being zero and the other being non-zero. Thus causing the revert when the auction tries to settle.

All an attacker would need to do in this case is monitor Axis auction creations, create the pair ahead of time and donate 1 wei of the quote token. By doing this the auction will not be able to settle.

Recommendation

Calculate the amount to be minted to the pair and call the `mint` function instead of `addLiquidity` to avoid this DOS case.

Resolution

Axis Team: The issue was resolved in PR#13.

H-01 | Curator Fees Will DoS Auction Settlement

Category	Severity	Location	Status
Logical Error	High	BaselineAxisLaunch.sol: 456	Resolved

Description PoC

Auction seller can opt to set a curator in charge of approving a batch auction lot. This curator will receive some base tokens. The issue relies on Baseline auctions, as curator fees were not expected to be enabled, but fees in bAssets are actually minted during the `onCurate` callback, as this permission flag is enabled in the callback contract.

This additional bAssets minted will cause the `onSettle` callback to revert, as curator fees increases the bAsset spot supply, breaking the capacity invariant (capacity ratio < 100%) and causing a DoS on auction settlement

Recommendation

If curator fees are enabled for Baseline Auctions, consider accounting for these fees during `onSettle` callback. Otherwise, revert during the `onCurate` callback.

Resolution

Axis Team: The issue was resolved in commit 47a44a0.

H-02 | Pool Percentage Causes DoS

Category	Severity	Location	Status
DoS	High	BaselineAxisLaunch: 313	Resolved

Description PoC

For Baseline, allowing a user to set the `poolPercent`, can cause a revert in `_onSettle()`. If the `poolPercent` is not configured to 100%, the revert will occur due to the `capacityRatio` not being within a tolerable range.

The `capacityRatio` is not satisfied because there is not enough backing liquidity to support the Baseline system. Since the revert is not caught in `_onCreate()`, an auction will be carried out, but then it will become unsettleable.

Recommendation

Do not allow the user to configure `poolPercent`, it should be set to 100%.

Resolution

Axis Team: The issue was resolved in commit 47a44a0.

H-03 | Baseline Settlement DoS Via External Liquidity

Category	Severity	Location	Status
DoS	High	BaselineAxisLaunch.sol: 592	Resolved

Description PoC

Reserves added as liquidity during Baseline token launches must have enough capacity to buy back all circulating bAssets at the floor price. This verification is conducted in the _onSettle function, and the total capacity must fall within the range of 100% to 102%.

The calculation of capacity involves the total liquidity amounts of the BPOOL contract in floor, anchor, and discovery ranges. However, anyone can add liquidity on behalf of the BPOOL before settlement, increasing the total capacity beyond 102% and causing the _onSettle callback to fail.

Recommendation

It is suggested to verify the LP token balance of the BPOOL before adding reserves/liquidity to the underlying pool and burn these tokens. Alternatively, consider allowing any capacity ratio above 100% without restricting it to 102%.

Resolution

Axis Team: The issue was resolved in commit 3e4061c.

H-04 | bAsset Price Set Below Floor

Category	Severity	Location	Status
Gaming	High	Global	Partially Resolved

Description PoC

The Uniswap pool is set up and initialized with the BPOOL contract before the auction begins.
The pool will initially have 0 liquidity, as a result, a malicious actor may trigger a swap with 0 bAsset input amount and a price limit to set the price where they wish. The pool price can then be assigned to below the floor tick, perturbing the launch of the system.

Recommendation

Consider implementing logic in the `_onSettle` callback to re-adjust the price back to the upper tick of the anchor range by swapping with 1 wei reserves in with a priceLimit at the desired price to assign the price to the bottom of the upper tick range of the Anchor.

Resolution

Axis Team: The issue was resolved in commit d37f1b2.

Guardian Team: If a malicious actor initializes a large tick for a Uniswap V3 launch, they can then add single sided liquidity to the pool to perform a DoS. The single sided liquidity, provided in the quote token, will prevent the swap in `_createAndInitializePoolIfNecessary()` from reaching the target price for the pool.

This will lead to a Callback_Slippage revert in `_mintAndDeposit()`, and will prevent an auction from being settled. Ultimately, an attacker can still DoS both Baseline and Uni V3 launches with enough assets, though this action will be costly and not profitable.

H-05 | Malicious Sellers Can Steal Auction Proceeds

Category	Severity	Location	Status
Gaming	High	UniswapV2DTL.sol: 119	Resolved

Description PoC

Uniswap V2 `onCreate` callback requires that the pair does not exist, but the `onSettle` callback will not revert if someone creates it before the auction settlement. Therefore, a malicious auction seller can create the pair just before the auction ends, adds liquidity in a proportion that results in a higher pool price, settle the auction and steal most of the quote tokens from the proceeds as a refund.

Recommendation

Consider creating the UniswapV2 pair during auction settlement.
Alternatively, make sure the AuctionHouse receives both the capacity and the tokens required for liquidity up front. Be advice that some sellers will be deterred as they may need these tokens in their protocol treasury.

Resolution

Axis Team: The issue was resolved in PR#13.

.

H-06 | Anchor Width Param Will DoS Auction Settlement

Category	Severity	Location	Status
DoS	High	BaselineAxisLaunch.sol: 367	Resolved

Description

Auction seller can set some initial params for the Baseline callback, like the `anchorTickWidth`. The `onCreate` function requires this value to be between 0 and 10.
However, setting a high anchor width increases the `ANCHOR` capacity, increasing the `capacityRatio`, DoS'ing the auctions settlement.

Recommendation

Instead of having a fixed anchor width, follow the same pattern as the `BaselineInit` policy, passing the desired `floorTickL` and calculating the `anchorTickL` with:
`int24 anchorTickL = max(activeTS - (ANCHOR_WIDTH * T_S), _floorTickL + T_S);`.

Resolution

Axis Team: The issue was resolved in commit 47a44a0.

H-07 | Lot Creation DoS Through Uniswap Pair

Category	Severity	Location	Status
DoS	High	UniswapV2DTL.sol: 83, UniswapV3DTL.sol: 110	Resolved

Description

Upon creating a lot which uses a DTL callback, the `onCreate` callback is invoked. For the `UniswapV2DTL` and `UniswapV3DTL` callbacks, the `__onCreate` function validates that the pair for the baseToken and quoteToken combination does not already exist.

However a malicious actor may observe the transaction to create a lot and frontrun this transaction to create a Uniswap V2 or V3 pair with the same baseToken and quoteToken. Notice that this action does not require owning either of the tokens in the pair. As a result the `onCreate` callback will revert, causing the lot creation to revert.

Recommendation

Remove the pair existence validation in the `__onCreate` function in the both the `UniswapV2DTL` and `UniswapV3DTL` files as it is unnecessary and introduces a DoS vector.

Resolution

Axis Team: The issue was resolved in commit d9a0fc1.

.

H-08 | Baseline Launches Can Not Be Done

Category	Severity	Location	Status
DoS	High	BaselineAxisLaunch.sol	Resolved

Description

When the BPOOL contract is deployed, token transfers are locked and must be unlocked via policy contracts. In the underlying Baseline protocol, the BaselineInit policy unlocks transfers after launch. The BaselineAxisLaunch contract serves as the BaselineInit policy. However, it does not have permission to call the `setTransferLock` function.

An auction with BPOOL tokens can be created since this action only requires minting and does not require a base token transfer. However, other actions such as cancelling an auction, settling an auction, or using base tokens for swaps are not possible due to transfers being locked.

Recommendation

Include the `setTransferLock.selector` in the `requestPermissions` function when configuring the policy, and unlock transfers after an auction is created.

Resolution

Axis Team: The issue was resolved in commit 301bbb2.

M-01 | discoveryTickWidth Configured Too Low

Category	Severity	Location	Status
Logical Error	Medium	BaselineAxisLaunch: 303	Resolved

Description PoC

Baseline sets DISCOVERY_WIDTH to 350 in MarketMaking.sol, however _onCreate() only validates that it is set to at least one. Setting it to 350 will ensure that there is enough liquidity spread out through a range that will not cause the price of the base token to increase in an unprecedented manner.

Having a larger discovery range will also mint more bAssets, making it harder to break through the discovery range without affecting the capacity ratio check. The protocol allows for anchor range widths between 1 and 10, as well as discovery range widths above 0. However, in the underlying Baseline protocol, anchor and discovery range widths are fixed values of 10 and 350, respectively.

When the sweep or slide functions in the MarketMaking contract are called, liquidity structure will be rebalanced based on these constant values from Baseline, regardless of the initial configuration. Additionally, reserves will be moved from the floor range to the anchor range during sweep in this case, which is an unwanted situation in Baseline. This happens because sweep function adds liquidity to anchor first, and then adds reserves to floor.

However, the same amount of liquidity for a much wider anchor range will require more reserves, causing floor reserves to decrease. Also, it is crucial for the discovery range to be wide enough and filled with BPOOL tokens in order to provide sufficient liquidity for swaps and ensure healthy price movements. A narrow discovery range could result in a single swap moving the current price tick well above the upper discovery tick, where there is minimal liquidity.

Other differences compared to Baseline are floorReservesPercent value being configurable (which affects liquidity thickness), and lack of gap between floor and anchor ranges. In baseline, floor has the thickest liquidity and anchor liquidity is much lower compared to floor.

Recommendation

Set the discovery range to 350 tick spacings. Consider using the Baseline range widths instead of allowing them to be configured by the seller. Also make sure that floor range has much more reserves compared to anchor.

Resolution

Axis Team: The issue was resolved in commit 8e1481c.

M-02 | Inconsistent Prices Between Pool and Auction

Category	Severity	Location	Status
Validation	Medium	Lack Of Present Code	Resolved

Description

The seller of the bAsset determines the open price for the Uniswap V3 Pool when creating the BPool. This is done by passing _initialActiveTick in the constructor. Fixed-Priced Auctions are intended to sell the asset at an opening price, but there is no validation that the Auction price matches the Uniswap V3 Pool price. Since no validation occurs, a malicious seller can perform two different dishonest actions:

1. Set the price of the bAsset to a lower price than the Auction. This will cause any users who bought tokens in an auction to immediately incur an unrealized loss. Additionally, the seller can purchase tokens at a discount.

2. Set the price of the bAsset to a higher price than the Auction. In this scenario, a seller can set themself as the curator, and sell the curator reward. This action can also push the Uniswap V3 Pool price back to the Fixed-Priced Auction price.

Recommendation

Verify that the price being initialized is the same price that is being used for Fixed-Price Auction.

Resolution

Axis Team: The issue was resolved in commit 8b5be94.

M-03 | Incorrect Liquidity Structure Deployed

Category	Severity	Location	Status
Logical Error	Medium	BaselineAxisLaunch.sol: 592	Resolved

Description PoC

Baseline auction settlement will revert if the `capacityRatio` is not between 100% and 102%. This makes sure that the total capacity on the system is greater than the spot supply of `bAssets`. However, the 102% check is specific for 1% fee tier pools. If the Baseline auction is used for a lower fee tier, the liquidity structure deployed might be invalid, but the `onSettle` callback might not revert.

This issue allows `MarketMaking.bump()` to be executed just after the liquidity is deployed, granting arbitrageurs an opportunity to extract quote tokens from the positions.

Recommendation

Consider calling MarketMaking.bump just after the capacity ratio check, inside a try/catch, and revert if the bump succeeds. Refer to: https://github.com/0xBaseline/baseline-v2/blob/main/test/TestFoundation.sol#L276

Resolution

Axis Team: Restricted to 1% fee tier / 200 tick spacing.

M-04 | Auction Settlement DoS'd By Creator

Category	Severity	Location	Status
DoS	Medium	BaseDTL.sol: 287	Acknowledged

Description

In the `BaseDTL` callback contract the `onSettle` callback assumes that the auction creator has the necessary balance of base tokens and has set the correct approval to the callback. However a malicious auction creator may neglect to approve the callback contract to transfer the base tokens, therefore DoSing the settlement of the auction and preventing users from claiming their bids.

This results in user's funds being held captive during the auction period until the auction can be aborted.

Recommendation

Consider requiring that the Axis core system or the callback contract is pre-funded with the base tokens which will be used to create the initial liquidity during settlement.

Resolution

Axis Team: Acknowledged.

M-05 | Blacklisted Addresses Halt Auction Settlement

Category	Severity	Location	Status
DoS	Medium	BaseDTL.sol: 368	Acknowledged

Description

In the `onSettle` callback any remaining quote tokens in the contract which weren't used up due to the `proceedsUtilisationPercent` or imbalance when providing liquidity are sent back to the auction seller. However since the seller never has to directly handle the quote tokens from the auction, it is possible that the seller is blacklisted for the quote token.

Thus the `onSettle` callback will always revert when there are leftover quote tokens that would be transferred to the seller. A malicious owner of a blacklisted address could leverage this to create an auction that can never be settled using a Uniswap callback. This would lock the funds of bidders and effectively create an auction which becomes a honey pot.

Recommendation

Consider leaving extra tokens in the callback contract and tracked in a mapping to be pulled by the seller address after settlement in a separate transaction.

Otherwise consider donating the additional quote tokens to a protocol address in the event that they cannot be transferred due to a blacklisted seller, thus allowing the onSettle callback to complete, similar to how GMX does this here: https://github.com/gmx-io/gmx-synthetics/blob/1938e365dc009342aa288aa6b42fc1fd3cd9e45d/contracts/token/TokenUtils.sol#L80

Resolution

Axis Team: We acknowledge this, but don't plan to fix it. This is possible in general on Axis without a DTL callback since the seller would be sent the funds directly by the AH.

M-06 | Outdated Vesting Modules Used

Category	Severity	Location	Status
Logical Error	Medium	BaseDTL.sol: 351	Acknowledged

Description

In the `_onSettle` function, if configured, a vest is created for the LP tokens created from the callback. The linearVestingModule is ensured to be the latest vesting module in the `_onCreate` callback when the `lotConfiguration` is set, however by the time the lot is settled the `latestVersion` or `isSunset` values are not checked.

Therefore it is possible that a lot is settled with a `linearVestingModule` that is not the latest version or is sunsetted. In the event that a vulnerability is identified in the `linearVestingModule` the protocol will not be able to wait for the current vests to end and be settled to update the module.

Recommendation

Fetch the latest `linearVestingModule` with the `_getLatestLinearVestingModule` when settling the lot.

Resolution

Axis Team: This follows the pattern we use with auctions for both auction and derivative modules. The reason is that we want to give sellers confidence that the contracts that are used for their auction are immutable. There is a tradeoff here between patching bugs and being “unruggable”, and we’ve leaned towards the latter.

L-01 | console.log Present in Code

Category	Severity	Location	Status
Best Practices	Low	BaselineAxisLaunch: 578-581, 591	Declined

Description

It is inadvisable to deploy code to a live blockchain with `console.logs` present. This will waste users' gas, with no benefit added.

Recommendation

Remove all occurrences of `console.log`.

Resolution

Axis Team: Will do. Several of the statements are still there from testing right before the audit.

L-02 | slide Inoperable Due To Floor Config

Category	Severity	Location	Status
Validation	Low	BaselineAxisLaunch: 308	Resolved

Description

When verifying the floorReservesPercent, the validation only checks if the value is less than or equal to 99%. This can allow a user to deploy a pool with 0% of tokens allocated to the floor. This means that even if all bAsset holders were to sell their tokens, the floor could never be reached. This will prevent the baseline market making feature slide() from being operable.

Consider the following scenario:
FLOOR 200 - 400
ANCHOR 400-600
DISCOVERY 600 - 5000
ACTIVE TICK = 500
SLIDE TICK = 500 - 200 = 300
Since 300 resides in the floor, slide() will not be callable.

Recommendation

Require the floorReservesPercent to be set to at least 50%.

Resolution

Axis Team: The issue was resolved in commit da9cf83.

L-03 | Early Unvesting Is Possible

Category	Severity	Location	Status
Logical Error	Low	Global	Resolved

Description

Auction sellers might need to vest their LP tokens after an auction has settled. Start time and expiry time of a vesting are provided by the seller. The protocol allows vesting start time to be before `block.timestamp` to prevent a DoS by delaying the settlements. However, this allows sellers to unvest most of their LP tokens immediately after auction.

Normal scenario (start is current timestamp):
• start: `block.timestamp`
• expiry: `block.timestamp` + 1 year
• LP amount: 100
In this 1 year vesting scenario, seller can unvest 25 tokens after 3 months.

Alternative scenario (seller sets vesting start 9 years ago):
• start: `block.timestamp` - 9 years
• expiry: `block.timestamp` + 1 year
• LP amount: 100
Here, seller can unvest 90 tokens right after settlement without waiting any time.

Recommendation

One option might be storing the total vesting period, and updating expiry based on this period after minting derivative tokens during settlement. Another option to consider is determining a maximum time that can be before `block.timestamp`. It would still allow malicious seller to perform this but can decrease the impact.

Resolution

Axis Team: The issue was resolved in commit 8eeb8d6.

L-04 | Unused active Flag

Category	Severity	Location	Status
Validation	Low	BaseDTL.sol: 224	Resolved

Description

In the `_onCancel` callback implementation the active flag is set to false for the `lotConfiguration`, however the active flag is not used to validate whether a lot is active anywhere.

Recommendation

Although there are mechanisms in place to prevent the `onSettle` and `onCurate` functions being called for a cancelled lot, consider implementing additional safeguards on the BaseDTL callback side to prevent these callbacks from being called for cancelled lots.

Resolution

Axis Team: The issue was resolved in commit 81e08e5.

L-05 | Discovery Range Loose Validations

Category	Severity	Location	Status
Validation	Low	BaselineAxisLaunch.sol: 383	Acknowledged

Description

In the onCreate callback the `discoveryRangeUpper` is validated to be within the `MAX_TICK`, however if the `discoveryRangeUpper` is assigned such that it is even close to the `MAX_TICK` this can lead to a DoS of the Baseline liquidity operations down the road.

For example:
TickSpacing: 20
Discovery ticks: [MaxTick - 200, MaxTick]
A sweep cannot be triggered in this case as it would cause the discovery upper tick to exceed the max tick.

Although it is unlikely that a configuration is made where the `discoveryUpperTick` is close to the `MAX_TICK`, such a configuration should not be allowed.

Recommendation

Consider restricting the validation on the `discoveryMaxTick` further to an expected range.

Resolution

Axis Team: The issue was resolved in PR#11.

L-06 | Unused Error

Category	Severity	Location	Status
Best Practices	Low	BaselineAxisLaunch.sol: 49	Resolved

Description

`Callback_Params_PoolTickMismatch` error is defined but never used.

Recommendation

Remove unused errors.

Resolution

Axis Team: The issue was resolved in commit b257373.

L-07 | High floorReservesPercent Causes DoS

Category	Severity	Location	Status
Validation	Low	BaselineAxisLaunch.sol: 308	Resolved

Description PoC

In the onCreate callback the floorReservesPercent is validated to be no more than 99%, however this loose validation allows users to create a liquidity structure where the majority of the liquidity is allocated to the Floor position.

As a result a malicious actor is able to buy through all of the Anchor and Discovery range liquidity as it can be very thin. The malicious actor can then set an LP outside of the discovery range in order to assign the price outside of the liquidity structure.
Once the active price is allowed to exceed the Discovery range liquidity operations will be DoS'd as a sweep cannot occur.
https://github.com/0xBaseline/baseline-v2/blob/60bed78b7bee28016321ddd8c590df6c61bae6e9/src/policies/MarketMaking.sol#L217

Recommendation

Consider making the floorReservesPercent more strict such that it is unlikely that a liquidity structure can be deployed where a malicious actor can easily buy through all of the available liquidity to DoS the system.

Resolution

Axis Team: The issue was resolved in commit 47e9cd5.

L-08 | Arbitrary Byte Length Risk

Category	Severity	Location	Status
Warning	Low	UniswapV3DTL.sol	Resolved

Description

`params.implParams` bytes are saved into storage within `_onCreate`, and operations may appear fine from the perspective of the user. During settlement callback when `mintAndDeposit` is executed, the `implParams` are loaded into memory:

```
(uint24 poolFee) = abi.decode(lotConfiguration[lotId].implParams, (uint24));
```

The first 24 bits can be decoded properly into a `uint24`, but there is no guarantee that the bytes do not have arbitrary padding afterwards that inflate the payload. Consequently, there will be extremely large gas costs associated with the load into memory upon settlement. It's important to note that there isn't a large risk since the callbacks do not have a predetermined max gas limit, but noteworthy for future-proofing operations.

Recommendation

Consider documenting this or placing a cap on the length of the bytes.

Resolution

Axis Team: The issue was resolved in commit 04e73e9.

L-09 | Esoteric Token Pairs Are Not Supported

Category	Severity	Location	Status
Warning	Low	SqrtPriceMath.sol: 34	Acknowledged

Description

The `getSqrtPriceX96` function used for the `UniswapV3DTL` callback computes the `ratioX192` and stores it as a `uint256` before taking the square root to compute the `sqrtPriceX96Temp`.
As a result with some esoteric token pairs which have a large difference in amounts, the intermediate `ratioX192` value can overflow the `uint256` size.

For example `uint160 price = SqrtPriceMath.getSqrtPriceX96(address(_baseToken), _BASELINE_QUOTE_TOKEN, 1e6, 1e26);` causes such an overflow.

Such a combination of tokens may arise when one token has low precision, such as USDC and another one has high precision such as YAMv2 or a very low price.

Recommendation

Be wary of this when supporting tokens and token prices in auctions. It is unlikely that supported token pairs would cause this issue. However it may be worth adding validations to disallow auctions which would cause this logic to revert based on token decimals and prices.

Resolution

Axis Team: In general, Axis supports tokens with between 6 and 18 decimals. However, given the price set can cause the difference to be larger, we acknowledge this could happen, but the likelihood is low.

L-10 | Launches Restrict Liquidity Structure

Category	Severity	Location	Status
Validation	Low	BaselineAxisLaunch.sol	Resolved

Description

In the `_onCreate` callback the validation correctly requires that the anchor range cannot exceed a width of 10 tick spacings, however the floor range upper tick is always assigned to the lower tick of the anchor range.

Therefore there can be no liquidity structures with a gap in between the upper floor tick and the lower anchor tick as will commonly be the case where there is a significant premium to the baseline value. As a result the initial liquidity structure configuration is limited in that the active price (upper tick range of the anchor position) cannot be more than 10 ticks above the floor position.

Recommendation

Consider allowing a separate configuration variable for the upper floor tick, where the liquidity structure can have a gap between the floor and the lower anchor range. Be sure to maintain the existing Anchor width validation and add relevant validation such that the anchor cannot collide with the floor and the anchor and floor are within a reasonable distance from each other.

Additionally, if the Anchor range is within 10 tick spacings of the floor range there should be no gap between the two positions.

Resolution

Axis Team: The issue was resolved in commit 9839ce1.

L-11 | Misleading Comment

Category	Severity	Location	Status
Best Practices	Low	BaseDTL.sol: 359	Resolved

Description

"Send the LP tokens to the seller" comment in L359 of the `BaseDTL` contract is misleading since LP tokens are transferred to the `recipient` not to the seller.

Recommendation

Consider updating the comment.

Resolution

Axis Team: Resolved.

L-12 | Seller May Receive All Proceeds

Category	Severity	Location	Status
Logical Error	Low	Global	Resolved

Description

Both Uniswap and Baseline auction callbacks allow the owner to set a percentage of the proceeds when adding liquidity during auction settlement. The issue relies on the validation for this percentage. Baseline auction requires this value to be within 1% and 100%, while Uniswap requires 0% to 100%.

This allows auction sellers to set a very small value, and receive most of the proceeds which were meant to be added as liquidity.

Recommendation

Consider increasing the lower limit for the proceeds used for liquidity providing.

Resolution

Axis Team: Resolved.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits