GA GUARDIAN

Baseline

MM Looping #1

Security Assessment

February 25th, 2025

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand

Client Firm Baseline

Final Report Date February 25, 2025

Audit Summary

Baseline engaged Guardian to review the security of their market making looping updates. From the 18th of January to the 22nd of January, a team of 2 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement 3 High/Critical issues were uncovered and promptly remediated by the Baseline team.

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

🔗 Blockchain network: **Base**

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

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/Baseline-Perps

Table of Contents

Project Overview

Project Summary

Project Name	Baseline
Language	Solidity
Codebase	https://github.com/0xBaseline/baseline-v2
Commit(s)	9664eb6679395d34e9bff36f8c919cb3f859ed3b

Audit Summary

Delivery Date	February 25, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

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

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	🔴 Critical	🟠 High	🟡 Medium
Likelihood: *Medium*	🟠 High	🟡 Medium	🟢 Low
Likelihood: *Low*	🟡 Medium	🟢 Low	🟢 Low

Impact

High Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.

Medium A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.

Low Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.

Medium An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.

Low Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

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.

Findings & Resolutions

ID	Title	Category	Severity	Status
H-01	Sqrt Price Compared With Price	Logical Error	High	Resolved
H-02	Liquidity Rebalance Arbitrage	Gaming	High	Partially Resolved
H-03	Third Party Liquidity Results In DoS	Logical Error	High	Resolved
M-01	Incorrect Bump Calculation	Logical Error	Medium	Acknowledged
L-01	Unnecessary Min	Optimization	Low	Resolved
L-02	Unused Variable	Gas Optimization	Low	Resolved
L-03	Lacking Threshold Liquidity Cap	Validation	Low	Acknowledged
L-04	_getUtilizationRate Reverts Near Min Tick	DoS	Low	Acknowledged
L-05	Rebalances Inconsistently Responsive	Unexpected Behavior	Low	Acknowledged
L-06	Lacking Balance Sweeps	Defensive Code	Low	Acknowledged
L-07	Outdated getBaselineValue Function	Warning	Low	Acknowledged
L-08	getCirculatingSupply Includes BPOOL Assets	Unexpected Behavior	Low	Acknowledged
L-09	Lacking blvTick Validation	Validation	Low	Acknowledged

H-01 | Sqrt Price Compared With Price

Category	Severity	Location	Status
Logical Error	High	MarketMaking.sol: 428	Resolved

Description

In the `_getUtilizationRate` function the `priceAdj` is computed as:

`FixedPointMathLib.divWad(TickMath.getSqrtRatioAtTick(activeTickAdj), FixedPoint96.Q96);`

Which has units of the square root of the price. However the `priceAdj` is compared against the result of `getBLV` to compute the `premiumRatio`.

The result of `getBLV` is a price, instead of a square root price. Therefore the `premiumRatio` is incorrect.

Recommendation

Square the `priceAdj` to compute the correct price.

Resolution

Baseline Team: Resolved.

H-02 | Liquidity Rebalance Arbitrage

Category	Severity	Location	Status
Gaming	High	MarketMaking.sol	Partially Resolved

Description

The `_updateTicks` logic intentionally assigns the `anchorTick` such that new `bAssets` are not minted within the anchor range when the `bAssets` minted would be in addition to the liquidity in the discovery range.

This is to avoid the following arbitrage attack:
• Anchor liquidity < Discovery liquidity
• An attacker makes a large sell through the discovery range and into the anchor range
• Due to the instantaneous large sell, and the leveraging of the anchor liquidity, the anchor liquidity becomes greater than the Discovery liquidity
• Now the attacker can buy back the same amount of bAssets but at a lower average price because of the increased Anchor liquidity relative to the liquidity they sold through.

There is however another similar arbitrage attack which is not protected against:
• Anchor liquidity < Discovery liquidity
• Instead of selling through the discovery range, the attacker sells from the top of the Anchor range
• After the sell, the rebalance causes the higher liquidity discovery range to come down closer to the new price
• Now the attacker can buy back the same amount of bAssets but at a lower average price because of the increased Discovery liquidity relative to the liquidity they sold through.
• As long as the leveraging of the Anchor position is not greater than the liquidity difference between the Anchor and Discovery then this arbitrage is profitable.

Recommendation

Consider rate limiting the amount of ticks that can be dropped at a time to limit the scale of this arbitrage vector.

Resolution

Baseline Team: Partially Resolved.

H-03 | Third Party Liquidity Results In DoS

Category	Severity	Location	Status
Logical Error	High	BPOOL.sol: 442	Resolved

Description

The `removeAllFrom` function in the BPOOL contract reports the entirety of third party liquidity as fees when the `liquidityToRemove > currentLiquidity`, however the excess liquidity is burned from the `msg.sender` in the `_removeLiquidity` function.

This means that the calling contract will think it has more `bAssets` than it does because the reported `bAssetFees_` includes an amount that was burnt from the sender.

During rebalances this results in an underflow DoS when ultimately attempting to send more `bAssets` then the contract holds to the fee receiver for `bAssetFees_`.

Recommendation

Do not burn the `bAssets` from the `msg.sender` in the `_removeLiquidity` function when removing third party liquidity.

Resolution

Baseline Team: Resolved.

M-01 | Incorrect Bump Calculation

Category	Severity	Location	Status
Logical Error	Medium	MarketMaking.sol: 287	Acknowledged

Description

The criteria for a bump is described as:

`That the total reserves in the system (inclusive of debt), when placed across the anchor position (with no reserves the floor), is enough to buy back the entire circulating supply.`

However, in the `_canBump` function the capacity is calculated with the `activeX96` as the upper to the Anchor range.

This is however flawed because the `activeX96` may not reside within the new Anchor range. Instead the `anchorTick` may be selected such that the `activeX96` is actually within the Discovery range.

This would not be an issue if the Discovery and Anchor ranges were guaranteed to have the same concentration of reserves.

However it is possible that the Discovery range liquidity is actually lower than the Anchor range liquidity, in which case assuming that the reserves were evenly spread out across this range would underestimate the capacity of the protocol and errantly indicate that a bump would not be possible when in fact it can be.

Recommendation

Consider executing the bump logic after the new `anchorTick` and liquidities of the Anchor and Discovery ranges have been defined. Then do not allow the bump logic to change the liquidity of the Anchor or discovery.

Resolution

Baseline Team: Acknowledged.

L-01 | Unnecessary Min

Category	Severity	Location	Status
Optimization	Low	MarketMaking.sol: 395	Resolved

Description

The `_getAnchorReserves` function uses the min function between `anchorReserves_` and `totalReserves - _getVirtualReserves()` inside the case where it is already determined that `totalReserves - _getVirtualReserves() < anchorReserves_`.

Therefore the `min` computation is unnecessary and the `anchorReserves_` can always just be assigned to the `totalReserves - _getVirtualReserves()` value.

Recommendation

Remove the `min` computation and always assign the `anchorReserves_` to `totalReserves - _getVirtualReserves()`.

Resolution

Baseline Team: Resolved.

L-02 | Unused Variable

Category	Severity	Location	Status
Gas Optimization	Low	MarketMaking.sol: 364	Resolved

Description

In the `_getACU` function the `activeX96` is declared but never used.

Recommendation

Remove the `activeX96` variable from the `_getACU` function.

Resolution

Baseline Team: Resolved.

L-03 | Lacking Threshold Liquidity Cap

Category	Severity	Location	Status
Validation	Low	MarketMaking.sol: 303	Acknowledged

Description

When deploying the threshold liquidity to the Discovery position in the `_deployLiquidity` function there is no cap on the amount of reserve assets that can be used.

It may be possible in some rare cases that the active price is within the Discovery range during a rebalance and the corresponding reserves requested by the `_getThresholdLiquidity` result are greater than the amount sitting in the `MarketMaking` contract due to most of the reserves being virtual reserves.

In this scenario the current logic will revert instead of gracefully handling the conditions, thus preventing a rebalance.

Recommendation

Consider how this edge case should be handled. If a revert is acceptable then consider explicitly reverting in this case.

Otherwise gracefully handle the case where the `_getThresholdLiquidity` result requests more reserves than are available in the `MarketMaking` contract, similarly to how this is handled in the `_getAnchorReserves` function.

Resolution

Baseline Team: Acknowledged.

L-04 | _getUtilizationRate Reverts Near Min Tick

Category	Severity	Location	Status
DoS	Low	MarketMaking.sol: 425	Acknowledged

Description

In the `_getUtilizationRate` function the active tick is reduced by the `BUMPABLE_PREMIUM` which is currently set at 1500 ticks.

In scenarios where the active tick is near the min tick, this will lead to a subsequent revert when attempting to do computations with a resulting tick that is under the min tick.

Recommendation

Ensure that system configurations never allow for any active price to be near the min tick.

Resolution

Baseline Team: Acknowledged.

L-05 | Rebalances Inconsistently Responsive

Category	Severity	Location	Status
Unexpected Behavior	Low	MarketMaking.sol	Acknowledged

Description

In `_updateTicks` the `rebalanceTicks` assignment does not take into account where the active price is, but rather purely where the `anchorTick` was assigned to.

This can result in scenarios where the active price has to travel a minimum distance of 100 ticks to trigger a rebalance, or a maximum distance of 300 ticks to trigger a rebalance.

This makes rebalances less or more responsive during different types of price action which may be unexpected and lead to unintended results.

Recommendation

Consider taking the `activeTick` into account when assigning the `rebalanceTicks` so that liquidity rebalances are triggered in a more uniform manner.

Resolution

Baseline Team: Acknowledged.

L-06 | Lacking Balance Sweeps

Category	Severity	Location	Status
Defensive Code	Low	MarketMaking.sol	Acknowledged

Description

In the `_removeLiquidity` function there is no logic to set aside the reserve assets which may be sitting in the `MarketMaking` contract before the liquidity positions are removed.

Any reserves which were artificially sent to the `MarketMaking` contract are able to affect the market making operations because `balanceOf` is used liberally throughout the logic.

This can potentially be used to manipulate a number of things, notably the `anchorTick` can be influenced to be the upper or lower tick based upon the buffer reserves which are based upon the `balanceOf`

Recommendation

Introduce a `bufferedReserves` approach similar to the previous iteration of the `MarketMaking` policy:

https://github.com/0xBaseline/baseline-v2/blob/6434202087be8278f09016f28be7dc9933d1085c/src/policies/MarketMaking.sol#L434

Resolution

Baseline Team: Acknowledged.

L-07 | Outdated getBaselineValue Function

Category	Severity	Location	Status
Warning	Low	BPOOL.sol	Acknowledged

Description

In the `BPOOL` contract the `getBaselineValue` function still returns the original baseline value of the lower floor tick instead of the upper floor tick.

Recommendation

Update this function to reflect the real baseline value of the upper floor tick.

Resolution

Baseline Team: Acknowledged.

L-08 | getCirculatingSupply Includes BPOOL Assets

Category	Severity	Location	Status
Unexpected Behavior	Low	MarketMaking.sol	Acknowledged

Description

The `getCirculatingSupply` function does not subtract the BPOOL contract balance from it's result and therefore reports any `bAssets` sitting in the BPOOL as circulating supply when in fact they should not be.

This does not cause any immediate issues in the market making logic because the BPOOL assets are burned before this function is used. However for integrators and public display via this view function the BPOOL bAsset amount should be removed from the circulating supply result.

Recommendation

Deduct the BPOOL `bAssets` from the result of the `getCirculatingSupply` function.

Resolution

Baseline Team: Acknowledged.

L-09 | Lacking blvTick Validation

Category	Severity	Location	Status
Validation	Low	MarketMaking.sol	Acknowledged

Description

The MarketMaking contract does not impose any validation on the starting value of the blvTick, therefore it is possible that a deployment of the market making policy is based on a blvTick that does not agree with the upper tick of the floor position.

This is currently the case in the TestFoundation as the BaselineInit.launch invocation uses the INITIAL_FLOOR_TICK as the initial floor lower tick while the MarketMaking deployment uses the same INITIAL_FLOOR_TICK as the initial blvTick.

This deployment allows for a scenario where the liquidity structure cannot absorb all supply since funds can initially be borrowed in the credit and looping facilities at a higher baseline value which is based on the upper tick of the floor position rather than the blvTick.

Recommendation

Consider adding validation in the constructor of the MarketMaking policy to ensure that no errant deployments can happen which do not have agreement between the blvTick and the range assigned in the BPOOL module.

Resolution

Baseline Team: Acknowledged.

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