CrocSwap V2

Download

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	AMM
Timeline	2023-03-23 through 2023-03-23
Language	Solidity, TypeScript
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	CrocSwap Whitepaper Tick Grids: An Illustration Decomposition of Liquidity: An Illustration Liquidity Curves: An Illustration Liquidity Fee Bounds CrocSwap Docs
Source Code	• CrocSwap/CrocSwap-protocol #7784d34
Auditors	• Kacper Bak Research Engineer • Alejandro Padilla Auditing Engineer

Documentation quality	High
Test quality	High
Total Findings	12 Fixed: 8 Acknowledged: 3 Mitigated: 1
High severity findings ⓘ	0
Medium severity findings ⓘ	0
Low severity findings ⓘ	9 Fixed: 6 Acknowledged: 2 Mitigated: 1
Undetermined severity findings ⓘ	3 Fixed: 2 Acknowledged: 1
Informational findings ⓘ	0

Summary of Findings

CrocSwap is a decentralized exchange protocol that allows for two-sided AMMs combining concentrated and ambient constant-product liquidity on any arbitrary pair of blockchain assets. Although the code well-documented, it is very complex. This audit provides additional findings and was performed after the previous Quantstamp audit and fix review. Hence, any unfixed findings from the previous report, still apply. In the current audit we have found a few issues, though they are either of low, informational, or undetermined severity. Low severity issues include the violation of checks-effects-interactions pattern, missing input validation, potential overlows and underflows. Although the code appears to be protected from reentrancies, it is important to keep in mind that the effectiveness of protections may change as the code evolves. Also, we point out a few places where return values are ignored. Finally, the code relies on oracles which may have significant impact on CrocSwap. We recommend addressing all the issues.

Update: CrocSwap addressed all the issues as of commit `511476`.

ID	DESCRIPTION	SEVERITY	STATUS
CRO-1	Checks-Effects-Interactions Pattern Violation	• Low ⓘ	Acknowledged
CRO-2	Greedy Contract	• Low ⓘ	Mitigated
CRO-3	Missing Input Validation	• Low ⓘ	Fixed
CRO-4	Unnecessary Dependencies	• Low ⓘ	Fixed
CRO-5	Integer Overflow / Underflow	• Low ⓘ	Fixed
CRO-6	Seeds Might Be Losing Value when a Pool Reaches Max Growth	• Low ⓘ	Acknowledged
CRO-7	The `seekMezzSpill()` Function May Return Incorrect Values for `MIN` / `MAX` Ticks	• Low ⓘ	Fixed
CRO-8	`recipQ64()` Could Overflow	• Low ⓘ	Fixed
CRO-9	`bumpConcentrated()` Can Overflow	• Low ⓘ	Fixed
CRO-10	Unchecked Return Value	• Undetermined ⓘ	Fixed
CRO-11	The `shaveForPrecision()` Buffer Might Be Insufficient After Assimilating Fees	• Undetermined ⓘ	Fixed
CRO-12	The `claim()` and `recover()` Knockout Commands Can Be Executed without Oracle's Approval	• Undetermined ⓘ	Acknowledged

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Disclaimer

If the final commit hash provided by the client contains features that are not within the scope of the audit or an associated fix review, those features are excluded from consideration in this report.

Also, it is important to note that the CrocSwap protocol has an upgradeable architecture. Privileged admins can add or upgrade specific parts of the code. However, this is a delicate operation, as upgrades can introduce bugs, regressions, and other undesirable code. We recommend conducting security audits and other detailed checks before performing any such upgrade on production code.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Findings

CRO-1 Checks-Effects-Interactions Pattern Violation

• Low ⓘ Acknowledged

Update

Response from CrocSwap: we acknowledge deviating from the checks-effects pattern opens up re-entrancy risk. However we believe the risk is minimal, since all public methods in the CrocSwapDex contract are entirely gated by a single re-entrancy guard for the entire dex contract. Since the re-entrancy guard is part of the core contract, even proxy contract upgrades cannot remove it. (In the case of the `CrocPolicy.transferGovernance()` , the code was refactored to conform to checks-effects pattern.) Commit: `5114761` .

File(s) affected: `mixins/SettleLayer.sol` , `mixins/MarketSequencer.sol` , `callpaths/LongPath.sol` , `governance/CrocPolicy.sol` , `CrocSwapDex.sol`

Related Issue(s): SWC-107

Description: The Checks-Effects-Interactions coding pattern is meant to mitigate any chance of other contracts manipulating the state of the blockchain in unexpected and possibly malicious ways before control is returned to the original contract. As its name implies, this pattern mandates that external calls to or interactions with other contracts be made only after checking whether appropriate conditions are met and acting internally on those conditions.

We found violations of the pattern in the following locations:

- `SettleLayer.settleFinal()` ,
- `SettleLayer.settleFlat()` ,
- `MarketSequencer.burnOverPool()` (both instances),
- `MarketSequencer.harvestOverPool()` (both instances),
- `MarketSequencer.mintOverPool()` ,
- `MarketSequencer.swapOverPool()` (both instances),
- `LongPath.settleFlows()` ,
- `LongPath.userCmd()` ,
- `CrocPolicy.transferGovernance()` , and
- `CrocSwapDex.userCmdRelayer()` .

Recommendation: We classified the issues as low severity since the way the code is used in the context of other contracts appears to be protected from the actual reentrancies via appropriate modifiers, although some effects do occur after interactions. If possible, we recommend refactoring the code so that it conforms to the Checks-Effects-Interactions pattern.

CRO-2 Greedy Contract

• Low ⓘ Mitigated

Update

Response from CrocSwap: payable was removed from `CrocPolicy` and related contracts. `payable` cannot be removed from `KnockoutPath.crossCurveFlags()` , because it is only invoked with `DELEGATECALL` . Therefore it will cause the transaction to fail on any top-level `CrocSwapDex` call with non-zero `msg.value` , since that's preserved through `DELEGATECALL` . In particular since `crossCurveFlags` is called in `swap()` , it would cause swaps to fail on pools with native ETH in the token pair. Documentation explaining this was added in the commit `226746d` .

File(s) affected: `callpaths/KnockoutPath.sol` , `governance/CrocPolicy.sol`

Description: A greedy contract is a contract that can receive ether (via functions marked as `payable` , such as `KnockoutPath.crossCurveFlag()`) which can never be redeemed.

Recommendation: We recommend adding function that allows for redeeming the assets and/or removing the modifier `payable` from functions that shall not receive any Ether.

CRO-3 Missing Input Validation

• Low ⓘ Fixed

Update

Fixed in commit `fb4801b` .

File(s) affected: `governance/CrocPolicy.sol` , `lens/CrocImpact.sol` , `lens/CrocQuery.sol` , `periphery/CrocLpErc20.sol` , `vendor/compound/Timelock.sol`

Related Issue(s): SWC-123

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. Specifically, in the following functions arguments of type `address` may be initialized with value `0x0` :

- `CrocPolicy.constructor()` ,
- `CrocImpact.constructor()` ,
- `CrocQuery.constructor()` ,
- `CrocLpErc20.constructor()` , and
- `Timelock.constructor()` .

Recommendation: We recommend adding the relevant checks.

CRO-4 Unnecessary Dependencies

• Low ⓘ Fixed

Update

Fixed in commit `4552b56` .

File(s) affected: `callpaths/BootPath.sol`

Description: The contract `BootPath` has unnecessary dependencies (both through the imports and base contracts it inherits from). Although we do not see an immediate vulnerability, the extra code contradicts the specification and adds unnecessary bytecode.

Recommendation: Remove the unnecessary dependencies.

CRO-5 Integer Overflow / Underflow

• Low ⓘ Fixed

Update

Fixed in commit `e3c4d1f` .

The functionality of `L286` depends on the assumption that `BUFFER_COLLATERAL` is always equal to `4` . If the value of `BUFFER_COLLATERAL` is ever refactored, using the literal `4` may lead to unexpected behavior. To avoid any such inconsistencies, it is recommended to replace the literal `4` on `L286` with the `BUFFER_COLLATERAL` constant.

File(s) affected: `libraries/Chaining.sol`

Related Issue(s): SWC-101

Description: Integer overflow/underflow occur when an integer hits its bit-size limit. Every integer has a set range; when that range is passed, the value loops back around. A clock is a good analogy: at 11:59, the minute hand goes to 0, not 60, because 59 is the largest possible minute.

Integer overflow and underflow may cause many unexpected kinds of behavior and was the core reason for the `batchOverflow` attack. Below is an example with `uint8` variables, meaning unsigned integers with a range of `0..255` .

```
        function under_over_flow() public {
            uint8 num_players = 0;
            num_players = num_players - 1;  // 0 - 1 now equals 255!
            if (num_players == 255) {
                emit LogUnderflow();        // underflow occurred
            }
            uint8 jackpot = 255;
            jackpot = jackpot + 1;          // 255 + 1 now equals 0!
            if (jackpot == 0) {
                emit LogOverflow();         // overflow occurred
            }
        }
```

The function `bufferCollateral()` may throw when overflow occurs. We singled out this specific instance since other instances seem to be addressed in the inline documentation.

Recommendation: We recommend fixing the issue or adding a comment explaining either why the overflow is irrelevant or intentional.

CRO-6 Seeds Might Be Losing Value when a Pool Reaches Max Growth

• Low ⓘ Acknowledged

Update

Response from CrocSwap: curves reaching max growth will stop accumulating rewards to ambient liquidity positions, but underlying collateral in the positions and collateralization of the curve will remain safe. Since reaching this point requires cumulative 6,500,000% rewards return to ambient liquidity, we believe this is unlikely to happen in economically meaningful pools. Documentation clearly outlining this possibility to end-users was added, along with advice on what should be done if a pool reaches this point (i.e. initialize a fresh pool for the same pair) Commit: `2be4a9` .

File(s) affected: `libraries/CurveAssimilate.sol`

Description: To assimilate fees into the curve, the `stepToLiquidity()` method calculates a new growth rate and the number of new ambient seeds. However, the growth rate is capped at a maximum value. Once this value is reached, the growth rate will not increase, but the ambient seeds may still increase. This could be problematic, as it would mean that instead of being rewarded, existing liquidity providers' positions would become less valuable over time.

Recommendation: Consider refactoring the code to handle this edge case properly, or add documentation to inform users what to do when a pool is close to maximizing its growth rate.

CRO-7 The `seekMezzSpill()` Function May Return Incorrect Values for `MIN` / `MAX` Ticks

• Low ⓘ Fixed

Update
Fixed in commit `df724a5` .

File(s) affected: `mixins/TickCensus.sol`

Description: The `seekMezzSpill()` method is used to determine the next tick bump boundary. Unfortunately, the method does not behave correctly when the `MIN` / `MAX` ticks are provided. If the `MIN` boundary is provided during a sell operation, the method will revert. Similarly, if `MAX` is provided during a buy operation and the last terminus tick has any other tick bookmarked, the method might return an incorrect value. This is because, unlike other upper ticks, the `MAX` tick is not past the end of the byte of the previous tick.

Recommendation: Consider handling the `MAX` and `MIN` ticks as special cases to ensure that the returned value is always correct.

CRO-8 `recipQ64()` Could Overflow

• Low ⓘ Fixed

Update
Fixed in commit `151224b` .

File(s) affected: `libraries/FixedPoint.sol`

Description: The `recipQ64()` method may lose precision if the input parameter `x` is ever `2^-64` . The correct reciprocal of that number is `2^129` . However, since that number is outside the valid `uint128` range, the method overflows and returns 0.

Recommendation: Either modify the return value from `uint128` to a larger numeric value or revert if that input value is ever used.

CRO-9 `bumpConcentrated()` Can Overflow

• Low ⓘ Fixed

Update
Fixed in commit `286e83f` .

File(s) affected: `mixins/LiquidityCurve.sol`

Description: The `bumpConcentrated()` method on line `L232` is converting the liquidity delta from a `uint128` to an `int128` without any checks. This means that if a liquidity delta has a value of `2^127` or higher, the result will overflow and return an incorrect value. Although it is unlikely that such high deltas will be encountered in practice, it is better to implement safety checks to prevent potential issues.

Recommendation: Consider using the `toInt128Sign()` method to safely cast the value.

CRO-10 Unchecked Return Value

• Undetermined ⓘ Fixed

Update
Fixed in commit `be2a2d1` . The CrocSwap team has refactored the `protocolCmd()` signature to not return any value and has also fixed the issue in `TradeMatcher` , as recommended. However, not tests have been added. To prevent future regressions, we recommend adding tests if possible.

File(s) affected: `governance/CrocPolicy.sol` , `mixins/TradeMatcher.sol`

Related Issue(s): SWC-104

Description: Some functions return data upon success (typically, but not always, as a boolean result). It is important to ensure that every necessary function is checked. We found unchecked return values in the following places:

- `CrocPolicy.sol#122, 137, 156, 159, 200` , and
- `TradeMatcher.sol#454` . The latter is worth paying attention to. Although the operation is currently a no-op (because `knockoutDelta` is always 0), this may not be the case as the code evolves.

Recommendation: We recommend either checking the return values or adding a comment that a check is unnecessary. Furthermore, remove any code that is unnecessary.

CRO-11 The `shaveForPrecision()` Buffer Might Be Insufficient After Assimilating Fees

• Undetermined ⓘ Fixed

Update
Fixed in commit `4b9a924` .

File(s) affected: `libraries/CurveMath.sol` , `libraries/CurveAssimilate.sol`

Description: Before accumulating liquidity fees into the curve, the `shaveForPrecision()` method calculates the number of tokens to use as a buffer to account for any price fixed point inaccuracies. However, the method only considers the curve's liquidity before incorporating the fees, and not after. If the liquidity changes significantly after incorporating the fees, the buffer might be insufficient and could potentially lead to under-collateralization issues.

Exploit Scenario: For example, the reserves needed for liquidity less than `2^64` are around 1 wei. However, reserves needed for liquidity of `2^70` are around 64 wei. If the liquidity changes so dramatically after incorporating fees, the buffer might be insufficient. It is unclear if such drastic liquidity changes are realistic in real-world scenarios or not.

Recommendation: Determine whether this edge case is important enough to handle. If it is, refactor the code as necessary to ensure that the reserves are always large enough to account for significant changes when assimilating liquidity.

CRO-12 The `claim()` and `recover()` Knockout Commands Can Be Executed without Oracle's Approval

• Undetermined ⓘ Acknowledged

Update
The post-knockout methods, such as `claim` and `recover` , operate on positions that are no longer active in the liquidity curve. As such, the CrocSwap team believes that they do not need to be gated by an oracle. Therefore, the team has acknowledged this issue and added in-code documentation to explain their design rationale.

Description: If a permission oracle is enabled, all operations in the Knockout Path and Warm Path can only be executed if authorized by the oracle. However, the `Claim` and `Recover` commands do not undergo this check. It is unclear whether this is by design or not.

Recommendation: Review whether the `Claim` and `Recover` commands require approval from the `ICrocPermitOracle` . If so, implement checks to ensure that these calls are only executed when authorized.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Code Documentation

1. **Update:** fixed. There is missing code documentation in the following locations:
   - `mixins/TradeMatcher.sol` (some functions)
   - `lens/CrocQuery.sol`
   - `lens/CrocImpact.sol`
2. **Update:** fixed. The Knockout Path online documentation contains some inconsistencies that we have identified:
   - Based on the `contracts/libraries/ProtocolCmd.sol` file, the command codes for the claim and recover knockout calls are 93 and 94, respectively. However, the code snippets in the online documentation have the command codes reversed.
   - The snippet for the burn call does not have the `inLiqQty` parameter.
3. **Update:** fixed. `contracts/mixins/SettleLayer.sol#L118` - The documentation for the `settleFlows()` method mentions a `useSurplus` parameter, but this parameter does not exist. Instead, the documented parameter should be `reserveFlags`.
4. **Update:** fixed. There are several spelling errors across the codebase.
5. **Update:** fixed. In `contracts/libraries/CurveMath.sol`, the comment on line `L223` is inaccurate. The `termOne` parameter is at most 192 bits.
6. **Update:** fixed. In `contracts/libraries/PoolSpecs.sol`, the comment on line `L23` is incorrect. The fees are given in 100ths of a basis point. Therefore, 0.25% would be 25 bps or 2,500 hundredths of a bps.
7. **Update:** fixed. It is important to document in the end-user documentation that the protocol fee can take up to 255/256 of the total fee. This means that in some pools, the majority of fees will be allocated to the protocol.
8. **Update:** fixed. The documentation for `bidLots_` and `askLots_` in the `contracts/mixins/LevelBook.sol` file is incorrect. Other sources of documentation and the code itself indicate that `bidLots` are added when the curve crosses the price from below, and `askLots` are added when the curve crosses the price from above. However, the description given in this file is different.
9. **Update:** fixed. In `contracts/mixins/PositionRegistrar.sol#L198`, there is an error. The `owner` parameter should be of type `address`, not `bytes32`.
10. **Update:** fixed. The comment in `contracts/mixins/PositionRegistrar.sol#L33` is incorrect. This method is actually used for ambient liquidity, not concentrated liquidity.

Adherence to Best Practices

1. **Update:** fixed. The functions `mixins/DepositDesk.depositSurplus()` and `mixins/SettleLayer.settleFinal()` are supposed to be called once per interaction. Although we have not found ways of violating this requirement, we recommend adding relevant `require()` statements to rule out such a possibility.
2. **Update:** fixed. The following methods are not used and are therefore dead code. Consider removing them if they are not needed:
   - `contracts/libraries/FixedPoint.sol - divSqQ64()`, and
   - `contracts/mixins/PositionRegistrar.sol - changePosOwner()`.
3. **Update:** fixed. The functionality in the Bitmaps library can operate on ticks outside the `MIN_TICK` and `MAX_TICK` defined in the `TickMath` library. It is important to note that it is the responsibility of the calling code to ensure that ticks remain within the valid range. Consider documenting this responsibility in the library to ensure proper usage.

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

- `730...38a ./contracts/CrocSwapDex.sol`
- `eb0...00f ./contracts/CrocEvents.sol`
- `71a...203 ./contracts/interfaces/ICrocLpConduit.sol`
- `954...3f6 ./contracts/interfaces/ICrocMinion.sol`
- `fcb...c1c ./contracts/interfaces/ICrocCondOracle.sol`
- `170...96a ./contracts/interfaces/ICrocPermitOracle.sol`
- `62c...148 ./contracts/interfaces/IERC20Minimal.sol`
- `271...3ca ./contracts/vendor/compound/Timelock.sol`
- `98f...875 ./contracts/callpaths/KnockoutPath.sol`
- `e2d...f22 ./contracts/callpaths/LongPath.sol`
- `706...26f ./contracts/callpaths/SafeModePath.sol`
- `dbe...0d4 ./contracts/callpaths/WarmPath.sol`
- `5f9...9a6 ./contracts/callpaths/BootPath.sol`
- `59b...d59 ./contracts/callpaths/HotPath.sol`
- `7d6...58c ./contracts/callpaths/ColdPath.sol`
- `c40...f5d ./contracts/callpaths/MicroPaths.sol`
- `bd4...89e ./contracts/governance/CrocPolicy.sol`
- `1a4...57d ./contracts/lens/CrocQuery.sol`
- `f77...e0f ./contracts/lens/CrocImpact.sol`
- `69a...278 ./contracts/periphery/CrocLpErc20.sol`
- `9a3...4ee ./contracts/libraries/KnockoutLiq.sol`
- `68f...a50 ./contracts/libraries/LiquidityMath.sol`
- `1d7...4ef ./contracts/libraries/Bitmaps.sol`
- `21a...405 ./contracts/libraries/Directives.sol`
- `b75...039 ./contracts/libraries/Encoding.sol`
- `57e...9fb ./contracts/libraries/BitMath.sol`
- `008...f24 ./contracts/libraries/FixedPoint.sol`
- `7cb...ec8 ./contracts/libraries/ProtocolCmd.sol`
- `8ba...935 ./contracts/libraries/TickMath.sol`
- `bfc...f04 ./contracts/libraries/PriceGrid.sol`
- `574...188 ./contracts/libraries/CurveCache.sol`
- `c52...980 ./contracts/libraries/TransferHelper.sol`
- `c02...04d ./contracts/libraries/CompoundMath.sol`
- `63f...47b ./contracts/libraries/SwapCurve.sol`
- `18c...61b ./contracts/libraries/CurveMath.sol`
- `ad1...b23 ./contracts/libraries/SafeCast.sol`
- `bdd...44a ./contracts/libraries/PoolSpecs.sol`
- `71e...85d ./contracts/libraries/CurveRoll.sol`
- `574...cc7 ./contracts/libraries/CurveAssimilate.sol`
- `104...fcc ./contracts/libraries/Chaining.sol`
- `628...42e ./contracts/libraries/TokenFlow.sol`
- `04a...edb ./contracts/mixins/PoolRegistry.sol`
- `b20...c31 ./contracts/mixins/LevelBook.sol`
- `136...05b ./contracts/mixins/StorageLayout.sol`
- `abe...c93 ./contracts/mixins/ProtocolAccount.sol`
- `15a...b25 ./contracts/mixins/KnockoutCounter.sol`
- `9fa...17e ./contracts/mixins/TickCensus.sol`
- `590...3d4 ./contracts/mixins/ProxyCaller.sol`
- `b9c...393 ./contracts/mixins/MarketSequencer.sol`
- `ed7...36b ./contracts/mixins/TradeMatcher.sol`
- `f08...b1e ./contracts/mixins/AgentMask.sol`
- `1b7...51d ./contracts/mixins/LiquidityCurve.sol`
- `892...299 ./contracts/mixins/PositionRegistrar.sol`
- `f0e...71e ./contracts/mixins/DepositDesk.sol`
- `a0d...81a ./contracts/mixins/SettleLayer.sol`

Tests

- `6dc...fa0 ./TestBitmaps.sol`
- `959...521 ./MockCond.sol`
- `38a...cba ./TestKnockoutCounter.sol`
- `c3a...94f ./TestAgentMask.sol`
- `85a...a7f ./MockPermit.sol`
- `d67...52c ./MockProxySidecar.sol`
- `7fd...dab ./TestTickCensus.sol`
- `032...112 ./TestEncoding.sol`
- `b7c...640 ./MockERC20.sol`
- `1f3...d57 ./TestCurveMath.sol`
- `d6b...98f ./TestLiquidityCurve.sol`

- `da7...bfa ./TestKnockoutLiq.sol`
- `c87...fcb ./TestProtocolAcct.sol`
- `0f5...a84 ./TestPositionRegistrar.sol`
- `df3...5ab ./MockTimelock.sol`
- `388...72e ./TestPriceGrid.sol`
- `e33...acc ./TestSettle.sol`
- `86b...b69 ./MockProxy.sol`
- `362...290 ./TestCompoundMath.sol`
- `8a4...b96 ./MockMinion.sol`
- `5a5...42a ./MockConduit.sol`
- `5ba...41f ./TestLevelBook.sol`
- `dcf...7ee ./QueryHelper.sol`
- `855...dbc ./CrocShell.sol`
- `437...1d9 ./TestTickMath.sol`
- `9c7...fdd ./TestLiquidityMath.sol`
- `cec...838 ./test/TestRoll.surplus.ts`
- `770...4e0 ./test/TestPool.permit.ts`
- `6ed...069 ./test/TestPool.conduit.ts`
- `f2d...4ae ./test/TestRoll.pairs.ts`
- `6cc...c9c ./test/TestKnockoutLiq.ts`
- `d96...6af ./test/TestPair.seq.ts`
- `280...643 ./test/TestPool.proxy.ts`
- `29c...bcf ./test/TestCompoundMath.ts`
- `4ca...3fd ./test/TestLiquidityMath.ts`
- `267...028 ./test/TestPool.lp.ts`
- `012...9c8 ./test/TestBitmaps.ts`
- `6f6...666 ./test/TestKnockoutCounter.ts`
- `a7e...9cf ./test/TestGas.eth.ts`
- `fbe...2d7 ./test/TestPool.harvest.ts`
- `ce1...715 ./test/TestCurveMath.ts`
- `aed...e28 ./test/TestSettle.ts`
- `971...f25 ./test/TestSwapCurve.ts`
- `935...5ac ./test/TestQuery.impact.ts`
- `4a3...2df ./test/TestGas.cold.ts`
- `641...011 ./test/FixedPoint.ts`
- `76f...195 ./test/TestPool.hotpath.ts`
- `f1c...abf ./test/TestRoll.pools.ts`
- `096...380 ./test/TestAgentMask.ts`
- `1b3...353 ./test/TestPool.sec.ts`
- `747...b45 ./test/TestPriceGrid.ts`
- `3fe...43f ./test/EncodeSimple.ts`
- `c6b...d38 ./test/TestPolicy.ts`
- `e4d...93f ./test/TestLevelBook.ts`
- `1f1...268 ./test/TestGas.knockout.ts`
- `712...b0e ./test/TestLpErc20.ts`
- `4d3...df2 ./test/TestPool.swap.ts`
- `296...552 ./test/TestLiqCurve.ts`
- `422...8e4 ./test/TestFixedMath.ts`
- `0e9...3a7 ./test/TestGas.ts`
- `16d...eb5 ./test/TestPair.ts`
- `0ea...447 ./test/TestRoll.ts`
- `ec0...6d1 ./test/TestPool.surplus.ts`
- `b65...998 ./test/TestProtocolAcct.ts`
- `4c7...94a ./test/TestPool.govern.ts`
- `d76...94e ./test/TestPool.deposit.ts`
- `f4e...90b ./test/TestPool.rebal.ts`
- `f63...c8d ./test/TestPool.knockout.ts`
- `056...c51 ./test/TestTickCensus.ts`
- `6fc...4d2 ./test/FacadePool.ts`
- `3fa...768 ./test/TestPool.eth.ts`
- `3a4...407 ./test/TestTickMath.ts`
- `2f8...fae ./test/TestPool.basic.ts`
- `c2e...202 ./test/TestInitPool.ts`
- `fc0...e64 ./test/TestPool.comp.ts`
- `d0c...a04 ./test/TestSettle.eth.ts`
- `933...106 ./test/TestGas.comp.ts`
- `e30...662 ./test/TestPool.grid.ts`
- `8cd...69e ./test/TestGas.proxy.ts`
- `6fe...f56 ./test/TestEncoding.ts`
- `581...e6c ./test/TestPool.jit.ts`
- `6b8...bb3 ./test/TestPositionRegistrar.ts`
- `2db...669 ./test/TestPool.agent.ts`
- `9a7...f18 ./test/EncodeOrder.ts`
- `43c...bdd ./test/TestTokenPrecision.ts`

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:
- Slither v0.8.3

Steps taken to run the tools:
1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Automated Analysis

Slither

1. The use of arbitrary `from` argument in `mixins/SettleLayer.sol#381`. After analyzing functions that make indirect and direct calls, we classified this issues as a false positive.
2. `delegatecall` to a input-controlled function id in:
   1. `mixins/ProxyCaller.sol#25-26`,
   2. `mixins/ProxyCaller.sol#34-35`,
   3. `mixins/ProxyCaller.sol#42-43`,
   4. `mixins/ProxyCaller.sol#72-81`,
   5. `mixins/ProxyCaller.sol#94-103`,
   6. `mixins/ProxyCaller.sol#117-126`,
   7. `mixins/ProxyCaller.sol#141-148`,
   8. `mixins/ProxyCaller.sol#162-166`, and
   9. `mixins/ProxyCaller.sol#189-193`. We classified the issues as false positives since the protocol validates the inputs or provides predefined inputs.
3. Uninitialized fields in `mixins/StorageLayout.sol`. We classified these issues as false positives since the fields are initialized externally.
4. Multiplication before division in:
   1. `PriceGrid.clipAbove()`, and
   2. `SwapCurve.calcFeeOverFlow()`. We classified these issues as false positives.
5. Strict equality in `libraries/KnockoutLiq.sol#224`. We classified it as a false positive.

Test Suite Results

There are some compilation issues when attempting to run the tests. In particular:

- `test/FacadePool.ts` - The import on `L16` is incorrect.
- `test/TestPool.proxy.ts` - The import on `L13` is incorrect. All tests passed after fixing the imports.

```
AgentMask
  ✓ pass relay cond
  ✓ pass relayer origin
  ✓ pass relayer origin
  ✓ fail relayer address
  ✓ fail deadline
  ✓ fail alive
  ✓ fail nonce early
  ✓ fail nonce late
  ✓ fail nonce salt
  ✓ repeat nonce cond (84ms)
  ✓ signature (46ms)

BitmapsLib
  ✓ truncateRight
  ✓ truncateLeft
  ✓ isBitSet
  ✓ indexPosLeft
  ✓ indexPosRight
  ✓ shiftPosLeft
  ✓ shiftPosRight
  ✓ spillPosLeft
  ✓ spillPosRight
  ✓ castIndex
  ✓ lobbyMezzTerm Decomp
  ✓ term shift

TestCompoundMath
  ✓ approx sqrt
  ✓ stack
  ✓ divide
  ✓ shrink
  ✓ price
  ✓ inflate
  ✓ deflate
  ✓ deflate precision

CurveMath
  ✓ active liquidity
  ✓ limit calc
  ✓ limit exhaust qty
  ✓ limit invert
  ✓ limit invert exhaust qty
  ✓ limit infinite
  ✓ limit inifinite invert
  ✓ vig flow
  ✓ vig flow sell
  ✓ vig flow quote denom
  ✓ vig flow limit
  ✓ vig protocol cut
  ✓ vig infinite max
  ✓ roll liq
  ✓ roll liq infinity
  ✓ assimilate liq
  ✓ assimilate liq denom
  ✓ assimilate zero fees
  ✓ assimilate zero liq
  ✓ assimilate one liq
  ✓ assimilate one liq denom
  ✓ assimilate one liq zero fees
  ✓ assimilate one liq denom zero fees
  ✓ assimilate at deflator bounds
  ✓ derive liq and price flow impact
  ✓ derive liq and price reserve rounding
  ✓ derive liq and price flow entire reserve

Encoding
  ✓ open settlement (45ms)
  ✓ hop settlement (42ms)
  ✓ hop improve (114ms)
  ✓ pool idx
  ✓ swap (41ms)
  ✓ ambient (76ms)
  ✓ concentrated
  ✓ chain flags (73ms)

TestFixedMath
  ✓ mulQ64
  ✓ mulQ64 Precision
  ✓ mulQ48
  ✓ mulQ48 Precision
  ✓ divQ64
  ✓ divQ64 Precision
  ✓ recipQ64
  ✓ recipQ64 size bounds

Gas Benchmarks Coldpath
  ✓ create pool [@gas-test]
  ✓ mint in virgin pool [@gas-test] (58ms)
  ✓ mint increase liq [@gas-test] (69ms)
  ✓ mint pre-init ticks [@gas-test] (67ms)
  ✓ mint one fresh init [@gas-test] (90ms)
  ✓ mint fresh ticks [@gas-test] (80ms)
  ✓ mint below price [@gas-test] (70ms)
  ✓ mint above price [@gas-test] (75ms)
  ✓ burn partial [@gas-test] (85ms)
  ✓ burn partial level left [@gas-test] (100ms)
  ✓ burn full [@gas-test] (73ms)
  ✓ burn full level left [@gas-test] (119ms)
  ✓ burn outside [@gas-test] (60ms)
  ✓ burn outside left [@gas-test] (107ms)
  ✓ burn liq rewards [@gas-test] (106ms)
  ✓ burn liq level left [@gas-test] (129ms)
  ✓ burn flipped [@gas-test] (163ms)
  ✓ burn flipped level left [@gas-test] (198ms)
  ✓ swap no pre-warm [@gas-test] (82ms)
  ✓ swap small [@gas-test] (101ms)
  ✓ swap tick w/o cross [@gas-test] (102ms)
  ✓ swap bitmap w/o cross [@gas-test] (104ms)
  ✓ swap cross tick [@gas-test] (157ms)
  ✓ swap cross two tick [@gas-test] (191ms)
  ✓ swap cross two tick and bitmap [@gas-test] (198ms)
  ✓ swap cross bitmap between two tick  [@gas-test] (181ms)
  ✓ swap cross many ticks [@gas-test] (359ms)
  ✓ swap cross many bitmap [@gas-test] (108ms)
  ✓ swap surplus [@gas-test] (122ms)
  ✓ mint surplus [@gas-test] (112ms)
  ✓ burn surplus [@gas-test] (149ms)

Gas Benchmarks Native Eth
  ✓ mint in virgin pool [@gas-test] (46ms)
  ✓ mint increase liq [@gas-test] (45ms)
  ✓ mint pre-init ticks [@gas-test] (48ms)
  ✓ mint one fresh init [@gas-test] (55ms)
  ✓ mint fresh ticks [@gas-test] (47ms)
  ✓ mint below price [@gas-test] (43ms)
  ✓ mint above price [@gas-test] (45ms)
  ✓ burn partial [@gas-test] (44ms)
  ✓ burn partial level left [@gas-test] (76ms)
  ✓ burn full [@gas-test] (44ms)
  ✓ burn full level left [@gas-test] (67ms)
  ✓ burn outside [@gas-test] (46ms)
  ✓ burn outside left [@gas-test] (60ms)
  ✓ burn liq rewards [@gas-test] (62ms)
  ✓ burn liq level left [@gas-test] (83ms)
  ✓ burn flipped [@gas-test] (115ms)
```

```
    ✓ burn flipped level left [@gas-test] (130ms)
    ✓ harvest fees [@gas-test] (165ms)
    ✓ swap no pre-warm [@gas-test] (47ms)
    ✓ swap small [@gas-test] (69ms)
    ✓ swap small [@gas-test] (69ms)
    ✓ swap small sell [@gas-test] (77ms)
    ✓ swap tick w/o cross [@gas-test] (66ms)
    ✓ swap bitmap w/o cross [@gas-test] (68ms)
    ✓ swap cross tick [@gas-test] (107ms)
    ✓ swap cross two tick [@gas-test] (118ms)
    ✓ swap cross two tick and bitmap [@gas-test] (138ms)
    ✓ swap cross bitmap between two tick  [@gas-test] (139ms)
    ✓ swap cross many ticks [@gas-test] (270ms)
    ✓ swap cross many bitmap [@gas-test] (79ms)

  Gas Benchmarks Knockout
    ✓ mint knockout (83ms)
    ✓ mint knockout pre-init pivot (86ms)
    ✓ swap cross full knockout [@gas-test] (101ms)
    ✓ swap cross end of knockout [@gas-test] (121ms)

  Gas Benchmarks Proxy Sidecars
    ✓ swap proxy unused [@gas-test] (58ms)
    ✓ swap proxy optimal - forced [@gas-test] (88ms)

  Gas Benchmarks
    ✓ mint in virgin pool [@gas-test]
    ✓ mint increase liq [@gas-test] (49ms)
    ✓ mint pre-init ticks [@gas-test] (44ms)
    ✓ mint one fresh init [@gas-test] (49ms)
    ✓ mint fresh ticks [@gas-test] (51ms)
    ✓ mint below price [@gas-test] (43ms)
    ✓ mint above price [@gas-test] (41ms)
    ✓ burn partial [@gas-test] (43ms)
    ✓ burn partial level left [@gas-test] (63ms)
    ✓ burn full [@gas-test] (56ms)
    ✓ burn full level left [@gas-test] (68ms)
    ✓ burn outside [@gas-test] (42ms)
    ✓ burn outside left [@gas-test] (60ms)
    ✓ burn liq rewards [@gas-test] (65ms)
    ✓ burn liq level left [@gas-test] (79ms)
    ✓ burn flipped [@gas-test] (115ms)
    ✓ burn flipped level left [@gas-test] (133ms)
    ✓ harvest fees [@gas-test] (727ms)
    ✓ swap no pre-warm [@gas-test] (48ms)
    ✓ swap small [@gas-test] (67ms)
    ✓ swap tick w/o cross [@gas-test] (77ms)
    ✓ swap bitmap w/o cross [@gas-test] (83ms)
    ✓ swap cross tick [@gas-test] (110ms)
    ✓ swap cross two tick [@gas-test] (142ms)
    ✓ swap cross two tick and bitmap [@gas-test] (131ms)
    ✓ swap cross bitmap between two tick  [@gas-test] (129ms)
    ✓ swap cross many ticks [@gas-test] (284ms)
    ✓ swap cross many bitmap [@gas-test] (88ms)
    ✓ swap surplus [@gas-test] (105ms)
    ✓ mint surplus [@gas-test] (90ms)
    ✓ burn surplus [@gas-test] (103ms)

  Initialize Pool
    ✓ init token pool (49ms)
    ✓ init ether pool (52ms)

  Knockout Counter Mixin
    ✓ mint position (44ms)
    ✓ mint ask position
    ✓ mint add pos
    ✓ mint pivot stack
    ✓ mint pivot arches
    ✓ merkle slot pre-warmed
    ✓ burn partial
    ✓ burn full
    ✓ burn over qty (53ms)
    ✓ burn ask position
    ✓ burn bid/ask criss-cross (38ms)
    ✓ burn add pos (42ms)
    ✓ burn pivot stack (49ms)
    ✓ cross position
    ✓ cross position sell
    ✓ cross multiple (78ms)
    ✓ claim position (42ms)
    ✓ claim multiple pos (59ms)
    ✓ claim stack (151ms)
    ✓ claim before knockout
    ✓ burn after knockout (38ms)
    ✓ bad claim proofs (162ms)
    ✓ recover (117ms)
    ✓ bad recovers (126ms)

  Knockout Liquidity
    ✓ encode pivot
    ✓ encode pos (55ms)
    ✓ proof no steps
    ✓ proof no steps history
    ✓ proof merkle one step
    ✓ proof merkle multi steps
    ✓ proof merkle middle step
    ✓ assert width
    ✓ assert grid
    ✓ assert disabled
    ✓ assert outside
    ✓ assert inside

  LevelBook
    ✓ empty init
    ✓ add fresh liq
    ✓ stack liq
    ✓ add above
    ✓ add below
    ✓ remove partial
    ✓ remove full
    ✓ remove over
    ✓ bookmark ticks (44ms)
    ✓ forget ticks (104ms)
    ✓ cross level liq (66ms)
    ✓ cross non level (64ms)
    ✓ odometer add
    ✓ odometer remove partial
    ✓ odometer remove full
    ✓ odometer add/rmove sequence (50ms)
    ✓ above re-clock
    ✓ odometer boundary
    ✓ odometer zero init
    ✓ below re-clock
    ✓ cross fee (68ms)
    ✓ cross up (47ms)
    ✓ cross sequence (85ms)

  LiquidityCurve
    ✓ liquidity receive ambient
    ✓ liquidity pay ambient
    ✓ liquidity receive concentrated
    ✓ liquidity pay concentrated
    ✓ liquidity oversized (68ms)
    ✓ multiple pools (55ms)
    ✓ liquidity below range (38ms)
    ✓ liquidity above range (39ms)
    ✓ liquidity below range (38ms)
    ✓ liquidity on lower bump (57ms)
    ✓ liquidity on lower bump wei
    ✓ liquidity on upper bump
    ✓ liquidity on upper bump wei (38ms)
    ✓ liquidity inside tick (46ms)
    ✓ liquidity rewards

  LiquidityMath
    ✓ add
    ✓ add signed
    ✓ minus

  Pool Conduit
    ✓ mint and burn ambient (52ms)
    ✓ transfer LP token (49ms)
    ✓ no accept concentrated LP (50ms)
    ✓ wrong pool token (291ms)
    ✓ wrong pool index (38ms)

  Sequence Pair
    ✓ two pair sequence (161ms)
    ✓ surplus settle (157ms)
    ✓ surplus partial (181ms)
    ✓ quote entry (140ms)
    ✓ settle mid (143ms)
    ✓ three pair sequence (672ms)

  Sequence Triangle
    ✓ triangle sequence (204ms)
    ✓ triangle arb (273ms)

  Pair
    ✓ two pool arbitrage (161ms)
    ✓ two pool arbitrage quote (158ms)
    ✓ three pools stacked flow (262ms)
    ✓ protocol fee baseline (226ms)
    ✓ protocol fee stack both sides (306ms)
    ✓ protocol fee stack base (292ms)
    ✓ protocol fee stack quote (301ms)
    ✓ pool settings individual (190ms)

  CrocPolicy
    ✓ constructor addresses
    ✓ transfer authority (53ms)
    ✓ authority for transfer authority (66ms)
    ✓ transfer not accepted (91ms)
    ✓ ops resolution
    ✓ ops resolution from treasury
    ✓ ops resolution from emergency
    ✓ ops resolution unauthorized
    ✓ treasury resolution
    ✓ treasury resolution unauthorized
    ✓ emergency halt
    ✓ emergency unauthorized
    ✓ policy invoke
    ✓ policy invoke flag pos
    ✓ policy non conduit
    ✓ policy flag off
    ✓ expired policy
    ✓ set policy unauthorized
    ✓ policy weaken
    ✓ expiry offset
    ✓ mandate weaken
    ✓ force weaken flags
    ✓ force weaken mandate
    ✓ force weaken unauthorized
    ✓ emergency policy
    ✓ emergency policy authorized

  Pool Router Agent
    ✓ router agent (39ms)
    ✓ router approve array
    ✓ router no cold path
    ✓ router not approved
    ✓ router unnapproved party
    ✓ router unapproved callpath
    ✓ router nonces (336ms)
    ✓ router nonces reset (675ms)

  Pool Relayer Agent
    ✓ relay call (71ms)
    ✓ nonce no repeat (58ms)
    ✓ nonce sequence (57ms)
    ✓ relayer address (41ms)
    ✓ unauthorized relayer
    ✓ deadline
    ✓ live time condition
    ✓ nonce reset (51ms)
    ✓ nonce reset wrong (41ms)
    ✓ nonce reset cond (58ms)
    ✓ nonce reset cond mock args (45ms)
    ✓ reset cond reject (40ms)
    ✓ reset cond bad oracle (41ms)
    ✓ relayer tip (42ms)
    ✓ tip sender (41ms)
    ✓ tip origin (41ms)
    ✓ tip protocol take (56ms)
    ✓ protocol take rate valid

  Pool
    ✓ mint collection (116ms)
    ✓ mint liquidity (144ms)
    ✓ swap simple (84ms)
    ✓ swap protocol fee (104ms)
    ✓ swap sell (78ms)
    ✓ swap sell protocol fee (107ms)
    ✓ swap wrong direction (95ms)
    ✓ swap buy quote output (83ms)
    ✓ swap sell quote output (101ms)
    ✓ swap buy quote proto fee (103ms)
    ✓ swap limit (83ms)
    ✓ swap tick step (181ms)
    ✓ swap tick sell (187ms)
    ✓ swap tick protocol fee (194ms)
    ✓ init protocol fee (386ms)
    ✓ burn payout full (76ms)
    ✓ burn payout sum full (133ms)
    ✓ burn payout tranche (182ms)
    ✓ burn liquidity (246ms)
    ✓ burn payout rewards (293ms)
    ✓ mint blends rewards (313ms)
    ✓ mint ambient (61ms)
    ✓ burn ambient (85ms)
    ✓ mint ambient seed inflator (238ms)
    ✓ burn ambient growth deflator (192ms)
    ✓ burn ambient post growth deflator (234ms)
    ✓ burn ambient over provision (200ms)

  Pool Compound
    ✓ swap->mint (115ms)
    ✓ swap defer (128ms)
    ✓ swap->burn  (119ms)
    ✓ mint concentrated (117ms)
    ✓ multiple range orders (167ms)

  Pool Compound Curve Cache
    ✓ swap curve cache (225ms)

  Pool Conduit
    ✓ mint ambient
    ✓ burn ambient (49ms)
    ✓ mint ambient deflator (101ms)
    ✓ mint concentrated
    ✓ burn concentrated (50ms)
    ✓ mint concentrated deflator (103ms)
    ✓ mint reject
    ✓ burn reject (78ms)

  Pool Surplus Deposits
    ✓ deposit
    ✓ deposit native (58ms)
    ✓ deposit native insufficient value
    ✓ deposit permit
    ✓ disburse
    ✓ disburse native (56ms)
    ✓ disburse full
    ✓ disburse full infer
    ✓ disburse over-size
    ✓ disburse all but
    ✓ transfer
    ✓ transfer full
    ✓ transfer all but
    ✓ transfer full infer
    ✓ transfer over
    ✓ side pocket
    ✓ side partial
    ✓ side zero full
    ✓ side all but
    ✓ side pocket protects capital

  Pool Ethereum
    ✓ mint
    ✓ balance client side
    ✓ burn (62ms)
    ✓ mint ambient
    ✓ burn ambient (47ms)
    ✓ swap protocol fee (134ms)

  Pool Ethereum Hotpath
    ✓ mint
    ✓ balance client side
    ✓ burn (46ms)
    ✓ mint ambient
    ✓ burn ambient
    ✓ swap protocol fee (81ms)

  Pool Governance
    ✓ transfer authority (89ms)
    ✓ set treasury (164ms)
    ✓ collect treasury (209ms)
    ✓ collect treasury time delay (155ms)
    ✓ safe mode (173ms)
    ✓ init liq valid bounds
    ✓ take rate (46ms)

  Pool Grid
    ✓ grid required (167ms)
    ✓ grid required hotpath (148ms)
    ✓ grid revised (66ms)
    ✓ burn after revised (130ms)
    ✓ burn after revised hotpath (103ms)
    ✓ price improve - no settings (69ms)
    ✓ price improve - settings (71ms)
    ✓ price improve burn full (77ms)
    ✓ price improve burn partial (43ms)
    ✓ price improve burn hot path (88ms)
    ✓ price improve - quote side (64ms)
    ✓ price improve - away (78ms)
    ✓ price improve - wrong base side
    ✓ price improve - wrong quote side

  Pool Harvest
    ✓ harvest rewards (85ms)
    ✓ harvest deplete (114ms)
    ✓ burn deplete (133ms)
    ✓ harvest re-fill (140ms)

  Pool HotPath
    ✓ mint collection (79ms)
    ✓ mint liquidity (107ms)
    ✓ swap simple (70ms)
    ✓ swap protocol fee (87ms)
    ✓ swap sell (58ms)
    ✓ swap sell protocol fee (86ms)
    ✓ swap wrong direction (53ms)
    ✓ swap buy quote output (64ms)
    ✓ swap sell quote output (69ms)
    ✓ swap buy quote proto fee (83ms)
    ✓ swap limit (64ms)
    ✓ swap tick step (143ms)
    ✓ swap tick sell (132ms)
    ✓ swap tick protocol fee (160ms)
    ✓ burn payout full (68ms)
    ✓ burn payout sum full (94ms)
    ✓ burn payout tranche (132ms)
    ✓ burn liquidity (159ms)
    ✓ burn payout rewards (195ms)
    ✓ mint blends rewards (229ms)
    ✓ mint ambient (43ms)
    ✓ burn ambient (57ms)
    ✓ mint ambient seed inflator (195ms)
    ✓ burn ambient growth deflator (143ms)
    ✓ burn ambient post growth deflator (189ms)
    ✓ burn ambient over provision (166ms)

  Pool JIT
    ✓ jit window (74ms)
    ✓ mint in window (97ms)
    ✓ jit window too large (61ms)

  Pool Knockout Liq
    ✓ mint flow (48ms)
    ✓ mint flow ask (54ms)
    ✓ mint off-grid (49ms)
    ✓ mint bad width (84ms)
    ✓ mint inside mid (76ms)
    ✓ mint bad inside mid (38ms)
    ✓ burn partial (68ms)
    ✓ burn full liq (69ms)
    ✓ swap into active range (210ms)
    ✓ swap into active range ask (206ms)
    ✓ swap knockout (205ms)
    ✓ swap knockout ask (215ms)
    ✓ claim knockout (223ms)
    ✓ claim knockout ask (228ms)
    ✓ claim knockout proof (1300ms)
    ✓ bad proof (555ms)
    ✓ recover knockout (214ms)
    ✓ claim knockout twice (254ms)
    ✓ recover knockout twice (241ms)
    ✓ knockout no repeat (342ms)

  Pool Knockout Liq Native Eth
    ✓ swap knockout (201ms)

  Pool Warm LP Path
    ✓ mint ambient base
    ✓ mint ambient quote
    ✓ burn ambient base
    ✓ burn ambient quote (38ms)
    ✓ mint conc base
    ✓ mint conc qutoe
    ✓ burn conc base (48ms)
    ✓ burn conc quoe (55ms)
    ✓ out of range base
    ✓ out of range quote
```

```
permissioned pool
  ✓ permit oracle (107ms)
  ✓ mint/burn concentrated (82ms)
  ✓ mint/burn ambient (53ms)
  ✓ compound directive (107ms)

Pool Proxy Paths
  ✓ swap no proxy (41ms)
  ✓ swap proxy (89ms)
  ✓ swap proxy optional (75ms)
  ✓ swap force proxy (50ms)
  ✓ swap long path okay (90ms)
  ✓ cannot upgrade boot path
  ✓ upgrade requires contract address (39ms)
  ✓ requires proxy contract accept (54ms)

Pool Rebalance
  ✓ rebalance range (140ms)
  ✓ rebalance gas (98ms)
  ✓ rebalance liq (120ms)
  ✓ rebalance liq gas [@gas-test] (89ms)

Pool Security
  ✓ double initialize (112ms)
  ✓ template disabled
  ✓ pre-initialize (83ms)
  ✓ mint outside tick range (133ms)
  ✓ over burn (379ms)
  ✓ burn steal (372ms)

Pool Surplus
  ✓ balance and withdraw
  ✓ debit entry (39ms)
  ✓ debit partial entry (44ms)
  ✓ credit entry (65ms)
  ✓ debit exit (38ms)
  ✓ debit partial exit (50ms)
  ✓ credit exit (79ms)
  ✓ swap hotpath (47ms)
  ✓ mint hotpath (52ms)
  ✓ burn hotpath (77ms)
  ✓ mint ambient hotpath (56ms)
  ✓ burn ambient hotpath (66ms)
  ✓ swap base settle (53ms)
  ✓ swap quote settle (47ms)
  ✓ mint base settle (44ms)
  ✓ mint quote settle (55ms)

Pool Surplus Ether
  ✓ balance and withdraw (59ms)

PositionRegistrar
  ✓ empty init
  ✓ add liq
  ✓ add stack
  ✓ add multi pos (39ms)
  ✓ burn partial
  ✓ burn full
  ✓ burn position only
  ✓ burn rewards

Price Improve
  ✓ ticks out of order
  ✓ non-improvable
  ✓ clip inside positive
  ✓ clip inside negative
  ✓ clip inside over zero
  ✓ clip below
  ✓ clip above
  ✓ unit tick
  ✓ scale thresh
  ✓ grid pin ask wings
  ✓ grid pin bid wings
  ✓ grid pin wings both sides
  ✓ on grid
  ✓ verify (43ms)

Protocol Account
  ✓ accum (75ms)
  ✓ disburse
  ✓ ethereum token
  ✓ disburse post

Query Impact
  ✓ small buy (61ms)
  ✓ small sell (62ms)
  ✓ buy denom (57ms)
  ✓ sell denom (63ms)
  ✓ large buy (310ms)
  ✓ large sell (346ms)
  ✓ bump ticks (352ms)
  ✓ multiple bump ticks (425ms)

Roll between pairs
  ✓ two pairs (176ms)
  ✓ mint -> swap (128ms)
  ✓ quote entry (132ms)
  ✓ three pair sequence (155ms)

Pair roll triangle
  ✓ triangle arb (264ms)
  ✓ roll surplus (201ms)
  ✓ roll surplus diff sides (201ms)

Roll Pools
  ✓ roll between pools (184ms)
  ✓ roll on exit token (187ms)
  ✓ different sides pools (175ms)

Roll Surplus
  ✓ roll surplus (57ms)
  ✓ roll stacked (71ms)
  ✓ surplus exit (58ms)
  ✓ surplus entry+exit (65ms)

Rolling Back Fill
  ✓ swap->mint ambient (77ms)
  ✓ ambient seed deflator (232ms)
  ✓ exit at base (72ms)
  ✓ swap quote->mint ambient (68ms)
  ✓ entry at quote (76ms)
  ✓ swap->burn ambient (67ms)
  ✓ mint ambient -> swap (69ms)
  ✓ burn ambient -> swap (72ms)
  ✓ swap roll flip direction (83ms)
  ✓ swap roll flip direction reverse (68ms)
  ✓ swap->mint range (83ms)
  ✓ swap->burn range (88ms)
  ✓ quote -> mint range (75ms)
  ✓ swap->mint below range (86ms)
  ✓ quote -> mint below range (78ms)
  ✓ swap -> mint wrong side (90ms)
  ✓ reposition range (86ms)

Settle Layer Ethereum
  ✓ settle debit (76ms)
  ✓ settle credit (66ms)
  ✓ settle flat (66ms)
  ✓ settle final on non-ethereum token (78ms)
  ✓ settle debit surplus (91ms)
```

```
    ✓ settle debit surplus partial (98ms)
    ✓ settle credit (75ms)
    ✓ settle refund (85ms)

  Settle Layer
    ✓ debit
    ✓ credit
    ✓ debit shortfall
    ✓ credit shortfall
    ✓ zero
    ✓ debit ether
    ✓ credit ether
    ✓ debit ether shortfall
    ✓ debit ether overpay
    ✓ debit ether overpay surplus
    ✓ debit ether double spend
    ✓ credit ether shortfall
    ✓ credit surplus
    ✓ debit surplus
    ✓ limit qty credit
    ✓ limit qty debit
    ✓ limit sign (46ms)
    ✓ dust
    ✓ no dust on debit

  Swap Curve
    ✓ swap full qty (181ms)
    ✓ swap fee full qty
    ✓ swap fee+proto full qty
    ✓ swap paid cumulative
    ✓ swap sell
    ✓ swap sell fee
    ✓ swap quote denom
    ✓ swap quote denom fees
    ✓ swap quote sell
    ✓ swap quote sell fee
    ✓ swap bump price
    ✓ swap bump sell
    ✓ swap bump denom
    ✓ swap limit price
    ✓ swap limit fee
    ✓ swap limit sell
    ✓ swap bump infinity
    ✓ swap bump sell infinity
    ✓ swap infinity
    ✓ swap infinity sell
    ✓ swap infinity quote
    ✓ swap infinity quote sell
    ✓ swap zero liq base buy
    ✓ swap zero liq quote buy
    ✓ swap zero liq base sell
    ✓ swap zero liq quote buy

  TickCensus
    ✓ empty bitmap
    ✓ bookmark tick
    ✓ bookmark repeat
    ✓ bookmark two shared
    ✓ bookmark multiple across
    ✓ forget reset
    ✓ forget repeat
    ✓ forget shared
    ✓ forget multiple across
    ✓ pin buy
    ✓ pin sell
    ✓ pin sell at
    ✓ pin edge
    ✓ pin edge barrier
    ✓ pin buy spill
    ✓ pin sell spill
    ✓ pin buy zero point
    ✓ pin sell zero point
    ✓ seek empty (95ms)
    ✓ seek over cliff (115ms)
    ✓ seek terminus neighbor
    ✓ seek immediate neighbor (40ms)
    ✓ seek through mezz
    ✓ seek immediate mezz
    ✓ seek mezz caged
    ✓ seek mezz inner caged (57ms)
    ✓ seek through lobby (42ms)
    ✓ seek lobby lookback
    ✓ seek lobby lookback reverse
    ✓ seek extreme ticks (191ms)
    ✓ seek extreme ticks bookmark in middle (103ms)
    ✓ seek extreme ticks terminus bookmark (113ms)

  Tick Math
    ✓ tick to ratio
    ✓ ratio to tick
    ✓ min tick
    ✓ max tick
    ✓ outside bounds

  Token Precision
    ✓ base token medium low liquidity
    ✓ base token low liquidity
    ✓ base token very low liquidity
    ✓ base token medium liquidity
    ✓ base token medium high liquidity
    ✓ base token high liquidity
    ✓ base token very high liquidity
    ✓ quote token medium low liquidity
    ✓ quote token low liquidity
    ✓ quote token very low liquidity
    ✓ quote token medium liquidity
    ✓ quote token medium high liquidity
    ✓ quote token high liquidity
    ✓ quote token very high liquidity


  690 passing (3m)
```

Code Coverage

The test coverage and the tests themselves are good. However, they mostly do not cover edge cases or other scenarios that could be problematic. Consider adding additional test cases to validate proper operation when handling those edge cases.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
CrocEvents.sol	100	100	100	100	
CrocSwapDex.sol	100	100	100	100	
contracts/callpaths/	94.6	89.83	86.3	94.3	
BootPath.sol	80	83.33	50	80	43,48,75
ColdPath.sol	94.21	92.86	92.59	94.06	... 313,315,336
HotPath.sol	96.3	50	100	96.3	94

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
KnockoutPath.sol	93.02	83.33	77.78	92.5	46,99,189
LongPath.sol	97.14	100	75	97.14	111
MicroPaths.sol	97.56	100	83.33	97.56	208
SafeModePath.sol	66.67	100	66.67	66.67	25
WarmPath.sol	97.01	96.43	92.31	96.3	111,298
contracts/governance/	100	100	100	100	
CrocPolicy.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
ICrocCondOracle.sol	100	100	100	100	
ICrocLpConduit.sol	100	100	100	100	
ICrocMinion.sol	100	100	100	100	
ICrocPermitOracle.sol	100	100	100	100	
IERC20Minimal.sol	100	100	100	100	
contracts/libraries/	98.77	90.54	99.42	99.54	
BitMath.sol	100	100	100	100	
Bitmaps.sol	100	100	100	100	
Chaining.sol	95.52	88.24	100	98.28	431
CompoundMath.sol	100	75	100	100	
CurveAssimilate.sol	100	100	100	100	
CurveCache.sol	100	100	100	100	
CurveMath.sol	100	100	100	100	
CurveRoll.sol	98.36	92.31	100	100	
Directives.sol	100	100	100	100	
Encoding.sol	100	50	100	100	
FixedPoint.sol	100	100	100	100	
KnockoutLiq.sol	100	87.5	100	100	
LiquidityMath.sol	96	90	100	100	
PoolSpecs.sol	100	50	100	100	
PriceGrid.sol	100	96.67	100	100	
ProtocolCmd.sol	100	100	100	100	
SafeCast.sol	100	100	83.33	83.33	11
SwapCurve.sol	100	66.67	100	100	
TickMath.sol	100	100	100	100	
TokenFlow.sol	100	100	100	100	
TransferHelper.sol	100	100	100	100	
contracts/mixins/	98.31	88.08	98.64	98.56	
AgentMask.sol	93.9	77.27	92.31	95.29	249,251,311,363
DepositDesk.sol	100	100	100	100	
KnockoutCounter.sol	100	93.33	100	100	
LevelBook.sol	100	100	100	100	
LiquidityCurve.sol	100	80.77	100	100	
MarketSequencer.sol	90.91	80	100	92.59	315,316,317,318
PoolRegistry.sol	100	84.78	100	100	
PositionRegistrar.sol	100	100	100	100	
ProtocolAccount.sol	100	83.33	100	100	
ProxyCaller.sol	91.89	81.82	90.91	91.89	41,42,44
SettleLayer.sol	100	95.24	100	100	
StorageLayout.sol	100	100	100	100	
TickCensus.sol	100	100	100	100	
TradeMatcher.sol	100	95	100	100	
contracts/periphery/	100	87.5	100	100	
CrocLpErc20.sol	100	87.5	100	100	
contracts/vendor/compound/	0	0	0	0	
Timelock.sol	0	0	0	0	... 103,105,110
All files	95.28	84.78	95.45	95.62	

Changelog

- 2023-04-25 - Initial report
- 2023-05-30 - Fixes review based on commit `511476`.

About Quantstamp

Timeliness of content

Notice of confidentiality

Links to other websites

Disclaimer