HALBORN

BubbleSwap - Concentrated Liquidity Pool AMM

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **February 27th, 2023 - March 31st, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	02/24/2023	Luis Buendia
0.2	Document Edit	02/24/2023	Luis Buendia
0.3	Document Edit	03/28/2023	Luis Buendia
0.4	Draft Review	04/03/2023	Gokberk Gulgun
0.5	Draft Review	04/03/2023	Gabi Urrutia
1.0	Remediation Plan	04/05/2023	Luis Buendia
1.1	Remediation Plan Review	04/05/2023	Gokberk Gulgun
1.2	Remediation Plan Review	04/05/2023	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com
Luis Buendia	Halborn	Luis.Buendia@halborn.com
Alejandro Taibo	Halborn	Alejandro.Taibo@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

BubbleSwap engaged Halborn to conduct a security audit on their smart contracts beginning on February 27th, 2023 and ending on March 31st, 2023. The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided four weeks for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified some security risks that were accepted and addressed by the BubbleSwap team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the contracts' solidity code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Manual testing with custom scripts. (Foundry).
- Static Analysis of security for scoped contract, and imported functions manually.
- Testnet deployment (Anvil).

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

1. IN-SCOPE TREE & COMMIT :

The security assessment was scoped to the following smart contracts on the halborn-audit branch.

Commit ID: 64ea300adef72a705a00c3e1e2f452eea44c1ef6

During the audit, there were other several commits with corrections and other considerations that were fund during the audit.

The main commits included during the audit time are:

- 5c6c0315490e9b63e32b2dd21fb13ae358ba6641
- 8e4d926575469cce51d8dd87398e8fc7be8b6ca3
- 32c81a6e7356a6c21a630295dfdbad2811fa4839
- 07c873d250d3e5ffed5b8c0df7baaea2f72ee13d
- 8ab288b849d763b28a9782c1cb8a9f12f1f90d15

2. REMEDIATION PR/COMMITS:

- 07c873d250d3e5ffed5b8c0df7baaea2f72ee13d
- 38755a333ae2eb46b6bf0ee94e659752a7356097
- a520bb7cf8162171c8d3fb18d17d10c07edb2b80

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	2	2	5

LIKELIHOOD

IMPACT

			(HAL-01)	
		(HAL-02) (HAL-03)		
	(HAL-04) (HAL-05)			
(HAL-06) (HAL-07) (HAL-08) (HAL-09) (HAL-10)				

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - ARITHMETIC ERROR CAN RESULT IN LOCKED USER FUNDS	High	SOLVED - 03/23/2023
HAL02 - IMPROPER SWAP AMOUNTS HANDLING	Medium	SOLVED - 03/23/2023
HAL03 - UNCONTROLLED FEEGROWTH ERROR WHEN ADDING LIQUIDITY	Medium	SOLVED - 03/23/2023
HAL04 - POSSIBILITY TO CREATE AN INCENTIVE FOR A DELETED POOL	Low	RISK ACCEPTED
HAL05 - POOL SHOULD RETURN FUNDS WHEN ADDING EXTRA LIQUIDITY VIA DIRECT MINT	Low	RISK ACCEPTED
HAL06 - POOL DOES NOT SUPPORT FRACTIONAL FEE TOKEN TRANSFERS	Informational	ACKNOWLEDGED
HAL07 - PAYER FIELD NEVER USED ON MINTPARAMS STRUCTS	Informational	ACKNOWLEDGED
HAL08 - SQRTPRICELIMITX96 IS NOT USED ON V3SWAPROUTER	Informational	ACKNOWLEDGED
HAL09 - CREATE CUSTOM ERRORS TO SAVE GAS	Informational	ACKNOWLEDGED
HAL10 - INCOMPLETE NATSPEC DOCUMENTATION	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) ARITHMETIC ERROR CAN RESULT IN LOCKED USER FUNDS - HIGH

Description:

The collect function of the ContentratedLiquidityPoolManager.sol contract does not withdraw the correct amount from the vaults. By withdrawing an improper amount, when a user under certain conditions tries to burn part or all of his position the contract reverts due to an arithmetic error in the withdraw function of the ConcetratedLiquidityPool.sol contract.

Code Location:

Location - ConcentratedLiquidityPoolManager.sol#L210

Listing 1: ConcentratedLiquidityPoolManager.sol (Line 210)

```
192 function collect(int64 tokenId) public returns (uint256
 ↳ token0amount, uint256 token1amount) {
193     require(msg.sender == ownerOf(uint256(int256(tokenId))), "
 ↳ NOT_ID_OWNER");
194     Position storage position = positions[tokenId];
195
196     PositionFeesData memory positionFeesData = positionFees(
 ↳ tokenId);
197
198     token0amount = positionFeesData.token0amount + position.
 ↳ unclaimedFees0;
199     token1amount = positionFeesData.token1amount + position.
 ↳ unclaimedFees1;
200
201     position.unclaimedFees0 = 0;
202     position.unclaimedFees1 = 0;
203
204     position.feeGrowthInside0 = positionFeesData.feeGrowthInside0;
205     position.feeGrowthInside1 = positionFeesData.feeGrowthInside1;
206
207     uint256 balance0 = position.pool.vault0(address(this));
208     uint256 balance1 = position.pool.vault1(address(this));
209
```

```
210     if (balance0 < token0amount || balance1 < token1amount) {
211         (uint256 amount0fees, uint256 amount1fees) = position.pool
 ↳ .collect(position.lower, position.upper);
212 
213         uint256 newBalance0 = amount0fees + balance0;
214         uint256 newBalance1 = amount1fees + balance1;
215 
216         // Rounding errors due to frequent claiming of other users
 ↳  in the same position may cost us some wei units.
217         if (token0amount > newBalance0) token0amount = newBalance0
 ↳ ;
218         if (token1amount > newBalance1) token1amount = newBalance1
 ↳ ;
219     }
220 
221     _transferBoth(position.pool, msg.sender, token0amount,
 ↳ token1amount);
222 
223     emit Collect(uint256(int256(tokenId)), msg.sender,
 ↳ token0amount, token1amount);
224 }
```

Risk Level:

Likelihood - 4
Impact - 4

Proof Of Concept:

The next test found through fuzzing triggered the previously described behavior.

Location - Invariant.t.sol#L379

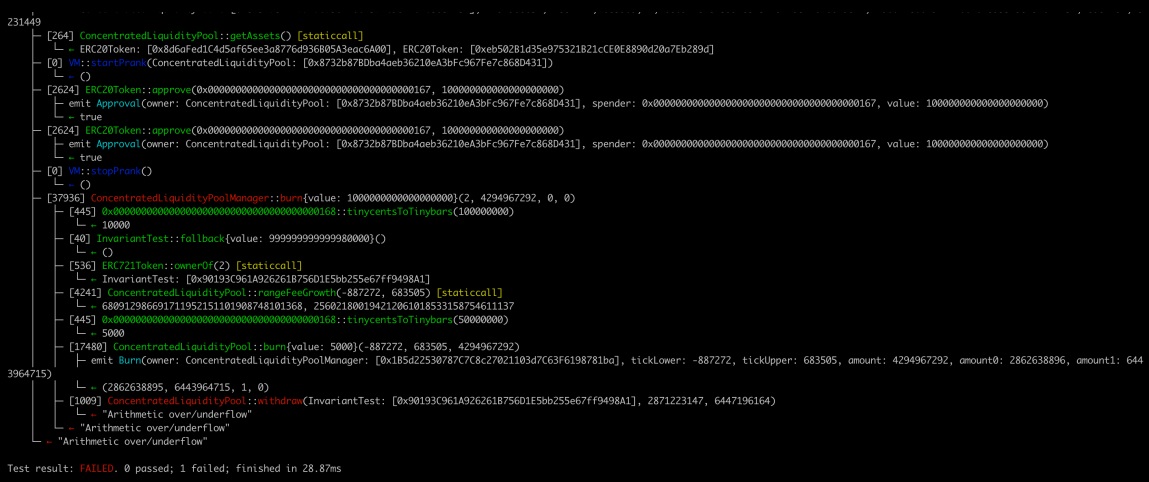
Listing 2

```
1 function testWithdrawRevert() public {
2     setupEnv();
3     State memory state;
4 
```

```
5     state.pool = Pool( 0, false );
6     state.mints = [Mint(0, 0, 4294965963, 4294965963, 0,
↳ 14615016108150356721930896253681831785095923749 88), Mint(0, 0, 0,
↳ 0, 0, 10238289572590543442296 4234), Mint(0, 0, 0, 0, 4727003,
↳ 4727003), Mint(0, 0, 5, 5,
↳ 14615016108150356721930896253681831785095923749 88, 0), Mint(0, 0,
↳ 0, 0, 57866823689572764762770813391297409132144668,
↳ 50066467198429302992085375296889841655755873), Mint(0, 0, 0, 0, 0,
↳  0), Mint(0, 0, 0, 0,
↳ 14610083571463544741599688418218697707556320070 64,
↳ 79347087983666005040985389675), Mint(0, 0, 0, 0,
↳ 10253657761813300067737 6439, 0), Mint(0, 0, 0, 0,
↳ 17081729108157914270742 59, 0), Mint(0, 0, 0, 0, 0, 0), Mint(0, 0,
↳ 0, 0, 14615016108150356721930896253681831785095923749 88, 0), Mint
↳ (0, 0, 0, 0, 14610083571463544741599688418218697707556320070 64,
↳ 19861840684872087267822641043486), Mint(0, 0, 0, 0, 0, 1718308),
↳ Mint(0, 0, 0, 0, 0, 0), Mint(0, 0, 0, 0, 0,
↳ 50257238252836758370807292880), Mint(0, 0, 0, 0,
↳ 54907329825568905816573164194356135163533386, 4727003), Mint(0, 0,
↳  0, 0, 14615016108150356721930896253681831785095923749 88, 0), Mint
↳ (0, 0, 0, 0, 11896287852404439590668368 7737,
↳ 40564819207303340847894502572031), Mint(982160, 0, 6279209, 1,
↳ 318130942495380,
↳ 14615016373309029182036848327162830196559325429 75), Mint(784,
↳ 8388605, 4294967292, 4294967293,
↳ 14309888427379450500257148252842651169046293 1,
↳ 14615016373309029182036848327162830196559325429 74)];
7     state.swaps = [Swap(false, 6728842), Swap(true, 2256334), Swap
↳ (false, 1601746), Swap(false, 0), Swap(true, 1), Swap(true,
↳ 4294967294), Swap(false, 4294967295), Swap(false, 15), Swap(false,
↳  13), Swap(false, 1), Swap(false, 2), Swap(false, 481), Swap(false
↳ , 231), Swap(true, 4294967293), Swap(false, 3), Swap(false,
↳ 730302967), Swap(true, 288226250), Swap(false, 4294967292), Swap(
↳ true, 2), Swap(true, 214470), Swap(false, 0), Swap(false, 1), Swap
↳ (false, 17), Swap(true, 789862), Swap(false, 3), Swap(true, 2),
↳ Swap(false, 2), Swap(false, 3), Swap(true, 2044119666), Swap(false
↳ , 814237), Swap(false, 0), Swap(true, 9844803), Swap(false, 2),
↳ Swap(true, 14889), Swap(true, 4294967294), Swap(true, 11944), Swap
↳ (false, 17337554), Swap(false, 4294967294), Swap(false, 7), Swap(
↳ false, 3), Swap(true, 4294967293), Swap(true, 22), Swap(true,
↳ 4294967295), Swap(true, 249862), Swap(false, 244), Swap(true, 0),
↳ Swap(false, 0), Swap(false, 1537), Swap(false, 0), Swap(true,
↳ 64239484)];
```

```
 8      state.burns = [Burn(2), Burn(514532), Burn(0), Burn
↳ (4294967292), Burn(250585), Burn(1), Burn(3), Burn(46378427), Burn
↳ (4294967292), Burn(72158)];
 9      state.collects = 64339;
10
11      state = _validateStateInternal(state);
12      setUpStateMoreRandom(state);
13  }
```

As explained before, this test reverts when trying to withdraw liquidity from the pool through the manager contract. The revert error can be observed in the next screenshot.



Recommendation:

Consider removing the if flow statement on the collect function of the manager who creates an error that accumulates after several transactions makes it not possible for users to withdraw their liquidity.

Remediation Plan:

SOLVED: The BubbleSwap team removed the if statement in the following commitId 07c873d250d3e5ffed5b8c0df7baaea2f72ee13d.

3.2 (HAL-02) IMPROPER SWAP AMOUNTS HANDLING - MEDIUM

Description:

The ConcentratedLiquidityPool.sol contract does not handle properly the errors when performing a swap that requires giving away a higher amount of tokens than the pool currently has.

This ends in a revert arithmetic overflow error.

Code Location:

Location - ConcentratedLiquidityPool.sol

Listing 3: ConcentratedLiquidityPool.sol

```
559 if (swapParams.zeroForOne) {
560     if (swapSnapshot.isAmountInGiven) {
561         require(swapSnapshot.end.balance0 >= swapSnapshot.start.
 ↳ balance0, "IIA-0a");
562     } else {
563         require((swapSnapshot.end.balance0 - swapSnapshot.start.
 ↳ balance0) >= actualIn, "IIA-0b");
564     }
565
566     require((swapSnapshot.start.balance1 - swapSnapshot.end.
 ↳ balance1) <= amountOut, "IIA-0c");
567 } else {
568     if (swapSnapshot.isAmountInGiven) {
569         require(swapSnapshot.end.balance1 >= swapSnapshot.start.
 ↳ balance1, "IIA-1a");
570     } else {
571         require((swapSnapshot.end.balance1 - swapSnapshot.start.
 ↳ balance1) >= actualIn, "IIA-1b");
572     }
573
574     require((swapSnapshot.start.balance0 - swapSnapshot.end.
 ↳ balance0) <= amountOut, "IIA-1c");
575 }
```

Proof Of Concept:

The next code snippet reproduces the previously described issue.

Listing 4

```
function testSwapRevert() public {

    address pool = deployPool( address(token0), address(token1),
↳ 1500, 792281625142643375935439503360 );
    addLiquidityManager(pool, address(token0), address(token1),
↳ 1000, 1000, -887272, 887272);

    uint256 _amountOutFinal = swapExactInputSingleRouter(pool,
↳ address(token0), address(token1), 1500, address(this), 1001, 0, 0)
↳ ;
}
```

As described before, when executing the test, it triggers the arithmetic overflow/underflow revert error.

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Consider adding a mechanism to control the error and swapping the corresponding number of tokens.

Remediation Plan:

SOLVED: The BubbleSwap team solved the issue by modifying the conditions in the following commitId 38755a333ae2eb46b6bf0ee94e659752a7356097.

3.3 (HAL-03) UNCONTROLLED FEEGROWTH ERROR WHEN ADDING LIQUIDITY - MEDIUM

Description:

The function rangeFeeGrowth of the contract ConcentratedLiquidityPool.sol does not handle the feegrowth variables properly. Thus, when adding liquidity, the function might revert with an arithmetic error.

Code Location:

Location - ConcentratedLiquidityPool.sol

Listing 5: ConcentratedLiquidityPool.sol

```
700 function rangeFeeGrowth(int24 lowerTick, int24 upperTick) public
 ↳ view returns (uint256 feeGrowthInside0, uint256 feeGrowthInside1)
 ↳ {
701     int24 currentTick = TickMath.getTickAtSqrtRatio(price); //
 ↳ https://github.com/sushiswap/trident/issues/338
702
703     RichTick storage lower = richTicks[lowerTick];
704     RichTick storage upper = richTicks[upperTick];
705
706     // Calculate fee growth below & above.
707     uint256 _feeGrowthGlobal0 = feeGrowthGlobal0;
708     uint256 _feeGrowthGlobal1 = feeGrowthGlobal1;
709     uint256 feeGrowthBelow0;
710     uint256 feeGrowthBelow1;
711     uint256 feeGrowthAbove0;
712     uint256 feeGrowthAbove1;
713
714
715     if (lowerTick <= currentTick) {
716         feeGrowthBelow0 = lower.feeGrowthOutside0;
717         feeGrowthBelow1 = lower.feeGrowthOutside1;
718     } else {
```

```
719         feeGrowthBelow0 = _feeGrowthGlobal0 - lower.
 ↳ feeGrowthOutside0;
720         feeGrowthBelow1 = _feeGrowthGlobal1 - lower.
 ↳ feeGrowthOutside1;
721     }
722
723     if (currentTick < upperTick) {
724         feeGrowthAbove0 = upper.feeGrowthOutside0;
725         feeGrowthAbove1 = upper.feeGrowthOutside1;
726     } else {
727         feeGrowthAbove0 = _feeGrowthGlobal0 - upper.
 ↳ feeGrowthOutside0;
728         feeGrowthAbove1 = _feeGrowthGlobal1 - upper.
 ↳ feeGrowthOutside1;
729     }
730
731     feeGrowthInside0 = _feeGrowthGlobal0 - feeGrowthBelow0 -
 ↳ feeGrowthAbove0;
732     feeGrowthInside1 = _feeGrowthGlobal1 - feeGrowthBelow1 -
 ↳ feeGrowthAbove1;
733 }
```

Risk Level:

Likelihood - 3
Impact - 3

Proof Of Concept:

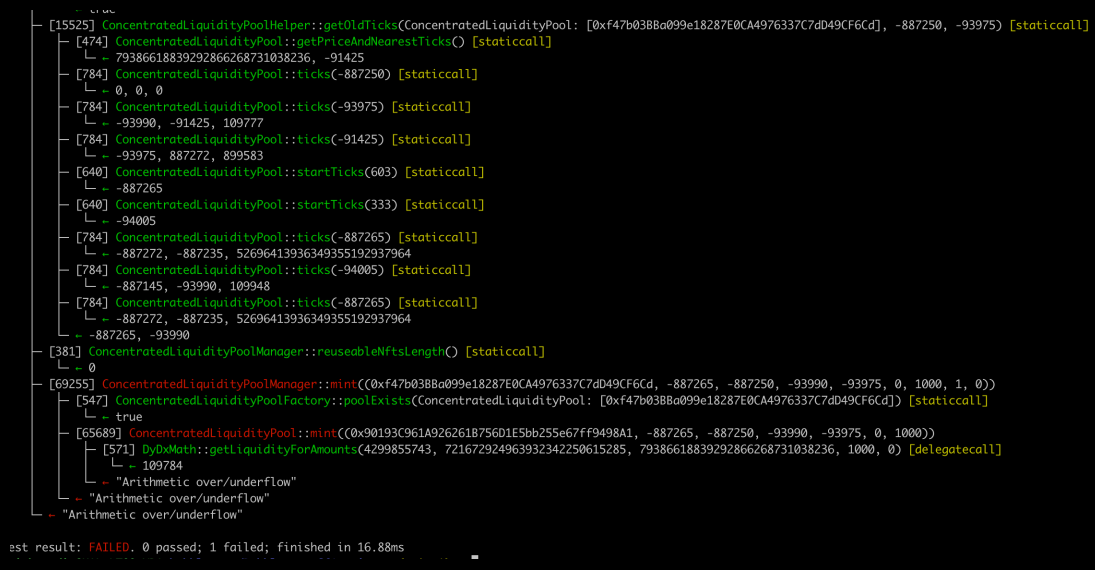
The next code snippet triggers the previously described issue.

Listing 6

```
1 function testAddLiquidityRevert() public {
2     setupEnv();
3     State memory state;
4
5     state.pool = Pool( 0, false );
6     state.mints = [Mint(0, 0, 0, 0, 0,
 ↳ 818616501176165588061273257), Mint(0, 0, 0, 0, 8168092, 0), Mint
 ↳ (0, 0, 0, 0, 72113190023062947720513039 9,
```

```
↳ 818616501176165588061273257), Mint(0, 0, 0, 0, 4727003, 674644605)
↳ , Mint(0, 0, 0, 0, 71954723305517046747822862 8, 0), Mint(0, 0, 0,
↳ 0, 0, 14627558), Mint(0, 0, 0, 0, 0, 21096719), Mint(0, 0, 0, 0,
↳ 0, 0), Mint(0, 0, 0, 0, 720627309756840161963903827, 0), Mint(0,
↳ 0, 0, 0, 0, 1718308), Mint(0, 0, 0, 0,
↳ 720627309756840161963903827, 4727003), Mint(0, 0, 0, 0, 0, 0),
↳ Mint(0, 0, 0, 0, 0, 0), Mint(0, 0, 0, 0, 0, 0), Mint(0, 0, 0, 0,
↳ 0, 0), Mint(0, 0, 0, 0, 0, 0), Mint(0, 0, 0, 0, 0, 0), Mint(0, 0,
↳ 0, 0, 0, 0), Mint(0, 0, 0, 0, 0, 0), Mint(0, 0, 0, 0, 0, 0)];
 7      state.swaps = [Swap(false, 2), Swap(false, 0), Swap(false, 0),
↳  Swap(false, 0), Swap(false, 0), Swap(false, 0), Swap(false, 0),
↳ Swap(false, 0), Swap(false, 0), Swap(false, 0), Swap(false, 0),
↳ Swap(false, 0), Swap(false, 0), Swap(false, 0), Swap(false, 0),
↳ Swap(false, 0), Swap(false, 0), Swap(false, 0), Swap(false, 0),
↳ Swap(false, 0), Swap(false, 0), Swap(false, 0), Swap(false, 0),
↳ Swap(false, 0), Swap(false, 0), Swap(false, 0), Swap(false, 0),
↳ Swap(false, 0), Swap(false, 0), Swap(false, 0), Swap(false, 0),
↳ Swap(false, 0), Swap(false, 0), Swap(false, 0), Swap(false, 63),
↳ Swap(false, 4294967295), Swap(true, 2), Swap(false, 27978), Swap(
↳ true, 326545111), Swap(true, 4294967293), Swap(false, 79310), Swap
↳ (false, 0), Swap(true, 1596228), Swap(true, 31), Swap(false, 1),
↳ Swap(true, 268219)];
 8      state.burns = [Burn(7), Burn(2), Burn(4294967293), Burn(2),
↳ Burn(3), Burn(703), Burn(3), Burn(3), Burn(3), Burn(2)];
 9      state.collects = 108;
10
11      state = _validateStateInternal(state);
12      setUpStateMoreRandom(state);
13  }
```

As it can be observed, the execution ends with the revert error previously mentioned.



Recommendation:

Consider allowing the range fee growth to overflow/underflow the same way **UniswapV3** does.

Remediation Plan:

SOLVED: The BubbleSwap team solved the issue in the following commitId a520bb7cf8162171c8d3fb18d17d10c07edb2b80.

3.4 (HAL-04) POSSIBILITY TO CREATE AN INCENTIVE FOR A DELETED POOL - LOW LowRISK ACCEPTED

Description:

The Hedera network deletes a contract if it does not pay a rent to the network. All the pools are supposed to receive money when adding/burning funds. However, there is a chance that a pool gets deleted over time.

However, if a pool is deleted, the Factory will still keep the address as an existing one. Thus, making it possible to create an incentive through the staker contract for a deleted pool.

Code Location:

Location - ConcentratedLiquidityPoolStaker.sol

Listing 7: ConcentratedLiquidityPoolStaker.sol

```
73 function addIncentive(address _pool, Incentive memory incentive)
↳ public payable costsCentiCents(1_000, 2) {
74     require(factory.poolExists(_pool), "INVALID_POOL");
75
76     require(incentive.minSpacing > 0 && incentive.maxSpacing > 0,
↳ "NULL_TICK_SPACING");
77     require(incentive.minSpacing <= MAX_SPACING && incentive.
↳ maxSpacing <= MAX_SPACING, "EXCEEDS_MAX_TICK_SPACING");
78     require(incentive.minSpacing <= incentive.maxSpacing, "
↳ INVALID_MIN_MAX_SPACING");
79
80     incentive.owner = msg.sender; // force msg.sender to be the
↳ owner
81
82     IConcentratedLiquidityPool pool = IConcentratedLiquidityPool(
↳ _pool);
83
84     uint32 current = uint32(block.timestamp);
85     require(current <= incentive.startTime, "ALREADY_STARTED");
```

```
86      require(incentive.startTime < incentive.endTime, "
↳ START_PAST_END");
87      require(incentive.endTime + 30 days < incentive.expiry, "
↳ END_PAST_BUFFER"); // allow users to claimRewards for at least 30
↳ days after endTime; after which the incentive may have expired and
↳  reclaimIncentive may be called
88      require(incentive.rewardsUnclaimed != 0, "NO_REWARDS");
89      incentive.secondsClaimed = 0;
90      incentives[pool][incentiveCount[pool]++] = incentive;
91      SafeHTS.safeTransferToken(incentive.token, msg.sender, address
↳ (this), int64(incentive.rewardsUnclaimed));
92      emit AddIncentive(pool, incentiveCount[pool], incentive.token)
↳ ;
93  }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Consider creating an administrator remove function, so when Hedera network deletes a pool, is it also possible to remove it from the manager.

Currently, the BubbleSwap team created the next HIP to try to ensure a network mechanism that prevents this from happening.

Remediation Plan:

RISK ACCEPTED: The BubbleSwap team accepted the risk of this issue.

3.5 (HAL-05) POOL SHOULD RETURN FUNDS WHEN ADDING EXTRA LIQUIDITY VIA DIRECT MINT - LOW

Description:

The ConcentratedLiquidityPool.sol contracts allows sending the funds before executing the mint function. However, if more funds than the required ones by the pool are sent, the pool will keep those extra funds and not return them to the user.

Code Location:

Code Section - ConcentratedLiquidityPool.sol#L176

Listing 8

```
176 function _mint(MintParams memory mintParams) internal returns (
 ↳ uint256 liquidityMinted, uint128 amount0Actual, uint128
 ↳ amount1Actual) {
177     _ensureTickSpacing(mintParams.lower, mintParams.upper);
178
179     uint256 priceLower = uint256(TickMath.getSqrtRatioAtTick(
 ↳ mintParams.lower));
180     uint256 priceUpper = uint256(TickMath.getSqrtRatioAtTick(
 ↳ mintParams.upper));
181     uint256 currentPrice = uint256(price);
182
183     liquidityMinted = DyDxMath.getLiquidityForAmounts(
184         priceLower,
185         priceUpper,
186         currentPrice,
187         uint256(mintParams.amount1Desired),
188         uint256(mintParams.amount0Desired)
189     );
190
191     // Ensure no overflow happens when we cast from uint256 to
 ↳ int128.
```

```
192     if (liquidityMinted > uint128(type(int128).max)) revert
 ↳ Overflow();
193
194     if (liquidityMinted == 0) {
195         revert NullLiquidity();
196     }
197
198     _updateSecondsPerLiquidity(uint256(liquidity));
199
200     TickParams memory tickParams = TickParams({
201         lowerOld: mintParams.lowerOld,
202         lower: mintParams.lower,
203         upperOld: mintParams.upperOld,
204         upper: mintParams.upper
205     });
206
207     /// @dev insert() is called before _updatePosition() as
 ↳ _updatePosition() requires the latest tick growth outside data
 ↳ including that for newly initialised ticks
208     nearestTick = Ticks.insert(
209         startTicks,
210         ticks,
211         richTicks,
212         feeGrowthGlobal0,
213         feeGrowthGlobal1,
214         secondsGrowthGlobal,
215         tickParams,
216         uint128(liquidityMinted),
217         nearestTick,
218         uint160(currentPrice)
219     );
220
221     unchecked {
222         (uint256 amount0Fees, uint256 amount1Fees) =
 ↳ _updatePosition(
223             msg.sender,
224             mintParams.lower,
225             mintParams.upper,
226             int128(uint128(liquidityMinted))
227         );
228         if (amount0Fees > 0) {
229             _deposit(true, msg.sender, amount0Fees);
230             /// @dev Unlike the original Trident implementation
 ↳ since the amount{0,1}Fees are still being custodied by the pool's
```

```
 ↳ vault{0,1} there's no reason to decrease reserve{0,1}
231             // reserve0 -= uint128(amount0Fees);
232         }
233         if (amount1Fees > 0) {
234             _deposit(false, msg.sender, amount1Fees);
235             // reserve1 -= uint128(amount1Fees);
236         }
237 
238         if (priceLower <= currentPrice && currentPrice <
 ↳ priceUpper) liquidity += uint128(liquidityMinted);
239     }
240 
241     (amount0Actual, amount1Actual) = DyDxMath.
 ↳ getAmountsForLiquidity(priceLower, priceUpper, currentPrice,
 ↳ liquidityMinted, true);
242 
243     IPositionManager(msg.sender).mintCallback(token0, token1,
 ↳ amount0Actual, amount1Actual);
244 
245     if (amount0Actual != 0) {
246         uint128 _reserve0 = reserve0;
247         _reserve0 += amount0Actual;
248         reserve0 = _reserve0;
249         uint128 actualReserves0 = uint128(_balance(token0));
250 
251         /// @dev SLOAD reserve{0,1} for gas savings
252         if (_reserve0 > actualReserves0) revert Token0Missing(
 ↳ _reserve0, actualReserves0, amount0Actual);
253     }
254 
255     if (amount1Actual != 0) {
256         uint128 _reserve1 = reserve1;
257         _reserve1 += amount1Actual;
258         reserve1 = _reserve1;
259         uint128 actualReserves1 = uint128(_balance(token1));
260 
261         if (_reserve1 > actualReserves1) revert Token1Missing(
 ↳ _reserve1, actualReserves1, amount1Actual);
262     }
263 
264     emit Mint(msg.sender, msg.sender, mintParams.lower, mintParams
 ↳ .upper, uint128(liquidityMinted), amount0Actual, amount1Actual);
265 }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Consider implementing a mechanism to give back the extra funds to the user.

Remediation Plan:

RISK ACCEPTED: The BubbleSwap team accepted the risk of this issue. The reason behind this is that users should be forced to implement the callback function where the amount to transfer funds is specified from the pool. Adding liquidity should not be done by sending funds directly to the pool.

3.6 (HAL-06) POOL DOES NOT SUPPORT FRACTIONAL FEE TOKEN TRANSFERS - INFORMATIONAL

Description:

Hedera network allows creating token contracts with a fee on transfer (docs). There are two different ways to create this fee, that the fee is taken from the payer, or that the fee is taken from the received amount by the receiver.

The ConcentratedLiquidityPool.sol contract, does not support tokens created with a fee on the receiver side, and any attempt to do add liquidity or a swap will revert.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider adding support for tokens with fee on the receiver side.

Remediation Plan:

ACKNOWLEDGED: The BubbleSwap team acknowledged this issue, assuming it as a design constraint.

3.7 (HAL-07) PAYER FIELD NEVER USED ON MINTPARAMS STRUCTS - INFORMATIONAL

Description:

The field `payer` declared as address on the `MintParams struct` of the `IConcentratedLiquidityPool.sol` is never used along the code base.

Although the data structure is set as a required parameter on the `mint` function and that the manager does set this value to the `msg.sender`, the field is not used on the pool.

Code Location:

Code location - IConcentratedLiquidityPool.sol

Listing 9: (Line 28)

```
27 struct MintParams {
28     address payer;
29     int24 lowerOld;
30     int24 lower;
31     int24 upperOld;
32     int24 upper;
33     uint128 amount0Desired;
34     uint128 amount1Desired;
35 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider removing the unused value or implementing the required code to use it.

Remediation Plan:

ACKNOWLEDGED: The BubbleSwap team acknowledged this finding.

3.8 (HAL-08) SQRTPRICELIMITX96 IS NOT USED ON V3SWAPROUTER - INFORMATIONAL

Description:

The parameter sqrtPriceLimitX96 declared as uint160 primitive type is set as a required input on exactInputSingle and exactInput functions. However, the parameter is not used.

Code Location:

Code Location -

Listing 10

```
function exactInputSingle(ExactInputSingleParams memory params)
external payable override returns (uint256 amountOut) {
    amountOut = exactInputInternal(
        params.amountIn,
        params.recipient,
        params.sqrtPriceLimitX96,
        SwapCallbackData({path: abi.encodePacked(params.tokenIn,
params.fee, params.tokenOut), payer: msg.sender})
    );
    require(amountOut >= params.amountOutMinimum, "Too little
received");
}
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider removing the unused parameter in order to call the aforementioned functions.

Remediation Plan:

ACKNOWLEDGED: The BubbleSwap team wants to keep the parameter to ensure interface compatibility with other projects.

3.9 (HAL-09) CREATE CUSTOM ERRORS TO SAVE GAS - INFORMATIONAL

Description:

Custom errors are available from Solidity version 0.8.4. Custom errors save ~50 gas each time they are hit by avoiding having to allocate and store the revert string. Not defining strings also saves deployment gas.

Although the project uses custom on errors on several contracts, the next contracts do not use them:

- V3SwapRouter.sol.
- ConcentratedLiquidityPoolStaker.sol.
- ConcentratedLiquidityPoolHelper.sol.
- ConcentratedLiquidityPoolManager.sol.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider replacing all revert strings with custom errors.

Remediation Plan:

ACKNOWLEDGED: The BubbleSwap team acknowledged this finding.

3.10 (HAL-10) INCOMPLETE NATSPEC DOCUMENTATION - INFORMATIONAL

Description:

Natspec documentation is useful for internal developers that need to work on the project, external developers that need to integrate with the project, auditors that have to review it but also for end users given that Snow trace has officially integrated the support for it directly on their site.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider adding the missing **Natspec** documentation.

Remediation Plan:

ACKNOWLEDGED: The BubbleSwap team acknowledged this finding.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' ABIs across the entire code-base.

Slither Results:

All the results provided by the tool are either false positives or not discarded once performed the dynamic analysis

- ConcentratedLiquidityPool.sol

```
~ lib/amm-v2-core » slither contracts/pool/concentrated/ConcentratedLiquidityPool.sol --solc-remaps "@openzeppelin=../openzeppelin-contracts/"
Compilation warnings/errors on contracts/pool/concentrated/ConcentratedLiquidityPool.sol:
Warning: Contract code size is 30444 bytes and exceeds 24576 bytes (a limit introduced in Spurious Dragon). This contract may not be deployable on mainnet. Consider enabling the optimizer (with a lo
s" value!), turning off revert strings, or using libraries.
  --> contracts/pool/concentrated/ConcentratedLiquidityPool.sol:26:1:
   |
26 | contract ConcentratedLiquidityPool is IConcentratedLiquidityPoolStruct, SelfFunding {
   | ^ (Relevant source part starts here and spans across multiple lines).


FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv = (3 * denominator) ^ 2 (contracts/libraries/FullMath.sol#82)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#86)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#87)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#88)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#89)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#90)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#91)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (contracts/libraries/FullMath.sol#68)
        -result = prod0 * inv (contracts/libraries/FullMath.sol#98)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

ConcentratedLiquidityPool.swap(IConcentratedLiquidityPoolStruct.SwapParams,bytes) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#330-575) uses a dangerous strict equality:
        - cache.currentLiquidity == 0 (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#470)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

ConcentratedLiquidityPool.swap(IConcentratedLiquidityPoolStruct.SwapParams,bytes).swapSnapshot (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#331) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

AUTOMATED TESTING

```
ConcentratedLiquidityPool.swap(IConcentratedLiquidityPoolStruct.SwapParams,bytes) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#330-575) uses a dangerous strict equality:
        - cache.currentLiquidity == 0 (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#470)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

ConcentratedLiquidityPool.swap(IConcentratedLiquidityPoolStruct.SwapParams,bytes).swapSnapshot (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#331) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ConcentratedLiquidityPoolFactory.cachedDeployData (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#24) is written in both
        cachedDeployData = _deployData (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#55)
        cachedDeployData =  (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write

Reentrancy in ConcentratedLiquidityPool._mint(IConcentratedLiquidityPoolStruct.MintParams) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#167-256):
        External calls:
        - IPositionManager(msg.sender).mintCallback(token0,token1,amount0Actual,amount1Actual) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#234)
        State variables written after the call(s):
        - reserve0 = _reserve0 (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#239)
        - reserve1 = _reserve1 (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#249)
Reentrancy in ConcentratedLiquidityPoolFactory.deployPool(address,address,uint24,uint160) (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#39-72):
        External calls:
        - costsCentiCents(MIN_SUPPLY,2) (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#44)
                - (success) = address(recipient).call{value: tinybars}() (contracts/_precompile-exchange-rate/SelfFunding.sol#41)
                - (success,result) = PRECOMPILE_ADDRESS.call(abi.encodeWithSelector(IExchangeRate.tinycentsToTinybars.selector,tinycents)) (contracts/_precompile-exchange-rate/SelfFunding.sol#15-17)
        External calls sending eth:
        - costsCentiCents(MIN_SUPPLY,2) (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#44)
                - (success) = address(recipient).call{value: tinybars}() (contracts/_precompile-exchange-rate/SelfFunding.sol#41)
        State variables written after the call(s):
        - cachedDeployData = _deployData (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#55)
Reentrancy in ConcentratedLiquidityPoolFactory.deployPool(address,address,uint24,uint160) (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#39-72):
        External calls:
        - pool = address(new ConcentratedLiquidityPool{value: centiCentsToTinybars(99_000),salt: salt}()) (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#62)
                - (success,result) = PRECOMPILE_ADDRESS.call(abi.encodeWithSelector(IExchangeRate.tinycentsToTinybars.selector,tinycents)) (contracts/_precompile-exchange-rate/SelfFunding.sol#15-17)
        - costsCentiCents(MIN_SUPPLY,2) (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#44)
                - (success) = address(recipient).call{value: tinybars}() (contracts/_precompile-exchange-rate/SelfFunding.sol#41)
                - (success,result) = PRECOMPILE_ADDRESS.call(abi.encodeWithSelector(IExchangeRate.tinycentsToTinybars.selector,tinycents)) (contracts/_precompile-exchange-rate/SelfFunding.sol#15-17)
        External calls sending eth:
        - pool = address(new ConcentratedLiquidityPool{value: centiCentsToTinybars(99_000),salt: salt}()) (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#62)
        - costsCentiCents(MIN_SUPPLY,2) (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#44)
                - (success) = address(recipient).call{value: tinybars}() (contracts/_precompile-exchange-rate/SelfFunding.sol#41)
        State variables written after the call(s):
        - cachedDeployData =  (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#64)
        - getPool[token0][token1][swapFee] = pool (contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol#66)
```

```
ConcentratedLiquidityPool.swap(IConcentratedLiquidityPoolStruct.SwapParams,bytes).swapSnapshot (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#331) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Reentrancy in ConcentratedLiquidityPool._mint(IConcentratedLiquidityPoolStruct.MintParams) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#167-256):
        External calls:
        - IPositionManager(msg.sender).mintCallback(token0,token1,amount0Actual,amount1Actual) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#234)
        State variables written after the call(s):
        - reserve0 = _reserve0 (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#239)
        - reserve1 = _reserve1 (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#249)
Reentrancy in ConcentratedLiquidityPool.withdraw(address,uint256,uint256) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#695-713):
        External calls:
        - SafeHTS.safeTransferToken(token0,address(this),recipient,int64(_shares0)) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#704)
        State variables written after the call(s):
        - reserve1 -= uint128(shares1) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#710)
        - vault1[msg.sender] = vault1Balance - _shares1 (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#709)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in ConcentratedLiquidityPool._mint(IConcentratedLiquidityPoolStruct.MintParams) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#167-256):
        External calls:
        - IPositionManager(msg.sender).mintCallback(token0,token1,amount0Actual,amount1Actual) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#234)
        Event emitted after the call(s):
        - Mint(msg.sender,msg.sender,mintParams.lower,mintParams.upper,uint128(liquidityMinted),amount0Actual,amount1Actual) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#255)
Reentrancy in ConcentratedLiquidityPool.swap(IConcentratedLiquidityPoolStruct.SwapParams,bytes) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#330-575):
        External calls:
        - SafeHTS.safeTransferToken(token1,address(this),swapParams.recipient,int64(int256(amountOut))) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#511)
        - ISwapCallback(msg.sender).swapCallback(0,amountOut,data) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#513)
        - ISwapCallback(msg.sender).swapCallback(actualIn,amountOut,data) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#513)
        Event emitted after the call(s):
        - Swap(msg.sender,swapParams.recipient,token0,token1,actualIn,amountOut,price,uint128(cache.currentLiquidity),TickMath.getTickAtSqrtRatio(price)) (contracts/pool/concentrated/ConcentratedLiquidity
Pool.sol#515-525)
Reentrancy in ConcentratedLiquidityPool.swap(IConcentratedLiquidityPoolStruct.SwapParams,bytes) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#330-575):
        External calls:
        - SafeHTS.safeTransferToken(token0,address(this),swapParams.recipient,int64(int256(amountOut))) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#527)
        - ISwapCallback(msg.sender).swapCallback(amountOut,0,data) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#529)
        - ISwapCallback(msg.sender).swapCallback(amountOut,actualIn,data) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#529)
        Event emitted after the call(s):
        - Swap(msg.sender,swapParams.recipient,token1,token0,actualIn,amountOut,price,uint128(cache.currentLiquidity),TickMath.getTickAtSqrtRatio(price)) (contracts/pool/concentrated/ConcentratedLiquidity
Pool.sol#531-541)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ConcentratedLiquidityPool.setPrice(uint160) (contracts/pool/concentrated/ConcentratedLiquidityPool.sol#150-155) uses timestamp for comparisons
        Dangerous comparisons:
```

- ConcentratedLiquidityPoolManager.sol

```
lib/amm-v2-core » slither contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol --solc-remaps "@openzeppelin=../openzeppelin-contracts/"  82
Compilation warnings/errors on contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol:
Warning: Function state mutability can be restricted to pure
  --> contracts/_precompile-hts/KeyHelper.sol:30:5:
   |
30 |     function createKeyValueType(
   |     ^ (Relevant source part starts here and spans across multiple lines).

Warning: Function state mutability can be restricted to pure
 --> contracts/_precompile-hts/ExpiryHelper.sol:9:5:
  |
9 |     function getAutoRenewExpiry(address autoRenewAccount, uint32 autoRenewPeriod)
  |     ^ (Relevant source part starts here and spans across multiple lines).

Warning: Function state mutability can be restricted to pure
  --> contracts/_precompile-hts/ExpiryHelper.sol:18:5:
   |
18 |     function getSecondExpiry(uint32 second) internal view returns (IHederaTokenService.Expiry memory expiry) {
   |     ^ (Relevant source part starts here and spans across multiple lines).


Multicall.multicall(bytes[]) (contracts/base/Multicall.sol#11-22) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/base/Multicall.sol
#14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-functions-using-delegatecall-inside-a-loop

Reentrancy in ConcentratedLiquidityPoolManager.burn(int64,uint128,uint256,uint256) (contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol#130-190):
        External calls:
        - burnResult = position.pool.burn{value: centiCentsToTinybars(50)}(position.lower,position.upper,amount) (contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol#144-148)
                - (success,result) = PRECOMPILE_ADDRESS.call(abi.encodeWithSelector(IExchangeRate.tinycentsToTinybars.selector,tinycents)) (contracts/_precompile-exchange-rate/SelfFunding.sol#15-17)
        - costsCentiCents(100,2) (contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol#135)
                - (success) = address(recipient).call{value: tinybars}() (contracts/_precompile-exchange-rate/SelfFunding.sol#41)
                - (success,result) = PRECOMPILE_ADDRESS.call(abi.encodeWithSelector(IExchangeRate.tinycentsToTinybars.selector,tinycents)) (contracts/_precompile-exchange-rate/SelfFunding.sol#15-17)
        External calls sending eth:
        - burnResult = position.pool.burn{value: centiCentsToTinybars(50)}(position.lower,position.upper,amount) (contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol#144-148)
        - costsCentiCents(100,2) (contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol#135)
                - (success) = address(recipient).call{value: tinybars}() (contracts/_precompile-exchange-rate/SelfFunding.sol#41)
        State variables written after the call(s):
        - positions[tokenId].feeGrowthInside0 = positionFeesData.feeGrowthInside0 (contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol#152)
        - positions[tokenId].feeGrowthInside1 = positionFeesData.feeGrowthInside1 (contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol#153)
        - positions[tokenId].liquidity -= amount (contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol#154)
        - positions[tokenId].unclaimedFees0 = 0 (contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol#159)
        - positions[tokenId].unclaimedFees1 = 0 (contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol#160)
```

- ConcentratedLiquidityPoolHelper.sol

```
lib/amm-v2-core » slither contracts/pool/concentrated/ConcentratedLiquidityPoolHelper.sol --solc-remaps "@openzeppelin=../openzeppelin-contracts/"  255

FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv = (3 * denominator) ^ 2 (contracts/libraries/FullMath.sol#82)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#86)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#87)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#88)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#89)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#90)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#91)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (contracts/libraries/FullMath.sol#68)
        -result = prod0 * inv (contracts/libraries/FullMath.sol#98)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Contract locking ether found:
        Contract ConcentratedLiquidityPoolHelper (contracts/pool/concentrated/ConcentratedLiquidityPoolHelper.sol#15-713) has payable functions:
         - ConcentratedLiquidityPoolHelper.constructor() (contracts/pool/concentrated/ConcentratedLiquidityPoolHelper.sol#23)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

ConcentratedLiquidityPoolHelper._getOldTicks(IConcentratedLiquidityPool,int24,int24).upperTick (contracts/pool/concentrated/ConcentratedLiquidityPoolHelper.sol#215) is a local variable never initialized
ConcentratedLiquidityPoolHelper._getOldTicks(IConcentratedLiquidityPool,int24,int24).segInfo (contracts/pool/concentrated/ConcentratedLiquidityPoolHelper.sol#253) is a local variable never initialized
ConcentratedLiquidityPoolHelper.getTickState(IConcentratedLiquidityPool,int24,uint24).i (contracts/pool/concentrated/ConcentratedLiquidityPoolHelper.sol#41) is a local variable never initialized
ConcentratedLiquidityPoolHelper._getOldTicks(IConcentratedLiquidityPool,int24,int24).lowerTick (contracts/pool/concentrated/ConcentratedLiquidityPoolHelper.sol#214) is a local variable never initialized
ConcentratedLiquidityPoolHelper._getOldTicks(IConcentratedLiquidityPool,int24,int24).gotUpper (contracts/pool/concentrated/ConcentratedLiquidityPoolHelper.sol#198) is a local variable never initialized
ConcentratedLiquidityPoolHelper._getOldTicks(IConcentratedLiquidityPool,int24,int24).gotLower (contracts/pool/concentrated/ConcentratedLiquidityPoolHelper.sol#197) is a local variable never initialized
ConcentratedLiquidityPoolHelper.getAmountIn(IConcentratedLiquidityPool,address,uint256).cache (contracts/pool/concentrated/ConcentratedLiquidityPoolHelper.sol#425) is a local variable never initialized
ConcentratedLiquidityPoolHelper.getSurroundingTicks(IConcentratedLiquidityPool,uint24).i (contracts/pool/concentrated/ConcentratedLiquidityPoolHelper.sol#92) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

- ConcentratedLiquidityPoolStaker.sol

```
. lib/amm-v2-core » slither contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol --solc-remaps "@openzeppelin=../openzeppelin-contracts/"                    255 ↵

ConcentratedLiquidityPoolStaker.subscribe(uint256,uint256[]) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#107-144) uses a dangerous strict equality:
        - ! (position.latestAddition > stake.timestamp || ! (stakeData.pool == stake.pool && stakeData.lower == stake.lower && stakeData.upper == stake.upper)) (contracts/pool/concentrated/ConcentratedLiq
uidityPoolStaker.sol#135-136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

ConcentratedLiquidityPoolStaker.subscribe(uint256,uint256[]).i (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#122) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Reentrancy in ConcentratedLiquidityPoolStaker.addIncentive(address,ConcentratedLiquidityPoolStaker.Incentive) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#73-93):
        External calls:
        - costsCentiCents(1_000,2) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#73)
                - (success) = address(recipient).call{value: tinybars}() (contracts/_precompile-exchange-rate/SelfFunding.sol#41)
                - (success,result) = PRECOMPILE_ADDRESS.call(abi.encodeWithSelector(IExchangeRate.tinycentsToTinybars.selector,tinycents)) (contracts/_precompile-exchange-rate/SelfFunding.sol#15-17)
        External calls sending eth:
        - costsCentiCents(1_000,2) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#73)
                - (success) = address(recipient).call{value: tinybars}() (contracts/_precompile-exchange-rate/SelfFunding.sol#41)
        State variables written after the call(s):
        - incentives[pool][incentiveCount[pool] ++] = incentive (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#90)
        - incentives[pool][incentiveCount[pool] ++] = incentive (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#90)
Reentrancy in ConcentratedLiquidityPoolStaker.associate(address) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#67-71):
        External calls:
        - SafeHTS.safeAssociateToken(token,address(this)) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#68)
        State variables written after the call(s):
        - associated[token] = true (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#70)
Reentrancy in ConcentratedLiquidityPoolStaker.subscribe(uint256,uint256[]) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#107-144):
        External calls:
        - costsCentiCents(100,2) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#107)
                - (success) = address(recipient).call{value: tinybars}() (contracts/_precompile-exchange-rate/SelfFunding.sol#41)
                - (success,result) = PRECOMPILE_ADDRESS.call(abi.encodeWithSelector(IExchangeRate.tinycentsToTinybars.selector,tinycents)) (contracts/_precompile-exchange-rate/SelfFunding.sol#15-17)
        External calls sending eth:
        - costsCentiCents(100,2) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#107)
                - (success) = address(recipient).call{value: tinybars}() (contracts/_precompile-exchange-rate/SelfFunding.sol#41)
        State variables written after the call(s):
        - stakes[positionId][incentiveId[i]] = stakeData (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in ConcentratedLiquidityPoolStaker.addIncentive(address,ConcentratedLiquidityPoolStaker.Incentive) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#73-93):
        External calls:
        - SafeHTS.safeTransferToken(incentive.token,msg.sender,address(this),int64(incentive.rewardsUnclaimed)) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#91)
        - costsCentiCents(1_000,2) (contracts/pool/concentrated/ConcentratedLiquidityPoolStaker.sol#73)
                - (success) = address(recipient).call{value: tinybars}() (contracts/_precompile-exchange-rate/SelfFunding.sol#41)
```

- ConcentratedLiquidityPoolFactory.sol

```
. lib/amm-v2-core » slither contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol --solc-remaps "@openzeppelin=../openzeppelin-contracts/"                    255 ↵
Compilation warnings/errors on contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol:
Warning: Contract code size is 30444 bytes and exceeds 24576 bytes (a limit introduced in Spurious Dragon). This contract may not be deployable on mainnet. Consider enabling the optimizer (with a low "run
s" value!), turning off revert strings, or using libraries.
  --> contracts/pool/concentrated/ConcentratedLiquidityPool.sol:26:1:
   |
26 | contract ConcentratedLiquidityPool is IConcentratedLiquidityPoolStruct, SelfFunding {
   | ^ (Relevant source part starts here and spans across multiple lines).

Warning: Contract code size is 39721 bytes and exceeds 24576 bytes (a limit introduced in Spurious Dragon). This contract may not be deployable on mainnet. Consider enabling the optimizer (with a low "run
s" value!), turning off revert strings, or using libraries.
  --> contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol:16:1:
   |
16 | contract ConcentratedLiquidityPoolFactory is IConcentratedLiquidityPoolFactory, SelfFunding {
   | ^ (Relevant source part starts here and spans across multiple lines).



FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv = (3 * denominator) ^ 2 (contracts/libraries/FullMath.sol#82)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#86)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#87)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#88)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#89)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#90)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#64)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#91)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#15-101) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (contracts/libraries/FullMath.sol#68)
        -result = prod0 * inv (contracts/libraries/FullMath.sol#98)
```

- No major issues were found by Slither.

Mythx Results:

All the results provided by the tool are either false positives or not discarded once performed the dynamic analysis

- ConcentratedLiquidityPool.sol

Report for amm-v2-core/contracts/pool/concentrated/ConcentratedLiquidityPool.sol
https://dashboard.mythx.io/#/console/analyses/5a3aa11f-34d5-460d-846d-6924ac67eb63
https://dashboard.mythx.io/#/console/analyses/867fec75-c240-4e76-9195-fbe816c2ab29

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
83	(SWC-110) Assert Violation	Unknown	Public state variable with array type causing reacheable exception by default.
142	(SWC-110) Assert Violation	Unknown	Out of bounds array access
143	(SWC-110) Assert Violation	Unknown	Out of bounds array access
229	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
238	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
248	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
272	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
292	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
293	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
321	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
324	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
324	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
340	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
341	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
371	(SWC-115) Authorization through tx.origin	Low	Use of "tx.origin" as a part of authorization control.
398	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

- ConcentratedLiquidityPoolManager.sol

Report for amm-v2-core/contracts/pool/concentrated/ConcentratedLiquidityPoolManager.sol
https://dashboard.mythx.io/#/console/analyses/92a49691-e441-4c64-bd68-4b2b35323bee

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
101	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
102	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
104	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
154	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
156	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
157	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
171	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
172	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
183	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
184	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
198	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
199	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
213	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
214	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
243	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
249	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

- ConcentratedLiquidityPoolHelper.sol

Report for pool/concentrated/ConcentratedLiquidityPoolHelper.sol
https://dashboard.mythx.io/#/console/analyses/30ddcd1a-ce1c-411a-90af-fd09e7b868b7

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
15	(SWC-123) Requirement Violation	Low	Requirement violation.
32	(SWC-123) Requirement Violation	Low	Requirement violation.

- ConcentratedLiquidityPoolStaker.sol

Report for pool/concentrated/ConcentratedLiquidityPoolStaker.sol
https://dashboard.mythx.io/#/console/analyses/474f59c0-558d-4faa-a6d3-e8647b09447f

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
21	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
22	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
24	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
25	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
87	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
90	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
101	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
120	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
122	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
123	(SWC-110) Assert Violation	Unknown	Out of bounds array access
124	(SWC-110) Assert Violation	Unknown	Out of bounds array access
138	(SWC-110) Assert Violation	Unknown	Out of bounds array access
141	(SWC-110) Assert Violation	Unknown	Out of bounds array access
142	(SWC-110) Assert Violation	Unknown	Out of bounds array access
156	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered

- ConcentratedLiquidityPoolFactory.sol

Report for amm-v2-core/contracts/pool/concentrated/ConcentratedLiquidityPoolFactory.sol
https://dashboard.mythx.io/#/console/analyses/5a3aa11f-34d5-460d-846d-6924ac67eb63

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

- No major issues were found by Slither.

THANK YOU FOR CHOOSING

// HALBORN