// HALBORN

HighStreetMarket - Staking

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **December 10th, 2021 - December 22nd, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 5

CONTACTS 5

1 EXECUTIVE OVERVIEW 6

1.1 INTRODUCTION 7

1.2 AUDIT SUMMARY 7

1.3 TEST APPROACH & METHODOLOGY 7

RISK METHODOLOGY 8

1.4 SCOPE 10

2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 11

3 FINDINGS & TECH DETAILS 12

3.1 (HAL-01) DENIAL OF SERVICE CAUSED BY NONREENTRANT MODIFIER - CRITICAL 14

Description 14

Risk Level 15

Recommendation 15

Remediation Plan 16

3.2 (HAL-02) SUBVAULTREWARDS ARE NOT UPDATED CORRECTLY WHEN CALLING PROCESSREWARDS - HIGH 17

Description 17

Proof of Concept 17

Risk Level 18

Recommendation 18

Remediation Plan 19

3.3 (HAL-03) UNSTAKE FUNCTION CAN BE LOCKED IF TOTALWEIGHT EQUALS TO ZERO - MEDIUM 20

Description 20

Risk Level 21

Recommendation 21

Remediation Plan 22

3.4 (HAL-04) UPDATEHIGHPERBLOCK IMPACTS NEGATIVELY ALL THE UNCLAIMED REWARDS - MEDIUM 23

Description 23

Proof of Concept 23

Risk Level 24

Recommendation 24

Remediation Plan 25

3.5 (HAL-05) MISSING REQUIRE STATEMENT IN UPDATESTAKELOCK FUNCTION - LOW 26

Description 26

Code Location 26

Risk Level 26

Recommendation 27

Remediation Plan 27

3.6 (HAL-06) USERS CAN STAKE IN A FLASHPOOL EVEN IF THE POOL IS DISABLED - LOW 28

Description 28

Code Location 28

Risk Level 28

Recommendation 28

Remediation Plan 29

3.7 (HAL-07) INCORRECT AMOUNT IN STAKED EVENT EMISSION - LOW 30

Description 30

Code Location 30

Risk Level 32
Recommendation 32
Remediation Plan 32
3.8 (HAL-08) SOLC 0.8.0 COMPILER VERSION CONTAINS MULTIPLE BUGS - INFORMATIONAL 33
Description 33
Risk Level 33
Recommendation 33
Remediation Plan 33
3.9 (HAL-09) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL 34
Description 34
Risk Level 34
Recommendation 34
Remediation Plan 34
3.10 (HAL-10) UINT32/UINT64 TYPES ARE LESS GAS EFFICIENT - INFORMATIONAL 35
Description 35
Risk Level 35
Recommendation 35
Remediation Plan 36
3.11 (HAL-11) WRONG COMMENT - INFORMATIONAL 37
Description 37
Risk Level 37
Recommendation 38
Remediation Plan 38
4 AUTOMATED TESTING 39
4.1 STATIC ANALYSIS REPORT 40

Description 40

Slither results 40

4.2 AUTOMATED SECURITY SCAN 48

Description 48

MythX results 48

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	12/10/2021	Roberto Reigada
0.2	Document Updates	12/12/2021	Roberto Reigada
0.3	Document Updates	12/12/2021	Aidan Marlin
0.4	Draft Review	12/15/2021	Gabi Urrutia
1.0	Remediation Plan	12/17/2021	Roberto Reigada
1.1	Remediation Plan Review	12/19/2021	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
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com
Aidan Marlin	Halborn	Aidan.Marlin@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

HighStreetMarket engaged Halborn to conduct a security audit on their staking smart contracts beginning on December 10th, 2021 and ending on December 22nd, 2021. The security assessment was scoped to the smart contract provided in the Github repository Highstreet-World/StakingPool

1.2 AUDIT SUMMARY

The team at Halborn was provided a week for the engagement and assigned two full time security engineer to audit the security of the smart contract. The security engineers are blockchain and smart-contract security experts 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 addressed by HighStreetMarket 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 bridge 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
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

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

IN-SCOPE:
The security assessment was scoped to the following smart contracts:

- HighStreetCorePool.sol
- HighStreetFlashPool.sol
- HighStreetPoolBase.sol
- HighStreetPoolFactory.sol

Commit ID: ff826505c538bbfb7b84a898361e7d1bc0e921b0

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	1	2	3	4

LIKELIHOOD

IMPACT

(HAL-03)			(HAL-02)	(HAL-01)
	(HAL-04)			
(HAL-06)				
	(HAL-05)			
(HAL-08) (HAL-09) (HAL-10) (HAL-11)		(HAL-07)		

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - DENIAL OF SERVICE CAUSED BY NONREENTRANT MODIFIER	Critical	SOLVED - 12/16/2021
HAL02 - SUBVAULTREWARDS ARE NOT UPDATED CORRECTLY WHEN CALLING PROCESSREWARDS	High	SOLVED - 12/16/2021
HAL03 - UNSTAKE FUNCTION CAN BE LOCKED IF TOTALWEIGHT EQUALS TO ZERO	Medium	SOLVED - 12/16/2021
HAL04 - UPDATEHIGHPERBLOCK IMPACTS NEGATIVELY ALL THE UNCLAIMED REWARDS	Medium	SOLVED - 12/16/2021
HAL05 - MISSING REQUIRE STATEMENT IN UPDATESTAKELOCK FUNCTION	Low	SOLVED - 12/16/2021
HAL06 - USERS CAN STAKE IN A FLASHPOOL EVEN IF THE POOL IS DISABLED	Low	SOLVED - 12/16/2021
HAL07 - INCORRECT AMOUNT IN STAKED EVENT EMISSION	Low	SOLVED - 12/16/2021
HAL08 - SOLC 0.8.0 COMPILER VERSION CONTAINS MULTIPLE BUGS	Informational	SOLVED - 12/16/2021
HAL09 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	SOLVED - 12/16/2021
HAL10 - UINT32/UINT64 TYPES ARE LESS GAS EFFICIENT	Informational	SOLVED - 12/16/2021
HAL11 - WRONG COMMENT	Informational	SOLVED - 12/16/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) DENIAL OF SERVICE CAUSED BY NONREENTRANT MODIFIER - CRITICAL

Description:

The following functions contain the nonReentrant modifier:
- HighStreetPoolBase.stake()
- HighStreetPoolBase.unstake()
- HighStreetPoolBase.updateStakeLock()
- HighStreetPoolBase.processRewards()
- HighStreetCorePool.transferHighToken()
- HighStreetCorePool.transferHighTokenFrom()

When receiveVaultRewards() function is called the pendingVaultRewards of the stakers will be always different from 0. In this case, when HighStreetPoolBase.unstake() is called, the flow below will be executed:

1. HighStreetPoolBase.unstake() (nonReentrant)
2. ---- HighStreetCorePool._unstake()
3. -------- HighStreetPoolBase._unstake()
4. ---------- HighStreetPoolBase._sync()
5. ---------- HighStreetCorePool._processRewards()
6. ---------- HighStreetCorePool._processVaultRewards()
7. ---------- HighStreetCorePool.transferHighToken() (nonReentrant again)

As we can see, in this case, HighStreetCorePool.transferHighToken() will be called by HighStreetPoolBase.unstake(), which is not possible as both functions got the nonReentrant modifier. This only happens when pendingVaultRewards > 0 as per the code below:

Listing 1: HighStreetCorePool.sol (Lines 275)

```
258 function _processVaultRewards(address _staker) private {
259     User storage user = users[_staker];
```

```
260     uint256 pendingVaultClaim = pendingVaultRewards(_staker);
261     if (pendingVaultClaim == 0) return;
262     // read HIGH token balance of the pool via standard ERC20
            interface
263     uint256 highBalance = IERC20(HIGH).balanceOf(address(this));
264     require(highBalance >= pendingVaultClaim, "contract HIGH
            balance too low");
265
266     // update `poolTokenReserve` only if this is a HIGH Core Pool
267     if (poolToken == HIGH) {
268         // protects against rounding errors
269         poolTokenReserve -= pendingVaultClaim > poolTokenReserve ?
                poolTokenReserve : pendingVaultClaim;
270     }
271
272     user.subVaultRewards = weightToReward(user.totalWeight,
            vaultRewardsPerWeight);
273
274     // transfer fails if pool HIGH balance is not enough - which
            is a desired behavior
275     transferHighToken(_staker, pendingVaultClaim);
276
277     emit VaultRewardsClaimed(msg.sender, _staker,
            pendingVaultClaim);
278 }
```

This would cause every single call to HighStreetPoolBase.unstake() to revert with the following error message: ReentrancyGuard: reentrant call

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

It is recommended to remove the nonReentrant modifier from the HighStreetCorePool.transferHighToken() function. The same is recommended for the HighStreetCorePool.transferHighTokenFrom() function. Both functions are internal.

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID ffef2b7402a0e5308b8019727688a242a220443e

3.2 (HAL-02) SUBVAULTREWARDS ARE NOT UPDATED CORRECTLY WHEN CALLING PROCESSREWARDS - HIGH

Description:

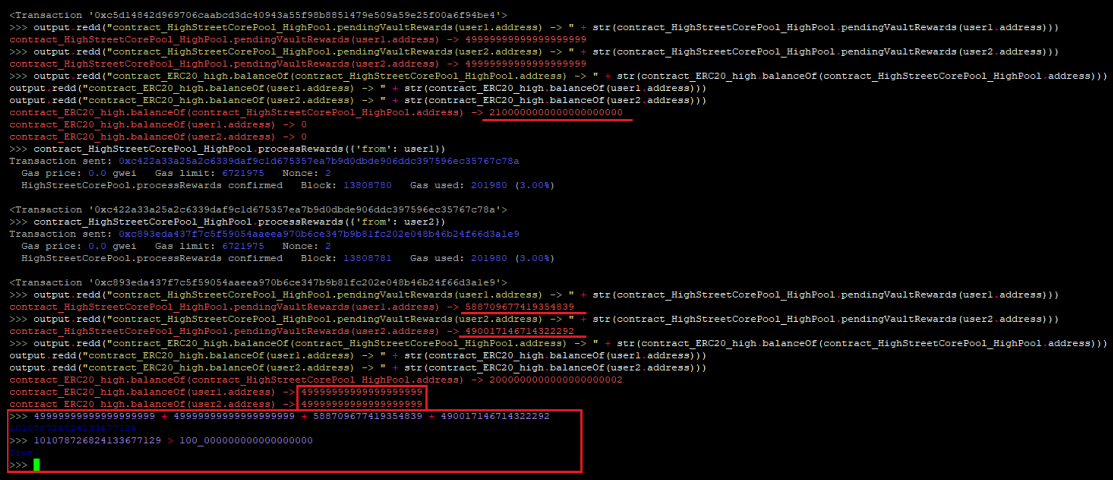
In the HighStreetCorePool contract, the function receiveVaultRewards() is executed by the vault to transfer vault rewards HIGH from the vault into the pool.

The function processRewards() and _processVaultRewards() handle the distribution of these rewards but this is not done correctly, and the users are sent a higher amount of reward tokens than the initial amount sent by the vault to the pool.

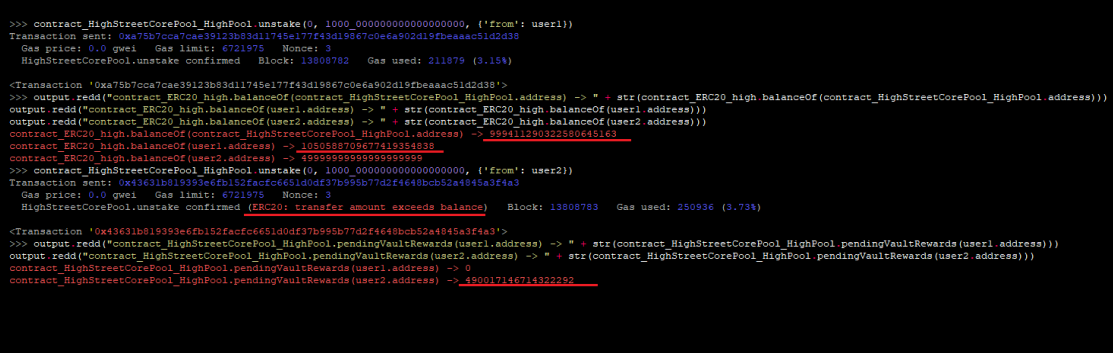
This causes that the smart contract eventually gets a lower reward token balance that what it actually should not allow users to unstake as the balance of the contract is not enough to transfer the users their reward tokens.

Proof of Concept:

```
>>> contract_HighStreetCorePool_HighPool.stake(1000_000000000000000000, chain.time() + (24*60*60*7), {'from': user1})
Transaction sent: 0x10bbcf7793ffb701c11d2ef3de1bad20bc43e8322228f3c77f3d606aae981945
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  HighStreetCorePool.stake confirmed   Block: 13808775   Gas used: 230907 (3.44%)

<Transaction '0x10bbcf7793ffb701c11d2ef3de1bad20bc43e8322228f3c77f3d606aae981945'>
>>> contract_HighStreetCorePool_HighPool.stake(1000_000000000000000000, chain.time() + (24*60*60*7), {'from': user2})
Transaction sent: 0x335ab9c9c4186366e961bc4d074d78a90e4d3d5a172cf1535fcf5ceed63a24dc
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  HighStreetCorePool.stake confirmed   Block: 13808776   Gas used: 237248 (3.53%)

<Transaction '0x335ab9c9c4186366e961bc4d074d78a90e4d3d5a172cf1535fcf5ceed63a24dc'>
>>> output.greenn("\nSleeping " + str(604800) + " seconds / 1 week...\n")
chain.sleep(24*60*60*7)
chain.mine(1)

Sleeping 604800 seconds / 1 week...

>>> contract_ERC20_high.approve(contract_HighStreetCorePool_HighPool.address, 100_000000000000000000, {'from': vault})
Transaction sent: 0x8ce27bcffbdfa7678d8fe6a7c9511d8cbfb69a9d63ef280ed1bed5df086483f6
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 0
  MockERC20.approve confirmed   Block: 13808778   Gas used: 44225 (0.66%)

<Transaction '0x8ce27bcffbdfa7678d8fe6a7c9511d8cbfb69a9d63ef280ed1bed5df086483f6'>
>>> contract_HighStreetCorePool_HighPool.receiveVaultRewards(100_000000000000000000, {'from': vault})
Transaction sent: 0xc5d14842d969706caabcd3dc40943a55f98b8851479e509a59e25f00a6f94be4
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  HighStreetCorePool.receiveVaultRewards confirmed   Block: 13808779   Gas used: 50696 (0.75%)
```



As we can see, 100 tokens were sent by the vault through the call to receiveVaultRewards() functions, but the total vault rewards that will be given are higher than that. In this case, this will cause that the smart contract will not have enough balance to allow user2 to unstake:



Risk Level:

Likelihood - 4

Impact - 5

Recommendation:

It is recommended to fix the logic in the processRewards() and _processVaultRewards() functions so the stakers are not sent a higher amount of reward tokens than the initial amount sent by the vault to the pool through receiveVaultRewards() function.

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID 99c5bf0cc40bc76fce973756a45634dfafaa3e01

3.3 (HAL-03) UNSTAKE FUNCTION CAN BE LOCKED IF TOTALWEIGHT EQUALS TO ZERO - MEDIUM

Description:

The contract HighStreetPoolBase contains the following function:

Listing 2: HighStreetPoolBase.sol (Lines 565)

```
538 function _sync() internal virtual {
539     // update HIGH per block value in factory if required
540     if (factory.shouldUpdateRatio()) {
541         factory.updateHighPerBlock();
542     }
543
544     // check bound conditions and if these are not met -
545     // exit silently, without emitting an event
546     uint256 endBlock = factory.endBlock();
547     if (lastYieldDistribution >= endBlock) {
548         return;
549     }
550     if (blockNumber() <= lastYieldDistribution) {
551         return;
552     }
553     // if locking weight is zero - update only `
            lastYieldDistribution` and exit
554     if (usersLockingWeight == 0) {
555         lastYieldDistribution = uint64(blockNumber());
556         return;
557     }
558
559     // to calculate the reward we need to know how many blocks
            passed, and reward per block
560     uint256 currentBlock = blockNumber() > endBlock ? endBlock :
            blockNumber();
561     uint256 blocksPassed = currentBlock - lastYieldDistribution;
562     uint256 highPerBlock = factory.highPerBlock();
563
564     // calculate the reward
```

```
565     uint256 highReward = (blocksPassed * highPerBlock * weight) /
            factory.totalWeight();
566
567     // update rewards per weight and `lastYieldDistribution`
568     yieldRewardsPerWeight += rewardToWeight(highReward,
            usersLockingWeight);
569     lastYieldDistribution = uint64(currentBlock);
570
571     // emit an event
572     emit Synchronized(msg.sender, yieldRewardsPerWeight,
            lastYieldDistribution);
573 }
```

If factory.totalWeight() equals to zero, a division by zero occurs and the call reverts. As this _sync() function is called every time by the unstake() function, if this value is zero all the rewards and staked tokens will be locked.

This scenario where factory.totalWeight() equals to zero can only occur if all the registered pools are HighStreetFlashPool and if all of them have expired/are disabled. In that case, when the users call the unstake() function to retrieve their staked tokens and their rewards the operation will revert.

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is recommended to modify HighStreetPoolFactory.registerPool() function so it does not allow anyone to add a HighStreetFlashPool unless there is already at least one HighStreetCorePool added. This way, the lock would never be possible, as that scenario where the factory.totalWeight() is zero can only happen if all the registered pools are HighStreetFlashPool.

Remediation Plan:

SOLVED: The HighStreetMarket team fixed the issue in the following commit IDs:

- 8f0b49cb90cbad93ed0586e6b05f54bac228e68b
  HighStreetPoolFactory.changePoolWeight() function can now be called by the owner of the contract or by a pool.
- 53dff6a43b47cdc21d83bb3b8232cbbc96e238b5
  HighStreetMarket team added an emergencyWithdraw() function that will allow users to retrieve their staked tokens and the yield rewards in case that factory.totalWeight() equals to zero.

3.4 (HAL-04) UPDATEHIGHPERBLOCK IMPACTS NEGATIVELY ALL THE UNCLAIMED REWARDS - MEDIUM

Description:

In the contract HighStreetPoolFactory the function updateHighPerBlock() decreases every blocksPerUpdate the highPerBlock rewards by a 3%. This could result, as shown in the Proof of Concept below, in a user losing a part of his reward by calling unstake() right after the execution of updateHighPerBlock().

Proof of Concept:

Based on a HighStreetPoolFactory contract with the following initial parameters:
- _highPerBlock = 10_000000000000000000
- _blocksPerUpdate = 50
- endBlockNumber - initBlockNumber = 200

Current yield block: 48/200
contract_poolFactory.highPerBlock() -> 10000000000000000000
contract_HighStreetCorePool_HighPool.pendingYieldRewards(user1.address)
-> 479999999999999999999

Call to updateHighPerBlock() - REVERTED: (too frequent)
Current yield block: 49/200
contract_poolFactory.highPerBlock() -> 10000000000000000000
contract_HighStreetCorePool_HighPool.pendingYieldRewards(user1.address)
-> 489999999999999999999

Call to updateHighPerBlock() - CONFIRMED
Current yield block: 50/200
contract_poolFactory.highPerBlock() -> 9700000000000000000
contract_HighStreetCorePool_HighPool.pendingYieldRewards(user1.address)
-> 484999999999999999999

As we can see, if the user calls unstake in the block 49/200 he will get a higher reward than if he called it in the block 50/200.

This happens because when updateHighPerBlock() is called it does not pay out the pending rewards to the users at the previous rate.

The difference can be really high as shown in the example below:
Based on a HighStreetPoolFactory contract with the following initial parameters:
- _highPerBlock = 10_000000000000000000
- _blocksPerUpdate = 1
- endBlockNumber - initBlockNumber = 40

Amount of reward tokens received by the user if he just calls unstake() after reaching endBlockNumber: 118284914959653117599
Amount of reward tokens received by the user after calling 40 times processRewards() generating 40 deposits and after retrieving those 40 deposits by calling unstake(): 1227719693740947062768

Difference:
227719693740947062768 - 118284914959653117599 = 109434778781293945169 = 109 reward tokens.

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

It is recommended to loop through the users generating a deposit with the previous rate that they can use to unstake the rewards at the previous rate. Although this solution may eventually require a lot of gas reaching the block gas limit and totally blocking the updateHighPerBlock() function.

Remediation Plan:

SOLVED: The HighStreetMarket team will mitigate this issue by:
- Incentivating their users to claim the rewards more frequently to maximize their profit by compound mechanism.
- Creating a script that executes automatically once before each updateHighPerBlock() call. This script will trigger the sync() function updating the yieldRewardsPerWeigh variable.

3.5 (HAL-05) MISSING REQUIRE STATEMENT IN UPDATESTAKELOCK FUNCTION - LOW

Description:

In the contract HighStreetPoolBase the function updateStakeLock() allows a user to extend the locking period for a given deposit. This function does not check that the tokenAmount is different from zero allowing users to extend deposits that were already unstaked, which makes no sense:

```
contract_HighStreetCorePool_HighPool.getDeposit(user1.address, 0) -> (0, 0, 0, 0, False)
>>> contract_HighStreetCorePool_HighPool.updateStakeLock(0, 1671611232, {'from': user1})
Transaction sent: 0xb0128385775c11350d3def57c382aa344e9c9cd7507fc7937af7ba6e90f3cfea
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 3
  HighStreetCorePool.updateStakeLock confirmed   Block: 13802206   Gas used: 180859 (2.69%)

<Transaction '0xb0128385775c11350d3def57c382aa344e9c9cd7507fc7937af7ba6e90f3cfea'>
>>> output.redd("contract_HighStreetCorePool_HighPool.getDeposit(user1.address, 0) -> " + str(contract_HighStreetCorePool_HighPool.getDeposit(user1.address, 0)))
contract_HighStreetCorePool_HighPool.getDeposit(user1.address, 0) -> (0, 0, 1640075337, 1671611232, False)
```

Code Location:

Listing 3: HighStreetPoolBase.sol

```
307 function updateStakeLock(
308     uint256 depositId,
309     uint64 lockedUntil
310 ) external nonReentrant {
311     // sync and call processRewards
312     _sync();
313     _processRewards(msg.sender, false);
314     // delegate call to an internal function
315     _updateStakeLock(msg.sender, depositId, lockedUntil);
316 }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to add a require statement in the updateStakeLock() function that checks that the tokenAmount in the deposit is > 0.

Remediation Plan:

SOLVED: The HighStreetMarket team fixed the issue in the commit ID f2562ad8f6e8cc39eb1ec17471368e7058b0df92

3.6 (HAL-06) USERS CAN STAKE IN A FLASHPOOL EVEN IF THE POOL IS DISABLED - LOW

Description:

The contract HighStreetFlashPool allows user to stake tokens even if the pool is disabled. This makes no sense as the users will not get any reward for staking their tokens in a disabled HighStreetFlashPool.

Code Location:

Listing 4: HighStreetFlashPool.sol

```
73 function _stake(
74     address _staker,
75     uint256 _amount,
76     uint64 _lockedUntil
77 ) internal override {
78     // override the `_lockedUntil` and execute parent
79     // we set "locked period" to 365 days only to have correct
           calculation of locking weights,
80     // the tokens are not really locked since _unstake in the core
           pool doesn't check the "locked period"
81     super._stake(_staker, _amount, uint64(now256() + 365 days));
82 }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to not allow staking if the HighStreetFlashPool is disabled:

Listing 5: HighStreetFlashPool.sol (Lines 78)

```
73 function _stake(
74     address _staker,
75     uint256 _amount,
76     uint64 _lockedUntil
77 ) internal override {
78     require (!isPoolDisabled(), "Pool disabled");
79     // override the `_lockedUntil` and execute parent
80     // we set "locked period" to 365 days only to have correct
           calculation of locking weights,
81     // the tokens are not really locked since _unstake in the core
           pool doesn't check the "locked period"
82     super._stake(_staker, _amount, uint64(now256() + 365 days));
83 }
```

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID dc058f113a6009585f7a47280c9b5cf2930a087b

3.7 (HAL-07) INCORRECT AMOUNT IN STAKED EVENT EMISSION - LOW

Description:

In the contract HighStreetPoolBase the event Stake will emit a wrong amount in case that the tokens staked are deflationary.

Code Location:

Listing 6: HighStreetPoolBase.sol (Lines 462)

```
394 function _stake(
395     address _staker,
396     uint256 _amount,
397     uint64 _lockUntil
398 ) internal virtual {
399     // validate the inputs
400     require(_amount > 0, "zero amount");
401     require(
402         _lockUntil == 0 || (_lockUntil > now256() && _lockUntil -
            now256() <= 365 days),
403         "invalid lock interval"
404     );
405
406     // update smart contract state
407     _sync();
408
409     // get a link to user data struct, we will write to it later
410     User storage user = users[_staker];
411     // process current pending rewards if any
412     if (user.tokenAmount > 0) {
413         _processRewards(_staker, false);
414     }
415
416     // in most of the cases added amount `addedAmount` is simply `
        _amount`
417     // however for deflationary tokens this can be different
418
419     // read the current balance
```

```
420     uint256 previousBalance = IERC20(poolToken).balanceOf(address(
            this));
421     // transfer `_amount`; note: some tokens may get burnt here
422     transferPoolTokenFrom(msg.sender, address(this), _amount);
423     // read new balance, usually this is just the difference `
            previousBalance - _amount`
424     uint256 newBalance = IERC20(poolToken).balanceOf(address(this)
            );
425     // calculate real amount taking into account deflation
426     uint256 addedAmount = newBalance - previousBalance;
427
428     // set the `lockFrom` and `lockUntil` taking into account that
429     // zero value for `_lockUntil` means "no locking" and leads to
             zero values
430     // for both `lockFrom` and `lockUntil`
431     uint64 lockFrom = _lockUntil > 0 ? uint64(now256()) : 0;
432     uint64 lockUntil = _lockUntil;
433
434     // stake weight formula rewards for locking
435     uint256 stakeWeight =
436         (((lockUntil - lockFrom) * WEIGHT_MULTIPLIER) / 365 days +
                WEIGHT_MULTIPLIER) * addedAmount;
437
438     // makes sure stakeWeight is valid
439     require(stakeWeight > 0, "invalid stakeWeight");
440
441     // create and save the deposit (append it to deposits array)
442     Deposit memory deposit =
443         Deposit({
444             tokenAmount: addedAmount,
445             weight: stakeWeight,
446             lockedFrom: lockFrom,
447             lockedUntil: lockUntil,
448             isYield: false
449         });
450     // deposit ID is an index of the deposit in `deposits` array
451     user.deposits.push(deposit);
452
453     // update user record
454     user.tokenAmount += addedAmount;
455     user.totalWeight += stakeWeight;
456     user.subYieldRewards = weightToReward(user.totalWeight,
            yieldRewardsPerWeight);
457
```

```
458     // update global variable
459     usersLockingWeight += stakeWeight;
460
461     // emit an event
462     emit Staked(msg.sender, _staker, _amount);
463 }
```

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

It is recommended to use addedAmount instead of _amount:

Listing 7

```
1 emit Staked(msg.sender, _staker, addedAmount);
```

Remediation Plan:

SOLVED: The HighStreetMarket team fixed the issue in the commit ID 3255a48e93ce0558de9ba33651f943b8f83b6b9f

3.8 (HAL-08) SOLC 0.8.0 COMPILER VERSION CONTAINS MULTIPLE BUGS - INFORMATIONAL

Description:

Solidity compiler version 0.8.3, 0.8.4 and 0.8.9 fixed important bugs in the compiler. The version 0.8.0 is missing all these fixes:

- 0.8.3
- 0.8.4
- 0.8.9

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use the most tested and stable versions, such as 0.6.12 or 0.7.6. Otherwise, if ^0.8.0 is still chosen to be used because of the new functionality it provides, it is recommended to use 0.8.10 version.

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID ec469c0e14949a991a5383ed836ba7dbf5534a7a

3.9 (HAL-09) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In the following contracts there are functions marked as public but they are never directly called within the same contract or in any of their descendants:

HighStreetPoolFactory.sol
- getPoolData() (HighStreetPoolFactory.sol#169-183)
- registerPool() (HighStreetPoolFactory.sol#209-227)

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

If the functions are not intended to be called internally or by their descendants, it is better to mark all of these functions as external to reduce gas costs.

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID c31e617d3aec0115855a50c8b39b8e31fbabe7c4

3.10 (HAL-10) UINT32/UINT64 TYPES ARE LESS GAS EFFICIENT - INFORMATIONAL

Description:

uint32, uint64... variables are less gas efficient than uint256. Due to how the EVM natively works on 256 bit numbers, using for example 32 bit number introduces additional costs as the EVM has to properly enforce the limits of this smaller type.

This happens in multiple functions across the different smart contracts. For example:

Listing 8: HighStreetPoolBase.sol

```
157 constructor(
158     address _high,
159     HighStreetPoolFactory _factory,
160     address _poolToken,
161     uint64 _initBlock,
162     uint32 _weight
163 )
```

In general, the usage of these smaller types only improves the gas costs in cases where the variables can be packed together, for example structs.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to make use of uint256 variables, for example, in state variables like HighStreetPoolFactory.lastRatioUpdate that are not

part/packed in any struct to reduce gas costs.

Remediation Plan:

SOLVED: The HighStreetMarket team fixed the issue in the commit ID 769e9773783a517a4389da5e602abe7595bb8454

3.11 (HAL-11) WRONG COMMENT - INFORMATIONAL

Description:

In the contract HighStreetPoolFactory, in the function changePoolWeight() the first comment describes that the function verifies that it is executed by the factory owner or by the pool itself when actually it only verifies and allows the pool to call it.

Listing 9: HighStreetPoolFactory.sol (Lines 272,273)

```
271 function changePoolWeight(address poolAddr, uint32 weight)
        external {
272     // verify function is executed either by factory owner or by
            the pool itself
273     require(poolExists[msg.sender], "access denied");
274
275     bool isFlashPool = IPool(poolAddr).isFlashPool();
276     // only flash pool can change weight
277     require(isFlashPool, "invalid poolAddr");
278
279     // recalculate total weight
280     totalWeight = totalWeight + weight - IPool(poolAddr).weight();
281
282     // set the new pool weight
283     IPool(poolAddr).setWeight(weight);
284
285     // emit an event
286     emit WeightUpdated(msg.sender, poolAddr, weight);
287 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to correct the comment in the code.

Remediation Plan:

SOLVED: The HighStreetMarket team fixed the issue in the commit ID 8f0b49cb90cbad93ed0586e6b05f54bac228e68b HighStreetPoolFactory.changePoolWeight() function can now be called by the owner of the contract or by a pool.

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' APIs across the entire code-base.

Slither results:

HighStreetCorePool.sol

```
Reentrancy in HighStreetCorePool._processRewards(address,bool) (contracts/HighStreetCorePool.sol#240-251):
        External calls:
        - _processVaultRewards(_staker) (contracts/HighStreetCorePool.sol#244)
                - SafeERC20.safeTransfer(IERC20(HIGH),_to,_value) (contracts/HighStreetCorePool.sol#287)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        - pendingYield = super._processRewards(_staker,_withUpdate) (contracts/HighStreetCorePool.sol#245)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        External calls sending eth:
        - _processVaultRewards(_staker) (contracts/HighStreetCorePool.sol#244)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        State variables written after the call(s):
        - poolTokenReserve += pendingYield (contracts/HighStreetCorePool.sol#249)
        - pendingYield = super._processRewards(_staker,_withUpdate) (contracts/HighStreetCorePool.sol#245)
                - user.deposits.push(newDeposit) (contracts/HighStreetPoolBase.sol#615)
                - user.tokenAmount += pendingYield (contracts/HighStreetPoolBase.sol#618)
                - user.totalWeight += depositWeight (contracts/HighStreetPoolBase.sol#619)
                - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#632)
Reentrancy in HighStreetCorePool._stake(address,uint256,uint64) (contracts/HighStreetCorePool.sol#203-213):
        External calls:
        - super._stake(_staker,_amount,_lockedUntil) (contracts/HighStreetCorePool.sol#208)
                - SafeERC20.safeTransferFrom(IERC20(poolToken),_from,_to,_value) (contracts/HighStreetPoolBase.sol#758)
                - SafeERC20.safeTransfer(IERC20(HIGH),_to,_value) (contracts/HighStreetCorePool.sol#287)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        External calls sending eth:
        - super._stake(_staker,_amount,_lockedUntil) (contracts/HighStreetCorePool.sol#208)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        State variables written after the call(s):
        - poolTokenReserve += _amount (contracts/HighStreetCorePool.sol#212)
        - user.subVaultRewards = weightToReward(user.totalWeight,vaultRewardsPerWeight) (contracts/HighStreetCorePool.sol#210)
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - SafeERC20.safeTransferFrom(IERC20(poolToken),_from,_to,_value) (contracts/HighStreetPoolBase.sol#758)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        State variables written after the call(s):
        - user.deposits.push(deposit) (contracts/HighStreetPoolBase.sol#451)
        - user.tokenAmount += addedAmount (contracts/HighStreetPoolBase.sol#454)
        - user.totalWeight += stakeWeight (contracts/HighStreetPoolBase.sol#455)
        - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#456)
        - usersLockingWeight += stakeWeight (contracts/HighStreetPoolBase.sol#459)
Reentrancy in HighStreetCorePool._unStake(address,uint256,uint256) (contracts/HighStreetCorePool.sol#221-232):
        External calls:
        - super._unstake(_staker,_depositId,_amount) (contracts/HighStreetCorePool.sol#230)
                - SafeERC20.safeTransfer(IERC20(HIGH),_to,_value) (contracts/HighStreetCorePool.sol#287)
                - SafeERC20.safeTransfer(IERC20(poolToken),_to,_value) (contracts/HighStreetPoolBase.sol#746)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
                - factory.mintYieldTo(msg.sender,_amount) (contracts/HighStreetPoolBase.sol#522)
        External calls sending eth:
        - super._unstake(_staker,_depositId,_amount) (contracts/HighStreetCorePool.sol#230)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        State variables written after the call(s):
        - user.subVaultRewards = weightToReward(user.totalWeight,vaultRewardsPerWeight) (contracts/HighStreetCorePool.sol#231)
Reentrancy in HighStreetCorePool.stakeAsPool(address,uint256) (contracts/HighStreetCorePool.sol#166-195):
        External calls:
        - _sync() (contracts/HighStreetCorePool.sol#170)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetCorePool.sol#173)
                - SafeERC20.safeTransfer(IERC20(HIGH),_to,_value) (contracts/HighStreetCorePool.sol#287)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        External calls sending eth:
        - _processRewards(_staker,false) (contracts/HighStreetCorePool.sol#173)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        State variables written after the call(s):
        - poolTokenReserve += _amount (contracts/HighStreetCorePool.sol#194)
        - user.tokenAmount += _amount (contracts/HighStreetCorePool.sol#184)
        - user.totalWeight += depositWeight (contracts/HighStreetCorePool.sol#185)
        - user.deposits.push(newDeposit) (contracts/HighStreetCorePool.sol#186)
        - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetCorePool.sol#190)
        - user.subVaultRewards = weightToReward(user.totalWeight,vaultRewardsPerWeight) (contracts/HighStreetCorePool.sol#191)
        - usersLockingWeight += depositWeight (contracts/HighStreetCorePool.sol#188)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```

```
HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637) uses a dangerous strict equality:
        - pendingYield == 0 (contracts/HighStreetPoolBase.sol#595)
HighStreetCorePool._processVaultRewards(address) (contracts/HighStreetCorePool.sol#258-278) uses a dangerous strict equality:
        - pendingVaultClaim == 0 (contracts/HighStreetCorePool.sol#261)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        State variables written after the call(s):
        - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#632)
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
                - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reentrancy in HighStreetPoolBase._unstake(address,uint256,uint256) (contracts/HighStreetPoolBase.sol#472-530):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#492)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
                - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - user.deposits.push(newDeposit) (contracts/HighStreetPoolBase.sol#615)
                - user.tokenAmount += pendingYield (contracts/HighStreetPoolBase.sol#618)
                - user.totalWeight += depositWeight (contracts/HighStreetPoolBase.sol#619)
                - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#632)
        - user.tokenAmount -= _amount (contracts/HighStreetPoolBase.sol#512)
        - user.totalWeight = user.totalWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#513)
        - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#514)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
        - usersLockingWeight = usersLockingWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#517)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reentrancy in HighStreetPoolBase.updateStakeLock(uint256,uint64) (contracts/HighStreetPoolBase.sol#307-316):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#312)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
                - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - _updateStakeLock(msg.sender,depositId,lockedUntil) (contracts/HighStreetPoolBase.sol#315)
                - user.totalWeight = user.totalWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#683)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
        - _updateStakeLock(msg.sender,depositId,lockedUntil) (contracts/HighStreetPoolBase.sol#315)
                - usersLockingWeight = usersLockingWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#684)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        State variables written after the call(s):
        - user.deposits.push(newDeposit) (contracts/HighStreetPoolBase.sol#615)
        - user.tokenAmount += pendingYield (contracts/HighStreetPoolBase.sol#618)
        - user.totalWeight += depositWeight (contracts/HighStreetPoolBase.sol#619)
Reentrancy in HighStreetPoolBase._sync() (contracts/HighStreetPoolBase.sol#538-573):
        External calls:
        - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        State variables written after the call(s):
        - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
        - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reentrancy in HighStreetCorePool.receiveVaultRewards(uint256) (contracts/HighStreetCorePool.sol#118-136):
        External calls:
        - transferHighTokenFrom(msg.sender,address(this),_rewardsAmount) (contracts/HighStreetCorePool.sol#126)
                - SafeERC20.safeTransferFrom(IERC20(HIGH),_from,_to,_value) (contracts/HighStreetCorePool.sol#301)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferHighTokenFrom(msg.sender,address(this),_rewardsAmount) (contracts/HighStreetCorePool.sol#126)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        State variables written after the call(s):
        - poolTokenReserve += _rewardsAmount (contracts/HighStreetCorePool.sol#132)
        - vaultRewardsPerWeight += rewardToWeight(_rewardsAmount,usersLockingWeight) (contracts/HighStreetCorePool.sol#128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in HighStreetCorePool._processRewards(address,bool) (contracts/HighStreetCorePool.sol#240-251):
        External calls:
        - _processVaultRewards(_staker) (contracts/HighStreetCorePool.sol#246)
                - SafeERC20.safeTransfer(IERC20(HIGH),_to,_value) (contracts/HighStreetCorePool.sol#287)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        - pendingYield = super._processRewards(_staker,_withUpdate) (contracts/HighStreetCorePool.sol#245)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        External calls sending eth:
        - _processVaultRewards(_staker) (contracts/HighStreetCorePool.sol#246)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
                - pendingYield = super._processRewards(_staker,_withUpdate) (contracts/HighStreetCorePool.sol#245)
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
                - pendingYield = super._processRewards(_staker,_withUpdate) (contracts/HighStreetCorePool.sol#245)
Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
Reentrancy in HighStreetCorePool._processVaultRewards(address) (contracts/HighStreetCorePool.sol#258-278):
        External calls:
        - transferHighToken(_staker,pendingVaultClaim) (contracts/HighStreetCorePool.sol#275)
                - SafeERC20.safeTransfer(IERC20(HIGH),_to,_value) (contracts/HighStreetCorePool.sol#287)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferHighToken(_staker,pendingVaultClaim) (contracts/HighStreetCorePool.sol#275)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - VaultRewardsClaimed(msg.sender,_staker,pendingVaultClaim) (contracts/HighStreetCorePool.sol#277)
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - SafeERC20.safeTransferFrom(IERC20(poolToken),_from,_to,_value) (contracts/HighStreetPoolBase.sol#758)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - Staked(msg.sender,_staker,_amount) (contracts/HighStreetPoolBase.sol#462)
Reentrancy in HighStreetPoolBase._sync() (contracts/HighStreetPoolBase.sol#538-573):
        External calls:
        - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        Event emitted after the call(s):
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
```

```
Reentrancy in HighStreetPoolBase._unstake(address,uint256,uint256) (contracts/HighStreetPoolBase.sol#472-530):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#492)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
Reentrancy in HighStreetPoolBase._unstake(address,uint256,uint256) (contracts/HighStreetPoolBase.sol#472-530):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#492)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        - factory.mintYieldTo(msg.sender,_amount) (contracts/HighStreetPoolBase.sol#522)
        - transferPoolToken(msg.sender,_amount) (contracts/HighStreetPoolBase.sol#525)
                - SafeERC20.safeTransfer(IERC20(poolToken),_to,_value) (contracts/HighStreetPoolBase.sol#746)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferPoolToken(msg.sender,_amount) (contracts/HighStreetPoolBase.sol#525)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - Unstaked(msg.sender,_staker,_amount) (contracts/HighStreetPoolBase.sol#529)
Reentrancy in HighStreetPoolFactory.changePoolWeight(address,uint32) (contracts/HighStreetPoolFactory.sol#271-287):
        External calls:
        - IPool(poolAddr).setWeight(weight) (contracts/HighStreetPoolFactory.sol#283)
        Event emitted after the call(s):
        - WeightUpdated(msg.sender,poolAddr,weight) (contracts/HighStreetPoolFactory.sol#286)
Reentrancy in HighStreetCorePool.receiveVaultRewards(uint256) (contracts/HighStreetCorePool.sol#118-136):
        External calls:
        - transferHighTokenFrom(msg.sender,address(this),_rewardsAmount) (contracts/HighStreetCorePool.sol#126)
                - SafeERC20.safeTransferFrom(IERC20(HIGH),_from,_to,_value) (contracts/HighStreetCorePool.sol#301)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferHighTokenFrom(msg.sender,address(this),_rewardsAmount) (contracts/HighStreetCorePool.sol#126)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - VaultRewardsReceived(msg.sender,_rewardsAmount) (contracts/HighStreetCorePool.sol#135)
Reentrancy in HighStreetCorePool.stakeAsPool(address,uint256) (contracts/HighStreetCorePool.sol#166-195):
        External calls:
        - _sync() (contracts/HighStreetCorePool.sol#170)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetCorePool.sol#173)
                - SafeERC20.safeTransfer(IERC20(HIGH),_to,_value) (contracts/HighStreetCorePool.sol#287)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        External calls sending eth:
        - _processRewards(_staker,false) (contracts/HighStreetCorePool.sol#173)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
                - _processRewards(_staker,false) (contracts/HighStreetCorePool.sol#173)
        - VaultRewardsClaimed(msg.sender,_staker,pendingVaultClaim) (contracts/HighStreetCorePool.sol#277)
                - _processRewards(_staker,false) (contracts/HighStreetCorePool.sol#173)

        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
                - _processRewards(_staker,false) (contracts/HighStreetCorePool.sol#173)
Reentrancy in HighStreetPoolBase.updateStakeLock(uint256,uint64) (contracts/HighStreetPoolBase.sol#307-316):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#312)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - StakeLockUpdated(_staker,_depositId,stakeDeposit.lockedFrom,_lockedUntil) (contracts/HighStreetPoolBase.sol#687)
                - _updateStakeLock(msg.sender,depositId,lockedUntil) (contracts/HighStreetPoolBase.sol#315)
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
                - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
                - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

HighStreetCorePool.receiveVaultRewards(uint256) (contracts/HighStreetCorePool.sol#118-136) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(usersLockingWeight > 0,zero locking weight) (contracts/HighStreetCorePool.sol#124)
HighStreetCorePool._unstake(address,uint256,uint256) (contracts/HighStreetCorePool.sol#221-232) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(stakeDeposit.lockedFrom == 0 || now256() > stakeDeposit.lockedUntil,deposit not yet unlocked) (contracts/HighStreetCorePool.sol#228)
HighStreetCorePool._processVaultRewards(address) (contracts/HighStreetCorePool.sol#258-278) uses timestamp for comparisons
        Dangerous comparisons:
        - pendingVaultClaim == 0 (contracts/HighStreetCorePool.sol#261)
        - require(bool,string)(highBalance >= pendingVaultClaim,contract HIGH balance too low) (contracts/HighStreetCorePool.sol#264)
        - pendingVaultClaim > poolTokenReserve (contracts/HighStreetCorePool.sol#266)
HighStreetPoolBase.pendingYieldRewards(address) (contracts/HighStreetPoolBase.sol#193-217) uses timestamp for comparisons
        Dangerous comparisons:
        - blockNumber() > lastYieldDistribution && usersLockingWeight != 0 (contracts/HighStreetPoolBase.sol#199)
HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lockUntil == 0 || (_lockUntil > now256() && _lockUntil - now256() <= 31536000),invalid lock interval) (contracts/HighStreetPoolBase.sol#401-404)
        - require(bool,string)(stakeWeight > 0,invalid stakeWeight) (contracts/HighStreetPoolBase.sol#439)
HighStreetPoolBase._sync() (contracts/HighStreetPoolBase.sol#538-573) uses timestamp for comparisons
        Dangerous comparisons:
        - usersLockingWeight == 0 (contracts/HighStreetPoolBase.sol#554)
HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637) uses timestamp for comparisons
        Dangerous comparisons:
        - pendingYield == 0 (contracts/HighStreetPoolBase.sol#595)
HighStreetPoolBase._updateStakeLock(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#646-688) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lockedUntil > now256(),lock should be in the future) (contracts/HighStreetPoolBase.sol#652)
        - require(bool,string)(_lockedUntil > stakeDeposit.lockedUntil,invalid new lock) (contracts/HighStreetPoolBase.sol#660)
        - require(bool,string)(_lockedUntil - now256() <= 31536000,max lock period is 365 days) (contracts/HighStreetPoolBase.sol#664)
        - require(bool,string)(_lockedUntil - stakeDeposit.lockedFrom <= 31536000,max lock period is 365 days) (contracts/HighStreetPoolBase.sol#667)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.5.1', '^0.5.0']
        - ^0.5.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.5.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#3)
        - ^0.5.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.5.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3)
        - ^0.5.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.5.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - 0.5.1 (contracts/HighStreetCorePool.sol#3)
        - 0.5.1 (contracts/HighStreetPoolBase.sol#2)
        - 0.5.1 (contracts/HighStreetPoolFactory.sol#2)
        - 0.5.1 (contracts/interfaces/ICorePool.sol#3)
        - 0.5.1 (contracts/interfaces/IPool.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#108-114) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-70) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#59-66) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.5.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.5.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.5.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.5.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.5.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.5.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.5.1 (contracts/HighStreetCorePool.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.5.1 (contracts/HighStreetPoolBase.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.5.1 (contracts/HighStreetPoolFactory.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.5.1 (contracts/interfaces/ICorePool.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.5.1 (contracts/interfaces/IPool.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.5.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter HighStreetCorePool.pendingVaultRewards(address)._staker (contracts/HighStreetCorePool.sol#85) is not in mixedCase
Parameter HighStreetCorePool.setVault(address)._vault (contracts/HighStreetCorePool.sol#96) is not in mixedCase
Parameter HighStreetCorePool.receiveVaultRewards(uint256)._rewardsAmount (contracts/HighStreetCorePool.sol#118) is not in mixedCase
Parameter HighStreetCorePool.stakeAsPool(address,uint256)._staker (contracts/HighStreetCorePool.sol#166) is not in mixedCase
Parameter HighStreetCorePool.stakeAsPool(address,uint256)._amount (contracts/HighStreetCorePool.sol#166) is not in mixedCase
Parameter HighStreetCorePool.transferHighToken(address,uint256)._to (contracts/HighStreetCorePool.sol#285) is not in mixedCase
Parameter HighStreetCorePool.transferHighToken(address,uint256)._value (contracts/HighStreetCorePool.sol#285) is not in mixedCase
Parameter HighStreetCorePool.transferHighTokenFrom(address,address,uint256)._from (contracts/HighStreetCorePool.sol#296) is not in mixedCase
Parameter HighStreetCorePool.transferHighTokenFrom(address,address,uint256)._to (contracts/HighStreetCorePool.sol#297) is not in mixedCase
Parameter HighStreetCorePool.transferHighTokenFrom(address,address,uint256)._value (contracts/HighStreetCorePool.sol#298) is not in mixedCase
Constant HighStreetCorePool.isFlashPool (contracts/HighStreetCorePool.sol#18) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter HighStreetPoolBase.pendingYieldRewards(address)._staker (contracts/HighStreetPoolBase.sol#193) is not in mixedCase
Parameter HighStreetPoolBase.balanceOf(address)._user (contracts/HighStreetPoolBase.sol#225) is not in mixedCase
Parameter HighStreetPoolBase.getDeposit(address,uint256)._user (contracts/HighStreetPoolBase.sol#239) is not in mixedCase
Parameter HighStreetPoolBase.getDeposit(address,uint256)._depositId (contracts/HighStreetPoolBase.sol#239) is not in mixedCase
Parameter HighStreetPoolBase.getDepositsLength(address)._user (contracts/HighStreetPoolBase.sol#252) is not in mixedCase
Parameter HighStreetPoolBase.stake(uint256,uint64)._amount (contracts/HighStreetPoolBase.sol#269) is not in mixedCase
Parameter HighStreetPoolBase.stake(uint256,uint64)._lockUntil (contracts/HighStreetPoolBase.sol#270) is not in mixedCase
Parameter HighStreetPoolBase.unstake(uint256,uint256)._depositId (contracts/HighStreetPoolBase.sol#287) is not in mixedCase
Parameter HighStreetPoolBase.unstake(uint256,uint256)._amount (contracts/HighStreetPoolBase.sol#288) is not in mixedCase
Parameter HighStreetPoolBase.setWeight(uint32)._weight (contracts/HighStreetPoolBase.sol#360) is not in mixedCase
Parameter HighStreetPoolBase.weightToReward(uint256,uint256)._weight (contracts/HighStreetPoolBase.sol#690) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolToken(address,uint256)._to (contracts/HighStreetPoolBase.sol#745) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolToken(address,uint256)._value (contracts/HighStreetPoolBase.sol#745) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolTokenFrom(address,address,uint256)._from (contracts/HighStreetPoolBase.sol#754) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolTokenFrom(address,address,uint256)._to (contracts/HighStreetPoolBase.sol#755) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolTokenFrom(address,address,uint256)._value (contracts/HighStreetPoolBase.sol#756) is not in mixedCase
Variable HighStreetPoolBase.HIGH (contracts/HighStreetPoolBase.sol#45) is not in mixedCase
Parameter HighStreetPoolFactory.getPoolData(address)._poolToken (contracts/HighStreetPoolFactory.sol#169) is not in mixedCase
Parameter HighStreetPoolFactory.mintYieldTo(address,uint256)._to (contracts/HighStreetPoolFactory.sol#256) is not in mixedCase
Parameter HighStreetPoolFactory.mintYieldTo(address,uint256)._amount (contracts/HighStreetPoolFactory.sol#256) is not in mixedCase
Parameter HighStreetPoolFactory.transferHighToken(address,uint256)._to (contracts/HighStreetPoolFactory.sol#304) is not in mixedCase
Parameter HighStreetPoolFactory.transferHighToken(address,uint256)._value (contracts/HighStreetPoolFactory.sol#304) is not in mixedCase
Variable HighStreetPoolFactory.HIGH (contracts/HighStreetPoolFactory.sol#32) is not in mixedCase
Function IPool.HIGH() (contracts/interfaces/IPool.sol#20) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
getPoolData(address) should be declared external:
        - HighStreetPoolFactory.getPoolData(address) (contracts/HighStreetPoolFactory.sol#169-183)
registerPool(address) should be declared external:
        - HighStreetPoolFactory.registerPool(address) (contracts/HighStreetPoolFactory.sol#209-227)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

HighStreetFlashPool.sol

```
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - SafeERC20.safeTransferFrom(IERC20(poolToken),_from,_to,_value) (contracts/HighStreetPoolBase.sol#758)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        State variables written after the call(s):
        - user.deposits.push(deposit) (contracts/HighStreetPoolBase.sol#451)
        - user.tokenAmount += addedAmount (contracts/HighStreetPoolBase.sol#454)
        - user.totalWeight += stakeWeight (contracts/HighStreetPoolBase.sol#455)
        - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#456)
        - usersLockingWeight += stakeWeight (contracts/HighStreetPoolBase.sol#459)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637) uses a dangerous strict equality:
        - pendingYield == 0 (contracts/HighStreetPoolBase.sol#595)
HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463) uses a dangerous strict equality:
        - require(bool,string)(_lockUntil == 0 || (_lockUntil > now256() && _lockUntil - now256() <= 31536000),invalid lock interval) (contracts/HighStreetPoolBase.sol#401-404)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        State variables written after the call(s):
        - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#632)
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
                - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reentrancy in HighStreetPoolBase._unstake(address,uint256,uint256) (contracts/HighStreetPoolBase.sol#472-530):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#492)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
                - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - user.deposits.push(newDeposit) (contracts/HighStreetPoolBase.sol#615)
                - user.tokenAmount += pendingYield (contracts/HighStreetPoolBase.sol#618)
                - user.totalWeight += depositWeight (contracts/HighStreetPoolBase.sol#619)
                - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#632)
        - user.tokenAmount -= _amount (contracts/HighStreetPoolBase.sol#512)
        - user.totalWeight = user.totalWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#513)
        - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#514)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
        - usersLockingWeight = usersLockingWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#517)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reentrancy in HighStreetPoolBase.updateStakeLock(uint256,uint64) (contracts/HighStreetPoolBase.sol#307-316):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#312)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
                - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - _updateStakeLock(msg.sender,depositId,lockedUntil) (contracts/HighStreetPoolBase.sol#315)
                - user.totalWeight = user.totalWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#683)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
        - _updateStakeLock(msg.sender,depositId,lockedUntil) (contracts/HighStreetPoolBase.sol#315)
                - usersLockingWeight = usersLockingWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#684)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        State variables written after the call(s):
        - user.deposits.push(newDeposit) (contracts/HighStreetPoolBase.sol#615)
        - user.tokenAmount += pendingYield (contracts/HighStreetPoolBase.sol#618)
        - user.totalWeight += depositWeight (contracts/HighStreetPoolBase.sol#619)
Reentrancy in HighStreetPoolBase._sync() (contracts/HighStreetPoolBase.sol#538-573):
        External calls:
        - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        State variables written after the call(s):
        - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
        - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - SafeERC20.safeTransferFrom(IERC20(poolToken),_from,_to,_value) (contracts/HighStreetPoolBase.sol#758)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - Staked(msg.sender,_staker,_amount) (contracts/HighStreetPoolBase.sol#462)
Reentrancy in HighStreetPoolBase._sync() (contracts/HighStreetPoolBase.sol#538-573):
        External calls:
        - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        Event emitted after the call(s):
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
Reentrancy in HighStreetPoolBase._unstake(address,uint256,uint256) (contracts/HighStreetPoolBase.sol#472-530):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#492)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
Reentrancy in HighStreetPoolBase._unstake(address,uint256,uint256) (contracts/HighStreetPoolBase.sol#472-530):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#492)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        - factory.mintYieldTo(msg.sender,_amount) (contracts/HighStreetPoolBase.sol#522)
        - transferPoolToken(msg.sender,_amount) (contracts/HighStreetPoolBase.sol#525)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - SafeERC20.safeTransfer(IERC20(poolToken),_to,_value) (contracts/HighStreetPoolBase.sol#746)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferPoolToken(msg.sender,_amount) (contracts/HighStreetPoolBase.sol#525)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - Unstaked(msg.sender,_staker,_amount) (contracts/HighStreetPoolBase.sol#529)
Reentrancy in HighStreetPoolFactory.changePoolWeight(address,uint32) (contracts/HighStreetPoolFactory.sol#271-287):
        External calls:
        - IPool(poolAddr).setWeight(weight) (contracts/HighStreetPoolFactory.sol#283)
        Event emitted after the call(s):
        - WeightUpdated(msg.sender,poolAddr,weight) (contracts/HighStreetPoolFactory.sol#286)
Reentrancy in HighStreetPoolBase.updateStakeLock(uint256,uint64) (contracts/HighStreetPoolBase.sol#307-316):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#312)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - StakeLockUpdated(_staker,_depositId,stakeDeposit.lockedFrom,_lockedUntil) (contracts/HighStreetPoolBase.sol#687)
                - _updateStakeLock(msg.sender,depositId,lockedUntil) (contracts/HighStreetPoolBase.sol#315)
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
                - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
                - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

HighStreetPoolBase.pendingYieldRewards(address) (contracts/HighStreetPoolBase.sol#193-217) uses timestamp for comparisons
        Dangerous comparisons:
        - blockNumber() > lastYieldDistribution && usersLockingWeight != 0 (contracts/HighStreetPoolBase.sol#199)
HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lockUntil == 0 || (_lockUntil > now256() && _lockUntil - now256() <= 31536000),invalid lock interval) (contracts/HighStreetPoolBase.sol#401-404)
        - require(bool,string)(stakeWeight > 0,invalid stakeWeight) (contracts/HighStreetPoolBase.sol#439)
HighStreetPoolBase._sync() (contracts/HighStreetPoolBase.sol#538-573) uses timestamp for comparisons
        Dangerous comparisons:
        - usersLockingWeight == 0 (contracts/HighStreetPoolBase.sol#554)
HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637) uses timestamp for comparisons
        Dangerous comparisons:
        - pendingYield == 0 (contracts/HighStreetPoolBase.sol#595)
HighStreetPoolBase._updateStakeLock(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#646-688) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lockedUntil > now256(),lock should be in the future) (contracts/HighStreetPoolBase.sol#652)
        - require(bool,string)(_lockedUntil > stakeDeposit.lockedUntil,invalid new lock) (contracts/HighStreetPoolBase.sol#660)
        - require(bool,string)(_lockedUntil - now256() <= 31536000,max lock period is 365 days) (contracts/HighStreetPoolBase.sol#664)
        - require(bool,string)(_lockedUntil - stakeDeposit.lockedFrom <= 31536000,max lock period is 365 days) (contracts/HighStreetPoolBase.sol#667)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.8.1', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - 0.8.1 (contracts/HighStreetFlashPool.sol#3)
        - 0.8.1 (contracts/HighStreetPoolBase.sol#2)
        - 0.8.1 (contracts/HighStreetPoolFactory.sol#2)
        - 0.8.1 (contracts/interfaces/ICorePool.sol#3)
        - 0.8.1 (contracts/interfaces/IPool.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#108-114) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-79) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#59-66) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (contracts/HighStreetFlashPool.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (contracts/HighStreetPoolBase.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (contracts/HighStreetPoolFactory.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (contracts/interfaces/ICorePool.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (contracts/interfaces/IPool.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Constant HighStreetFlashPool.isFlashPool (contracts/HighStreetFlashPool.sol#25) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter HighStreetPoolBase.pendingYieldRewards(address)._staker (contracts/HighStreetPoolBase.sol#193) is not in mixedCase
Parameter HighStreetPoolBase.balanceOf(address)._user (contracts/HighStreetPoolBase.sol#225) is not in mixedCase
Parameter HighStreetPoolBase.getDeposit(address,uint256)._user (contracts/HighStreetPoolBase.sol#239) is not in mixedCase
Parameter HighStreetPoolBase.getDeposit(address,uint256)._depositId (contracts/HighStreetPoolBase.sol#239) is not in mixedCase
Parameter HighStreetPoolBase.getDepositsLength(address)._user (contracts/HighStreetPoolBase.sol#252) is not in mixedCase
Parameter HighStreetPoolBase.stake(uint256,uint64)._amount (contracts/HighStreetPoolBase.sol#269) is not in mixedCase
Parameter HighStreetPoolBase.stake(uint256,uint64)._lockUntil (contracts/HighStreetPoolBase.sol#270) is not in mixedCase
Parameter HighStreetPoolBase.unstake(uint256,uint256)._depositId (contracts/HighStreetPoolBase.sol#287) is not in mixedCase
Parameter HighStreetPoolBase.unstake(uint256,uint256)._amount (contracts/HighStreetPoolBase.sol#288) is not in mixedCase
Parameter HighStreetPoolBase.setWeight(uint32)._weight (contracts/HighStreetPoolBase.sol#360) is not in mixedCase
Parameter HighStreetPoolBase.weightToReward(uint256,uint256)._weight (contracts/HighStreetPoolBase.sol#698) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolToken(address,uint256)._to (contracts/HighStreetPoolBase.sol#745) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolToken(address,uint256)._value (contracts/HighStreetPoolBase.sol#745) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolTokenFrom(address,address,uint256)._from (contracts/HighStreetPoolBase.sol#754) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolTokenFrom(address,address,uint256)._to (contracts/HighStreetPoolBase.sol#755) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolTokenFrom(address,address,uint256)._value (contracts/HighStreetPoolBase.sol#756) is not in mixedCase
Variable HighStreetPoolBase.HIGH (contracts/HighStreetPoolBase.sol#45) is not in mixedCase
Parameter HighStreetPoolFactory.getPoolData(address)._poolToken (contracts/HighStreetPoolFactory.sol#169) is not in mixedCase
Parameter HighStreetPoolFactory.mintYieldTo(address,uint256)._to (contracts/HighStreetPoolFactory.sol#256) is not in mixedCase
Parameter HighStreetPoolFactory.mintYieldTo(address,uint256)._amount (contracts/HighStreetPoolFactory.sol#256) is not in mixedCase
Parameter HighStreetPoolFactory.transferHighToken(address,uint256)._to (contracts/HighStreetPoolFactory.sol#304) is not in mixedCase
Parameter HighStreetPoolFactory.transferHighToken(address,uint256)._value (contracts/HighStreetPoolFactory.sol#304) is not in mixedCase
Variable HighStreetPoolFactory.HIGH (contracts/HighStreetPoolFactory.sol#32) is not in mixedCase
Function IPool.HIGH() (contracts/interfaces/IPool.sol#29) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
getPoolData(address) should be declared external:
        - HighStreetPoolFactory.getPoolData(address) (contracts/HighStreetPoolFactory.sol#169-183)
registerPool(address) should be declared external:
        - HighStreetPoolFactory.registerPool(address) (contracts/HighStreetPoolFactory.sol#209-227)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

HighStreetPoolBase.sol

```
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - SafeERC20.safeTransferFrom(IERC20(poolToken),_from,_to,_value) (contracts/HighStreetPoolBase.sol#758)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        State variables written after the call(s):
        - user.deposits.push(deposit) (contracts/HighStreetPoolBase.sol#451)
        - user.tokenAmount += addedAmount (contracts/HighStreetPoolBase.sol#454)
        - user.totalWeight += stakeWeight (contracts/HighStreetPoolBase.sol#455)
        - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#456)
        - usersLockingWeight += stakeWeight (contracts/HighStreetPoolBase.sol#459)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637) uses a dangerous strict equality:
        - pendingYield == 0 (contracts/HighStreetPoolBase.sol#598)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        State variables written after the call(s):
        - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#632)
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
                - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reentrancy in HighStreetPoolBase._unstake(address,uint256,uint256) (contracts/HighStreetPoolBase.sol#472-530):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#492)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
                - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - user.deposits.push(newDeposit) (contracts/HighStreetPoolBase.sol#615)
                - user.tokenAmount += pendingYield (contracts/HighStreetPoolBase.sol#618)
                - user.totalWeight += depositWeight (contracts/HighStreetPoolBase.sol#619)
                - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#632)
        - user.tokenAmount -= _amount (contracts/HighStreetPoolBase.sol#512)
        - user.totalWeight = user.totalWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#513)
        - user.subYieldRewards = weightToReward(user.totalWeight,yieldRewardsPerWeight) (contracts/HighStreetPoolBase.sol#514)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
        - usersLockingWeight = usersLockingWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#517)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reentrancy in HighStreetPoolBase.updateStakeLock(uint256,uint64) (contracts/HighStreetPoolBase.sol#307-316):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#312)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        State variables written after the call(s):
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
                - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - _updateStakeLock(msg.sender,depositId,lockedUntil) (contracts/HighStreetPoolBase.sol#315)
                - user.totalWeight = user.totalWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#683)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - usersLockingWeight += depositWeight (contracts/HighStreetPoolBase.sol#622)
        - _updateStakeLock(msg.sender,depositId,lockedUntil) (contracts/HighStreetPoolBase.sol#315)
                - usersLockingWeight = usersLockingWeight - previousWeight + newWeight (contracts/HighStreetPoolBase.sol#684)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        State variables written after the call(s):
        - user.deposits.push(newDeposit) (contracts/HighStreetPoolBase.sol#615)
        - user.tokenAmount += pendingYield (contracts/HighStreetPoolBase.sol#618)
        - user.totalWeight += depositWeight (contracts/HighStreetPoolBase.sol#619)
Reentrancy in HighStreetPoolBase._sync() (contracts/HighStreetPoolBase.sol#538-573):
        External calls:
        - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        State variables written after the call(s):
        - lastYieldDistribution = uint64(blockNumber()) (contracts/HighStreetPoolBase.sol#555)
        - lastYieldDistribution = uint64(currentBlock) (contracts/HighStreetPoolBase.sol#569)
        - yieldRewardsPerWeight += rewardToWeight(highReward,usersLockingWeight) (contracts/HighStreetPoolBase.sol#568)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Reentrancy in HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#588)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
Reentrancy in HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#407)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#413)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - SafeERC20.safeTransferFrom(IERC20(poolToken),_from,_to,_value) (contracts/HighStreetPoolBase.sol#758)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferPoolTokenFrom(msg.sender,address(this),_amount) (contracts/HighStreetPoolBase.sol#422)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - Staked(msg.sender,_staker,_amount) (contracts/HighStreetPoolBase.sol#462)
Reentrancy in HighStreetPoolBase._sync() (contracts/HighStreetPoolBase.sol#538-573):
        External calls:
        - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        Event emitted after the call(s):
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
Reentrancy in HighStreetPoolBase._unstake(address,uint256,uint256) (contracts/HighStreetPoolBase.sol#472-530):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#492)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
                - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
Reentrancy in HighStreetPoolBase._unstake(address,uint256,uint256) (contracts/HighStreetPoolBase.sol#472-530):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#492)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(_staker,false) (contracts/HighStreetPoolBase.sol#494)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        - factory.mintYieldTo(msg.sender,_amount) (contracts/HighStreetPoolBase.sol#522)
        - transferPoolToken(msg.sender,_amount) (contracts/HighStreetPoolBase.sol#525)
                - SafeERC20.safeTransfer(IERC20(poolToken),_to,_value) (contracts/HighStreetPoolBase.sol#746)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        External calls sending eth:
        - transferPoolToken(msg.sender,_amount) (contracts/HighStreetPoolBase.sol#525)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
        Event emitted after the call(s):
        - Unstaked(msg.sender,_staker,_amount) (contracts/HighStreetPoolBase.sol#529)
Reentrancy in HighStreetPoolFactory.changePoolWeight(address,uint32) (contracts/HighStreetPoolFactory.sol#271-287):
        External calls:
        - IPool(poolAddr).setWeight(weight) (contracts/HighStreetPoolFactory.sol#283)
        Event emitted after the call(s):
        - WeightUpdated(msg.sender,poolAddr,weight) (contracts/HighStreetPoolFactory.sol#286)
Reentrancy in HighStreetPoolBase.updateStakeLock(uint256,uint64) (contracts/HighStreetPoolBase.sol#307-316):
        External calls:
        - _sync() (contracts/HighStreetPoolBase.sol#312)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
        - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
                - factory.updateHighPerBlock() (contracts/HighStreetPoolBase.sol#541)
                - ICorePool(highPool).stakeAsPool(_staker,pendingYield) (contracts/HighStreetPoolBase.sol#627)
        Event emitted after the call(s):
        - StakeLockUpdated(_staker,_depositId,stakeDeposit.lockedFrom,_lockedUntil) (contracts/HighStreetPoolBase.sol#687)
                - _updateStakeLock(msg.sender,depositId,lockedUntil) (contracts/HighStreetPoolBase.sol#315)
        - Synchronized(msg.sender,yieldRewardsPerWeight,lastYieldDistribution) (contracts/HighStreetPoolBase.sol#572)
                - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
        - YieldClaimed(msg.sender,_staker,pendingYield) (contracts/HighStreetPoolBase.sol#636)
                - _processRewards(msg.sender,false) (contracts/HighStreetPoolBase.sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

HighStreetPoolBase.pendingYieldRewards(address) (contracts/HighStreetPoolBase.sol#193-217) uses timestamp for comparisons
        Dangerous comparisons:
        - blockNumber() > lastYieldDistribution && usersLockingWeight != 0 (contracts/HighStreetPoolBase.sol#199)
HighStreetPoolBase._stake(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#394-463) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lockUntil == 0 || (_lockUntil > now256() && _lockUntil - now256() <= 31536000),invalid lock interval) (contracts/HighStreetPoolBase.sol#401-404)
        - require(bool,string)(stakeWeight > 0,invalid stakeWeight) (contracts/HighStreetPoolBase.sol#439)
HighStreetPoolBase._sync() (contracts/HighStreetPoolBase.sol#538-573) uses timestamp for comparisons
        Dangerous comparisons:
        - usersLockingWeight == 0 (contracts/HighStreetPoolBase.sol#554)
HighStreetPoolBase._processRewards(address,bool) (contracts/HighStreetPoolBase.sol#582-637) uses timestamp for comparisons
        Dangerous comparisons:
        - pendingYield == 0 (contracts/HighStreetPoolBase.sol#595)
HighStreetPoolBase._updateStakeLock(address,uint256,uint64) (contracts/HighStreetPoolBase.sol#646-688) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lockedUntil > now256(),lock should be in the future) (contracts/HighStreetPoolBase.sol#652)
        - require(bool,string)(_lockedUntil > stakeDeposit.lockedUntil,invalid new lock) (contracts/HighStreetPoolBase.sol#660)
        - require(bool,string)(_lockedUntil - now256() <= 31536000,max lock period is 365 days) (contracts/HighStreetPoolBase.sol#664)
        - require(bool,string)(_lockedUntil - stakeDeposit.lockedFrom <= 31536000,max lock period is 365 days) (contracts/HighStreetPoolBase.sol#667)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.1', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - 0.8.1 (contracts/HighStreetPoolBase.sol#2)
        - 0.8.1 (contracts/HighStreetPoolFactory.sol#2)
        - 0.8.1 (contracts/interfaces/ICorePool.sol#3)
        - 0.8.1 (contracts/interfaces/IPool.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#108-114) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#68-79) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#59-66) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (contracts/HighStreetPoolBase.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (contracts/HighStreetPoolFactory.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (contracts/interfaces/ICorePool.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (contracts/interfaces/IPool.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter HighStreetPoolBase.pendingYieldRewards(address)._staker (contracts/HighStreetPoolBase.sol#193) is not in mixedCase
Parameter HighStreetPoolBase.balanceOf(address)._user (contracts/HighStreetPoolBase.sol#225) is not in mixedCase
Parameter HighStreetPoolBase.getDeposit(address,uint256)._user (contracts/HighStreetPoolBase.sol#239) is not in mixedCase
Parameter HighStreetPoolBase.getDeposit(address,uint256)._depositId (contracts/HighStreetPoolBase.sol#239) is not in mixedCase
Parameter HighStreetPoolBase.getDepositsLength(address)._user (contracts/HighStreetPoolBase.sol#252) is not in mixedCase
Parameter HighStreetPoolBase.stake(uint256,uint64)._amount (contracts/HighStreetPoolBase.sol#269) is not in mixedCase
Parameter HighStreetPoolBase.stake(uint256,uint64)._lockUntil (contracts/HighStreetPoolBase.sol#270) is not in mixedCase
Parameter HighStreetPoolBase.unstake(uint256,uint256)._depositId (contracts/HighStreetPoolBase.sol#287) is not in mixedCase
Parameter HighStreetPoolBase.unstake(uint256,uint256)._amount (contracts/HighStreetPoolBase.sol#288) is not in mixedCase
Parameter HighStreetPoolBase.setWeight(uint32)._weight (contracts/HighStreetPoolBase.sol#360) is not in mixedCase
Parameter HighStreetPoolBase.weightToReward(uint256,uint256)._weight (contracts/HighStreetPoolBase.sol#698) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolToken(address,uint256)._to (contracts/HighStreetPoolBase.sol#745) is not in mixedCase
```

AUTOMATED TESTING

```
Parameter HighStreetPoolBase.transferPoolToken(address,uint256)._value (contracts/HighStreetPoolBase.sol#745) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolTokenFrom(address,address,uint256)._from (contracts/HighStreetPoolBase.sol#754) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolTokenFrom(address,address,uint256)._to (contracts/HighStreetPoolBase.sol#755) is not in mixedCase
Parameter HighStreetPoolBase.transferPoolTokenFrom(address,address,uint256)._value (contracts/HighStreetPoolBase.sol#756) is not in mixedCase
Variable HighStreetPoolBase.HIGH (contracts/HighStreetPoolBase.sol#45) is not in mixedCase
Parameter HighStreetPoolFactory.getPoolData(address)._poolToken (contracts/HighStreetPoolFactory.sol#169) is not in mixedCase
Parameter HighStreetPoolFactory.mintYieldTo(address,uint256)._to (contracts/HighStreetPoolFactory.sol#256) is not in mixedCase
Parameter HighStreetPoolFactory.mintYieldTo(address,uint256)._amount (contracts/HighStreetPoolFactory.sol#256) is not in mixedCase
Parameter HighStreetPoolFactory.transferHighToken(address,uint256)._to (contracts/HighStreetPoolFactory.sol#304) is not in mixedCase
Parameter HighStreetPoolFactory.transferHighToken(address,uint256)._value (contracts/HighStreetPoolFactory.sol#304) is not in mixedCase
Variable HighStreetPoolFactory.HIGH (contracts/HighStreetPoolFactory.sol#32) is not in mixedCase
Function IPool.HIGH() (contracts/interfaces/IPool.sol#29) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

HighStreetPoolBase (contracts/HighStreetPoolBase.sol#29-761) does not implement functions:
        - IPool.isFlashPool() (contracts/interfaces/IPool.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
getPoolData(address) should be declared external:
        - HighStreetPoolFactory.getPoolData(address) (contracts/HighStreetPoolFactory.sol#169-183)
registerPool(address) should be declared external:
        - HighStreetPoolFactory.registerPool(address) (contracts/HighStreetPoolFactory.sol#209-227)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

HighStreetPoolFactory.sol

```
Reentrancy in HighStreetPoolFactory.changePoolWeight(address,uint32) (contracts/HighStreetPoolFactory.sol#271-287):
        External calls:
        - IPool(poolAddr).setWeight(weight) (contracts/HighStreetPoolFactory.sol#283)
        Event emitted after the call(s):
        - WeightUpdated(msg.sender,poolAddr,weight) (contracts/HighStreetPoolFactory.sol#286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.1', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - 0.8.1 (contracts/HighStreetPoolFactory.sol#2)
        - 0.8.1 (contracts/interfaces/IPool.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#108-114) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#44-57) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#68-79) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#59-66) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#28-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (contracts/HighStreetPoolFactory.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.1 (contracts/interfaces/IPool.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter HighStreetPoolFactory.getPoolData(address)._poolToken (contracts/HighStreetPoolFactory.sol#169) is not in mixedCase
Parameter HighStreetPoolFactory.mintYieldTo(address,uint256)._to (contracts/HighStreetPoolFactory.sol#256) is not in mixedCase
Parameter HighStreetPoolFactory.mintYieldTo(address,uint256)._amount (contracts/HighStreetPoolFactory.sol#256) is not in mixedCase
Parameter HighStreetPoolFactory.transferHighToken(address,uint256)._to (contracts/HighStreetPoolFactory.sol#304) is not in mixedCase
Parameter HighStreetPoolFactory.transferHighToken(address,uint256)._value (contracts/HighStreetPoolFactory.sol#304) is not in mixedCase
Variable HighStreetPoolFactory.HIGH (contracts/HighStreetPoolFactory.sol#32) is not in mixedCase
Function IPool.HIGH() (contracts/interfaces/IPool.sol#29) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
getPoolData(address) should be declared external:
        - HighStreetPoolFactory.getPoolData(address) (contracts/HighStreetPoolFactory.sol#169-183)
registerPool(address) should be declared external:
        - HighStreetPoolFactory.registerPool(address) (contracts/HighStreetPoolFactory.sol#209-227)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- No major issues found by Slither.

4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

HighStreetCorePool.sol

Report for contracts/HighStreetCorePool.sol
https://dashboard.mythx.io/#/console/analyses/e4c502ef-4a23-470a-b845-ddf97078a774

Line	SWC Title	Severity	Short Description
88	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
128	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
132	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
175	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
180	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
184	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
185	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
188	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
194	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
212	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
227	(SWC-110) Assert Violation	Unknown	Out of bounds array access
229	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
249	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
269	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered

HighStreetFlashPool.sol

Report for contracts/HighStreetFlashPool.sol
https://dashboard.mythx.io/#/console/analyses/d09f569f-9529-48b0-a635-814a6cfbff0c

Line	SWC Title	Severity	Short Description
81	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

HighStreetPoolBase.sol

Report for contracts/HighStreetPoolBase.sol
https://dashboard.mythx.io/#/console/analyses/e4c502ef-4a23-470a-b845-ddf97078a774

Line	SWC Title	Severity	Short Description
85	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
202	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
203	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
203	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
206	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
214	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
241	(SWC-110) Assert Violation	Unknown	Out of bounds array access
384	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
402	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
426	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
436	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
436	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
436	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
436	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
454	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
455	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
459	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
483	(SWC-110) Assert Violation	Unknown	Out of bounds array access
499	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
499	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
499	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
499	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
501	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
504	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
505	(SWC-110) Assert Violation	Unknown	Out of bounds array access
507	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
512	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
513	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
513	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
517	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
517	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
561	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
565	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
565	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
568	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
603	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
611	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
618	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
619	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
622	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
657	(SWC-110) Assert Violation	Unknown	Out of bounds array access
664	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
667	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
673	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
673	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
673	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
673	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
683	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
683	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
684	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
684	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
700	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
700	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
716	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
716	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
727	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

HighStreetPoolFactory.sol

Report for contracts/HighStreetPoolFactory.sol
https://dashboard.mythx.io/#/console/analyses/e4c502ef-4a23-470a-b845-ddf97078a774

Line	SWC Title	Severity	Short Description
199	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
223	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
238	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
238	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
280	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
280	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
297	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

- Integer Overflows and Underflows flagged by MythX are false positives, as the contract is using Solidity 0.8.9 version. After the Solidity version 0.8.0 Arithmetic operations revert to underflow and overflow by default.
- Assert violations are false positives.

THANK YOU FOR CHOOSING

// HALBORN