MythX

REPORT 64231429B44969001A1A3D04

Created	Tue Mar 28 2023 16:22:01 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	60be4a3d43f2c340d312dbe0

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
538d8c0f-8f27-473f-a7cc-3d2f1d3389bb	BaseFactory.sol	1

Analysis 538d8c0f-8f27-473f-a7cc-3d2f1d3389bb

MythX

Started	Tue Mar 28 2023 16:22:03 GMT+0000 (Coordinated Universal Time)
Finished	Tue Mar 28 2023 17:18:27 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	BaseFactory.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

BasePair.sol

Locations

```
mapping(address => uint256) public nonces;

uint256 internal constant MINIMUM_LIQUIDITY = 10**3;

address public immutable token0;
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

BasePair.sol

Locations

```
}

decimals0 = 10**IERC20(_token0).decimals();
decimals1 = 10**IERC20(_token1).decimals();
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```

decimals0 = 10**IERC20(_token0).decimals();
decimals1 = 10**IERC20(_token1).decimals();

observations.push(Observation(block.timestamp, 0, 0));
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```

function lastObservation() public view returns (Observation memory) {
return observations[observations.length - 1];
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _update0(uint256 amount) internal {
_safeTransfer(token0, fees, amount); // transfer the fees out to BaseV1Fees
uint256 _ratio = (amount * 1e18) / totalSupply; // 1e18 adjustment is removed during claim
if (_ratio > 0) {
index0 += _ratio;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _update0(uint256 amount) internal {
_safeTransfer(token0, fees, amount); // transfer the fees out to BaseV1Fees
uint256 _ratio = (amount * 1e18) / totalSupply; // 1e18 adjustment is removed during claim
if (_ratio > 0) {
index0 += _ratio;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _ratio = (amount * 1e18) / totalSupply; // 1e18 adjustment is removed during claim
if (_ratio > 0) {
index0 += _ratio;
}
emit Fees(msg.sender, amount, 0);
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _update1(uint256 amount) internal {
_safeTransfer(token1, fees, amount);
uint256 _ratio = (amount * 1e18) / totalSupply;
if (_ratio > 0) {
index1 += _ratio;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _update1(uint256 amount) internal {
_safeTransfer(token1, fees, amount);
uint256 _ratio = (amount * 1e18) / totalSupply;
if (_ratio > 0) {
index1 += _ratio;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _ratio = (amount * 1e18) / totalSupply;
if (_ratio > 0) {
index1 += _ratio;
}
emit Fees(msg.sender, 0, amount);
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
supplyIndex0[recipient] = _index0; // update user current position to global position
supplyIndex1[recipient] = _index1;
uint256 _delta0 = _index0 - _supplyIndex0; // see if there is any difference that need to be accrued
uint256 _delta1 = _index1 - _supplyIndex1;
if (_delta0 > 0) {
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
supplyIndex1[recipient] = _index1;
uint256 _delta0 = _index0 - _supplyIndex0; // see if there is any difference that need to be accrued
uint256 _delta1 = _index1 - _supplyIndex1;
if (_delta0 > 0) {
uint256 _share = (_supplied * _delta0) / 1e18; // add accrued difference for each supplied token
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _delta1 = _index1 - _supplyIndex1;
if (_delta0 > 0) {
uint256 _share = (_supplied * _delta0) / 1e18; // add accrued difference for each supplied token
claimable0[recipient] += _share;
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _delta1 = _index1 - _supplyIndex1;
if (_delta0 > 0) {
uint256 _share = (_supplied * _delta0) / 1e18; // add accrued difference for each supplied token
claimable0[recipient] += _share;
}
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
if (_delta0 > 0) {
uint256 _share = (_supplied * _delta0) / 1e18; // add accrued difference for each supplied token
claimable0[recipient] += _share;
}
if (_delta1 > 0) {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
}
if (_delta1 > 0) {
uint256 _share = (_supplied * _delta1) / 1e18;
claimable1[recipient] += _share;
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
}
if (_delta1 > 0) {
uint256 _share = (_supplied * _delta1) / 1e18;
claimable1[recipient] += _share;
}
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
if (_delta1 > 0) {
uint256 _share = (_supplied * _delta1) / 1e18;
claimable1[recipient] += _share;
}
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
) internal {
uint256 blockTimestamp = block.timestamp;
uint256 timeElapsed = blockTimestamp - blockTimestampLast; // overflow is desired
if (timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0) {
reserve0CumulativeLast += _reserve0 * timeElapsed;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 timeElapsed = blockTimestamp - blockTimestampLast; // overflow is desired
if (timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0) {
reserve0CumulativeLast += _reserve0 * timeElapsed;
reserve1CumulativeLast += _reserve1 * timeElapsed;
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 timeElapsed = blockTimestamp - blockTimestampLast; // overflow is desired
if (timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0) {
reserve0CumulativeLast += _reserve0 * timeElapsed;
reserve1CumulativeLast += _reserve1 * timeElapsed;
}
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
if (timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0) {
reserve0CumulativeLast += _reserve0 * timeElapsed;
reserve1CumulativeLast += _reserve1 * timeElapsed;
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
if (timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0) {
reserve0CumulativeLast += _reserve0 * timeElapsed;
reserve1CumulativeLast += _reserve1 * timeElapsed;
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```

Observation memory _point = lastObservation();
timeElapsed = blockTimestamp - _point.timestamp; // compare the last observation with current timestamp, if greater than 30 minutes, record a new event
if (timeElapsed > periodSize) {
observations.push(
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
if (_blockTimestampLast != blockTimestamp) {
// subtraction overflow is desired
uint256 timeElapsed = blockTimestamp - _blockTimestampLast;
reserve0Cumulative += _reserve0 * timeElapsed;
reserve1Cumulative += _reserve1 * timeElapsed;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
// subtraction overflow is desired
uint256 timeElapsed = blockTimestamp - _blockTimestampLast;
reserve0Cumulative += _reserve0 * timeElapsed;
reserve1Cumulative += _reserve1 * timeElapsed;
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
// subtraction overflow is desired
uint256 timeElapsed = blockTimestamp - _blockTimestampLast;
reserve0Cumulative += _reserve0 * timeElapsed;
reserve1Cumulative += _reserve1 * timeElapsed;
}
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 timeElapsed = blockTimestamp - _blockTimestampLast;
reserve0Cumulative += _reserve0 * timeElapsed;
reserve1Cumulative += _reserve1 * timeElapsed;
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 timeElapsed = blockTimestamp - _blockTimestampLast;
reserve0Cumulative += _reserve0 * timeElapsed;
reserve1Cumulative += _reserve1 * timeElapsed;
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
) = currentCumulativePrices();
if (block.timestamp == _observation.timestamp) {
_observation = observations[observations.length - 2];
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
}

uint256 timeElapsed = block.timestamp - _observation.timestamp;
uint256 _reserve0 = (reserve0Cumulative -
_observation.reserve0Cumulative) / timeElapsed;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```

uint256 timeElapsed = block.timestamp - _observation.timestamp;
uint256 _reserve0 = (reserve0Cumulative -
_observation.reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (reserve1Cumulative -
_observation.reserve1Cumulative) / timeElapsed;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```

uint256 timeElapsed = block.timestamp - _observation.timestamp;
uint256 _reserve0 = (reserve0Cumulative -
_observation.reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (reserve1Cumulative -
_observation.reserve1Cumulative) / timeElapsed;
```

UNKNOWN

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

BasePair.sol

Locations

```
uint256 _reserve0 = (reserve0Cumulative -
_observation.reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (reserve1Cumulative -
_observation.reserve1Cumulative) / timeElapsed;
amountOut = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
}
```

UNKNOWN

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

BasePair.sol

Locations

```
uint256 _reserve0 = (reserve0Cumulative -
_observation.reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (reserve1Cumulative -
_observation.reserve1Cumulative) / timeElapsed;
amountOut = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
}
```

UNKNOWN

Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

BasePair.sol

Locations

```
uint256[] memory _prices = sample(tokenIn, amountIn, granularity, 1);
uint256 priceAverageCumulative;
for (uint256 i = 0; i < _prices.length; i++) {
priceAverageCumulative += _prices[i];
}
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

BasePair.sol

Locations

```
uint256 priceAverageCumulative;
for (uint256 i = 0; i < _prices.length; i++) {
priceAverageCumulative += _prices[i];
}
return priceAverageCumulative / granularity;
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

BasePair.sol

Locations

```
priceAverageCumulative += _prices[i];
}
return priceAverageCumulative / granularity;
}
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

BasePair.sol

Locations

```
uint256[] memory _prices = new uint256[](points);

uint256 length = observations.length - 1;
uint256 i = length - (points * window);
uint256 nextIndex = 0;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```

uint256 length = observations.length - 1;
uint256 i = length - (points * window);
uint256 nextIndex = 0;
uint256 index = 0;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```

uint256 length = observations.length - 1;
uint256 i = length - (points * window);
uint256 nextIndex = 0;
uint256 index = 0;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 index = 0;

for (; i < length; i += window) {
nextIndex = i + window;
uint256 timeElapsed = observations[nextIndex].timestamp -
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```

for (; i < length; i += window) {
nextIndex = i + window;
uint256 timeElapsed = observations[nextIndex].timestamp -
observations[i].timestamp;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
for (; i < length; i += window) {
nextIndex = i + window;
uint256 timeElapsed = observations[nextIndex].timestamp -
observations[i].timestamp;
uint256 _reserve0 = (observations[nextIndex].reserve0Cumulative -
observations[i].reserve0Cumulative) / timeElapsed;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 timeElapsed = observations[nextIndex].timestamp -
observations[i].timestamp;
uint256 _reserve0 = (observations[nextIndex].reserve0Cumulative -
observations[i].reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (observations[nextIndex].reserve1Cumulative -
observations[i].reserve1Cumulative) / timeElapsed;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
uint256 timeElapsed = observations[nextIndex].timestamp -
observations[i].timestamp;
uint256 _reserve0 = (observations[nextIndex].reserve0Cumulative -
observations[i].reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (observations[nextIndex].reserve1Cumulative -
observations[i].reserve1Cumulative) / timeElapsed;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
uint256 _reserve0 = (observations[nextIndex].reserve0Cumulative -
observations[i].reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (observations[nextIndex].reserve1Cumulative -
observations[i].reserve1Cumulative) / timeElapsed;
_prices[index] = _getAmountOut(
amountIn,
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
uint256 _reserve0 = (observations[nextIndex].reserve0Cumulative -
observations[i].reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (observations[nextIndex].reserve1Cumulative -
observations[i].reserve1Cumulative) / timeElapsed;
_prices[index] = _getAmountOut(
amountIn,
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
_reserve1
);
index = index + 1;
}
return _prices;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _balance0 = IERC20(token0).balanceOf(address(this));
uint256 _balance1 = IERC20(token1).balanceOf(address(this));
uint256 _amount0 = _balance0 - _reserve0;
uint256 _amount1 = _balance1 - _reserve1;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _balance1 = IERC20(token1).balanceOf(address(this));
uint256 _amount0 = _balance0 - _reserve0;
uint256 _amount1 = _balance1 - _reserve1;

uint256 _totalSupply = totalSupply; // gas savings, must be defined here since totalSupply can update in _mintFee
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _totalSupply = totalSupply; // gas savings, must be defined here since totalSupply can update in _mintFee
if (_totalSupply == 0) {
liquidity = Math.sqrt(_amount0 * _amount1) - MINIMUM_LIQUIDITY;
_mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _totalSupply = totalSupply; // gas savings, must be defined here since totalSupply can update in _mintFee
if (_totalSupply == 0) {
liquidity = Math.sqrt(_amount0 * _amount1) - MINIMUM_LIQUIDITY;
_mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
} else {
liquidity = Math.min(
(_amount0 * _totalSupply) / _reserve0,
(_amount1 * _totalSupply) / _reserve1
);
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
} else {
liquidity = Math.min(
(_amount0 * _totalSupply) / _reserve0,
(_amount1 * _totalSupply) / _reserve1
);
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
liquidity = Math.min(
(_amount0 * _totalSupply) / _reserve0,
(_amount1 * _totalSupply) / _reserve1
);
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
liquidity = Math.min(
(_amount0 * _totalSupply) / _reserve0,
(_amount1 * _totalSupply) / _reserve1
);
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```

uint256 _totalSupply = totalSupply; // gas savings, must be defined here since totalSupply can update in _mintFee
amount0 = (_liquidity * _balance0) / _totalSupply; // using balances ensures proportionate distribution
amount1 = (_liquidity * _balance1) / _totalSupply; // using balances ensures proportionate distribution
require(amount0 > 0 && amount1 > 0, "ILB"); // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```

uint256 _totalSupply = totalSupply; // gas savings, must be defined here since totalSupply can update in _mintFee
amount0 = (_liquidity * _balance0) / _totalSupply; // using balances ensures proportionate distribution
amount1 = (_liquidity * _balance1) / _totalSupply; // using balances ensures proportionate distribution
require(amount0 > 0 && amount1 > 0, "ILB"); // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
uint256 _totalSupply = totalSupply; // gas savings, must be defined here since totalSupply can update in _mintFee
amount0 = (_liquidity * _balance0) / _totalSupply; // using balances ensures proportionate distribution
amount1 = (_liquidity * _balance1) / _totalSupply; // using balances ensures proportionate distribution
require(amount0 > 0 && amount1 > 0, "ILB"); // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
_burn(address(this), _liquidity);
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
483 uint256 _totalSupply = totalSupply; // gas savings, must be defined here since totalSupply can update in _mintFee
484 amount0 = (_liquidity * _balance0) / _totalSupply; // using balances ensures proportionate distribution
485 amount1 = (_liquidity * _balance1) / _totalSupply; // using balances ensures proportionate distribution
486 require(amount0 > 0 && amount1 > 0, "ILB"); // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
487 _burn(address(this), _liquidity);
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
525 _balance1 = IERC20(_token1).balanceOf(address(this));
526 }
527 uint256 amount0In = _balance0 > _reserve0 - amount0Out
528 ? _balance0 - (_reserve0 - amount0Out)
529 : 0;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
526 }
527 uint256 amount0In = _balance0 > _reserve0 - amount0Out
528 ? _balance0 - (_reserve0 - amount0Out)
529 : 0;
530 uint256 amount1In = _balance1 > _reserve1 - amount1Out
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
}
uint256 amount0In = _balance0 > _reserve0 - amount0Out
? _balance0 - (_reserve0 - amount0Out)
: 0;
uint256 amount1In = _balance1 > _reserve1 - amount1Out
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
? _balance0 - (_reserve0 - amount0Out)
: 0;
uint256 amount1In = _balance1 > _reserve1 - amount1Out
? _balance1 - (_reserve1 - amount1Out)
: 0;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
: 0;
uint256 amount1In = _balance1 > _reserve1 - amount1Out
? _balance1 - (_reserve1 - amount1Out)
: 0;
require(amount0In > 0 || amount1In > 0, "IIA"); // BaseV1: INSUFFICIENT_INPUT_AMOUNT
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
: 0;
uint256 amount1In = _balance1 > _reserve1 - amount1Out
? _balance1 - (_reserve1 - amount1Out)
: 0;
require(amount0In > 0 || amount1In > 0, "IIA"); // BaseV1: INSUFFICIENT_INPUT_AMOUNT
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
(address _token0, address _token1) = (token0, token1);

if (amount0In > 0) _update0(amount0In / fee); // accrue fees for token0 and move them out of pool
if (amount1In > 0) _update1(amount1In / fee); // accrue fees for token1 and move them out of pool
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```

if (amount0In > 0) _update0(amount0In / fee); // accrue fees for token0 and move them out of pool
if (amount1In > 0) _update1(amount1In / fee); // accrue fees for token1 and move them out of pool

_balance0 = IERC20(_token0).balanceOf(address(this)); // since we removed tokens, we need to reconfirm balances, can also simply use previous balance - amountIn/ 10000, but doing balanceOf again as safety check
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
_token0,
to,
IERC20(_token0).balanceOf(address(this)) - (reserve0)
);
_safeTransfer(
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
_token1,
to,
IERC20(_token1).balanceOf(address(this)) - (reserve1)
);
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _f(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
1e18;
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _f(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
1e18;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _f(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _f(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _f(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _f(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _f(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
1e18;
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
1e18;
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
1e18;
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
1e18;
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
1e18;
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
(x0 * ((((y * y) / 1e18) * y) / 1e18)) /
1e18 +
(((((x0 * x0) / 1e18) * x0) / 1e18) * y) /
1e18;
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
function _d(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(3 * x0 * ((y * y) / 1e18)) /
1e18 +
((((x0 * x0) / 1e18) * x0) / 1e18);
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
function _d(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(3 * x0 * ((y * y) / 1e18)) /
1e18 +
((((x0 * x0) / 1e18) * x0) / 1e18);
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
function _d(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(3 * x0 * ((y * y) / 1e18)) /
1e18 +
((((x0 * x0) / 1e18) * x0) / 1e18);
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _d(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(3 * x0 * ((y * y) / 1e18)) /
1e18 +
((((x0 * x0) / 1e18) * x0) / 1e18);
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _d(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(3 * x0 * ((y * y) / 1e18)) /
1e18 +
((((x0 * x0) / 1e18) * x0) / 1e18);
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _d(uint256 x0, uint256 y) internal pure returns (uint256) {
return
(3 * x0 * ((y * y) / 1e18)) /
1e18 +
((((x0 * x0) / 1e18) * x0) / 1e18);
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
(3 * x0 * ((y * y) / 1e18)) /
1e18 +
((((x0 * x0) / 1e18) * x0) / 1e18);
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
(3 * x0 * ((y * y) / 1e18)) /
1e18 +
((((x0 * x0) / 1e18) * x0) / 1e18);
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
(3 * x0 * ((y * y) / 1e18)) /
1e18 +
((((x0 * x0) / 1e18) * x0) / 1e18);
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
(3 * x0 * ((y * y) / 1e18)) /
1e18 +
((((x0 * x0) / 1e18) * x0) / 1e18);
}
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 y
) internal pure returns (uint256) {
for (uint256 i = 0; i < 255; i++) {
uint256 y_prev = y;
uint256 k = _f(x0, y);
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 k = _f(x0, y);
if (k < xy) {
uint256 dy = ((xy - k) * 1e18) / _d(x0, y);
y = y + dy;
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 k = _f(x0, y);
if (k < xy) {
uint256 dy = ((xy - k) * 1e18) / _d(x0, y);
y = y + dy;
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 k = _f(x0, y);
if (k < xy) {
uint256 dy = ((xy - k) * 1e18) / _d(x0, y);
y = y + dy;
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
if (k < xy) {
uint256 dy = ((xy - k) * 1e18) / _d(x0, y);
y = y + dy;
} else {
uint256 dy = ((k - xy) * 1e18) / _d(x0, y);
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
y = y + dy;
} else {
uint256 dy = ((k - xy) * 1e18) / _d(x0, y);
y = y - dy;
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
y = y + dy;
} else {
uint256 dy = ((k - xy) * 1e18) / _d(x0, y);
y = y - dy;
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
y = y + dy;
} else {
uint256 dy = ((k - xy) * 1e18) / _d(x0, y);
y = y - dy;
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
} else {
uint256 dy = ((k - xy) * 1e18) / _d(x0, y);
y = y - dy;
}
if (y > y_prev) {
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
}
if (y > y_prev) {
if (y - y_prev <= 1) {
return y;
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
}
} else {
if (y_prev - y <= 1) {
return y;
}
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
{
(uint256 _reserve0, uint256 _reserve1) = (reserve0, reserve1);
amountIn -= amountIn / fee; // remove fee from amount received
return _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
{
(uint256 _reserve0, uint256 _reserve1) = (reserve0, reserve1);
amountIn -= amountIn / fee; // remove fee from amount received
return _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
if (stable) {
uint256 xy = _k(_reserve0, _reserve1);
_reserve0 = (_reserve0 * 1e18) / decimals0;
_reserve1 = (_reserve1 * 1e18) / decimals1;
(uint256 reserveA, uint256 reserveB) = tokenIn == token0
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
if (stable) {
uint256 xy = _k(_reserve0, _reserve1);
_reserve0 = (_reserve0 * 1e18) / decimals0;
_reserve1 = (_reserve1 * 1e18) / decimals1;
(uint256 reserveA, uint256 reserveB) = tokenIn == token0
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 xy = _k(_reserve0, _reserve1);
_reserve0 = (_reserve0 * 1e18) / decimals0;
_reserve1 = (_reserve1 * 1e18) / decimals1;
(uint256 reserveA, uint256 reserveB) = tokenIn == token0
? (_reserve0, _reserve1)
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 xy = _k(_reserve0, _reserve1);
_reserve0 = (_reserve0 * 1e18) / decimals0;
_reserve1 = (_reserve1 * 1e18) / decimals1;
(uint256 reserveA, uint256 reserveB) = tokenIn == token0
? (_reserve0, _reserve1)
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
: (_reserve1, _reserve0);
amountIn = tokenIn == token0
? (amountIn * 1e18) / decimals0
: (amountIn * 1e18) / decimals1;
uint256 y = reserveB - _get_y(amountIn + reserveA, xy, reserveB);
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
: (_reserve1, _reserve0);
amountIn = tokenIn == token0
? (amountIn * 1e18) / decimals0
: (amountIn * 1e18) / decimals1;
uint256 y = reserveB - _get_y(amountIn + reserveA, xy, reserveB);
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
amountIn = tokenIn == token0
? (amountIn * 1e18) / decimals0
: (amountIn * 1e18) / decimals1;
uint256 y = reserveB - _get_y(amountIn + reserveA, xy, reserveB);
return (y * (tokenIn == token0 ? decimals1 : decimals0)) / 1e18;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
amountIn = tokenIn == token0
? (amountIn * 1e18) / decimals0
: (amountIn * 1e18) / decimals1;
uint256 y = reserveB - _get_y(amountIn + reserveA, xy, reserveB);
return (y * (tokenIn == token0 ? decimals1 : decimals0)) / 1e18;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
? (amountIn * 1e18) / decimals0
: (amountIn * 1e18) / decimals1;
uint256 y = reserveB - _get_y(amountIn + reserveA, xy, reserveB);
return (y * (tokenIn == token0 ? decimals1 : decimals0)) / 1e18;
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
? (amountIn * 1e18) / decimals0
: (amountIn * 1e18) / decimals1;
uint256 y = reserveB - _get_y(amountIn + reserveA, xy, reserveB);
return (y * (tokenIn == token0 ? decimals1 : decimals0)) / 1e18;
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
: (amountIn * 1e18) / decimals1;
uint256 y = reserveB - _get_y(amountIn + reserveA, xy, reserveB);
return (y * (tokenIn == token0 ? decimals1 : decimals0)) / 1e18;
} else {
(uint256 reserveA, uint256 reserveB) = tokenIn == token0
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
: (amountIn * 1e18) / decimals1;
uint256 y = reserveB - _get_y(amountIn + reserveA, xy, reserveB);
return (y * (tokenIn == token0 ? decimals1 : decimals0)) / 1e18;
} else {
(uint256 reserveA, uint256 reserveB) = tokenIn == token0
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
? (_reserve0, _reserve1)
: (_reserve1, _reserve0);
return (amountIn * reserveB) / (reserveA + amountIn);
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
? (_reserve0, _reserve1)
: (_reserve1, _reserve0);
return (amountIn * reserveB) / (reserveA + amountIn);
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
? (_reserve0, _reserve1)
: (_reserve1, _reserve0);
return (amountIn * reserveB) / (reserveA + amountIn);
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _k(uint256 x, uint256 y) internal view returns (uint256) {
if (stable) {
uint256 _x = (x * 1e18) / decimals0;
uint256 _y = (y * 1e18) / decimals1;
uint256 _a = (_x * _y) / 1e18;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
function _k(uint256 x, uint256 y) internal view returns (uint256) {
if (stable) {
uint256 _x = (x * 1e18) / decimals0;
uint256 _y = (y * 1e18) / decimals1;
uint256 _a = (_x * _y) / 1e18;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
if (stable) {
uint256 _x = (x * 1e18) / decimals0;
uint256 _y = (y * 1e18) / decimals1;
uint256 _a = (_x * _y) / 1e18;
uint256 _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
if (stable) {
uint256 _x = (x * 1e18) / decimals0;
uint256 _y = (y * 1e18) / decimals1;
uint256 _a = (_x * _y) / 1e18;
uint256 _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
uint256 _x = (x * 1e18) / decimals0;
uint256 _y = (y * 1e18) / decimals1;
uint256 _a = (_x * _y) / 1e18;
uint256 _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
return (_a * _b) / 1e18; // x3y+y3x >= k
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _x = (x * 1e18) / decimals0;
uint256 _y = (y * 1e18) / decimals1;
uint256 _a = (_x * _y) / 1e18;
uint256 _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
return (_a * _b) / 1e18; // x3y+y3x >= k
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _y = (y * 1e18) / decimals1;
uint256 _a = (_x * _y) / 1e18;
uint256 _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
return (_a * _b) / 1e18; // x3y+y3x >= k
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _y = (y * 1e18) / decimals1;
uint256 _a = (_x * _y) / 1e18;
uint256 _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
return (_a * _b) / 1e18; // x3y+y3x >= k
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _y = (y * 1e18) / decimals1;
uint256 _a = (_x * _y) / 1e18;
uint256 _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
return (_a * _b) / 1e18; // x3y+y3x >= k
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _y = (y * 1e18) / decimals1;
uint256 _a = (_x * _y) / 1e18;
uint256 _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
return (_a * _b) / 1e18; // x3y+y3x >= k
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _y = (y * 1e18) / decimals1;
uint256 _a = (_x * _y) / 1e18;
uint256 _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
return (_a * _b) / 1e18; // x3y+y3x >= k
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _a = (_x * _y) / 1e18;
uint256 _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
return (_a * _b) / 1e18; // x3y+y3x >= k
} else {
return x * y; // xy >= k
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256 _a = (_x * _y) / 1e18;
uint256 _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
return (_a * _b) / 1e18; // x3y+y3x >= k
} else {
return x * y; // xy >= k
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
return (_a * _b) / 1e18; // x3y+y3x >= k
} else {
return x * y; // xy >= k
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
function _mint(address dst, uint256 amount) internal {
_updateFor(dst); // balances must be updated on mint/burn/transfer
totalSupply += amount;
balanceOf[dst] += amount;
emit Transfer(address(0), dst, amount);
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
_updateFor(dst); // balances must be updated on mint/burn/transfer
totalSupply += amount;
balanceOf[dst] += amount;
emit Transfer(address(0), dst, amount);
}
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`BasePair.sol`

Locations

```
function _burn(address dst, uint256 amount) internal {
_updateFor(dst);
totalSupply -= amount;
balanceOf[dst] -= amount;
emit Transfer(dst, address(0), amount);
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
_updateFor(dst);
totalSupply -= amount;
balanceOf[dst] -= amount;
emit Transfer(dst, address(0), amount);
}
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
spender,
value,
nonces[owner]++,
deadline
)
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```

if (spender != src && spenderAllowance != type(uint256).max) {
uint256 newAllowance = spenderAllowance - amount;
allowance[src][spender] = newAllowance;
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
_updateFor(dst); // update fee position for dst

balanceOf[src] -= amount;
balanceOf[dst] += amount;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```

balanceOf[src] -= amount;
balanceOf[dst] += amount;

emit Transfer(src, dst, amount);
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```

function lastObservation() public view returns (Observation memory) {
return observations[observations.length - 1];
}
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

BasePair.sol

Locations

```
uint256[] memory _prices = new uint256[](points);

uint256 length = observations.length - 1;
uint256 i = length - (points * window);
uint256 nextIndex = 0;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`Math.sol`

Locations

```
if (y > 3) {
z = y;
uint256 x = y / 2 + 1;
while (x < z) {
z = x;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`Math.sol`

Locations

```
if (y > 3) {
z = y;
uint256 x = y / 2 + 1;
while (x < z) {
z = x;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`Math.sol`

Locations

```
while (x < z) {
z = x;
x = (y / x + x) / 2;
}
} else if (y != 0) {
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`Math.sol`

Locations

```
while (x < z) {
z = x;
x = (y / x + x) / 2;
}
} else if (y != 0) {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`Math.sol`

Locations

```
while (x < z) {
z = x;
x = (y / x + x) / 2;
}
} else if (y != 0) {
```

LOW

SWC-135

Usage of equality comparison instead of assignment

This equality comparison doesn't have any effect. Did you mean to do assignment instead?

Source file

`BasePair.sol`

Locations

```
)
);
chainid == block.chainid;
}
bytes32 digest = keccak256(
```

UNKNOWN

SWC-110

Public state variable with array type causing reacheable exception by default.

The public state variable "allPairs" in "BaseFactory" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source file

BaseFactory.sol

Locations

```

mapping(address => mapping(address => mapping(bool => address))) public getPair;
address[] public allPairs;
mapping(address => bool) public isPair; // simplified check if its a pair, given that `stable` flag might not be available in peripherals
```

UNKNOWN

SWC-110

Public state variable with array type causing reacheable exception by default.

The public state variable "observations" in "BasePair" contract has type "struct BasePair.Observation[]" and can cause an exception in case of use of invalid array index value.

Source file

BasePair.sol

Locations

```
uint256 constant periodSize = 1800;

Observation[] public observations;

uint256 internal immutable decimals0;
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

BasePair.sol

Locations

```

function lastObservation() public view returns (Observation memory) {
return observations[observations.length - 1];
}
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

BasePair.sol

Locations

```
) = currentCumulativePrices();
if (block.timestamp == _observation.timestamp) {
_observation = observations[observations.length - 2];
}
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

BasePair.sol

Locations

```
uint256 priceAverageCumulative;
for (uint256 i = 0; i < _prices.length; i++) {
priceAverageCumulative += _prices[i];
}
return priceAverageCumulative / granularity;
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

BasePair.sol

Locations

```
for (; i < length; i += window) {
nextIndex = i + window;
uint256 timeElapsed = observations[nextIndex].timestamp -
observations[i].timestamp;
uint256 _reserve0 = (observations[nextIndex].reserve0Cumulative -
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

BasePair.sol

Locations

```
nextIndex = i + window;
uint256 timeElapsed = observations[nextIndex].timestamp -
observations[i].timestamp;
uint256 _reserve0 = (observations[nextIndex].reserve0Cumulative -
observations[i].reserve0Cumulative) / timeElapsed;
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

`BasePair.sol`

Locations

```
uint256 timeElapsed = observations[nextIndex].timestamp -
observations[i].timestamp;
uint256 _reserve0 = (observations[nextIndex].reserve0Cumulative -
observations[i].reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (observations[nextIndex].reserve1Cumulative -
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

`BasePair.sol`

Locations

```
observations[i].timestamp;
uint256 _reserve0 = (observations[nextIndex].reserve0Cumulative -
observations[i].reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (observations[nextIndex].reserve1Cumulative -
observations[i].reserve1Cumulative) / timeElapsed;
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

`BasePair.sol`

Locations

```
uint256 _reserve0 = (observations[nextIndex].reserve0Cumulative -
observations[i].reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (observations[nextIndex].reserve1Cumulative -
observations[i].reserve1Cumulative) / timeElapsed;
_prices[index] = _getAmountOut(
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

BasePair.sol

Locations

```
observations[i].reserve0Cumulative) / timeElapsed;
uint256 _reserve1 = (observations[nextIndex].reserve1Cumulative -
observations[i].reserve1Cumulative) / timeElapsed;
_prices[index] = _getAmountOut(
amountIn,
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

BasePair.sol

Locations

```
uint256 _reserve1 = (observations[nextIndex].reserve1Cumulative -
observations[i].reserve1Cumulative) / timeElapsed;
_prices[index] = _getAmountOut(
amountIn,
tokenIn,
```