

REPORT 628D67EF74BE140019ADE146

Created Tue May 24 2022 23:19:11 GMT+0000 (Coordinated Universal Time)

Number of analyses 1

User 6197960e3494e9c8c076e89b

REPORT SUMMARY

| Analyses ID | Main source file | Detected |
|-------------|------------------|-----------------|
| | | vulnerabilities |

a1876108-a8d4-4a10-8556-3f805af7945d

Pair.sol

1

Started Tue May 24 2022 23:19:20 GMT+0000 (Coordinated Universal Time)

Finished Tue May 24 2022 23:19:25 GMT+0000 (Coordinated Universal Time)

Mode Deep

Client Tool Remythx

Main Source File Pair.Sol

DETECTED VULNERABILITIES

| (HIGH | (MEDIUM | (LOW |
|-------|---------|------|
| 0 | 0 | 1 |

ISSUES

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

SWC-101

Source file Pair.sol

Locations

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

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

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

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

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

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
function _update0(uint amount) internal {
    _safeTransfer(token0, fees, amount); // transfer the fees out to PairFees

uint256 _ratio = amount * 1e18 / totalSupply; // 1e18 adjustment is removed during claim

if (_ratio > 0) {
    index0 += _ratio;
}
```

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

SWC-101

Source file

Pair.sol Locations

```
function _update0(uint amount) internal {

_safeTransfer(token0, fees, amount); // transfer the fees out to PairFees

uint256 _ratio = amount * le18 / totalSupply; // le18 adjustment is removed during claim

if (_ratio > 0) {

index0 += _ratio;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Pair.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 Arithmetic operation "/" discovered

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

SWC-101

Source file

```
Locations
```

```
function _update1(uint amount) internal {
    _safeTransfer(token1, fees, amount);

uint256 _ratio = amount * le18 / totalSupply;

if (_ratio > 0) {
    index1 += _ratio;
}
```

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

SWC-101

Source file Pair.sol

Locations

```
function _update1(uint amount) internal {
    _safeTransfer(token1, fees, amount);
    uint256 _ratio = amount * le18 / totalSupply;
    if (_ratio > 0) {
        index1 += _ratio;
    }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
uint256 _ratio = amount * 1e18 / totalSupply;

if (_ratio > 0) {
    index1 += _ratio;
}

femit Fees(msg.sender, 0, amount);
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

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

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

SWC-101

Source file

Pair.sol Locations

```
supplyIndex1[recipient] = _index1;

uint _delta0 = _index0 - _supplyIndex0; // see if there is any difference that need to be accrued

uint _delta1 = _index1 - _supplyIndex1;

if (_delta0 > 0) {

uint _share = _supplied * _delta0 / 1e18; // add accrued difference for each supplied token
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

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

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

```
Locations
```

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

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

SWC-101

Source file

Pair.sol Locations

```
if (_delta0 > 0) {

uint _share = _supplied * _delta0 / le18; // add accrued difference for each supplied token

claimable0 recipient _ += _share;
}

if (_delta1 > 0) {
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

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

SWC-101

Source file Pair.sol

Locations

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
function _update(uint balance0, uint balance1, uint _reserve0, uint _reserve1) internal {

uint blockTimestamp = block.timestamp;

uint timeElapsed = blockTimestamp - blockTimestampLast; // overflow is desired

if (timeElapsed > 0.88 _reserve0 != 0.88 _reserve1 != 0) {

reserve0CumulativeLast += _reserve0 * timeElapsed;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

```
uint timeElapsed = blockTimestamp - blockTimestampLast; // overflow is desired
if (timeElapsed > 0 &8 _reserve0 != 0 &8 _reserve1 != 0) {

reserve0CumulativeLast += _reserve0 * timeElapsed;

reserve1CumulativeLast += _reserve1 * timeElapsed;
}
```

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

SWC-101

Source file

Pair.sol Locations

```
uint timeElapsed = blockTimestamp - blockTimestamplast; // overflow is desired
if (timeElapsed > 0 &5 _reserve0 != 0 &5 _reserve1 != 0) {
reserve0CumulativeLast += _reserve0 * timeElapsed;
reserve1CumulativeLast += _reserve1 * timeElapsed;
}
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
207  if (timeElapsed > 0 88 _reserve0 != 0 88 _reserve1 != 0) {
208    reserve0CumulativeLast += _reserve0 * timeElapsed;
209    reserve1CumulativeLast += _reserve1 * timeElapsed;
210  }
211
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

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

SWC-101

Source file

Pair.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(Observation(blockTimestamp, reserve0CumulativeLast, reserve1CumulativeLast));
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
if (_blockTimestampLast != blockTimestamp) {

// subtraction overflow is desired

uint timeElapsed = blockTimestamp - _blockTimestampLast;

reserve0Cumulative += _reserve0 * timeElapsed;

reserve1Cumulative += _reserve1 * timeElapsed;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

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

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

SWC-101

Source file

Pair.sol

```
Locations
```

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

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
uint timeElapsed = blockTimestamp - _blockTimestamplast;
reserve0Cumulative += _reserve0 * timeElapsed;
reserve1Cumulative += _reserve1 * timeElapsed;
}
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

```
Locations
```

```
uint timeElapsed = blockTimestamp - _blockTimestampLast;
reserve0Cumulative += _reserve0 * timeElapsed;
reserve1Cumulative += _reserve1 * timeElapsed;
}
235
}
```

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

SWC-101

Source file

Pair.sol Locations

```
242  (uint reserve0Cumulative, uint reserve1Cumulative,) = currentCumulativePrices();
243  if (block.timestamp == _observation.timestamp) {
    observation = observations[observations length-2];
244  }
245  }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
245
246
247
uint timeElapsed = block timestamp - _observation.timestamp;
248
uint _reserve0 = (reserve0Cumulative - _observation.reserve0Cumulative) / timeElapsed;
249
uint _reserve1 = (reserve1Cumulative - _observation.reserve1Cumulative) / timeElapsed;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
uint timeElapsed = block.timestamp - _observation.timestamp;
uint _reserve0 = _reserve0Cumulative - _observation.reserve0Cumulative) / timeElapsed;
uint _reserve1 = (reserve1Cumulative - _observation.reserve1Cumulative) / timeElapsed;
amountOut = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
```

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

SWC-101

Source file Pair.sol

Locations

```
uint timeElapsed = block.timestamp - _observation.timestamp;

uint _reserve0 = (reserve0Cumulative - _observation reserve0Cumulative) / timeElapsed;

uint _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

Pair.sol

Locations

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

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

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

SWC-101

Source file

Pair.sol Locations

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

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
uint priceAverageCumulative;

for (uint i = 0; i < _prices.length; i++) {
    priceAverageCumulative += _prices i;
}

return priceAverageCumulative / granularity;</pre>
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
priceAverageCumulative += _prices[i];
}

return priceAverageCumulative / granularity;
}

261
}
```

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

SWC-101

Source file Pair.sol

Locations

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

uint length = observations length-1;

uint i = length - (points * window);

uint nextIndex = 0;

uint nextIndex = 0;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
270
271    uint length = observations.length-1;
272    uint i = length - points * window ;
273    uint nextIndex = 0;
274    uint index = 0;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

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

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

SWC-101

Source file

Pair.sol Locations

```
uint index = 0;

for (; i < length; i==window) {
    nextIndex = i + window;

uint timeElapsed = observations[nextIndex].timestamp - observations[i].timestamp;
}</pre>
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Pair.sol

Locations

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

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

```
Locations
```

```
for (; i < length; i+=window) {

nextIndex = i + window;

uint timeElapsed = observations nextIndex timestamp - observations i timestamp;

uint _reserve0 = (observations[nextIndex].reserve0Cumulative - observations[i].reserve0Cumulative) / timeElapsed;

uint _reserve1 = (observations[nextIndex].reserve1Cumulative - observations[i].reserve1Cumulative) / timeElapsed;
```

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

SWC-101

Source file Pair.sol

Locations

```
nextIndex = i + window;

uint timeElapsed = observations[nextIndex].timestamp - observations[i].timestamp;

uint _reserve0 = observations nextIndex] reserve0Cumulative - observations i reserve0Cumulative / timeElapsed;

uint _reserve1 = (observations[nextIndex].reserve1Cumulative - observations[i].reserve1Cumulative) / timeElapsed;

_prices[index] = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
nextIndex = i + window;

uint timeElapsed = observations[nextIndex].timestamp - observations[i].timestamp;

uint _reserve0 = (observations nextIndex reserve0Cumulative - observations i reserve0Cumulative) / timeElapsed;

uint _reserve1 = (observations[nextIndex].reserve1Cumulative - observations[i].reserve1Cumulative) / timeElapsed;

_prices[index] = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
uint timeElapsed = observations[nextIndex].timestamp - observations[i].timestamp;
uint _reserve0 = (observations[nextIndex].reserve0Cumulative - observations[i].reserve0Cumulative) / timeElapsed;
uint _reserve1 = observations nextIndex _reserve1Cumulative - observations i _reserve1Cumulative / timeElapsed;
prices[index] = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
index = index + 1;
```

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

SWC-101

Source file

Pair.sol Locations

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Pair.sol

```
Locations
```

```
uint _reserve1 = (observations[nextIndex].reserve1Cumulative - observations[i].reserve1Cumulative) / timeElapsed;

prices[index] = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);

index = index + 1;

return _prices;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

```
Locations
```

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

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

SWC-101

Source file

Pair.sol Locations

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

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

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
uint _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 Arithmetic operation "*" discovered

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

SWC-101

Source file

```
uint _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 {
```

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

SWC-101

Source file

Pair.sol Locations

```
299    __mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens
300  } else {
301    liquidity = Math.min(_amount0 * _totalSupply / _reserve0, _amount1 * _totalSupply / _reserve1);
302  }
303    require(liquidity > 0, 'ILM'); // Pair: INSUFFICIENT_LIQUIDITY_MINIED
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

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

SWC-101

Source file Pair.sol

Locations

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

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

amount0 = _liquidity *_balance0 / _totalSupply; // using balances ensures pro-rata distribution

amount1 = _liquidity *_balance1 / _totalSupply; // using balances ensures pro-rata distribution

require(amount0 > 0 && amount1 > 0, 'ILB'); // Pair: INSUFFICIENT_LIQUIDITY_BURNED
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

```
Locations
```

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

amount0 = _liquidity * _balance0 / _totalSupply; // using balances ensures pro-rata distribution

amount1 = _liquidity * _balance1 / _totalSupply; // using balances ensures pro-rata distribution

require(amount0 > 0 &8 amount1 > 0, 'ILB'); // Pair: INSUFFICIENT_LIQUIDITY_BURNED
```

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

SWC-101

Source file

Pair.sol Locations

```
uint _totalSupply = totalSupply; // gas savings, must be defined here since totalSupply can update in _mintFee
amount0 = _liquidity * _balance0 / _totalSupply; // using balances ensures pro-rata distribution
amount1 = _liquidity * _balance1 / _totalSupply; // using balances ensures pro-rata distribution
require(amount0 > 0 &8 amount1 > 0, 'ILB'); // Pair: INSUFFICIENT_LIQUIDITY_BURNED
_burn(address(this), _liquidity);
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

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

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

```
__balance1 = IERC20(_token1).balanceOf(address(this));

solution = __balance0 > __reserve0 - amount0Out ? __balance0 - (_reserve0 - amount0Out) : 0;

uint amount1In = __balance1 > __reserve1 - amount1Out ? __balance1 - (_reserve1 - amount1Out) : 0;

require(amount0In > 0 || amount1In > 0, 'IIA'); // Pair: INSUFFICIENT_INPUT_AMOUNT
```

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

SWC-101

Source file Pair.sol

Locations

```
__balance1 = IERC20(_token1).balanceOf(address(this));

350

351

uint amount0In = _balance0 > _reserve0 - amount0Out ? __balance0 - __reserve0 - amount0Out : 0;

352

uint amount1In = _balance1 > _reserve1 - amount1Out ? _balance1 - (_reserve1 - amount1Out) : 0;

353

require(amount0In > 0 || amount1In > 0, 'IIA'); // Pair: INSUFFICIENT_INPUT_AMOUNT
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
__balance1 = IERC20(_token1).balanceOf(address(this));

350  }

351  uint amount0In = _balance0 > _reserve0 - amount0Out ? _balance0 - (_reserve0 - amount0Out) : 0;

352  uint amount1In = _balance1 > _reserve1 - amount1Out ? _balance1 - (_reserve1 - amount1Out) : 0;

353  require(amount0In > 0 || amount1In > 0, 'IIA'); // Pair: INSUFFICIENT_INPUT_AMOUNT
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

```
uint amount0In = _balance0 > _reserve0 - amount0Out ? _balance0 - (_reserve0 - amount0Out) : 0;
uint amount1In = _balance1 > _reserve1 - amount1Out ? _balance1 - (_reserve1 - amount1Out) : 0;
require(amount0In > 0 || amount1In > 0, 'IIA'); // Pair: INSUFFICIENT_INPUT_AMOUNT
{ // scope for reserve{0,1}Adjusted, avoids stack too deep errors
```

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

SWC-101

Source file Pair.sol

Locations

```
350  }
351  uint amount0In = _balance0 > _reserve0 - amount0Out ? _balance0 - (_reserve0 - amount0Out) : 0;
352  uint amount1In = _balance1 > _reserve1 - amount1Out ? __balance1 - (_reserve1 - amount1Out) : 0;
353  require(amount0In > 0 || amount1In > 0, 'IIA'); // Pair: INSUFFICIENT_INPUT_AMOUNT
354  { // scope for reserve(0,1)Adjusted, avoids stack too deep errors
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
int amount0In = _balance0 > _reserve0 - amount0Out ? _balance0 - (_reserve0 - amount0Out) : 0;

int amount1In = _balance1 > _reserve1 - amount1Out ? _balance1 - (_reserve1 - amount1Out) : 0;

require(amount0In > 0 || amount1In > 0, 'IIA'); // Pair: INSUFFICIENT_INPUT_AMOUNT

/// scope for reserve(0,1)Adjusted, avoids stack too deep errors
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
{ // scope for reserve(0,1)Adjusted, avoids stack too deep errors

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

if (amount0In > 0) _update0(amount0In * PairFactory factory) getFee stable / 10000); // accrue fees for token0 and move them out of pool

if (amount1In > 0) _update1(amount1In * PairFactory(factory).getFee(stable) / 10000); // 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
```

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

SWC-101

Source file Pair.sol

Locations

```
{ // scope for reserve{0,1}Adjusted, avoids stack too deep errors

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

if (amount0In > 0) _update0(amount0In * PairFactory factory getFee stable / 10000); // accrue fees for token0 and move them out of pool

if (amount1In > 0) _update1(amount1In * PairFactory(factory).getFee(stable) / 10000); // 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

| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| Address _token0, address _token1 = (token0, token1); |
| A
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file Pair.sol Locations

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

if (amount0In > 0) _update0(amount0In * PairFactory(factory).getFee(stable) / 10000); // accrue fees for token0 and move them out of pool

if (amount1In > 0) _update1(amount1In * PairFactory factory_getFee(stable) / 10000); // accrue fees for token1 and move them out of pool

_balance0 = IERC20(_token0).balance0f(address(this)); // since we removed tokens, we need to reconfirm balances, can also simply use previous balance - amountIn/ 10000, but doing

balance0 = IERC20(_token1).balance0f(address(this));
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

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

if (amount0In > 0) _update0(amount0In * PairFactory(factory).getFee(stable) / 10000); // accrue fees for token0 and move them out of pool

if (amount1In > 0) _update1(amount1In * PairFactory factory _getFee stable / 10000); // accrue fees for token1 and move them out of pool

_balance0 = IERC20(_token0).balance0f(address(this)); // since we removed tokens, we need to reconfirm balances, can also simply use previous balance - amountIn/ 10000, but doing

balance1 = IERC20(_token1).balance0f(address(this));
```

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

SWC-101

Source file

Pair.sol Locations

```
function skim(address to) external lock {

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

_safeTransfer(_token0, to, __IERC20(_token0 _balanceOf(address(this)) - __reserve0 _);

_safeTransfer(_token1, to, __IERC20(_token1).balanceOf(address(this)) - __(reserve1));

373 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

```
379
380 function _f(uint x0, uint y) internal pure returns (uint) {
381 return x0* y*y/le18*y/le18 /le18* x0*x0/le18*x0/le18 *y/le18;
382 }
383
```

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

SWC-101

Source file

Pair.sol Locations

```
379
380 function _f(uint x0, uint y) internal pure returns (uint) {
381 return x0* y*y/le18*y/le18*y/le18*v0/le18*x0/le18*x0/le18*y/le18;
382 }
383
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
379
380 function _f(uint x0, uint y) internal pure returns (uint) {
381 return x0* y*y/le18*y/le18 /le18+(x0*x0/le18*x0/le18)*y/le18;
382 }
383
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
Locations
```

```
579

580

function _f(uint x0, uint y) internal pure returns (uint) {

return x0*(y*y/1e18*y/1e18)/1e18+(x0*x0/1e18)*y/1e18;

582

583
```

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

SWC-101

Source file

Pair.sol Locations

```
379
380
function _f(uint x0, uint y) internal pure returns (uint) {
return x0*(y'y'/1e18*y'/1e18)/1e18+(x0*x0/1e18*x0/1e18)*y/1e18;
}
382
383
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

```
379

function _f(uint x0, uint y) internal pure returns (uint) {

return x0*(\vec{y}^*\vec{y}^*\vec{y}^*)/1e18*\vec{y}/1e18\vec{y}/1e18*\vec{x0}/1e18*\vec{x0}/1e18\vec{x0}/1e18\vec{y}^*\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\vec{y}/1e18\
```

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

SWC-101

Source file

Pair.sol Locations

```
379
380 function _f(uint x0, uint y) internal pure returns (uint) {
    return x0*(y*y/1e18*y/1e18)/1e18+\u00edx0/\u00edrel8*x0/\u00edrel8*x0/\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00edrel8\u00
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
380 function _f(uint x0, uint y) internal pure returns (uint) {
381 return x0*(y*y/1e18*y/1e18)/1e18+ x0*x0/le18*x0/le18*y/1e18;
382 }
383
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
379
380 function _f(uint x0, uint y) internal pure returns (uint) {
381 return x0*(y*y/1e18*y/1e18)/1e18+(x0*x0/1e18)*y/1e18;
382 }
383
```

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

SWC-101

Source file

Pair.sol Locations

```
379
380 function _f(uint x0, uint y) internal pure returns (uint) {
381 return x0*(y*y/1e18*y/1e18)/1e18+(x0*x0/1e18*x0/1e18)*y/1e18;
382
}
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
379
380 function _f(uint x0, uint y) internal pure returns (uint) {
381 return x0*(y*y/le18*y/le18)/le18*(x0*x0/le18*x0/le18)*y/le18;
382
}
383
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
379

380

function _f(uint x0, uint y) internal pure returns (uint) {

return x0*(y*y/1e18*y/1e18)/1e18*x0/1e18)*y/1e18;

382

383
```

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

SWC-101

Source file

Pair.sol

Locations

```
383 |
384 | function _d(uint x0, uint y) internal pure returns (uint) {
385 | return | 5*x0* | y*y/1e18 | /1e18+ x0*x0/1e18*x0/1e18 |;
386 | }
387
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

```
383 | function _d(uint x0, uint y) internal pure returns (uint) {
385 | return 5*x0* y*y/1e18 /1e18+(x0*x0/1e18*x0/1e18);
386 | }
387
```

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

SWC-101

Source file

Pair.sol Locations

```
383

384

function _d(uint x0, uint y) internal pure returns (uint) {

return 5*x0*(y*y/1e18)/1e18+(x0*x0/1e18*x0/1e18);

386

}
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
383

384 function _d(uint x0, uint y) internal pure returns (uint) {

785 return 3*x0*(y*y/1e18)/1e18+(x0*x0/1e18*x0/1e18);

786 }

387
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
383

384 function _d(uint x0, uint y) internal pure returns (uint) {

385 return 3*x0*(ȳ*ȳ/1e18)/1e18+(x0*x0/1e18*x0/1e18);

386 }

387
```

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

SWC-101

Source file

Pair.sol Locations

```
383

384 function _d(uint x0, uint y) internal pure returns (uint) {

385 return 3*x0*(y*y/le18)/le18+(x0*x0/le18*x0/le18);

386 }

387
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
383 | function _d(uint x0, uint y) internal pure returns (uint) {
385 | return 3*x0*(y*y/1e18)/1e18+(x0*x0/1e18*x0/1e18);
386 | }
387
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
383 | function _d(uint x0, uint y) internal pure returns (uint) {
385 | return 3*x0*(y*y/1e18)/1e18+(x0*x0/1e18*x0/1e18);
386 | }
387
```

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

SWC-101

Source file

Pair.sol Locations

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

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
387

388 function _get_y(uint x0, uint xy, uint y) internal pure returns (uint) {

389 for (uint i = 0; i < 255; i++) {

390 uint y_prev = y;

391 uint k = _f(x0, y);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
Locations
```

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

SWC-101

Source file

Pair.sol Locations

```
391    uint k = _f(x0, y);
392    if (k < xy) {
393       uint dy = _xy - k_*^1e18/_d(x0, y);
394    y = y + dy;
395    } else {</pre>
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
391 | uint k = _f(x0, y);

392 | if (k < xy) {

uint dy = (xy - k)*1e18/_d(x0, y);

394 | y = y + dy;

395 | else {
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

```
392 if (k < xy) {
393    uint dy = (xy - k)*1e18/_d(x0, y);
394    y = y + dy;
395 } else {
396    uint dy = (k - xy)*1e18/_d(x0, y);
```

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

SWC-101

Source file

Pair.sol

Locations

```
394 | y = y + dy;

395 | else {

396 | uint dy = | k - xy| + 1e18/_d(x0 y);

397 | y = y - dy;

398 | }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
394 | y = y + dy;

395 | else {

396 | uint dy = | k - xy| + 1els/_d(x0, y);

397 | y = y - dy;

398 | }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

```
Locations
```

```
394 | y = y + dy;

395 | else {

396 | uint dy = (x - xy)*1e18/_d(x0, y);

397 | y = y - dy;

398 | }
```

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

SWC-101

Source file

Pair.sol Locations

```
395 | } else {
396 | uint dy = (k - xy)*1e18/_d(x0, y);
397 | y = y - dy;
398 | }
399 | if (y > y_prev) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

```
462 | }
463 | } else {
464 | if (y_prev - y <= 1) {
465 | return y;
466 | }
```

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

SWC-101

Source file

Pair.sol Locations

```
function getAmountOut(uint amountIn, address tokenIn) external view returns (uint) {

(uint _reserve0, uint _reserve1) = (reserve1);

amountIn _= amountIn * PairFactory factory getFee stable / 10000; // remove fee from amount received

return _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);

}
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
function getAmountOut(uint amountIn, address tokenIn) external view returns (uint) {

(uint _reserve0, uint _reserve1) = (reserve0, reserve1);

amountIn -= amountIn |* PairFactory factory getFee stable |/ 18080; // remove fee from amount received

return _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);

}
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

```
Locations
```

```
function getAmountOut(uint amountIn, address tokenIn) external view returns (uint) {

(uint _reserve0, uint _reserve1) = (reserve0, reserve1);

amountIn -= amountIn * PairFactory factory. getFee stable / 10000; // remove fee from amount received

return _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);

416 }
```

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

SWC-101

Source file

Pair.sol Locations

```
419  if (stable) {
420    uint xy = _k(_reserve0, _reserve1);
421    _reserve0 = _reserve0 * 1e18 / decimals0;
422    _reserve1 = _reserve1 * 1e18 / decimals1;
423    (uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, _reserve1) : (_reserve0);
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
419  if (stable) {
420    uint xy = _k(_reserve0, _reserve1);
421    __reserve0 = __reserve0 * _le18 / decimals0;
422    __reserve1 = __reserve1 * _le18 / decimals1;
423    (uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, _reserve1) : (_reserve0);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
Locations

420 | uint xy = _k(_reserve0, _reserve1);
```

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

SWC-101

Source file

Pair.sol Locations

```
uint xy = _k(_reserve0, _reserve1);

reserve0 = _reserve0 * 1e18 / decimals0;

reserve1 = _reserve1 * 1e18 / decimals1;

(uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, _reserve1) : (_reserve1, _reserve0);

amountIn = tokenIn == token0 ? amountIn * 1e18 / decimals0 : amountIn * 1e18 / decimals1;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

```
Locations
```

```
422     _reserve1 = _reserve1 * 1e18 / decimals1;
423     (uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, _reserve1) : (_reserve1, _reserve0);
424     amountIn = tokenIn == token0 ? amountIn * 1e18 / decimals0 : amountIn * 1e18 / decimals1;
425     uint y = reserveB - _get_y(amountIn+reserveA, xy, reserveB);
426     return y * (tokenIn == token0 ? decimals1 : decimals0) / 1e18;
```

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

SWC-101

Source file

Pair.sol

Locations

```
422    __reserve1 = __reserve1 * 1e18 / decimals1;
423    (uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, __reserve1) : (_reserve1, __reserve0);
424    amountIn = tokenIn == token0 ? amountIn * 1e18 / decimals0 : amountIn | * 1e18 / decimals1;
425    uint y = reserveB - __get_y(amountIn+reserveA, xy, reserveB);
426    return y * (tokenIn == token0 ? decimals1 : decimals0) / 1e18;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
422    _reserve1 = _reserve1 * 1e18 / decimals1;
423    (uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, _reserve1) : (_reserve1, _reserve0);
424    amountIn = tokenIn == token0 ? amountIn * 1e18 / decimals0 : amountIn * 1e18 / decimals1;
425    uint y = reserveB - _get_y(amountIn+reserveA, xy, reserveB);
426    return y * (tokenIn == token0 ? decimals1 : decimals0) / 1e18;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol Locations

```
(uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, _reserve1) : (_reserve1, _reserve0);

amountIn = tokenIn == token0 ? amountIn * 1e18 / decimals0 : amountIn * 1e18 / decimals1;

uint y = reserveB - __get_y amountIn+reserveA xy reserveB;

return y * (tokenIn == token0 ? decimals1 : decimals0) / 1e18;

} else {
```

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

SWC-101

Source file

Pair.sol Locations

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
amountIn = tokenIn == token0 ? amountIn * 1e18 / decimals0 : amountIn * 1e18 / decimals1;

uint y = reserveB - _get_y(amountIn+reserveA, xy, reserveB);

return y tokenIn == token0 ? decimals1 decimals0 / 1e18;

} else {

(uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, _reserve1) : (_reserve0);
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

```
Locations
```

```
amountIn = tokenIn == token0 ? amountIn * 1e18 / decimals0 : amountIn * 1e18 / decimals1;

uint y = reserveB - _get_y(amountIn+reserveA, xy, reserveB);

return y * tokenIn == token0 ? decimals1 decimals0 / 1e18;

27 } else {

uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, _reserve1) : (_reserve0);
```

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

SWC-101

Source file

Pair.sol Locations

```
427 | } else {
428    (uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, _reserve1) : (_reserve1, _reserve0);
429    return amountIn * reserveB / reserveA + amountIn ;
430    }
431 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

```
Locations
```

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

SWC-101

Source file

Pair.sol Locations

```
function _k(uint x, uint y) internal view returns (uint) {

if (stable) {

uint _x = x * 1e18 / decimals0;

uint _y = y * 1e18 / decimals1;

uint _a = (_x * _y) / 1e18;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
function _k(uint x, uint y) internal view returns (uint) {

if (stable) {

uint _x = x " 1e18 / decimals0;

uint _y = y * 1e18 / decimals1;

uint _a = (_x * _y) / 1e18;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
Locations
```

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

SWC-101

Source file Pair.sol

Locations

```
if (stable) {

uint _x = x * 1e18 / decimals0;

uint _y = y * 1e18 / decimals1;

uint _a = (_x * _y) / 1e18;

uint _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

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

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

```
Locations
```

```
uint _x = x * 1e18 / decimals0;

uint _y = y * 1e18 / decimals1;

uint _a = (_x * _y) / 1e18;

uint _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);

return _a * _b / 1e18; // x3y+y3x >= k
```

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

SWC-101

Source file

Pair.sol Locations

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
436     uint _y = y * 1e18 / decimals1;
437     uint _a = (_x * _y) / 1e18;
438     uint _b = (_x * _x / 1e18 + (_y * _y) / 1e18);
439     return _a * _b / 1e18; // x3y+y3x >= k
440     } else {
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol Locations

```
436     uint _y = y * 1e18 / decimals1;
437     uint _a = (_x * _y) / 1e18;
438     uint _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
439     return _a * _b / 1e18; // x3y+y3x >= k
440     ) else {
```

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

SWC-101

Source file

Pair.sol Locations

```
436     uint _y = y * 1e18 / decimals1;
437     uint _a = (_x * _y) / 1e18;
438     uint _b = ((_x * _x) / 1e18 + _y * _y / 1e18);
439     return _a * _b / 1e18; // x3y+y3x >= k
440     } else {
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
436    uint _y = y * 1e18 / decimals1;
437    uint _a = (_x * _y) / 1e18;
438    uint _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
439    return _a * _b / 1e18; // x3y+y3x >= k
440    } else {
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Pair.sol

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

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

SWC-101

Source file

Pair.sol Locations

```
437    uint _a = (_x * _y) / 1e18;
438    uint _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
439    return _a * _b / 1e18; // x3y+y3x >= k
440    } else {
441    return x * y; // xy >= k
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
439 | return _a * _b / 1e18; // x3y+y3x >= k

440 | } else {

441 | return x * y; // xy >= k

442 | }

443 | }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

```
Locations
```

```
function _mint(address dst, uint amount) internal {
__updateFor(dst); // balances must be updated on mint/burn/transfer

totalSupply |== amount;

balanceOf(dst) += amount;

emit Transfer(address(0), dst, amount);
```

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

SWC-101

Source file

Pair.sol Locations

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
function _burn(address dst, uint amount) internal {
    _updateFor(dst);
    totalSupply -= amount;
    balanceOf[dst] -= amount;

emit Transfer(dst, address(0), amount);
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

```
Locations
```

```
453    _updateFor(dst);
454    totalSupply -= amount;
455    balanceOf dst |-= amount;
456    emit Transfer(dst, address(0), amount);
457 }
```

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

SWC-101

Source file

Pair.sol Locations

```
479 '\x19\x01',
480 DOMAIN_SEPARATOR,
481 keccak256(abi.encode(PERMIT_TYPEHASH, owner, spender, value, nonces owner ++, deadline))
482 )
483 );
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Pair.sol

Locations

```
if (spender != src 88 spenderAllowance != type(uint).max) {
uint newAllowance = spenderAllowance |- amount;
allowance[src][spender] = newAllowance;
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

Pair.sol Locations

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

balanceOf src | -= amount;

balanceOf[dst] += amount;
```

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

SWC-101

Source file

Pair.sol Locations

```
514
515
     balanceOf[src] -= amount;
516
     balanceOf[dst] += amount;
517
     emit Transfer(src, dst, amount);
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

Pair.sol Locations

```
120
```

```
function lastObservation() public view returns (Observation memory) {
122 return observations[observations.length-1];
123
124
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

Pair.sol Locations

```
269 | uint[] memory _prices = new uint[](points);
270
     uint length = observations.length-1;
     uint i = length - (points * window);
272
     uint nextIndex = 0;
```

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

SWC-101

Source file

libraries/Math.sol

Locations

```
11 | if (y > 3) {
12 | z = y;
13 | uint x = y / 2 + 1;
14 | white (x < z) {
15 | z = x;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

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

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

libraries/Math.sol

```
14 | while (x < z) {
15 | z = x;
16 | x = | y / x + x | / 2;
17 | }
18 | } else if (y != 0) {
```

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

SWC-101

Source file

libraries/Math.sol

Locations

```
14 | while (x < z) {
15 | z = x;
16 | x = (y / | x + x) / 2;
17 | }
18 | } else if (y != 0) {
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
14  while (x < z) {
15  z = x;
16  x = (y / x + x) / 2;
17  }
18  } else if (y != 0) {</pre>
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

libraries/Math.sol

```
for (uint256 y = 1 << 255; y > 0; y >>= 3) {
    x <<= 1;
    uint256 z = 3 * x * (x + 1) + 1;
    if (n / y >= z) {
        n -= y * z;
    }
```

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

SWC-101

Source file

libraries/Math.sol

Locations

```
24 | for (uint256 y = 1 << 255; y > 0; y >>= 3) {
25 | x <<= 1;
26 | uint256 z = 5 * x * x * x + 1 + 1;
27 | if (n / y >= z) {
28 | n -= y * z;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
for (uint256 y = 1 << 255; y > 0; y >>= 3) {
    x <<= 1;
    uint256 z = 3 * x * (x + 1) + 1;
    if (n / y >= z) {
        n == y * z;
    }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

libraries/Math.sol

```
24  for (uint256 y = 1 << 255; y > 0; y >>= 3) {
25     x <<= 1;
26     uint256 z = 3 * x * (x + 1) + 1;
27     if (n / y >= z) {
28         n -= y * z;
```

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

SWC-101

Source file

libraries/Math.sol

Locations

```
25 | x <<= 1;

26 | uint256 z = 3 * x * (x + 1) + 1;

27 | if (n/y >= z) {

28 | n -= y * z;

29 | x += 1;
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
26 | uint256 z = 3 * x * (x + 1) + 1;

27 | if (n / y >= z) {

28 | n -= y |* z;

29 | x += 1;

30 | }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

libraries/Math.sol

```
26 | uint256 z = 3 * x * (x + 1) + 1;

27 | if (n / y >= z) {

28 | n -= y * z;

29 | x += 1;

30 | }
```

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

SWC-101

Source file

libraries/Math.sol

Locations

```
27 | if (n / y >= z) {
28 | n -= y * z;
29 | x += 1;
30 | }
31 | }
```

LOW State variable visibility is not set.

It is best practice to set the visibility of state variables explicitly. The default visibility for "factory" is internal. Other possible visibility settings are public and private.

SWC-108

Source file

Pair.sol

Locations

```
address public immutable token1;
address public immutable fees;
address immutable factory;

// Structure to capture time period obervations every 30 minutes, used for local oracles
```

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

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

SWC-110

Source file

Pair.sol

```
uint constant periodSize = 1800;

d6

d7
   Observation    public observations;

d8

d9   uint internal immutable decimals0;
```

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

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

SWC-110

Source file

factories/PairFactory.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 Out of bounds array access

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

SWC-110

Source file

Pair.sol

Locations

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

124
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Pair.sol

```
(uint reserve0Cumulative, uint reserve1Cumulative,) = currentCumulativePrices();
if (block.timestamp == _observation.timestamp) {
    _observation = observations observations length-2;
}
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

```
Pair.sol
Locations
```

```
256 | uint priceAverageCumulative;
     for (uint i = 0; i < _prices.length; i++) {
258  priceAverageCumulative += _prices[i];
259
     return priceAverageCumulative / granularity;
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Pair.sol

Locations

```
276 | for (; i < length; i+=window) {
     nextIndex = i + window;
278 uint timeElapsed = observations[nextIndex].timestamp - observations[i].timestamp;
uint _reserve0 = (observations[nextIndex].reserve0Cumulative - observations[i].reserve0Cumulative) / timeElapsed;
uint _reserve1 = (observations[nextIndex].reserve1Cumulative - observations[i].reserve1Cumulative) / timeElapsed;
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

```
Locations
```

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

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Pair.sol

Locations

```
nextIndex = i + window;

uint timeElapsed = observations[nextIndex].timestamp - observations[i].timestamp;

uint _reserve0 = (observations nextIndex_.reserve0Cumulative - observations[i].reserve0Cumulative) / timeElapsed;

uint _reserve1 = (observations[nextIndex].reserve1Cumulative - observations[i].reserve1Cumulative) / timeElapsed;

_prices[index] = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Pair.sol

Locations

```
nextIndex = i + window;

uint timeElapsed = observations[nextIndex].timestamp - observations[i].timestamp;

uint _reserve0 = (observations[nextIndex].reserve0Cumulative - observations_i _reserve0Cumulative) / timeElapsed;

uint _reserve1 = (observations[nextIndex].reserve1Cumulative - observations[i].reserve1Cumulative) / timeElapsed;

_prices[index] = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Pair.sol

```
uint timeElapsed = observations[nextIndex].timestamp - observations[i].timestamp;
uint _reserve0 = (observations[nextIndex].reserve0Cumulative - observations[i].reserve0Cumulative) / timeElapsed;
uint _reserve1 = (observations nextIndex_.reserve1Cumulative - observations[i].reserve1Cumulative) / timeElapsed;
prices[index] = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
index = index + 1;
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Pair.sol

```
Locations
```

```
uint timeElapsed = observations[nextIndex].timestamp - observations[i].timestamp;
uint _reserve0 = (observations[nextIndex].reserve0Cumulative - observations[i].reserve0Cumulative) / timeElapsed;
uint _reserve1 = (observations[nextIndex].reserve1Cumulative - observations i _.reserve1Cumulative) / timeElapsed;
_prices[index] = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
index = index + 1;
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Pair.sol

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

prices index = _getAmountOut(amountIn, tokenIn, _reserve0, _reserve1);
index = index + 1;
}
```