

### REPORT 61F78CD54594AB0018BEA2B0

Created Mon Jan 31 2022 07:16:37 GMT+0000 (Coordinated Universal Time)

Number of analyses 1

User 61f52e351fd393a0c51a34fe

# **REPORT SUMMARY**

Analyses ID	Main source file	Detected
		vulnerabilities

72627795-433d-4668-a9bb-0542f1183232

core.sol

0

Started Mon Jan 31 2022 07:16:41 GMT+0000 (Coordinated Universal Time)

Finished Mon Jan 31 2022 07:16:46 GMT+0000 (Coordinated Universal Time)

Mode Deep

Client Tool Remythx

Main Source File Core.Sol

#### **DETECTED VULNERABILITIES**

(HIGH	(MEDIUM	(LOW	
0	0	0	

#### **ISSUES**

```
UNKNOWN Arithmetic operation "+" discovered
```

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

SWC-101

Source file core.sol Locations

20 | if (y > 3) { 21 | z = y;

22 uint x = y / 2 + 1; 23 while (x < z) {

23 | while (x < z) { 24 | z = x;

### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol Locations

```
20 | if (y > 3) {
21 | z = y;
22 | uint x = y / 2 + 1;
23 | while (x < z) {
24 | z = x;
```

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

SWC-101

Source file

core.sol Locations

```
23 | while (x < z) {
24 | z = x;
25 | x = | y / x + x) / 2;
26 | }
27 | } else if (y != 0) {
```

### UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

core.sol

Locations

```
23 | white (x < z) {
24 | z = x;
25 | x = (y / x + x) / 2;
26 | }
27 | } else if (y != 0) {
```

## UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol

Locations

```
25 | while (x < z) {
24 | z = x;
25 | x = (y / x + x) / 2;
26 | }
27 | } else if (y != 0) {
```

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

SWC-101

Source file

core.sol Locations

```
84 | mapping(address => uint) public nonces;
85
    uint internal constant MINIMUM_LIQUIDITY = 10**3;
87
    event Transfer(address indexed from, address indexed to, uint amount);
```

## UNKNOWN Arithmetic operation "\*\*" discovered

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

SWC-101

Source file

core.sol

Locations

```
154 }
155
156 | decimals0 = 10**erc20(_token0).decimals();
     decimals1 = 10**erc20(_token1).decimals();
158
```

### UNKNOWN Arithmetic operation "\*\*" discovered

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

SWC-101

Source file

```
Locations
```

```
155
156
    decimals0 = 10**erc20(_token0).decimals();
     decimals1 = 10**erc20(_token1).decimals();
158
     observations. \\ push(Observation(block.timestamp, \ 0, \ 0)); \\
```

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

SWC-101

Source file

core.sol Locations

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

178
```

### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol

Locations

## UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.sol

Locations

```
function _update0(uint amount) internal {
    _safeTransfer(token0, fees, amount); // transfer the fees out to BaseV1Fees
    uint256 _ratio = amount * le18 / totalSupply; // le18 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 core.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

core.sol

Locations

```
function _update1(uint amount) internal {
    _safeTransfer(token1, fees, amount);
    uint256 _ratio = amount * 1e18 / 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

```
function _update1(uint amount) internal {

_safeTransfer(token1, fees, amount);

uint256 _ratio = amount |* le18 / totalSupply;

if (_ratio > 0) {

uindex1 += _ratio;
```

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

SWC-101

Source file

core.sol Locations

```
219     uint256 _ratio = amount * 1e18 / totalSupply;
220     if (_ratio > 0) {
221         index1 += _ratio;
222     }
223     emit Fees(msg, sender, 0, amount);
```

## UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

core.sol

Locations

```
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) {
```

## UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

```
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
```

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

SWC-101

Source file

core.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

core.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

core.sol

Locations

```
if (_delta0 > 0) {

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

claimable0 recipient | := _share;

if (_delta1 > 0) {
```

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

SWC-101

Source file

core.sol Locations

### UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.sol

Locations

```
242 | }
243 if (_delta1 > 0) {
244     uint _share = _supplied * _delta1 / 1e18;
245     claimable1[recipient] += _share;
246     }
```

# UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

core.sol Locations

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

SWC-101

Source file core.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 > 0.58 _reserve0 != 0.58 _reserve1 != 0) {
reserve0CumulativeLast += _reserve0 * timeElapsed;
```

#### UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

core.sol

Locations

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

if (timeElapsed > 0.58 _reserve0 != 0.58 _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

```
uint timeElapsed = blockTimestamp - blockTimestamplast; // overflow is desired
if (timeElapsed > 0 &δ _reserve0 != 0 &δ _reserve1 != 0) {
reserve0Cumulativelast += _reserve0 * timeElapsed;
reserve1Cumulativelast += _reserve1 * timeElapsed;
}
```

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

SWC-101

Source file core.sol Locations

```
reserve0CumulativeLast += _reserve0 * timeElapsed;
```

```
263 | if (timeElapsed > 0 88 _reserve0 != 0 88 _reserve1 != 0) {
265
     reserve1CumulativeLast += _reserve1 * timeElapsed;
266
```

## UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file core.sol

Locations

```
263 | if (timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0) {
     reserve0CumulativeLast += _reserve0 * timeElapsed;
     reserve1CumulativeLast += <mark>_reserve1 * timeElapsed</mark>;
265
266
267
```

## UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

core.sol Locations

```
267
    Observation memory _point = lastObservation();
268
     timeElapsed = blockTimestamp - _point timestamp; // compare the last observation with current timestamp, if greater than 30 minutes, record a new event
     if (timeElapsed > periodSize) {
270
     observations, \\ push (Observation (blockTimestamp, \ reserve0 CumulativeLast, \ reserve1 CumulativeLast)); \\
```

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

SWC-101

Source file

core.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

core.sol

Locations

```
// subtraction overflow is desired

uint timeElapsed = blockTimestamp - _blockTimestampLast;

reserve0Cumulative += _reserve0 * timeElapsed;

reserve1Cumulative += _reserve1 * timeElapsed;

292

}
```

### UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.sol

Locations

```
// 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 core.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

core.sol

Locations

```
uint timeElapsed = blockTimestamp - _blockTimestampLast;
reserveOcumulative += _reserveO * timeElapsed;
reserveIcumulative += _reserveI * timeElapsed;
}
```

### UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

```
Locations
```

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

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

SWC-101

Source file

```
core.sol
Locations
```

```
301
302
303
uint timeElapsed = block timestamp - _observation timestamp;
304
uint _reserve0 = (reserve0Cumulative - _observation.reserve0Cumulative) / timeElapsed;
305
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 core.sol Locations

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

uint _reserve0 = _reserve0Cumulative - _observation.reserve0Cumulative / timeElapsed;

uint _reserve1 = (reserve1Cumulative - _observation.reserve1Cumulative) / 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 core.sol

```
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 core.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

core.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

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

SWC-101

Source file

core.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

core.sol

Locations

```
priceAverageCumulative += _prices[i];

priceAverageCumulative / granularity;

return priceAverageCumulative / granularity;

317
318
```

### UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

```
Locations
```

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

uint length = observations length-1;

uint i = length - (points * window);

uint nextIndex = 0;
```

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

SWC-101

Source file

core.sol Locations

```
326
327  uint length = observations.length-1;
328  uint i = length - points * window;
329  uint nextIndex = 0;
330  uint index = 0;
```

### UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.sol

Locations

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

#### UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

```
Locations
```

```
uint index = 0;

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

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

SWC-101

Source file core.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

core.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;

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

#### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
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);
```

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

SWC-101

Source file core.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

core.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 Arithmetic operation "-" discovered

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

SWC-101

Source file

core.sol

Locations

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

SWC-101

Source file

core.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

core.sol

Locations

```
uint _balance0 = erc20(token0).balanceOf(address(this));
uint _balance1 = erc20(token1).balanceOf(address(this));
uint _amount0 = _balance0 - _reserve0;
uint _amount1 = _balance1 - _reserve1;
```

## UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

```
Locations
```

```
uint _balance1 = erc20(token1).balance0f(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
```

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

SWC-101

Source file

core.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

core.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

```
Locations
```

```
__mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens

set iquidity = Math.min(_amount0 * _totalSupply / _reserve0, _amount1 * _totalSupply / _reserve1);

require(liquidity > 0, 'ILM'); // BaseV1: INSUFFICIENT_LIQUIDITY_MINTED
```

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

SWC-101

Source file

core.sol

```
Locations
```

```
__mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens

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

require(liquidity > 0, 'ILM'); // BaseV1: INSUFFICIENT_LIQUIDITY_MINTED
```

### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.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 core.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 88 amount1 > 0, 'ILB'); // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
```

### UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.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 56 amount1 > 0, 'ILB'); // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
```

#### 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
amount0 = _liquidity * _balance0 / _totalSupply; // using balances ensures pro-rata distribution
amount1 = _liquidity * _balance1 / _totalSupply; // using balances ensures pro-rata distribution
require(amount0 > 0 88 amount1 > 0, 'ILB'); // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
_burn(address(this), _liquidity);
```

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

SWC-101

Source file

core.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 88 amount1 > 0, 'ILB'); // BaseV1: 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 core.sol

Locations

```
_balance1 = erc20(_token1).balanceOf(address(this));

}

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

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

require(amount0In > 0 || amount1In > 0, 'IIA'); // BaseV1: INSUFFICIENT_INPUT_AMOUNT
```

#### UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file core.sol

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

SWC-101

Source file core.sol

### UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file core.sol Locations

```
405 | }
406 | uint amount0In = _balance0 > _reserve0 - amount0Out ? _balance0 - (_reserve0 - amount0Out) : 0;
407 | uint amount1In = _balance1 > _reserve1 - amount1Out ? _balance1 - (_reserve1 - amount1Out) : 0;
408 | require(amount0In > 0 || amount1In > 0, 'IIA'); // BaseV1: INSUFFICIENT_INPUT_AMOUNT
409 | { // 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

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

SWC-101

Source file

core.sol

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

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

require(amount0In > 0 || amount1In > 0, 'IIA'); // BaseV1: 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

core.sol

Locations

```
409 { // scope for reserve{0,1}Adjusted, avoids stack too deep errors
410 (address _token0, address _token1) = (token0, token1);
411 if (amount0In > 0) _update0(amount0In / 10000); // accrue fees for token0 and move them out of pool
412 if (amount1In > 0) _update1(amount1In / 10000); // accrue fees for token1 and move them out of pool
413 _balance0 = erc20(_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 Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol Locations

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

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

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

_balance0 = erc20(_token0).balance0f(address(this)); // since we removed tokens, we need to reconfirm balances, can also simply use previous balance - amountIn/ 10000, but doing

balance0 = erc20(_token1).balance0f(address(this));
```

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

SWC-101

Source file

core.sol Locations

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

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

_safeTransfer(_token0, to, erc20(_token0) balanceOf(address(this)) - (reserve0);

_safeTransfer(_token1, to, erc20(_token1).balanceOf(address(this)) - (reserve1));

}
```

### UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

core.sol

Locations

#### UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

```
Locations
```

```
434
435

function _f(uint x0, uint y) internal pure returns (uint) {
436

return x0* y*y/le18*y/le18 /le18* x0*x0/le18*x0/le18 *y/le18;
437

}
```

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

SWC-101

Source file

core.sol Locations

```
434
435
function _f(uint x0, uint y) internal pure returns (uint) {
return x0° y°y/1e18°y/le18 /le18*x0/le18*x0/le18)*y/le18;
}
437
438
```

### UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.sol

Locations

```
434
435 function _f(uint x0, uint y) internal pure returns (uint) {
436 return x0° y°y/1e18°y/1e18 /1e18+(x0°x0/1e18°x0/1e18)°y/1e18;
437
438
```

### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
Locations
```

```
434
435 function _f(uint x0, uint y) internal pure returns (uint) {
436 return x0*(\frac{y}{y}^*\frac{y}{1e18}^*\frac{y}{1e18})/1e18+(x0*x0/1e18)*\frac{y}{1e18};
437 }
438
```

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

SWC-101

Source file

core.sol Locations

```
434
435 function _f(uint x0, uint y) internal pure returns (uint) {
436 return x0*(\frac{y}{y}^*\frac{y}{1e18}^*\frac{y}{2}/1e18)/1e18+(x0*x0/1e18*x0/1e18)*y/1e18;
437 }
438
```

### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol

Locations

```
434
435 function _f(uint x0, uint y) internal pure returns (uint) {
436 return x0*(y*y/le18**y/le18)/le18*x0/le18*x0/le18)*y/le18;
437 }
438
```

### UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

```
Locations
```

```
434
435 function _f(uint x0, uint y) internal pure returns (uint) {
436 return x0*(y*y/1e18*y/1e18)/1e18+(x0*x0/1e18*x0/1e18)*y/1e18;
437
438
```

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

SWC-101

Source file

core.sol

```
Locations
```

```
434
435
function _f(uint x0, uint y) internal pure returns (uint) {
7436
return x0*(y*y/le18*y/le18)/le18+ x0*x0/le18*x0/le18*y/le18;
7437
438
```

### UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.sol

Locations

```
434
435 function _f(uint x0, uint y) internal pure returns (uint) {
436 return x0*(y*y/1e18*y/1e18)/1e18+ x0*x0/[e18*x0/le18]*y/1e18;
437
438
```

### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol Locations

```
434 435 function _f(uint x0, uint y) internal pure returns (uint) {
436    return x0*(y*y/le18*y/le18)/le18+(x0*x0/le18*x0/le18)*y/le18;
437    }
438
```

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

SWC-101

Source file

core.sol Locations

```
434
435 function _f(uint x0, uint y) internal pure returns (uint) {
436 return x0*(y*y/1e18*y/1e18)/1e18+(x0*x0/1e18)*y/1e18;
437 }
438
```

### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol

Locations

```
434
435 function _f(uint x0, uint y) internal pure returns (uint) {
return x0*(y*y/1e18*y/1e18)/1e18+(x0*x0/1e18)*y/1e18;
437
}
```

### UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

```
Locations
```

```
434
435 function _f(uint x0, uint y) internal pure returns (uint) {
436 return x0*(y*y/1e18*y/1e18)/1e18*x0/1e18*x0/1e18)*y/1e18;
437
438
```

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

SWC-101

Source file

core.sol Locations

```
439
440
function _d(uint x0, uint y) internal pure returns (uint) {
441
return 3°x0° y°y/Te18 /Te18 x0°x0/Te18 x0°x0/Te18;
}
442
443
```

### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol

Locations

```
439

440

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

return 5*x0* y*y/le18 /le18+(x0*x0/le18*x0/le18);

}

442

443
```

### UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.sol

Locations

```
439

440

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

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

442

443
```

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

SWC-101

Source file

core.sol Locations

```
439
440 function _d(uint x0, uint y) internal pure returns (uint) {
441 return 5*x0*(y*y/1e18)/1e18+(x0*x0/1e18*x0/1e18);
442
443
```

### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol

Locations

```
439
440
function _d(uint x0, uint y) internal pure returns (uint) {
return 3*x0*(y*y/1e18)/1e18+(x0*x0/1e18*x0/1e18);
}

442
443
```

# UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

```
Locations
```

```
439
440 function _d(uint x0, uint y) internal pure returns (uint) {
441 return 3*x0*(y*y')/1e18)/1e18+(x0*x0/1e18*x0/1e18);
442
443
```

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

SWC-101

Source file

core.sol Locations

```
439
440 function _d(uint x0, uint y) internal pure returns (uint) {
441 return 3*x0*(y*y/le18)/le18*(x0*x0/le18*x0/le18);
442
443
```

### UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.sol

Locations

```
439
440 function _d(uint x0, uint y) internal pure returns (uint) {
441 return 3*x0*(y*y/1e18)/1e18+(x0*x0/1e18*x0/1e18);
}
442
443
```

### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol Locations

443

```
439 | function _d(uint x0, uint y) internal pure returns (uint) {
441 | return 3*x0*(y*y/1e18)/1e18+(x0*x0/1e18*x0/1e18);
442 | }
```

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

SWC-101

Source file

core.sol Locations

```
439
440
function _d(uint x0, uint y) internal pure returns (uint) {
441
return 3*x0*(y*y/le18)/le18*(x0*x0/le18);
}
442
443
```

### UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

core.sol

Locations

```
443

444

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

445

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

446

uint y_prev = y;

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

### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

y = y + dy;

} else {

Source file

core.sol Locations

450

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

448 | if (k < xy) {

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

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

SWC-101

Source file

core.sol

Locations

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

448 | if (k < xy) {

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

450 | y = y + dy;

451 | } else {
```

### UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

core.sol

Locations

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

448 | if (k < xy) {

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

450 | y = y + dy;

451 | } else {
```

## UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

core.sol

Locations

```
448 if (k < xy) {
449    uint dy = (xy - k)*1e18/_d(x0, y);
450    y = y + dy;
451 } else {
452    uint dy = (k - xy)*1e18/_d(x0, y);
```

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

SWC-101

Source file

core.sol

Locations

```
450 | y = y + dy;

451 | } else {

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

453 | y = y - dy;

454 | }
```

## UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.sol

Locations

```
450 | y = y + dy;

451 | } else {

452 | uint dy = | k - | xy | *| tels / _ d(x0, y);

453 | y = y - dy;

454 | }
```

# 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

core.sol

```
Locations
```

## UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

core.sol

Locations

```
454 | }
455 | if (y > y_prev) {
456 | if (y - y_prev <= 1) {
457 | return y;
458 | }
```

# UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

core.sol

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

SWC-101

Source file

core.sol Locations

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

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

amountIn _= amountIn / 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

core.sol

Locations

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

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

amountIn -= amountIn / 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

core.sol

```
475     if (stable) {
476         uint xy = _k(_reserve0, _reserve1);
477         _reserve0 = __reserve0 * lel8 / decimals0;
478         _reserve1 = _reserve1 * lel8 / decimals1;
479         (uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, _reserve1) : (_reserve0);
```

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

SWC-101

Source file

core.sol Locations

## UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol

Locations

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

reserve0 = _reserve0 * le18 / decimals0;

reserve1 = _reserve1 * le18 / decimals1;

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

amountIn = tokenIn == token0 ? amountIn * le18 / decimals0 : amountIn * le18 / decimals1;
```

#### UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.sol

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

reserve0 = _reserve0 * le18 / decimals0;

reserve1 = _reserve1 * le18 / decimals1;

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

amountIn = tokenIn == token0 ? amountIn * le18 / decimals0 : amountIn * le18 / decimals1;
```

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

SWC-101

Source file

core.sol Locations

```
__reserve1 = __reserve1 * 1e18 / decimals1;

479    (uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, __reserve1) : (_reserve1, __reserve0);

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

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

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

core.sol

Locations

```
__reserve1 = __reserve1 * 1e18 / decimals1;

(uint reserveA, uint reserveB) = tokenIn == token0 ? (__reserve0, __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;
```

#### UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol

```
__reserve1 = __reserve1 * 1e18 / decimals1;

dint reserveA, uint reserveB) = tokenIn == token0 ? (__reserve0, __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;
```

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

SWC-101

Source file

core.sol Locations

```
_reserve1 = _reserve1 * 1e18 / decimals1;

(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;
```

## UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

core.sol

```
Locations
```

```
479  (uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve1) : (_reserve1) : (_reserve1);
480  amountIn = tokenIn == token0 ? amountIn * 1e18 / decimals0 : amountIn * 1e18 / decimals1;
481  uint y = reserveB - _get_y(amountIn+reserveA xy reserveB);
482  return y * (tokenIn == token0 ? decimals1 : decimals0) / 1e18;
483  } else {
```

#### UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

```
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

core.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) : (_reserve1, _reserve0);
```

## UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.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

```
483 } else {
484 (uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve0, _reserve1) : (_reserve1, _reserve0);
485 return amountIn * reserveB / reserveA + amountIn ;
486 }
487 }
```

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

SWC-101

Source file

core.sol Locations

## UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

core.sol

Locations

```
483 | } else {
484    (uint reserveA, uint reserveB) = tokenIn == token0 ? (_reserve1) : (_reserve1, _reserve0);
485    return amountIn * reserveB / (reserveA + amountIn);
486    }
487 }
```

## UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

```
Locations
```

```
489     function _k(uint x, uint y) internal view returns (uint) {
490     if (stable) {
491         uint _x = x * 1e18 / decimals0;
492         uint _y = y * 1e18 / decimals1;
493         uint _a = (_x * _y) / 1e18;
```

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

SWC-101

Source file

core.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

core.sol

Locations

```
490     if (stable) {
491         uint _x = x * 1e18 / decimals0;
492         uint _y = y * 1e18 / decimals1;
493         uint _a = (_x * _y) / 1e18;
494         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

```
490 | if (stable) {

491     uint _x = x * 1e18 / decimals0;

492     uint _y = y * 1e18 / decimals1;

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

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

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

SWC-101

Source file

core.sol Locations

```
491    uint _x = x * 1e18 / decimals0;
492    uint _y = y * 1e18 / decimals1;
493    uint _a = _x * _y / 1e18;
494    uint _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
495    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

core.sol

Locations

```
491    uint _x = x * 1e18 / decimals0;
492    uint _y = y * 1e18 / decimals1;
493    uint _a = (_x * _y) / 1e18;
494    uint _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
495    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
```

```
492    uint _y = y * 1e18 / decimals1;
493    uint _a = (_x * _y) / 1e18;
494    uint _b = ( _x * _x / 1e18 + (_y * _y) / 1e18);
495    return _a * _b / 1e18; // x3y+y3x >= k
496    } else {
```

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

SWC-101

Source file

core.sol Locations

## UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

core.sol

Locations

```
492 | uint _y = y * 1e18 / decimals1;

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

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

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

496 | } else {
```

## UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

} else {

Source file

```
492 | uint _y = y * 1e18 / decimals1;

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

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

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

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

SWC-101

Source file

core.sol Locations

```
492     uint _y = y * 1e18 / decimals1;
493     uint _a = (_x * _y) / 1e18;
494     uint _b = ((_x * _x) / 1e18 + (_y * _y) / 1e18);
495     return _a * _b / 1e18; // x3y+y3x >= k
496     } else {
```

## UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

core.sol

Locations

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

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

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

496 | } else {

497 | return x * y; // xy >= k
```

## UNKNOWN Arithmetic operation "\*" discovered

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

SWC-101

Source file

```
Locations
```

```
493     uint _a = (_x * _y) / le18;
494     uint _b = ((_x * _x) / le18 + (_y * _y) / le18);
495     return _a * _b / le18; // x3y+y3x >= k
496     } else {
497     return x * y; // xy >= k
```

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

SWC-101

Source file

core.sol

```
Locations
```

```
495 | return _a * _b / 1e18; // x3y+y3x >= k
     } else {
496
     return x * y; // xy >= k
498
```

## UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

core.sol

Locations

```
501 | function _mint(address dst, uint amount) internal {
     _updateFor(dst); // balances must be updated on mint/burn/transfer
     totalSupply += amount;
503
     balanceOf[dst] += amount;
     emit Transfer(address(0), dst, amount);
```

#### UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

```
502 | _updateFor(dst); // balances must be updated on mint/burn/transfer
    totalSupply += amount;
```

```
balanceOf[dst] += amount;
     emit Transfer(address(0), dst, amount);
505
```

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

SWC-101

Source file

core.sol Locations

## UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

core.sol

Locations

## UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

```
Locations
```

```
535 '\x19\x01',
536 DOMAIN_SEPARATOR,
537 keccak256(abi.encode(PERMIT_TYPEHASH, owner, spender, value, nonces owner ++, deadline))
538 )
539 );
```

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

SWC-101

Source file

core.sol Locations

```
if (spender != src && 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

core.sol

Locations

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

570

balanceOf[src] -= amount;

balanceOf[dst] += amount;

573
```

#### UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

```
570
571 balanceOf[src] -= amount;
572 balanceOf dst |+= amount;
573
574 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

core.sol Locations

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

178
```

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

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

SWC-101

Source file

core.sol

Locations

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

uint length = observations length-1;

uint i = length - (points * window);

uint nextIndex = 0;
```

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

The public state variable "observations" in "BaseV1Pair" contract has type "struct BaseV1Pair.Observation[]" and can cause an exception in case of use of invalid array index value.

SWC-110

Source file

```
Locations
```

```
uint constant periodSize = 1800;

104

105

Observation public observations;

106

107

uint internal immutable decimals0;
```

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

The public state variable "allPairs" in "BaseV1Factory" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

SWC-110

Source file

core.sol

#### 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

core.sol

Locations

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

178
```

#### 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

core.sol

#### 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

core.sol

Locations

```
uint priceAverageCumulative;
for (uint i = 0; i < _prices.length; i++) {
    priceAverageCumulative += _prices i ;
}
return priceAverageCumulative / granularity;</pre>
```

#### 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

core.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;

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

#### 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

core.sol

```
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;
</pre>
```

## 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

core.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

core.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

core.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

core.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

core.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;
}
```