MythX

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

Analysis 72627795-433d-4668-a9bb-0542f1183232

MythX

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

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

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN
SWC-101

Arithmetic operation "**" discovered

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

Source file

`core.sol`

Locations

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

uint internal constant MINIMUM_LIQUIDITY = 10**3;

event Transfer(address indexed from, address indexed to, uint amount);
```

UNKNOWN
SWC-101

Arithmetic operation "**" discovered

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

Source file

`core.sol`

Locations

```
}

decimals0 = 10**erc20(_token0).decimals();
decimals1 = 10**erc20(_token1).decimals();

```

UNKNOWN
SWC-101

Arithmetic operation "**" discovered

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

Source file

`core.sol`

Locations

```

decimals0 = 10**erc20(_token0).decimals();
decimals1 = 10**erc20(_token1).decimals();

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

UNKNOWN SWC-101

Arithmetic operation "-" discovered

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

Source file

`core.sol`

Locations

```

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

```

UNKNOWN SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN Arithmetic operation "+=" discovered

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

core.sol

Locations

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

UNKNOWN SWC-101

Arithmetic operation "+=" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN SWC-101

Arithmetic operation "-" discovered

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

Source file

`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 SWC-101

Arithmetic operation "-" discovered

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

Source file

`core.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

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`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 && _reserve0 != 0 && _reserve1 != 0) {
reserve0CumulativeLast += _reserve0 * timeElapsed;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

core.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

core.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

SWC-101

Arithmetic operation "-" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

core.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`core.sol`

Locations

```
}

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

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

SWC-101

Arithmetic operation "-" discovered

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

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

SWC-101

Arithmetic operation "/" discovered

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

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

SWC-101

Arithmetic operation "-" discovered

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

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

SWC-101

Arithmetic operation "++" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

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 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

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

core.sol

Locations

```
uint index = 0;

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`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;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

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;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

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

SWC-101

Arithmetic operation "-" discovered

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

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

SWC-101

Arithmetic operation "/" discovered

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

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

SWC-101

Arithmetic operation "-" discovered

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

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

SWC-101

Arithmetic operation "+" discovered

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

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

SWC-101

Arithmetic operation "-" discovered

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

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

SWC-101

Arithmetic operation "-" discovered

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

Source file

`core.sol`

Locations

```
uint _balance1 = erc20(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

SWC-101

Arithmetic operation "-" discovered

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

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

SWC-101

Arithmetic operation "*" discovered

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

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

SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

```
_mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens
} else {
liquidity = Math.min(_amount0 * _totalSupply / _reserve0, _amount1 * _totalSupply / _reserve1);
}
require(liquidity > 0, 'ILM'); // BaseV1: INSUFFICIENT_LIQUIDITY_MINTED
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

```
_mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens
} else {
liquidity = Math.min(_amount0 * _totalSupply / _reserve0, _amount1 * _totalSupply / _reserve1);
}
require(liquidity > 0, 'ILM'); // BaseV1: INSUFFICIENT_LIQUIDITY_MINTED
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.sol

Locations

```
_mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens
} else {
liquidity = Math.min(_amount0 * _totalSupply / _reserve0, _amount1 * _totalSupply / _reserve1);
}
require(liquidity > 0, 'ILM'); // BaseV1: INSUFFICIENT_LIQUIDITY_MINTED
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

```
_mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens
} else {
liquidity = Math.min(_amount0 * _totalSupply / _reserve0, _amount1 * _totalSupply / _reserve1);
}
require(liquidity > 0, 'ILM'); // BaseV1: INSUFFICIENT_LIQUIDITY_MINTED
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

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 && amount1 > 0, 'ILB'); // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`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 && amount1 > 0, 'ILB'); // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`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 && amount1 > 0, 'ILB'); // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
_burn(address(this), _liquidity);
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

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 && amount1 > 0, 'ILB'); // BaseV1: INSUFFICIENT_LIQUIDITY_BURNED
_burn(address(this), _liquidity);
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

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

SWC-101

Arithmetic operation "-" discovered

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

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

SWC-101

Arithmetic operation "-" discovered

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

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

SWC-101

Arithmetic operation "-" discovered

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

Source file

`core.sol`

Locations

```
}
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

SWC-101

Arithmetic operation "-" discovered

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

Source file

`core.sol`

Locations

```
}
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

SWC-101

Arithmetic operation "-" discovered

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

Source file

core.sol

Locations

```
}
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

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.sol

Locations

```
{ // scope for reserve{0,1}Adjusted, avoids stack too deep errors
(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).balanceOf(address(this)); // since we removed tokens, we need to reconfirm balances, can also simply use previous balance - amountIn/ 10000, but doing
balanceOf again as safety check
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

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

SWC-101

Arithmetic operation "-" discovered

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

Source file

core.sol

Locations

```
(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

SWC-101

Arithmetic operation "+" discovered

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

Source file

core.sol

Locations

```

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;
}

```

UNKNOWN SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

```

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;
}

```

UNKNOWN SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.sol`

Locations

```

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;
}

```

UNKNOWN SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

```

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;
}

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.sol`

Locations

```

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;
}

```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

```

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;
}

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.sol`

Locations

```

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;
}

```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.sol

Locations

```

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;
}

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

```

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;
}

```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.sol

Locations

```

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;
}

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

```

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;
}

```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.sol

Locations

```

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;
}

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

```

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;
}

```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`core.sol`

Locations

```

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

```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

```

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

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.sol`

Locations

```

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

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

```

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

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.sol`

Locations

```

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

```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

```

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

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

```

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

```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

core.sol

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

`core.sol`

Locations

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

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

`core.sol`

Locations

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

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

`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

SWC-101

Arithmetic operation "/" discovered

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

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

SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.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

SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.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

SWC-101

Arithmetic operation "/" discovered

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

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

SWC-101

Arithmetic operation "*" discovered

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

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

SWC-101

Arithmetic operation "/" discovered

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

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

SWC-101

Arithmetic operation "*" discovered

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

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

SWC-101

Arithmetic operation "-" discovered

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

Source file

core.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 {
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

core.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 {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

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

SWC-101

Arithmetic operation "*" discovered

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

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

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

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

SWC-101

Arithmetic operation "*" discovered

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

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

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.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

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.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 SWC-101

Arithmetic operation "/" discovered

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

Source file

core.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 SWC-101

Arithmetic operation "*" discovered

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

Source file

core.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 SWC-101

Arithmetic operation "+" discovered

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

Source file

core.sol

Locations

```
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
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.sol

Locations

```
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
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

```
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
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

core.sol

Locations

```
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
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.sol`

Locations

```
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
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`core.sol`

Locations

```
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
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`core.sol`

Locations

```
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
```

UNKNOWN SWC-101

Arithmetic operation "*" discovered

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

Source file

core.sol

Locations

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

UNKNOWN SWC-101

Arithmetic operation "+=" discovered

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

Source file

core.sol

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);
```

UNKNOWN SWC-101

Arithmetic operation "+=" discovered

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

Source file

core.sol

Locations

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

UNKNOWN SWC-101

Arithmetic operation "-=" discovered

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

Source file

core.sol

Locations

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

UNKNOWN SWC-101

Arithmetic operation "-=" discovered

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

Source file

core.sol

Locations

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

UNKNOWN SWC-101

Arithmetic operation "++" discovered

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

Source file

core.sol

Locations

```
'\x19\x01',
DOMAIN_SEPARATOR,
keccak256(abi.encode(PERMIT_TYPEHASH, owner, spender, value, nonces[owner]++, deadline))
)
);
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

core.sol

Locations

```

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

```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

core.sol

Locations

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

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

```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

core.sol

Locations

```

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

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

core.sol

Locations

```

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

```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

core.sol

Locations

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

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

UNKNOWN

SWC-110

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.

Source file

core.sol

Locations

```
uint constant periodSize = 1800;

Observation[] public observations;

uint internal immutable decimals0;
```

UNKNOWN

SWC-110

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.

Source file

`core.sol`

Locations

```

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

```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`core.sol`

Locations

```

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

```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`core.sol`

Locations

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

```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

core.sol

Locations

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

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

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;
```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

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;
```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`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

SWC-110

Out of bounds array access

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

Source file

`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

SWC-110

Out of bounds array access

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

Source file

`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

SWC-110

Out of bounds array access

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

Source file

`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

SWC-110

Out of bounds array access

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

Source file

`core.sol`

Locations

```
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;
}
```