MythX

REPORT 628D687A5CF3170019C3CC75

Created	Tue May 24 2022 23:21:30 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	6197960e3494e9c8c076e89b

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
cc3b6d66-a570-499a-9252-beb02bdef65d	Router.sol	1

Analysis cc3b6d66-a570-499a-9252-beb02bdef65d MythX

Started	Tue May 24 2022 23:21:40 GMT+0000 (Coordinated Universal Time)
Finished	Tue May 24 2022 23:22:50 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	Router.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file
Router.sol
Locations

```
19 address public immutable factory;
20 IWETH public immutable weth;
21 uint internal constant MINIMUM_LIQUIDITY = 10**3;
22 bytes32 immutable pairCodeHash;
23
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file
Router.sol
Locations

```
58 require(amountA > 0, 'Router: INSUFFICIENT_AMOUNT');
59 require(reserveA > 0 && reserveB > 0, 'Router: INSUFFICIENT_LIQUIDITY');
60 amountB = amountA * reserveB / reserveA;
61 }
62
```

UNKNOWN

Arithmetic operation "*" discovered

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

SWC-101

Source file

Router.sol

Locations

```
58 require(amountA > 0, 'Router: INSUFFICIENT_AMOUNT');
59 require(reserveA > 0 && reserveB > 0, 'Router: INSUFFICIENT_LIQUIDITY');
60 amountB = amountA * reserveB / reserveA;
61 }
62
```

UNKNOWN

Arithmetic operation "+" discovered

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

SWC-101

Source file

Router.sol

Locations

```
86 function getAmountsOut(uint amountIn, route[] memory routes) public view returns (uint[] memory amounts) {
87 require(routes.length >= 1, 'Router: INVALID_PATH');
88 amounts = new uint[](routes.length+1);
89 amounts[0] = amountIn;
90 for (uint i = 0; i < routes.length; i++) {
```

UNKNOWN

Arithmetic operation "++" discovered

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

SWC-101

Source file

Router.sol

Locations

```
88 amounts = new uint[](routes.length+1);
89 amounts[0] = amountIn;
90 for (uint i = 0; i < routes.length; i++) {
91 address pair = pairFor(routes[i].from, routes[i].to, routes[i].stable);
92 if (IPairFactory(factory).isPair(pair)) {
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Router.sol

Locations

```
91  address pair = pairFor(routes[i].from, routes[i].to, routes[i].stable);
92  if (IPairFactory(factory).isPair(pair)) {
93  amounts[i+1] = IPair(pair).getAmountOut(amounts[i], routes[i].from);
94  }
95  }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Router.sol

Locations

```
117  if (reserveA == 0 && reserveB == 0) {
118  (amountA, amountB) = (amountADesired, amountBDesired);
119  liquidity = Math.sqrt(amountA * amountB) - MINIMUM_LIQUIDITY;
120  } else {
121
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Router.sol

Locations

```
117  if (reserveA == 0 && reserveB == 0) {
118  (amountA, amountB) = (amountADesired, amountBDesired);
119  liquidity = Math.sqrt(amountA * amountB) - MINIMUM_LIQUIDITY;
120  } else {
121
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Router.sol

Locations

```
123 if (amountBOptimal <= amountBDesired) {
124 (amountA, amountB) = (amountADesired, amountBOptimal);
125 liquidity = Math.min(amountA * _totalSupply / reserveA, amountB * _totalSupply / reserveB);
126 } else {
127 uint amountAOptimal = quoteLiquidity(amountBDesired, reserveB, reserveA);
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Router.sol

Locations

```
123 if (amountBOptimal <= amountBDesired) {
124 (amountA, amountB) = (amountADesired, amountBOptimal);
125 liquidity = Math.min(amountA * _totalSupply / reserveA, amountB * _totalSupply / reserveB);
126 } else {
127 uint amountAOptimal = quoteLiquidity(amountBDesired, reserveB, reserveA);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Router.sol

Locations

```
123 if (amountBOptimal <= amountBDesired) {
124 (amountA, amountB) = (amountADesired, amountBOptimal);
125 liquidity = Math.min(amountA * _totalSupply / reserveA, amountB * _totalSupply / reserveB);
126 } else {
127 uint amountAOptimal = quoteLiquidity(amountBDesired, reserveB, reserveA);
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Router.sol

Locations

```
123 if (amountBOptimal <= amountBDesired) {
124 (amountA, amountB) = (amountADesired, amountBOptimal);
125 liquidity = Math.min(amountA * _totalSupply / reserveA, amountB * _totalSupply / reserveB);
126 } else {
127 uint amountAOptimal = quoteLiquidity(amountBDesired, reserveB, reserveA);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Router.sol

Locations

```
127 uint amountAOptimal = quoteLiquidity(amountBDesired, reserveB, reserveA);
128 (amountA, amountB) = (amountAOptimal, amountBDesired);
129 liquidity = Math.min(amountA * _totalSupply / reserveA, amountB * _totalSupply / reserveB);
130 }
131 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Router.sol

Locations

```
127 uint amountAOptimal = quoteLiquidity(amountBDesired, reserveB, reserveA);
128 (amountA, amountB) = (amountAOptimal, amountBDesired);
129 liquidity = Math.min(amountA * _totalSupply / reserveA, amountB * _totalSupply / reserveB);
130 }
131 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Router.sol

Locations

```
127 | uint amountAOptimal = quoteLiquidity(amountBDesired, reserveB, reserveA);
128 | (amountA, amountB) = (amountAOptimal, amountBDesired);
129 | liquidity = Math.min(amountA * _totalSupply / reserveA, amountB * _totalSupply / reserveB);
130 | }
131 | }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Router.sol

Locations

```
127 | uint amountAOptimal = quoteLiquidity(amountBDesired, reserveB, reserveA);
128 | (amountA, amountB) = (amountAOptimal, amountBDesired);
129 | liquidity = Math.min(amountA * _totalSupply / reserveA, amountB * _totalSupply / reserveB);
130 | }
131 | }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Router.sol

Locations

```
148 | uint _totalSupply = IERC20(_pair).totalSupply();
149 |
150 | amountA = liquidity * reserveA / _totalSupply; // using balances ensures pro-rata distribution
151 | amountB = liquidity * reserveB / _totalSupply; // using balances ensures pro-rata distribution
152 |
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Router.sol

Locations

```
148 uint _totalSupply = IERC20(_pair).totalSupply();
149
150 amountA = liquidity * reserveA / _totalSupply; // using balances ensures pro-rata distribution
151 amountB = liquidity * reserveB / _totalSupply; // using balances ensures pro-rata distribution
152
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Router.sol

Locations

```
149
150 amountA = liquidity * reserveA / _totalSupply; // using balances ensures pro-rata distribution
151 amountB = liquidity * reserveB / _totalSupply; // using balances ensures pro-rata distribution
152
153 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Router.sol

Locations

```
149
150 amountA = liquidity * reserveA / _totalSupply; // using balances ensures pro-rata distribution
151 amountB = liquidity * reserveB / _totalSupply; // using balances ensures pro-rata distribution
152
153 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Router.sol

Locations

```
228 liquidity = IPair(pair).mint(to);
229 // refund dust eth, if any
230 if (msg.value > amountETH) _safeTransferETH(msg.sender, msg.value - amountETH);
231 }
232
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Router.sol

Locations

```
314 // requires the initial amount to have already been sent to the first pair
315 function _swap(uint[] memory amounts, route[] memory routes, address _to) internal virtual {
316 for (uint i = 0; i < routes.length; i++) {
317 (address token0,) = sortTokens(routes[i].from, routes[i].to);
318 uint amountOut = amounts[i + 1];
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Router.sol

Locations

```
316 for (uint i = 0; i < routes.length; i++) {
317 (address token0,) = sortTokens(routes[i].from, routes[i].to);
318 uint amountOut = amounts[i + 1];
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Router.sol

Locations

```
318 uint amountOut = amounts[i + 1];
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
322 amount0Out, amount1Out, to, new bytes(0)
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Router.sol

Locations

```
318 uint amountOut = amounts[i + 1];
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
322 amount0Out, amount1Out, to, new bytes(0)
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Router.sol

Locations

```
318 uint amountOut = amounts[i + 1];
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
322 amount0Out, amount1Out, to, new bytes(0)
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Router.sol

Locations

```
318 uint amountOut = amounts[i + 1];
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
322 amount0Out, amount1Out, to, new bytes(0)
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Router.sol

Locations

```
339 routes[0].stable = stable;
340 amounts = getAmountsOut(amountIn, routes);
341 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
342 _safeTransferFrom(
343 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Router.sol

Locations

```
354 ) external ensure(deadline) returns (uint[] memory amounts) {
355 amounts = getAmountsOut(amountIn, routes);
356 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
357 _safeTransferFrom(
358 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Router.sol

Locations

```
369 require(routes[0].from == address(weth), 'Router: INVALID_PATH');
370 amounts = getAmountsOut(msg.value, routes);
371 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
372 weth.deposit{value: amounts[0]}();
373 assert(weth.transfer(pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]));
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Router.sol

Locations

```
380 returns (uint[] memory amounts)
381 {
382 require(routes[routes.length - 1].to == address(weth), 'Router: INVALID_PATH');
383 amounts = getAmountsOut(amountIn, routes);
384 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Router.sol

Locations

```
382 require(routes[routes.length - 1].to == address(weth), 'Router: INVALID_PATH');
383 amounts = getAmountsOut(amountIn, routes);
384 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
385 _safeTransferFrom(
386 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Router.sol

Locations

```
387 );
388 _swap(amounts, routes, address(this));
389 weth.withdraw(amounts[amounts.length - 1]);
390 _safeTransferETH(to, amounts[amounts.length - 1]);
391 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Router.sol

Locations

```
388 _swap(amounts, routes, address(this));
389 weth.withdraw(amounts[amounts.length - 1]);
390 _safeTransferETH(to, amounts[amounts.length - 1]);
391 }
392
```

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

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

SWC-101

Source file

Router.sol

Locations

```
318 uint amountOut = amounts[i + 1];
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
322 amount0Out, amount1Out, to, new bytes(0)
```

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

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

SWC-101

Source file

Router.sol

Locations

```
339 routes[0].stable = stable;
340 amounts = getAmountsOut(amountIn, routes);
341 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
342 _safeTransferFrom(
343 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
```

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

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

SWC-101

Source file

Router.sol

Locations

```
354 ) external ensure(deadline) returns (uint[] memory amounts) {
355 amounts = getAmountsOut(amountIn, routes);
356 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
357 _safeTransferFrom(
358 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
```

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

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

SWC-101

Source file

Router.sol

Locations

```
369 require(routes[0].from == address(weth), 'Router: INVALID_PATH');
370 amounts = getAmountsOut(msg.value, routes);
371 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
372 weth.deposit{value: amounts[0]}();
373 assert(weth.transfer(pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]));
```

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

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

SWC-101

Source file

Router.sol

Locations

```
380 returns (uint[] memory amounts)
381 {
382 require(routes[routes.length - 1].to == address(weth), 'Router: INVALID_PATH');
383 amounts = getAmountsOut(amountIn, routes);
384 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
```

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

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

SWC-101

Source file

Router.sol

Locations

```
382 require(routes[routes.length - 1].to == address(weth), 'Router: INVALID_PATH');
383 amounts = getAmountsOut(amountIn, routes);
384 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
385 _safeTransferFrom(
386 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
```

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

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

SWC-101

Source file

Router.sol

Locations

```
387 );
388 _swap(amounts, routes, address(this));
389 weth.withdraw(amounts[amounts.length - 1]);
390 _safeTransferETH(to, amounts[amounts.length - 1]);
391 }
```

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

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

SWC-101

Source file

Router.sol

Locations

```
388 _swap(amounts, routes, address(this));
389 weth.withdraw(amounts[amounts.length - 1]);
390 _safeTransferETH(to, amounts[amounts.length - 1]);
391 }
392
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

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

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

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

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

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

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

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

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

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

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

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

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

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

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

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

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

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

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

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

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
26 | uint256 z = 3 * x * (x + 1) + 1;
27 | if (n / y >= z) {
28 | n -= y * z;
29 | x += 1;
30 | }
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

libraries/Math.sol

Locations

```
26 uint256 z = 3 * x * (x + 1) + 1;
27 if (n / y >= z) {
28 n -= y * z;
29 x += 1;
30 }
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

libraries/Math.sol

Locations

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

LOW

SWC-108

State variable visibility is not set.

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

Source file

Router.sol

Locations

```
20 IWETH public immutable weth;
21 uint internal constant MINIMUM_LIQUIDITY = 10**3;
22 bytes32 immutable pairCodeHash;
23
24 modifier ensure(uint deadline) {
```

UNKNOWN Out of bounds array access

SWC-110

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

Source file

Router.sol

Locations

```
87  require(routes.length >= 1, 'Router: INVALID_PATH');
88  amounts = new uint[](routes.length+1);
89  amounts[0] = amountIn;
90  for (uint i = 0; i < routes.length; i++) {
91  address pair = pairFor(routes[i].from, routes[i].to, routes[i].stable);
```

UNKNOWN Out of bounds array access

SWC-110

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

Source file

Router.sol

Locations

```
89  amounts[0] = amountIn;
90  for (uint i = 0; i < routes.length; i++) {
91  address pair = pairFor(routes[i].from, routes[i].to, routes[i].stable);
92  if (IPairFactory(factory).isPair(pair)) {
93  amounts[i+1] = IPair(pair).getAmountOut(amounts[i], routes[i].from);
```

UNKNOWN Out of bounds array access

SWC-110

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

Source file

Router.sol

Locations

```
89  amounts[0] = amountIn;
90  for (uint i = 0; i < routes.length; i++) {
91  address pair = pairFor(routes[i].from, routes[i].to, routes[i].stable);
92  if (IPairFactory(factory).isPair(pair)) {
93  amounts[i+1] = IPair(pair).getAmountOut(amounts[i], routes[i].from);
```

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

Router.sol

Locations

```
89  amounts[0] = amountIn;
90  for (uint i = 0; i < routes.length; i++) {
91  address pair = pairFor(routes[i].from, routes[i].to, routes[i].stable);
92  if (IPairFactory(factory).isPair(pair)) {
93  amounts[i+1] = IPair(pair).getAmountOut(amounts[i], routes[i].from);
```

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

Router.sol

Locations

```
91  address pair = pairFor(routes[i].from, routes[i].to, routes[i].stable);
92  if (IPairFactory(factory).isPair(pair)) {
93  amounts[i+1] = IPair(pair).getAmountOut(amounts[i], routes[i].from);
94  }
95  }
```

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

Router.sol

Locations

```
91  address pair = pairFor(routes[i].from, routes[i].to, routes[i].stable);
92  if (IPairFactory(factory).isPair(pair)) {
93  amounts[i+1] = IPair(pair).getAmountOut(amounts[i], routes[i].from);
94  }
95  }
```

UNKNOWN Out of bounds array access

SWC-110

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

Source file

Router.sol

Locations

```
91  address pair = pairFor(routes[i].from, routes[i].to, routes[i].stable);
92  if (IPairFactory(factory).isPair(pair)) {
93  amounts[i+1] = IPair(pair).getAmountOut(amounts[i], routes[i].from);
94  }
95  }
```

UNKNOWN Out of bounds array access

SWC-110

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

Source file

Router.sol

Locations

```
315  function _swap(uint[] memory amounts, route[] memory routes, address _to) internal virtual {
316  for (uint i = 0; i < routes.length; i++) {
317  (address token0,) = sortTokens(routes[i].from, routes[i].to);
318  uint amountOut = amounts[i + 1];
319  (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
```

UNKNOWN Out of bounds array access

SWC-110

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

Source file

Router.sol

Locations

```
315  function _swap(uint[] memory amounts, route[] memory routes, address _to) internal virtual {
316  for (uint i = 0; i < routes.length; i++) {
317  (address token0,) = sortTokens(routes[i].from, routes[i].to);
318  uint amountOut = amounts[i + 1];
319  (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
```

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

Router.sol

Locations

```
316 for (uint i = 0; i < routes.length; i++) {
317 (address token0,) = sortTokens(routes[i].from, routes[i].to);
318 uint amountOut = amounts[i + 1];
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
```

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

Router.sol

Locations

```
317 (address token0,) = sortTokens(routes[i].from, routes[i].to);
318 uint amountOut = amounts[i + 1];
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
```

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

Router.sol

Locations

```
318 uint amountOut = amounts[i + 1];
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
322 amount0Out, amount1Out, to, new bytes(0)
```

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

Router.sol

Locations

```
318 uint amountOut = amounts[i + 1];
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
322 amount0Out, amount1Out, to, new bytes(0)
```

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

Router.sol

Locations

```
318 uint amountOut = amounts[i + 1];
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
322 amount0Out, amount1Out, to, new bytes(0)
```

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

Router.sol

Locations

```
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
322 amount0Out, amount1Out, to, new bytes(0)
323 );
```

UNKNOWN Out of bounds array access

SWC-110

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

Source file

Router.sol

Locations

```
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
322 amount0Out, amount1Out, to, new bytes(0)
323 );
```

UNKNOWN Out of bounds array access

SWC-110

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

Source file

Router.sol

Locations

```
319 (uint amount0Out, uint amount1Out) = routes[i].from == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
320 address to = i < routes.length - 1 ? pairFor(routes[i+1].from, routes[i+1].to, routes[i+1].stable) : _to;
321 IPair(pairFor(routes[i].from, routes[i].to, routes[i].stable)).swap(
322 amount0Out, amount1Out, to, new bytes(0)
323 );
```

UNKNOWN Out of bounds array access

SWC-110

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

Source file

Router.sol

Locations

```
335 ) external ensure(deadline) returns (uint[] memory amounts) {
336 route[] memory routes = new route[](1);
337 routes[0].from = tokenFrom;
338 routes[0].to = tokenTo;
339 routes[0].stable = stable;
```

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

Router.sol

Locations

```
336 route[] memory routes = new route[](1);
337 routes[0].from = tokenFrom;
338 routes[0].to = tokenTo;
339 routes[0].stable = stable;
340 amounts = getAmountsOut(amountIn, routes);
```

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

Router.sol

Locations

```
337 routes[0].from = tokenFrom;
338 routes[0].to = tokenTo;
339 routes[0].stable = stable;
340 amounts = getAmountsOut(amountIn, routes);
341 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
```

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

Router.sol

Locations

```
339 routes[0].stable = stable;
340 amounts = getAmountsOut(amountIn, routes);
341 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
342 _safeTransferFrom(
343 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
```

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

Router.sol

Locations

```
341 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
342 _safeTransferFrom(
343 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
344 );
345 _swap(amounts, routes, to);
```

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

Router.sol

Locations

```
341 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
342 _safeTransferFrom(
343 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
344 );
345 _swap(amounts, routes, to);
```

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

Router.sol

Locations

```
341 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
342 _safeTransferFrom(
343 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
344 );
345 _swap(amounts, routes, to);
```

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

Router.sol

Locations

```
341 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
342 _safeTransferFrom(
343 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
344 );
345 _swap(amounts, routes, to);
```

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

Router.sol

Locations

```
341 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
342 _safeTransferFrom(
343 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
344 );
345 _swap(amounts, routes, to);
```

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

Router.sol

Locations

```
354 ) external ensure(deadline) returns (uint[] memory amounts) {
355 amounts = getAmountsOut(amountIn, routes);
356 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
357 _safeTransferFrom(
358 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
```

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

Router.sol

Locations

```
356 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
357 _safeTransferFrom(
358 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
359 );
360 _swap(amounts, routes, to);
```

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

Router.sol

Locations

```
356 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
357 _safeTransferFrom(
358 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
359 );
360 _swap(amounts, routes, to);
```

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

Router.sol

Locations

```
356 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
357 _safeTransferFrom(
358 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
359 );
360 _swap(amounts, routes, to);
```

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

Router.sol

Locations

```
356 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
357 _safeTransferFrom(
358 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
359 );
360 _swap(amounts, routes, to);
```

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

Router.sol

Locations

```
356 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
357 _safeTransferFrom(
358 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
359 );
360 _swap(amounts, routes, to);
```

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

Router.sol

Locations

```
367 returns (uint[] memory amounts)
368 {
369 require(routes[0].from == address(weth), 'Router: INVALID_PATH');
370 amounts = getAmountsOut(msg.value, routes);
371 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
```

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

Router.sol

Locations

```
369 require(routes[0].from == address(weth), 'Router: INVALID_PATH');
370 amounts = getAmountsOut(msg.value, routes);
371 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
372 weth.deposit{value: amounts[0]}();
373 assert(weth.transfer(pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]));
```

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

Router.sol

Locations

```
370 amounts = getAmountsOut(msg.value, routes);
371 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
372 weth.deposit{value: amounts[0]}();
373 assert(weth.transfer(pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]));
374 _swap(amounts, routes, to);
```

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

Router.sol

Locations

```
371 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
372 weth.deposit{value: amounts[0]}();
373 assert(weth.transfer(pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]));
374 _swap(amounts, routes, to);
375 }
```

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

Router.sol

Locations

```
371 | require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
372 | weth.deposit{value: amounts[0]}();
373 | assert(weth.transfer(pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]));
374 | _swap(amounts, routes, to);
375 | }
```

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

Router.sol

Locations

```
371 | require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
372 | weth.deposit{value: amounts[0]}();
373 | assert(weth.transfer(pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]));
374 | _swap(amounts, routes, to);
375 | }
```

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

Router.sol

Locations

```
371 | require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
372 | weth.deposit{value: amounts[0]}();
373 | assert(weth.transfer(pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]));
374 | _swap(amounts, routes, to);
375 | }
```

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

Router.sol

Locations

```
380 returns (uint[] memory amounts)
381 {
382 require(routes[routes.length - 1].to == address(weth), 'Router: INVALID_PATH');
383 amounts = getAmountsOut(amountIn, routes);
384 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
```

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

Router.sol

Locations

```
382 require(routes[routes.length - 1].to == address(weth), 'Router: INVALID_PATH');
383 amounts = getAmountsOut(amountIn, routes);
384 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
385 _safeTransferFrom(
386 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
```

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

Router.sol

Locations

```
384 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
385 _safeTransferFrom(
386 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
387 );
388 _swap(amounts, routes, address(this));
```

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

Router.sol

Locations

```
384 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
385 _safeTransferFrom(
386 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
387 );
388 _swap(amounts, routes, address(this));
```


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

Router.sol

Locations


```
384 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
385 _safeTransferFrom(
386 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
387 );
388 _swap(amounts, routes, address(this));
```


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

Router.sol

Locations


```
384 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
385 _safeTransferFrom(
386 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
387 );
388 _swap(amounts, routes, address(this));
```

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

Router.sol

Locations

```
384 require(amounts[amounts.length - 1] >= amountOutMin, 'Router: INSUFFICIENT_OUTPUT_AMOUNT');
385 _safeTransferFrom(
386 routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]
387 );
388 _swap(amounts, routes, address(this));
```

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

Router.sol

Locations

```
387 );
388 _swap(amounts, routes, address(this));
389 weth.withdraw(amounts[amounts.length - 1]);
390 _safeTransferETH(to, amounts[amounts.length - 1]);
391 }
```

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

Router.sol

Locations

```
388 _swap(amounts, routes, address(this));
389 weth.withdraw(amounts[amounts.length - 1]);
390 _safeTransferETH(to, amounts[amounts.length - 1]);
391 }
392
```

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

Router.sol

Locations

```
397 uint deadline
398 ) external ensure(deadline) returns (uint[] memory) {
399 _safeTransferFrom(routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]);
400 _swap(amounts, routes, to);
401 return amounts;
```

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

Router.sol

Locations

```
397 uint deadline
398 ) external ensure(deadline) returns (uint[] memory) {
399 _safeTransferFrom(routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]);
400 _swap(amounts, routes, to);
401 return amounts;
```

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

Router.sol

Locations

```
397 uint deadline
398 ) external ensure(deadline) returns (uint[] memory) {
399 _safeTransferFrom(routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]);
400 _swap(amounts, routes, to);
401 return amounts;
```

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

Router.sol

Locations

```
397 | uint deadline
398 | ) external ensure(deadline) returns (uint[] memory) {
399 | _safeTransferFrom(routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]);
400 | _swap(amounts, routes, to);
401 | return amounts;
```

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

Router.sol

Locations

```
397 | uint deadline
398 | ) external ensure(deadline) returns (uint[] memory) {
399 | _safeTransferFrom(routes[0].from, msg.sender, pairFor(routes[0].from, routes[0].to, routes[0].stable), amounts[0]);
400 | _swap(amounts, routes, to);
401 | return amounts;
```