MythX

Analysis b530aaff-b222-45c0-9c4f-1abe9c566a75 MythX

Started

Finished Sun Jan 02 2022 12:56:37 GMT+0000 (Coordinated Universal Time)

Mode Deep

Client Tool Remythx

Main Source File AAptitude.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	2

ISSUES

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
34
35 function add(uint256 a, uint256 b) internal pure returns (uint256) {
36 uint256 c = a + b;
37 require(c >= a, "SafeMath: addition overflow");
38
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
46 function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
47 require(b <= a, errorMessage);
48 uint256 c = a - b;
49
50 return c;
```

UNKNOWN

Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
56 }
57
58 uint256 c = a * b;
59 require(c / a == b, "SafeMath: multiplication overflow");
60
```

UNKNOWN

Arithmetic operation "/" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
57
58 uint256 c = a * b;
59 require(c / a == b, "SafeMath: multiplication overflow");
60
61 return c;
```

UNKNOWN

Arithmetic operation "/" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
69 function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
70 require(b > 0, errorMessage);
71 uint256 c = a / b;
72 // assert(a == b * c + a % b); // There is no case in which this doesn't hold
73
```

UNKNOWN Arithmetic operation "%" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
81 function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
82 require(b != 0, errorMessage);
83 return a % b;
84 }
85 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
394
395 uint256 private constant MAX = ~uint256(0);
396 uint256 private _tTotal = 10000 * 10**6 * 10**9;
397 uint256 private _rTotal = (MAX - (MAX % _tTotal));
398 uint256 private _tFeeTotal;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
394
395 uint256 private constant MAX = ~uint256(0);
396 uint256 private _tTotal = 10000 * 10**6 * 10**9;
397 uint256 private _rTotal = (MAX - (MAX % _tTotal));
398 uint256 private _tFeeTotal;
```

UNKNOWN

Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
394
395 uint256 private constant MAX = ~uint256(0);
396 uint256 private _tTotal = 10000 * 10**6 * 10**9;
397 uint256 private _rTotal = (MAX - (MAX % _tTotal));
398 uint256 private _tFeeTotal;
```

UNKNOWN

Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
394
395 uint256 private constant MAX = ~uint256(0);
396 uint256 private _tTotal = 10000 * 10**6 * 10**9;
397 uint256 private _rTotal = (MAX - (MAX % _tTotal));
398 uint256 private _tFeeTotal;
```

UNKNOWN

Arithmetic operation "-" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
395 uint256 private constant MAX = ~uint256(0);
396 uint256 private _tTotal = 10000 * 10**6 * 10**9;
397 uint256 private _rTotal = (MAX - (MAX % _tTotal));
398 uint256 private _tFeeTotal;
399
```

UNKNOWN Arithmetic operation "%" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
395 uint256 private constant MAX = ~uint256(0);
396 uint256 private _tTotal = 10000 * 10**6 * 10**9;
397 uint256 private _rTotal = (MAX - (MAX % _tTotal));
398 uint256 private _tFeeTotal;
399
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
411 uint256 public devAndMarketingDivisor = 3;
412
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
411 uint256 public devAndMarketingDivisor = 3;
412
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
411 uint256 public devAndMarketingDivisor = 3;
412
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
411 uint256 public devAndMarketingDivisor = 3;
412
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
412
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
```

UNKNOWN

Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
412
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
```

UNKNOWN

Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
412
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
```

UNKNOWN

Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
412
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
417
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
417
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
417
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
413 uint256 public _maxTxAmount = 100 * 10**6 * 10**9;
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
417
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
417
418 uint256 public percentToBuyBack = 1; // percent of fund to be used to buy back each time
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
417
418 uint256 public percentToBuyBack = 1; // percent of fund to be used to buy back each time
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
417
418 uint256 public percentToBuyBack = 1; // percent of fund to be used to buy back each time
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
414 uint256 private minimumTokensBeforeSwap = 2 * 10**6 * 10**9;
415 uint256 private buyBackUpperLimit = 1 * 10**4 * 10**9;
416 uint256 private minimumBeforeBuyBack = 1 * 10**4 * 10**9;
417
418 uint256 public percentToBuyBack = 1; // percent of fund to be used to buy back each time
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
584 function includeInReward(address account) external onlyOwner() {
585 require(_isExcluded[account], "Account is already excluded");
586 for (uint256 i = 0; i < _excluded.length; i++) {
587 if (_excluded[i] == account) {
588 _excluded[i] = _excluded[_excluded.length - 1];
```

UNKNOWN

Arithmetic operation "-" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
586 for (uint256 i = 0; i < _excluded.length; i++) {
587 if (_excluded[i] == account) {
588 _excluded[i] = _excluded[_excluded.length - 1];
589 _tOwned[account] = 0;
590 _isExcluded[account] = false;
```

UNKNOWN

Arithmetic operation "++" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
814 uint256 rSupply = _rTotal;
815 uint256 tSupply = _tTotal;
816 for (uint256 i = 0; i < _excluded.length; i++) {
817 if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
818 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
```

UNKNOWN

Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
833 function calculateTaxFee(uint256 _amount) private view returns (uint256) {
834 return _amount.mul(_taxFee).div(
835 10**2
836 );
837 }
```

UNKNOWN

Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
839 function calculateLiquidityFee(uint256 _amount) private view returns (uint256) {
840 return _amount.mul(_liquidityFee).div(
841 10**2
842 );
843 }
```

UNKNOWN

Arithmetic operation "*" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
915
916 function setBuybackUpperLimit(uint256 buyBackLimit) external onlyOwner() {
917 buyBackUpperLimit = buyBackLimit * 10**18;
918 }
919
```

UNKNOWN

Arithmetic operation "**" discovered

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

SWC-101

Source file

AAptitude.sol

Locations

```
915
916 function setBuybackUpperLimit(uint256 buyBackLimit) external onlyOwner() {
917 buyBackUpperLimit = buyBackLimit * 10**18;
918 }
919
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

AAptitude.sol

Locations

```
1168
1169 // query support of each interface in _interfaceIds
1170 for (uint256 i = 0; i < interfaceIds.length; i++) {
1171 if (!_supportsERC165Interface(account, interfaceIds[i])) {
1172 return false;
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

AAptitude.sol

Locations

```
586 for (uint256 i = 0; i < _excluded.length; i++) {
587 if (_excluded[i] == account) {
588 _excluded[i] = _excluded[_excluded.length - 1];
589 _tOwned[account] = 0;
590 _isExcluded[account] = false;
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.4"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

AAptitude.sol

Locations

```
7 // SPDX-License-Identifier: Unlicensed
8
9 pragma solidity ^0.8.4;
10
11 abstract contract Context {
```

LOW

State variable visibility is not set.

SWC-108

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

Source file

AAptitude.sol

Locations

```
421 address public immutable uniswapV2Pair;
422
423 bool inSwapAndLiquify;
424 bool public swapAndLiquifyEnabled = true;
425 bool public buyBackEnabled = true;
```

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

AAptitude.sol

Locations

```
585 require(_isExcluded[account], "Account is already excluded");
586 for (uint256 i = 0; i < _excluded.length; i++) {
587 if (_excluded[i] == account) {
588 _excluded[i] = _excluded[_excluded.length - 1];
589 _tOwned[account] = 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

AAptitude.sol

Locations

```
586 for (uint256 i = 0; i < _excluded.length; i++) {
587 if (_excluded[i] == account) {
588 _excluded[i] = _excluded[_excluded.length - 1];
589 _tOwned[account] = 0;
590 _isExcluded[account] = false;
```

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

AAptitude.sol

Locations

```
586 for (uint256 i = 0; i < _excluded.length; i++) {
587 if (_excluded[i] == account) {
588 _excluded[i] = _excluded[_excluded.length - 1];
589 _tOwned[account] = 0;
590 _isExcluded[account] = false;
```

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

AAptitude.sol

Locations

```
673 // generate the uniswap pair path of token -> weth
674 address[] memory path = new address[](2);
675 path[0] = address(this);
676 path[1] = uniswapV2Router.WETH();
677
```

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

AAptitude.sol

Locations

```
674 address[] memory path = new address[](2);
675 path[0] = address(this);
676 path[1] = uniswapV2Router.WETH();
677
678 _approve(address(this), address(uniswapV2Router), tokenAmount);
```

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

AAptitude.sol

Locations

```
693 // generate the uniswap pair path of token -> weth
694 address[] memory path = new address[](2);
695 path[0] = uniswapV2Router.WETH();
696 path[1] = address(this);
697
```

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

AAptitude.sol

Locations

```
694 address[] memory path = new address[](2);
695 path[0] = uniswapV2Router.WETH();
696 path[1] = address(this);
697
698 // make the 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

AAptitude.sol

Locations

```
815 uint256 tSupply = _tTotal;
816 for (uint256 i = 0; i < _excluded.length; i++) {
817 if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
818 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
819 tSupply = tSupply.sub(_tOwned[_excluded[i]]);
```

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

AAptitude.sol

Locations

```
815 uint256 tSupply = _tTotal;
816 for (uint256 i = 0; i < _excluded.length; i++) {
817 if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
818 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
819 tSupply = tSupply.sub(_tOwned[_excluded[i]]);
```

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

AAptitude.sol

Locations

```
816 for (uint256 i = 0; i < _excluded.length; i++) {
817 if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
818 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
819 tSupply = tSupply.sub(_tOwned[_excluded[i]]);
820 }
```

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

AAptitude.sol

Locations

```
817 if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
818 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
819 tSupply = tSupply.sub(_tOwned[_excluded[i]]);
820 }
821 if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
```

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

AAptitude.sol

Locations

```
1169 // query support of each interface in _interfaceIds
1170 for (uint256 i = 0; i < interfaceIds.length; i++) {
1171 if (!_supportsERC165Interface(account, interfaceIds[i])) {
1172 return false;
1173 }
```