MythX

REPORT 61396731D171E20018068691

Created	Thu Sep 09 2021 01:45:21 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	613960e4a6e18485c0c6ee6c

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
91bc881c-6d36-4e45-b33a-25c06c9f3bbe	SafeTits.sol	2

Analysis 91bc881c-6d36-4e45-b33a-25c06c9f3bbe MythX

Started	
Finished	Thu Sep 09 2021 01:45:21 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	SafeTits.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	2

ISSUES

UNKNOWN **Arithmetic operation "+" discovered**

SWC-101

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

Source file

SafeTits.sol

Locations

```
94  function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
95  unchecked {
96  uint256 c = a + b;
97  if (c < a) return (false, 0);
98  return (true, c);
```

UNKNOWN **Arithmetic operation "-" discovered**

SWC-101

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

Source file

SafeTits.sol

Locations

```
108 unchecked {
109 if (b > a) return (false, 0);
110 return (true, a - b);
111 }
112 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
// See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
if (a == 0) return (true, 0);
uint256 c = a * b;
if (c / a != b) return (false, 0);
return (true, c);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
if (a == 0) return (true, 0);
uint256 c = a * b;
if (c / a != b) return (false, 0);
return (true, c);
}
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
unchecked {
if (b == 0) return (false, 0);
return (true, a / b);
}
}
```

UNKNOWN Arithmetic operation "%" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
149 unchecked {
150 if (b == 0) return (false, 0);
151 return (true, a % b);
152 }
153 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
164 */
165 function add(uint256 a, uint256 b) internal pure returns (uint256) {
166 return a + b;
167 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
178 */
179 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
180 return a - b;
181 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
192 */
193 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
194 return a * b;
195 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
206 */
207 function div(uint256 a, uint256 b) internal pure returns (uint256) {
208 return a / b;
209 }
```

UNKNOWN Arithmetic operation "%" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
222 */
223 function mod(uint256 a, uint256 b) internal pure returns (uint256) {
224 return a % b;
225 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
241 unchecked {
242 require(b <= a, errorMessage);
243 return a - b;
244 }
245 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
264 unchecked {
265 require(b > 0, errorMessage);
266 return a / b;
267 }
268 }
```

UNKNOWN

SWC-101

Arithmetic operation "%" discovered

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

Source file

SafeTits.sol

Locations

```
286 unchecked {
287 require(b > 0, errorMessage);
288 return a % b;
289 }
290 }
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

SafeTits.sol

Locations

```
780
781 uint256 private constant MAX = ~uint256(0);
782 uint256 private _tTotal = 1000000000000 * 10**9;
783 uint256 private _rTotal = (MAX - (MAX % _tTotal));
784 uint256 private _tFeeTotal;
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

SafeTits.sol

Locations

```
780
781 uint256 private constant MAX = ~uint256(0);
782 uint256 private _tTotal = 1000000000000 * 10**9;
783 uint256 private _rTotal = (MAX - (MAX % _tTotal));
784 uint256 private _tFeeTotal;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

SafeTits.sol

Locations

```
781 uint256 private constant MAX = ~uint256(0);
782 uint256 private _tTotal = 1000000000000 * 10**9;
783 uint256 private _rTotal = (MAX - (MAX % _tTotal));
784 uint256 private _tFeeTotal;
```

UNKNOWN

SWC-101

Arithmetic operation "%" discovered

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

Source file

SafeTits.sol

Locations

```
781 uint256 private constant MAX = ~uint256(0);
782 uint256 private _tTotal = 1000000000000 * 10**9;
783 uint256 private _rTotal = (MAX - (MAX % _tTotal));
784 uint256 private _tFeeTotal;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

SafeTits.sol

Locations

```
806 bool public swapAndLiquifyEnabled = true;
807
808 uint256 public _maxTxAmount = 5000000000 * 10**9;
809 uint256 public numTokensSellToAddToLiquidity = 1 * 10**6 * 10**9;
810
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

SafeTits.sol

Locations

```
806 bool public swapAndLiquifyEnabled = true;
807
808 uint256 public _maxTxAmount = 5000000000 * 10**9;
809 uint256 public numTokensSellToAddToLiquidity = 1 * 10**6 * 10**9;
810
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
807
808 uint256 public _maxTxAmount = 5000000000 * 10**9;
809 uint256 public numTokensSellToAddToLiquidity = 1 * 10**6 * 10**9;
810
811 event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
807
808 uint256 public _maxTxAmount = 5000000000 * 10**9;
809 uint256 public numTokensSellToAddToLiquidity = 1 * 10**6 * 10**9;
810
811 event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
807
808 uint256 public _maxTxAmount = 5000000000 * 10**9;
809 uint256 public numTokensSellToAddToLiquidity = 1 * 10**6 * 10**9;
810
811 event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
```

UNKNOWN

SWC-101

Arithmetic operation "**" discovered

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

Source file

SafeTits.sol

Locations

```
807
808 uint256 public _maxTxAmount = 5000000000 * 10**9;
809 uint256 public numTokensSellToAddToLiquidity = 1 * 10**6 * 10**9;
810
811 event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

SafeTits.sol

Locations

```
946 function includeInReward(address account) external onlyOwner() {
947 require(_isExcluded[account], "Account is already included");
948 for (uint256 i = 0; i < _excluded.length; i++) {
949 if (_excluded[i] == account) {
950 _excluded[i] = _excluded[_excluded.length - 1];
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

SafeTits.sol

Locations

```
948 for (uint256 i = 0; i < _excluded.length; i++) {
949 if (_excluded[i] == account) {
950 _excluded[i] = _excluded[_excluded.length - 1];
951 _tOwned[account] = 0;
952 _isExcluded[account] = false;
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
990 function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner() {
991 _maxTxAmount = _tTotal.mul(maxTxPercent).div(
992 10**2
993 );
994 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
1038 uint256 rSupply = _rTotal;
1039 uint256 tSupply = _tTotal;
1040 for (uint256 i = 0; i < _excluded.length; i++) {
1041 if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
1042 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

SafeTits.sol

Locations

```
1065 function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1066 return _amount.mul(_taxFee).div(
1067 10**2
1068 );
1069 }
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
1071 function calculateCharityFee(uint256 _amount) private view returns (uint256) {
1072 return _amount.mul(_charityFee).div(
1073 10**2
1074 );
1075 }
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

SafeTits.sol

Locations

```
1077 function calculateLiquidityFee(uint256 _amount) private view returns (uint256) {
1078 return _amount.mul(_liquidityFee).div(
1079 10**2
1080 );
1081 }
```

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

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

SWC-101

Source file

SafeTits.sol

Locations

```
948 for (uint256 i = 0; i < _excluded.length; i++) {
949 if (_excluded[i] == account) {
950 _excluded[i] = _excluded[_excluded.length - 1];
951 _tOwned[account] = 0;
952 _isExcluded[account] = false;
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.3"". 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

SafeTits.sol

Locations

```
2
3
4  pragma solidity ^0.8.3;
5
```

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 "inSwapAndLiquify" is internal. Other possible visibility settings are public and private.

Source file

SafeTits.sol

Locations

```
803  address public uniswapV2Pair;
804
805  bool inSwapAndLiquify;
806  bool public swapAndLiquifyEnabled = true;
807
```

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

SafeTits.sol

Locations

```
947  require(_isExcluded[account], "Account is already included");
948  for (uint256 i = 0; i < _excluded.length; i++) {
949  if (_excluded[i] == account) {
950  _excluded[i] = _excluded[_excluded.length - 1];
951  _tOwned[account] = 0;
```

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

SafeTits.sol

Locations

```
948  for (uint256 i = 0; i < _excluded.length; i++) {
949  if (_excluded[i] == account) {
950  _excluded[i] = _excluded[_excluded.length - 1];
951  _tOwned[account] = 0;
952  _isExcluded[account] = false;
```

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

SafeTits.sol

Locations

```
948 for (uint256 i = 0; i < _excluded.length; i++) {
949 if (_excluded[i] == account) {
950 _excluded[i] = _excluded[_excluded.length - 1];
951 _tOwned[account] = 0;
952 _isExcluded[account] = false;
```

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

SafeTits.sol

Locations

```
1039 uint256 tSupply = _tTotal;
1040 for (uint256 i = 0; i < _excluded.length; i++) {
1041 if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
1042 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1043 tSupply = tSupply.sub(_tOwned[_excluded[i]]);
```

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

SafeTits.sol

Locations

```
1039 uint256 tSupply = _tTotal;
1040 for (uint256 i = 0; i < _excluded.length; i++) {
1041 if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
1042 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1043 tSupply = tSupply.sub(_tOwned[_excluded[i]]);
```

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

SafeTits.sol

Locations

```
1040 for (uint256 i = 0; i < _excluded.length; i++) {
1041 if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
1042 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1043 tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1044 }
```

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

SafeTits.sol

Locations

```
1041 if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
1042 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1043 tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1044 }
1045 if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
```

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

SafeTits.sol

Locations

```
1183 // generate the uniswap pair path of token -> weth
1184 address[] memory path = new address[](2);
1185 path[0] = address(this);
1186 path[1] = uniswapV2Router.WETH();
```

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

`SafeTits.sol`

Locations

```
1184 address[] memory path = new address[](2);
1185 path[0] = address(this);
1186 path[1] = uniswapV2Router.WETH();
1187
1188 _approve(address(this), address(uniswapV2Router), tokenAmount);
```