MythX

REPORT 6119F94BCAF8260018DAD663

Created	Mon Aug 16 2021 05:36:11 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	6036a12b68203600180f4af6

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
b33a3a0c-d51a-472e-a906-24c661bf5f1b	XTT-TokenTimeLock.sol	1

Analysis b33a3a0c-d51a-472e-a906-24c661bf5f1b MythX

Started	Mon Aug 16 2021 05:36:16 GMT+0000 (Coordinated Universal Time)
Finished	Mon Aug 16 2021 06:21:36 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	XTT-TokenTimeLock.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

UNKNOWN Arithmetic operation "+" discovered

SWC-101

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

Source file

SafeMath.sol

Locations

```
23 */
24 function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
25 uint256 c = a + b;
26 if (c < a) return (false, 0);
27 return (true, c);
```

UNKNOWN Arithmetic operation "-" discovered

SWC-101

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

Source file

SafeMath.sol

Locations

```
35 function trySub(uint256 a, uint256 b) internal pure returns (bool, uint256) {
36 if (b > a) return (false, 0);
37 return (true, a - b);
38 }
39
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

SafeMath.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

SWC-101

Arithmetic operation "/" discovered

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

Source file

SafeMath.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

SafeMath.sol

Locations

```
function tryDiv(uint256 a, uint256 b) internal pure returns (bool, uint256) {
if (b == 0) return (false, 0);
return (true, a / b);
}

```

UNKNOWN

SWC-101

Arithmetic operation "%" discovered

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

Source file

SafeMath.sol

Locations

```
70 function tryMod(uint256 a, uint256 b) internal pure returns (bool, uint256) {
71 if (b == 0) return (false, 0);
72 return (true, a % b);
73 }
74
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

SafeMath.sol

Locations

```
84 */
85 function add(uint256 a, uint256 b) internal pure returns (uint256) {
86 uint256 c = a + b;
87 require(c >= a, "SafeMath: addition overflow");
88 return c;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

SafeMath.sol

Locations

```
101 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
102 require(b <= a, "SafeMath: subtraction overflow");
103 return a - b;
104 }
105
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

SafeMath.sol

Locations

```
116 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
117 if (a == 0) return 0;
118 uint256 c = a * b;
119 require(c / a == b, "SafeMath: multiplication overflow");
120 return c;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

SafeMath.sol

Locations

```
117 if (a == 0) return 0;
118 uint256 c = a * b;
119 require(c / a == b, "SafeMath: multiplication overflow");
120 return c;
121 }
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

SafeMath.sol

Locations

```
135 function div(uint256 a, uint256 b) internal pure returns (uint256) {
136 require(b > 0, "SafeMath: division by zero");
137 return a / b;
138 }
139
```

UNKNOWN

SWC-101

Arithmetic operation "%" discovered

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

Source file

SafeMath.sol

Locations

```
152 function mod(uint256 a, uint256 b) internal pure returns (uint256) {
153 require(b > 0, "SafeMath: modulo by zero");
154 return a % b;
155 }
156
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

SafeMath.sol

Locations

```
170 function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
171 require(b <= a, errorMessage);
172 return a - b;
173 }
174
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

SafeMath.sol

Locations

```
190 function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
191 require(b > 0, errorMessage);
192 return a / b;
193 }
194
```

UNKNOWN

SWC-101

Arithmetic operation "%" discovered

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

Source file

SafeMath.sol

Locations

```
210 function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
211 require(b > 0, errorMessage);
212 return a % b;
213 }
214 }
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""><=0.6.0<0.8.0"". 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

IERC20.sol

Locations

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity >=0.6.0 <0.8.0;
4
5 /**
```