MythX

Analysis a1423e99-9070-4b9f-ba6e-960ac410e9e7 MythX

Started	Tue Sep 07 2021 07:31:17 GMT+0000 (Coordinated Universal Time)
Finished	Tue Sep 07 2021 07:31:23 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	Contracts/XttVault.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	8

ISSUES

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

contracts/XttVault.sol

Locations

```
122 */
123 function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
124 uint256 c = a + b;
125 if (c < a) return (false, 0);
126 return (true, c);
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

contracts/XttVault.sol

Locations

```
134 function trySub(uint256 a, uint256 b) internal pure returns (bool, uint256) {
135 if (b > a) return (false, 0);
136 return (true, a - b);
137 }
138
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`contracts/XttVault.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

`contracts/XttVault.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

`contracts/XttVault.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

contracts/XttVault.sol

Locations

```
169 function tryMod(uint256 a, uint256 b) internal pure returns (bool, uint256) {
170 if (b == 0) return (false, 0);
171 return (true, a % b);
172 }
173
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

contracts/XttVault.sol

Locations

```
183 */
184 function add(uint256 a, uint256 b) internal pure returns (uint256) {
185 uint256 c = a + b;
186 require(c >= a, "SafeMath: addition overflow");
187 return c;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

contracts/XttVault.sol

Locations

```
200 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
201 require(b <= a, "SafeMath: subtraction overflow");
202 return a - b;
203 }
204
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

contracts/XttVault.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

contracts/XttVault.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

contracts/XttVault.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "%" discovered

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

Source file

contracts/XttVault.sol

Locations

```
251 function mod(uint256 a, uint256 b) internal pure returns (uint256) {
252 require(b > 0, "SafeMath: modulo by zero");
253 return a % b;
254 }
255
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

contracts/XttVault.sol

Locations

```
273 ) internal pure returns (uint256) {
274 require(b <= a, errorMessage);
275 return a - b;
276 }
277
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

contracts/XttVault.sol

Locations

```
297 ) internal pure returns (uint256) {
298 require(b > 0, errorMessage);
299 return a / b;
300 }
301
```

UNKNOWN

SWC-101

Arithmetic operation "%" discovered

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

Source file

contracts/XttVault.sol

Locations

```
321 ) internal pure returns (uint256) {
322 require(b > 0, errorMessage);
323 return a % b;
324 }
325 }
```

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

contracts/XttVault.sol

Locations

```
7  // SPDX-License-Identifier: MIT
8
9  pragma solidity >=0.6.0 <0.8.0;
10
11 /*
```

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

contracts/XttVault.sol

Locations

```
32 // File: @openzeppelin/contracts/access/Ownable.sol
33
34 pragma solidity >=0.6.0 <0.8.0;
35
36 /**
```

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

`contracts/XttVault.sol`

Locations

```
100 // File: @openzeppelin/contracts/math/SafeMath.sol
101
102 pragma solidity >=0.6.0 <0.8.0;
103
104 /**
```

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

`contracts/XttVault.sol`

Locations

```
327 // File: @openzeppelin/contracts/token/ERC20/IERC20.sol
328
329 pragma solidity >=0.6.0 <0.8.0;
330
331 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""><=0.6.2<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

`contracts/XttVault.sol`

Locations

```
409 // File: @openzeppelin/contracts/utils/Address.sol
410
411 pragma solidity >=0.6.2 <0.8.0;
412
413 /**
```

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

`contracts/XttVault.sol`

Locations

```
// File: @openzeppelin/contracts/token/ERC20/SafeERC20.sol

pragma solidity >=0.6.0 <0.8.0;

/**
```

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

`contracts/XttVault.sol`

Locations

```
// File: @openzeppelin/contracts/utils/Pausable.sol

pragma solidity >=0.6.0 <0.8.0;

/**
```

LOW

SWC-115

Use of "tx.origin" as a part of authorization control.

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source file

`contracts/XttVault.sol`

Locations

```
modifier notContract() {
require(!_isContract(msg.sender), "contract not allowed");
require(msg.sender == tx.origin, "proxy contract not allowed");
_;
}
```