MythX

REPORT 62501CD72D60760018751BC3

Created	Fri Apr 08 2022 11:30:31 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	6135edf7a6e184c5d2c6ee1e

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
cf141aa2-0540-47f8-9b82-279a6fd09f05	/batcher/batcher.sol	1

Analysis cf141aa2-0540-47f8-9b82-279a6fd09f05

MythX

Started	Fri Apr 08 2022 11:30:33 GMT+0000 (Coordinated Universal Time)
Finished	Fri Apr 08 2022 11:30:39 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Vscode-Extension
Main Source File	/Batcher/Batcher.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

/batcher/batcher.sol

Locations

```

/**
* @notice Stores the deposits for future batching via periphery
* @param amountIn Value of Lp token to be deposited
* @param signature signature verifying that depositor has enough karma and is authorized to deposit by brahma
```

UNKNOWN Arithmetic operation "-" discovered

SWC-101

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

Source file

/batcher/batcher.sol

Locations

```
}

withdrawLedger[msg.sender] = withdrawLedger[msg.sender] + (amountIn);

vaultInfo.currentAmount -= amountIn;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```
vaultInfo.currentAmount -= amountIn;

emit WithdrawRequest(msg.sender, vaultInfo.vaultAddress, amountIn);
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```

/**
* @notice Allows user to withdraw LP tokens
* @param amount Amount of LP tokens to withdraw
* @param recipient Address to receive the LP tokens
*/
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```
/**
* @notice Allows user to withdraw LP tokens
* @param amount Amount of LP tokens to withdraw
* @param recipient Address to receive the LP tokens
*/
function claimTokens(uint256 amount, address recipient)
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```

/*///////////////////////////////////////////////////////////////
VAULT DEPOSIT/WITHDRAWAL LOGIC
//////////////////////////////////////////////////////////////*/
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```
}

require(amountToDeposit > 0, "NO_DEPOSITS");

uint256 lpTokensReportedByVault = vault.deposit(
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```
require(amountToDeposit > 0, "NO_DEPOSITS");

uint256 lpTokensReportedByVault = vault.deposit(
amountToDeposit,
address(this)
);
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

/batcher/batcher.sol

Locations

```
);

for (uint256 i = 0; i < users.length; i++) {
uint256 userAmount = depositLedger[users[i]];
if (processedAddresses[users[i]]) {
if (userAmount > 0) {
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

/batcher/batcher.sol

Locations

```
if (processedAddresses[users[i]]) {
if (userAmount > 0) {
uint256 userShare = (userAmount * (lpTokensReceived)) /
(amountToDeposit);
userTokens[users[i]] = userTokens[users[i]] + userShare;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

/batcher/batcher.sol

Locations

```
depositLedger[users[i]] = 0;
}
processedAddresses[users[i]] = false;
}
}
}

/**
* @notice Performs withdraws on the periphery for the supplied users in batch
* @param users array of users whose deposits must be resolved
*/
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/batcher/batcher.sol

Locations

```
depositLedger[users[i]] = 0;
}
processedAddresses[users[i]] = false;
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/batcher/batcher.sol

Locations

```

/**
* @notice Performs withdraws on the periphery for the supplied users in batch
* @param users array of users whose deposits must be resolved
*/
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

/batcher/batcher.sol

Locations

```
}

require(amountToWithdraw > 0, "NO_WITHDRAWS");

uint256 wantTokensReportedByVault = vault.withdraw(
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```
require(amountToWithdraw > 0, "NO_WITHDRAWS");

uint256 wantTokensReportedByVault = vault.withdraw(
amountToWithdraw,
address(this)
);
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```
);

for (uint256 i = 0; i < users.length; i++) {
uint256 userAmount = withdrawLedger[users[i]];
if (processedAddresses[users[i]]) {
if (userAmount > 0) {
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```
if (processedAddresses[users[i]]) {
if (userAmount > 0) {
uint256 userShare = (userAmount * wantTokensReceived) /
amountToWithdraw;
token.safeTransfer(users[i], userShare);
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

/batcher/batcher.sol

Locations

```
withdrawLedger[users[i]] = 0;
}
processedAddresses[users[i]] = false;
}
}
}

/*///////////////////////////////////////////////////////////////
INTERNAL HELPERS
//////////////////////////////////////////////////////////////*/
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/batcher/batcher.sol

Locations

```
withdrawLedger[users[i]] = 0;
}
processedAddresses[users[i]] = false;
}
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/batcher/batcher.sol

Locations

```

/// @notice Can be changed by keeper
uint256 public slippageForCurveLp = 30;

/// @notice Helper to convert Lp tokens into USDC
/// @dev Burns LpTokens on UST3-Wormhole pool on curve to get USDC
/// @param lpToken Curve Lp Token
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```
/*///////////////////////////////////////////////////////////////
MAINTAINANCE ACTIONS
//////////////////////////////////////////////////////////////*/

/// @notice Function to set authority address
/// @param authority New authority address
function setAuthority(address authority) public {
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```
/*///////////////////////////////////////////////////////////////
MAINTAINANCE ACTIONS
//////////////////////////////////////////////////////////////*/

/// @notice Function to set authority address
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`/batcher/batcher.sol`

Locations

```
/*///////////////////////////////////////////////////////////////
MAINTAINANCE ACTIONS
//////////////////////////////////////////////////////////////*/

/// @notice Function to set authority address
```

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

/batcher/batcher.sol

Locations

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.4;

import "@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol";
```

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

/batcher/batcher.sol

Locations

```
}

require(amountToDeposit > 0, "NO_DEPOSITS");

uint256 lpTokensReportedByVault = vault.deposit(
amountToDeposit,
address(this)
```

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

/batcher/batcher.sol

Locations

```

uint256 lpTokensReportedByVault = vault.deposit(
amountToDeposit,
address(this)
);
```

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

`/batcher/batcher.sol`

Locations

```
);

uint256 lpTokensReceived = IERC20(address(vault)).balanceOf(address(this)) -
(oldLPBalance);
```

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

`/batcher/batcher.sol`

Locations

```
uint256 userShare = (userAmount * (lpTokensReceived)) /
(amountToDeposit);
userTokens[users[i]] = userTokens[users[i]] + userShare;
depositLedger[users[i]] = 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

`/batcher/batcher.sol`

Locations

```
uint256 userShare = (userAmount * (lpTokensReceived)) /
(amountToDeposit);
userTokens[users[i]] = userTokens[users[i]] + userShare;
depositLedger[users[i]] = 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

/batcher/batcher.sol

Locations

```

/**
* @notice Performs withdraws on the periphery for the supplied users in batch
* @param users array of users whose deposits must be resolved
*/
```

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

/batcher/batcher.sol

Locations

```

/**
* @notice Performs withdraws on the periphery for the supplied users in batch
* @param users array of users whose deposits must be resolved
*/
```

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

/batcher/batcher.sol

Locations

```
/**
* @notice Performs withdraws on the periphery for the supplied users in batch
* @param users array of users whose deposits must be resolved
*/
function batchWithdraw(address[] memory users)
```

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

/batcher/batcher.sol

Locations

```
* @notice Performs withdraws on the periphery for the supplied users in batch
* @param users array of users whose deposits must be resolved
*/
function batchWithdraw(address[] memory users)
external
override
```

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

/batcher/batcher.sol

Locations

```
}

require(amountToWithdraw > 0, "NO_WITHDRAWS");

uint256 wantTokensReportedByVault = vault.withdraw(
amountToWithdraw,
address(this)
```

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

/batcher/batcher.sol

Locations

```

uint256 wantTokensReportedByVault = vault.withdraw(
amountToWithdraw,
address(this)
);
```

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

/batcher/batcher.sol

Locations

```
address(this)
);

uint256 wantTokensReceived = token.balanceOf(address(this)) -
(oldWantBalance);
```

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

/batcher/batcher.sol

Locations

```
uint256 userShare = (userAmount * wantTokensReceived) /
amountToWithdraw;
token.safeTransfer(users[i], userShare);

withdrawLedger[users[i]] = 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

/batcher/batcher.sol

Locations

```
token.safeTransfer(users[i], userShare);

withdrawLedger[users[i]] = 0;
}
processedAddresses[users[i]] = 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

/batcher/batcher.sol

Locations

```
}

/*///////////////////////////////////////////////////////////////
INTERNAL HELPERS
//////////////////////////////////////////////////////////////*/
```

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

/batcher/batcher.sol

Locations

```

/*///////////////////////////////////////////////////////////////
INTERNAL HELPERS
//////////////////////////////////////////////////////////////*/

/// @notice Helper to verify signature against verification authority
```

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

/batcher/batcher.sol

Locations

```
/*///////////////////////////////////////////////////////////////
INTERNAL HELPERS
//////////////////////////////////////////////////////////////*/

/// @notice Helper to verify signature against verification authority
```