MythX

REPORT 62B43DCA1CD79F00189373AD

| | |
|---|---|
| Created | Thu Jun 23 2022 10:17:46 GMT+0000 (Coordinated Universal Time) |
| Number of analyses | 1 |
| User | 6135edf7a6e184c5d2c6ee1e |

## REPORT SUMMARY

| Analyses ID | Main source file | Detected vulnerabilities |
|---|---|---|
| cc83138a-de27-4983-81a3-6fb762fd3608 | /batcher/batcher.sol | 1 |

Analysis cc83138a-de27-4983-81a3-6fb762fd3608 MythX

| | |
|---|---|
| Started | Thu Jun 23 2022 10:17:54 GMT+0000 (Coordinated Universal Time) |
| Finished | Thu Jun 23 2022 10:17:59 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

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

SWC-101

Source file
/batcher/batcher.sol
Locations

```
110 pendingDeposit -
111 pendingWithdrawal +
112 amountIn <=
113 vaultInfo.maxAmount,
114 "MAX_LIMIT_EXCEEDED"
115 );
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file
/batcher/batcher.sol
Locations

```
115 );
116
117 depositLedger[recipient] = depositLedger[recipient] + (amountIn);
118 pendingDeposit = pendingDeposit + amountIn;
119
120 emit DepositRequest(recipient, vaultInfo.vaultAddress, amountIn);
121 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
115 );
116
117 depositLedger[recipient] = depositLedger[recipient] + (amountIn);
118 pendingDeposit = pendingDeposit + amountIn;
119
120 emit DepositRequest(recipient, vaultInfo.vaultAddress, amountIn); emit DepositRequest(recipient, vaultInfo.vaultAddress, amountIn);
121 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
115 );
116
117 depositLedger[recipient] = depositLedger[recipient] + (amountIn);
118 pendingDeposit = pendingDeposit + amountIn;
119
120 emit DepositRequest(recipient, vaultInfo.vaultAddress, amountIn);
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
122
123 /**
124 * @notice User deposits vault LP tokens to be withdrawn. Stores the deposits for future batching via periphery
125 * @param amountIn Value of token to be deposited
126 */
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
123 /**
124 * @notice User deposits vault LP tokens to be withdrawn. Stores the deposits for future batching via periphery
125 * @param amountIn Value of token to be deposited
126 */
127 function initiateWithdrawal(uint256 amountIn)
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
145 }
146
147 withdrawLedger[msg.sender] = withdrawLedger[msg.sender] + (amountIn);
148
149 pendingWithdrawal = pendingWithdrawal + amountIn;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
149 pendingWithdrawal = pendingWithdrawal + amountIn;
150
151 emit WithdrawRequest(msg.sender, vaultInfo.vaultAddress, amountIn);
152 }
153
154 /**
155 * @notice Allows user to collect want token back after successfull batch withdrawal
156 * @param amountOut Amount of token to be withdrawn
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
153
154 /**
155 * @notice Allows user to collect want token back after successfull batch withdrawal
156 * @param amountOut Amount of token to be withdrawn
157 */
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
154 /**
155 * @notice Allows user to collect want token back after successfull batch withdrawal
156 * @param amountOut Amount of token to be withdrawn
157 */
158 function completeWithdrawal(uint256 amountOut, address recipient)
159 external
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
171
172 /**
173 * @notice User deposits vault LP tokens to be withdrawn. Stores the deposits for future batching via periphery
174 * @param cancellationAmount Value of token to be cancelled for withdrawal
175 */
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
196 pendingWithdrawal = pendingWithdrawal - cancellationAmount;
197
198 emit WithdrawRescinded(
199 msg.sender,
200 vaultInfo.vaultAddress,
201 cancellationAmount
202 );
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
199 msg.sender,
200 vaultInfo.vaultAddress,
201 cancellationAmount
202 );
203 }
204
205 /**
206 * @notice Can be used to send LP tokens owed to the recipient
207 * @param amount Amount of LP tokens to withdraw
208 * @param recipient Address to receive the LP tokens
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
204
205 /**
206 * @notice Can be used to send LP tokens owed to the recipient
207 * @param amount Amount of LP tokens to withdraw
208 * @param recipient Address to receive the LP tokens
209 */
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
220 /*////////////////////////////////////////////////////////////////
221 VAULT DEPOSIT/WITHDRAWAL LOGIC
222 ////////////////////////////////////////////////////////////////*/
223
224 /// @notice Ledger to maintain addresses and vault LP tokens which batcher owes them
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
252 depositValues[i] = userDeposit;
253
254 // deposit ledger for that address is set to zero
255 // Incase of duplicate address sent, new deposit amount used for same user will be 0
256 depositLedger[users[i]] = 0;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
254 // deposit ledger for that address is set to zero
255 // Incase of duplicate address sent, new deposit amount used for same user will be 0
256 depositLedger[users[i]] = 0;
257 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
272 uint256 totalUsersProcessed = 0;
273
274 for (uint256 i = 0; i < users.length; i++) {
275 uint256 userAmount = depositValues[i];
276
277 // Checks if userAmount is not 0, only then proceed to allocate LP tokens
278 if (userAmount > 0) {
279 uint256 userShare = (userAmount * (lpTokensReceived)) /
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
277 // Checks if userAmount is not 0, only then proceed to allocate LP tokens
278 if (userAmount > 0) {
279 uint256 userShare = (userAmount * (lpTokensReceived)) /
280 (amountToDeposit);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
281
282 // Allocating LP tokens to user, can be calimed by the user later by calling claimTokens
283 userLPTokens[users[i]] = userLPTokens[users[i]] + userShare;
284 ++totalUsersProcessed;
285 }
286 }
287
288 pendingDeposit = pendingDeposit - amountToDeposit;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
281
282 // Allocating LP tokens to user, can be calimed by the user later by calling claimTokens
283 userLPTokens[users[i]] = userLPTokens[users[i]] + userShare;
284 ++totalUsersProcessed;
285 }
286 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
289
290 emit BatchDepositSuccessful(lpTokensReceived, totalUsersProcessed);
291 }
292
293 /**
294 * @notice Performs withdraws on the periphery for the supplied users in batch
295 * @param users array of users whose deposits must be resolved
296 */
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
292
293 /**
294 * @notice Performs withdraws on the periphery for the supplied users in batch
295 * @param users array of users whose deposits must be resolved
296 */
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
293 /**
294 * @notice Performs withdraws on the periphery for the supplied users in batch
295 * @param users array of users whose deposits must be resolved
296 */
297 function batchWithdraw(address[] memory users)
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
316 withdrawValues[i] = userWithdraw;
317
318 // Withdrawal ledger for that address is set to zero
319 // Incase of duplicate address sent, new withdrawal amount used for same user will be 0
320 withdrawLedger[users[i]] = 0;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
317
318 // Withdrawal ledger for that address is set to zero
319 // Incase of duplicate address sent, new withdrawal amount used for same user will be 0
320 withdrawLedger[users[i]] = 0;
321 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
336
337 for (uint256 i = 0; i < users.length; i++) {
338 uint256 userAmount = withdrawValues[i];
339
340 // Checks if userAmount is not 0, only then proceed to allocate want tokens
341 if (userAmount > 0) {
342 uint256 userShare = (userAmount * wantTokensReceived) /
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
340 // Checks if userAmount is not 0, only then proceed to allocate want tokens
341 if (userAmount > 0) {
342 uint256 userShare = (userAmount * wantTokensReceived) /
343 amountToWithdraw;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
344
345 // Allocating want tokens to user. Can be claimed by the user by calling completeWithdrawal
346 userWantTokens[users[i]] = userWantTokens[users[i]] + userShare;
347 ++totalUsersProcessed;
348 }
349 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
344
345 // Allocating want tokens to user. Can be claimed by the user by calling completeWithdrawal
346 userWantTokens[users[i]] = userWantTokens[users[i]] + userShare;
347 ++totalUsersProcessed;
348 }
349 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
351 pendingWithdrawal = pendingWithdrawal - amountToWithdraw;
352
353 emit BatchWithdrawSuccessful(wantTokensReceived, totalUsersProcessed);
354 }
355
356 /*////////////////////////////////////////////////////
357 INTERNAL HELPERS
358 ////////////////////////////////////////////////////////////////*/
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/batcher/batcher.sol

Locations

```
354 }
355
356 /*////////////////////////////////////////////////////////////////
357 INTERNAL HELPERS
358 ////////////////////////////////////////////////////////////////*/
```

UNKNOWN

SWC-101

## Arithmetic operation "-" discovered

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

Source file

/batcher/batcher.sol

Locations

```
356 /*////////////////////////////////////////////////////////////////
357 INTERNAL HELPERS
358 //////////////////////////////////////////////////////////////////*/
359
360 /// @notice Helper to verify signature against verification authority
```

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

```
1 /// SPDX-License-Identifier: GPL-3.0-or-later
2 pragma solidity ^0.8.4;
3
4 import {IERC20Permit} from "@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.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

```
253
254 // deposit ledger for that address is set to zero
255 // Incase of duplicate address sent, new deposit amount used for same user will be 0
256 depositLedger[users[i]] = 0;
257 }
```

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

/batcher/batcher.sol

Locations

```
255 // Incase of duplicate address sent, new deposit amount used for same user will be 0
256 depositLedger[users[i]] = 0;
257 }
258
259 require(amountToDeposit > 0, "NO_DEPOSITS");
260
261 uint256 lpTokensReportedByVault = vault.deposit(
```

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

/batcher/batcher.sol

Locations

```
264 );
265
266 uint256 lpTokensReceived = IERC20(address(vault)).balanceOf(
267 address(this)
268 ) - (oldLPBalance);
```

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

/batcher/batcher.sol

Locations

```
278 if (userAmount > 0) {
279 uint256 userShare = (userAmount * (lpTokensReceived)) /
280 (amountToDeposit);
281
282 // Allocating LP tokens to user, can be calimed by the user later by calling claimTokens
283 userLPTokens[users[i]] = userLPTokens[users[i]] + userShare;
284 ++totalUsersProcessed;
```

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

/batcher/batcher.sol

Locations

```
288 pendingDeposit = pendingDeposit - amountToDeposit;
289
290 emit BatchDepositSuccessful(lpTokensReceived, totalUsersProcessed);
291 } }
292
293 /**
```

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

/batcher/batcher.sol

Locations

```
292
293 /**
294 * @notice Performs withdraws on the periphery for the supplied users in batch
295 * @param users array of users whose deposits must be resolved
296 */
```

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

/batcher/batcher.sol

Locations

```
317
318 // Withdrawal ledger for that address is set to zero
319 // Incase of duplicate address sent, new withdrawal amount used for same user will be 0
320 withdrawLedger[users[i]] = 0;
321 }
```

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

/batcher/batcher.sol

Locations

```
318 // Withdrawal ledger for that address is set to zero
319 // Incase of duplicate address sent, new withdrawal amount used for same user will be 0
320 withdrawLedger[users[i]] = 0;
321 }
322
323 require(amountToWithdraw > 0, "NO_WITHDRAWS");
324
325 uint256 wantTokensReportedByVault = vault.withdraw(
```

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

/batcher/batcher.sol

Locations

```
328 );
329
330 uint256 wantTokensReceived = token.balanceOf(address(this)) -
331 (oldWantBalance);
```

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

/batcher/batcher.sol

Locations

```
341 if (userAmount > 0) {
342 uint256 userShare = (userAmount * wantTokensReceived) /
343 amountToWithdraw;
344
345 // Allocating want tokens to user. Can be claimed by the user by calling completeWithdrawal
346 userWantTokens[users[i]] = userWantTokens[users[i]] + userShare;
347 ++totalUsersProcessed;
```

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

/batcher/batcher.sol

Locations

```
351 pendingWithdrawal = pendingWithdrawal - amountToWithdraw;
352
353 emit BatchWithdrawSuccessful(wantTokensReceived, totalUsersProcessed);
354 }
```

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

/batcher/batcher.sol

Locations

```
354 }
355
356 /*//////////////////////////////////////////////////////////////
357 INTERNAL HELPERS
358 //////////////////////////////////////////////////////////////*/
```