MythX

REPORT 65FDF14858BAA0001A239C80

Created	Fri Mar 22 2024 20:59:52 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	64ea50748288ab3d19a06f14

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
744b53d2-da8a-4c0f-983e-4ee9b6d24106	DiviBase.sol	8

Analysis 744b53d2-da8a-4c0f-983e-4ee9b6d24106 MythX

Started	Fri Mar 22 2024 20:59:59 GMT+0000 (Coordinated Universal Time)
Finished	Fri Mar 22 2024 21:02:07 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Remythx
Main Source File	DiviBase.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

DiviBase.sol

Locations

```
174 require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
175 unchecked {
176 _approve(sender, _msgSender(), currentAllowance - amount);
177 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
181
182 function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
183 _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
184 return true;
185 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
189 require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
190 unchecked {
191 _approve(_msgSender(), spender, currentAllowance - subtractedValue);
192 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
206 require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
207 unchecked {
208 _balances[sender] = senderBalance - amount;
209 }
210 _balances[recipient] += amount;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
208 _balances[sender] = senderBalance - amount;
209 }
210 _balances[recipient] += amount;
211
212 emit Transfer(sender, recipient, amount);
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
215 function _createInitialSupply(address account, uint256 amount) internal virtual {
216 require(account != address(0), "ERC20: mint to the zero address");
217 _totalSupply += amount;
218 _balances[account] += amount;
219 emit Transfer(address(0), account, amount);
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
216 require(account != address(0), "ERC20: mint to the zero address");
217 _totalSupply += amount;
218 _balances[account] += amount;
219 emit Transfer(address(0), account, amount);
220 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
297 */
298 function add(uint256 a, uint256 b) internal pure returns (uint256) {
299 uint256 c = a + b;
300 require(c >= a, "SafeMath: addition overflow");
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
329 function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
330 require(b <= a, errorMessage);
331 uint256 c = a - b;
332
333 return c;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
352 }
353
354 uint256 c = a * b;
355 require(c / a == b, "SafeMath: multiplication overflow");
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
353
354 uint256 c = a * b;
355 require(c / a == b, "SafeMath: multiplication overflow");
356
357 return c;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
388 function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
389 require(b > 0, errorMessage);
390 uint256 c = a / b;
391 // assert(a == b * c + a % b); // There is no case in which this doesn't hold
```

UNKNOWN Arithmetic operation "%" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
424 function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
425 require(b != 0, errorMessage);
426 return a % b;
427 }
428 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
488 */
489 function mul(int256 a, int256 b) internal pure returns (int256) {
490 int256 c = a * b;
491
492 // Detect overflow when multiplying MIN_INT256 with -1
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
492 // Detect overflow when multiplying MIN_INT256 with -1
493 require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
494 require((b == 0) || (c / b == a));
495 return c;
496 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
504
505 // Solidity already throws when dividing by 0.
506 return a / b;
507 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
511 */
512 function sub(int256 a, int256 b) internal pure returns (int256) {
513 int256 c = a - b;
514 require((b >= 0 && c <= a) || (b < 0 && c > a));
515 return c;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
520 */
521 function add(int256 a, int256 b) internal pure returns (int256) {
522 int256 c = a + b;
523 require((b >= 0 && c >= a) || (b < 0 && c < a));
524 return c;
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
691 // For more discussion about choosing the value of `magnitude`,
692 // see https://github.com/ethereum/EIPs/issues/1726#issuecomment-472352728
693 uint256 constant internal magnitude = 2**128;
694
695 mapping(address => uint256) internal magnifiedDividendPerShare;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
759 if (newBalance > 0) {
760 magnifiedDividendPerShare[nextRewardToken] = magnifiedDividendPerShare[nextRewardToken].add(
761 (newBalance).mul(magnitude) / totalBalance
762 );
763 emit DividendsDistributed(msg.sender, newBalance);
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
765 totalDividendsDistributed[nextRewardToken] = totalDividendsDistributed[nextRewardToken].add(newBalance);
766 }
767 rewardTokenCounter = rewardTokenCounter == rewardTokens.length - 1 ? 0 : rewardTokenCounter + 1;
768 nextRewardToken = rewardTokens[rewardTokenCounter];
769 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
765 totalDividendsDistributed[nextRewardToken] = totalDividendsDistributed[nextRewardToken].add(newBalance);
766 }
767 rewardTokenCounter = rewardTokenCounter == rewardTokens.length - 1 ? 0 : rewardTokenCounter + 1;
768 nextRewardToken = rewardTokens[rewardTokenCounter];
769 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
834 /// @return The amount of dividend in wei that `_owner` has earned in total.
835 function accumulativeDividendOf(address _owner, address _rewardToken) public view override returns(uint256) {
836 return magnifiedDividendPerShare[_rewardToken].mul(holderBalance[_owner]).toInt256Safe()
837 .add(magnifiedDividendCorrections[_rewardToken][_owner]).toUint256Safe() / magnitude;
838 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
843 /// @param value The amount that will be created.
844 function _increase(address account, uint256 value) internal {
845 for (uint256 i; i < rewardTokens.length; i++){
846 magnifiedDividendCorrections[rewardTokens[i]][account] = magnifiedDividendCorrections[rewardTokens[i]][account]
847 .sub((magnifiedDividendPerShare[rewardTokens[i]].mul(value)).toInt256Safe());
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
854 /// @param value The amount that will be burnt.
855 function _reduce(address account, uint256 value) internal {
856 for (uint256 i; i < rewardTokens.length; i++){
857 magnifiedDividendCorrections[rewardTokens[i]][account] = magnifiedDividendCorrections[rewardTokens[i]][account]
858 .add( (magnifiedDividendPerShare[rewardTokens[i]].mul(value)).toInt256Safe() );
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
866 uint256 increaseAmount = newBalance.sub(currentBalance);
867 _increase(account, increaseAmount);
868 totalBalance += increaseAmount;
869 } else if(newBalance < currentBalance) {
870 uint256 reduceAmount = currentBalance.sub(newBalance);
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
870 uint256 reduceAmount = currentBalance.sub(newBalance);
871 _reduce(account, reduceAmount);
872 totalBalance -= reduceAmount;
873 }
874 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
927
928 uint index = tokenHoldersMap.indexOf[key];
929 uint lastIndex = tokenHoldersMap.keys.length - 1;
930 address lastKey = tokenHoldersMap.keys[lastIndex];
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
955 constructor() {
956 claimWait = 1200;
957 minimumTokenBalanceForDividends = 100 * (10**18);
958 }
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
955 constructor() {
956 claimWait = 1200;
957 minimumTokenBalanceForDividends = 100 * (10**18);
958 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1097
1098 while(gasUsed < gas && iterations < numberOfTokenHolders) {
1099 _lastProcessedIndex++;
1100
1101 if(_lastProcessedIndex >= tokenHoldersMap.keys.length) {
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1107 if(canAutoClaim(lastClaimTimes[account])) {
1108 if(processAccount(payable(account), true)) {
1109 claims++;
1110 }
1111 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1111 }
1112
1113 iterations++;
1114
1115 uint256 newGasLeft = gasleft();
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1129 uint256 amount;
1130 bool paid;
1131 for (uint256 i; i < rewardTokens.length; i++){
1132 amount = _withdrawDividendOfUser(account, rewardTokens[i]);
1133 if(amount > 0) {
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1234 constructor() ERC20("DiviBase", "DIV") {
1235
1236 uint256 totalSupply = 500 * 1e3 * 1e18;
1237
1238 maxTransactionAmount = totalSupply * 10 / 1000; // 2% maxTransactionAmountTxn
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1234 constructor() ERC20("DiviBase", "DIV") {
1235
1236 uint256 totalSupply = 500 * 1e3 * 1e18;
1237
1238 maxTransactionAmount = totalSupply * 10 / 1000; // 2% maxTransactionAmountTxn
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1236 uint256 totalSupply = 500 * 1e3 * 1e18;
1237
1238 maxTransactionAmount = totalSupply * 10 / 1000; // 2% maxTransactionAmountTxn
1239 swapTokensAtAmount = totalSupply * 70 / 10000; // 0.7% swap tokens amount
1240 maxWallet = totalSupply * 20 / 1000; // 3% Max wallet
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1236 uint256 totalSupply = 500 * 1e3 * 1e18;
1237
1238 maxTransactionAmount = totalSupply * 10 / 1000; // 2% maxTransactionAmountTxn
1239 swapTokensAtAmount = totalSupply * 70 / 10000; // 0.7% swap tokens amount
1240 maxWallet = totalSupply * 20 / 1000; // 3% Max wallet
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1237
1238 maxTransactionAmount = totalSupply * 10 / 1000; // 2% maxTransactionAmountTxn
1239 swapTokensAtAmount = totalSupply * 70 / 10000; // 0.7% swap tokens amount
1240 maxWallet = totalSupply * 20 / 1000; // 3% Max wallet
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1237
1238 maxTransactionAmount = totalSupply * 10 / 1000; // 2% maxTransactionAmountTxn
1239 swapTokensAtAmount = totalSupply * 70 / 10000; // 0.7% swap tokens amount
1240 maxWallet = totalSupply * 20 / 1000; // 3% Max wallet
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1238 maxTransactionAmount = totalSupply * 10 / 1000; // 2% maxTransactionAmountTxn
1239 swapTokensAtAmount = totalSupply * 70 / 10000; // 0.7% swap tokens amount
1240 maxWallet = totalSupply * 20 / 1000; // 3% Max wallet
1241
1242 rewardsBuyFee = 40;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1238 maxTransactionAmount = totalSupply * 10 / 1000; // 2% maxTransactionAmountTxn
1239 swapTokensAtAmount = totalSupply * 70 / 10000; // 0.7% swap tokens amount
1240 maxWallet = totalSupply * 20 / 1000; // 3% Max wallet
1241
1242 rewardsBuyFee = 40;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1243 divBuyFee = 100;
1244 liquidityBuyFee = 10;
1245 totalBuyFees = rewardsBuyFee + divBuyFee + liquidityBuyFee;
1246 //launch buy 15 sell 25
1247 rewardsSellFee = 100;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1243 divBuyFee = 100;
1244 liquidityBuyFee = 10;
1245 totalBuyFees = rewardsBuyFee + divBuyFee + liquidityBuyFee;
1246 //launch buy 15 sell 25
1247 rewardsSellFee = 100;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1248 divSellFee = 120;
1249 liquiditySellFee = 30;
1250 totalSellFees = rewardsSellFee + divSellFee + liquiditySellFee;
1251
1252 dividendTracker = new DividendTracker();
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1248 divSellFee = 120;
1249 liquiditySellFee = 30;
1250 totalSellFees = rewardsSellFee + divSellFee + liquiditySellFee;
1251
1252 dividendTracker = new DividendTracker();
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1318 require(_msgSender() == dev);
1319
1320 require(newNum > (totalSupply() * 1 / 1000)/1e18, "Cannot set maxTransactionAmount lower than 0.1%");
1321 maxTransactionAmount = newNum * (10**18);
1322 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1318 require(_msgSender() == dev);
1319
1320 require(newNum > (totalSupply() * 1 / 1000)/1e18, "Cannot set maxTransactionAmount lower than 0.1%");
1321 maxTransactionAmount = newNum * (10**18);
1322 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1318 require(_msgSender() == dev);
1319
1320 require(newNum > (totalSupply() * 1 / 1000)/1e18, "Cannot set maxTransactionAmount lower than 0.1%");
1321 maxTransactionAmount = newNum * (10**18);
1322 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1319
1320 require(newNum > (totalSupply() * 1 / 1000)/1e18, "Cannot set maxTransactionAmount lower than 0.1%");
1321 maxTransactionAmount = newNum * (10**18);
1322 }
1323
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1319
1320 require(newNum > (totalSupply() * 1 / 1000)/1e18, "Cannot set maxTransactionAmount lower than 0.1%");
1321 maxTransactionAmount = newNum * (10**18);
1322 }
1323
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1325 require(_msgSender() == dev);
1326
1327 require(newNum > (totalSupply() * 1 / 100)/1e18, "Cannot set maxWallet lower than 1%");
1328 maxWallet = newNum * (10**18);
1329 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1325 require(_msgSender() == dev);
1326
1327 require(newNum > (totalSupply() * 1 / 100)/1e18, "Cannot set maxWallet lower than 1%");
1328 maxWallet = newNum * (10**18);
1329 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1325 require(_msgSender() == dev);
1326
1327 require(newNum > (totalSupply() * 1 / 100)/1e18, "Cannot set maxWallet lower than 1%");
1328 maxWallet = newNum * (10**18);
1329 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1326
1327  require(newNum > (totalSupply() * 1 / 100)/1e18, "Cannot set maxWallet lower than 1%");
1328  maxWallet = newNum * (10**18);
1329  }
1330
```

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1326
1327  require(newNum > (totalSupply() * 1 / 100)/1e18, "Cannot set maxWallet lower than 1%");
1328  maxWallet = newNum * (10**18);
1329  }
1330
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1333  rewardsBuyFee = _rewardsFee;
1334  liquidityBuyFee = _liquidityFee;
1335  totalBuyFees = divBuyFee + rewardsBuyFee + liquidityBuyFee;
1336  require(totalBuyFees <= 100, "Must keep fees at 10% or less");
1337  }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1333  rewardsBuyFee = _rewardsFee;
1334  liquidityBuyFee = _liquidityFee;
1335  totalBuyFees = divBuyFee + rewardsBuyFee + liquidityBuyFee;
1336  require(totalBuyFees <= 100, "Must keep fees at 10% or less");
1337  }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1341 | rewardsSellFee = _rewardsFee;
1342 | liquiditySellFee = _liquidityFee;
1343 | totalSellFees = divSellFee + rewardsSellFee + liquiditySellFee;
1344 | require(totalSellFees <= 100, "Must keep fees at 10% or less");
1345 | }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1341 | rewardsSellFee = _rewardsFee;
1342 | liquiditySellFee = _liquidityFee;
1343 | totalSellFees = divSellFee + rewardsSellFee + liquiditySellFee;
1344 | require(totalSellFees <= 100, "Must keep fees at 10% or less");
1345 | }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1357 |
1358 | function excludeMultipleAccountsFromFees(address[] calldata accounts, bool excluded) external onlyOwner {
1359 | for(uint256 i = 0; i < accounts.length; i++) {
1360 | _isExcludedFromFees[accounts[i]] = excluded;
1361 | }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1512 if (transferDelayEnabled){
1513 if (to != address(uniswapV2Router) && to != address(uniswapV2Pair)){
1514 require(_holderLastTransferTimestamp[tx.origin] < block.number + 1, "_transfer:: Transfer Delay enabled. Only one purchase per block allowed.");
1515 _holderLastTransferTimestamp[tx.origin] = block.number;
1516 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1520 if (automatedMarketMakerPairs[from] && !_isExcludedMaxTransactionAmount[to]) {
1521 require(amount <= maxTransactionAmount, "Buy transfer amount exceeds the maxTransactionAmount.");
1522 require(amount + balanceOf(to) <= maxWallet, "Unable to exceed Max Wallet");
1523 }
1524 //when sell
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1527 }
1528 else if(!_isExcludedMaxTransactionAmount[to]) {
1529 require(amount + balanceOf(to) <= maxWallet, "Unable to exceed Max Wallet");
1530 }
1531 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1560 // no taxes on transfers (non buys/sells)
1561 if(takeFee){
1562 if(tradingActiveBlock + 1 >= block.number && (automatedMarketMakerPairs[to] || automatedMarketMakerPairs[from])){
1563 fees = amount.mul(90).div(100);
1564 tokensForLiquidity += fees * 10 / 99;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1562 if(tradingActiveBlock + 1 >= block.number && (automatedMarketMakerPairs[to] || automatedMarketMakerPairs[from])){
1563 fees = amount.mul(90).div(100);
1564 tokensForLiquidity += fees * 10 / 99;
1565 tokensForRewards += fees * 78 / 99;
1566 tokensFordiv += fees * 2 / 99;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1562 if(tradingActiveBlock + 1 >= block.number && (automatedMarketMakerPairs[to] || automatedMarketMakerPairs[from])){
1563 fees = amount.mul(90).div(100);
1564 tokensForLiquidity += fees * 10 / 99;
1565 tokensForRewards += fees * 78 / 99;
1566 tokensFordiv += fees * 2 / 99;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1562 if(tradingActiveBlock + 1 >= block.number && (automatedMarketMakerPairs[to] || automatedMarketMakerPairs[from])){
1563 fees = amount.mul(90).div(100);
1564 tokensForLiquidity += fees * 10 / 99;
1565 tokensForRewards += fees * 78 / 99;
1566 tokensFordiv += fees * 2 / 99;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1563 fees = amount.mul(90).div(100);
1564 tokensForLiquidity += fees * 10 / 99;
1565 tokensForRewards += fees * 78 / 99;
1566 tokensFordiv += fees * 2 / 99;
1567 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1563 fees = amount.mul(90).div(100);
1564 tokensForLiquidity += fees * 10 / 99;
1565 tokensForRewards += fees * 78 / 99;
1566 tokensFordiv += fees * 2 / 99;
1567 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1563 fees = amount.mul(90).div(100);
1564 tokensForLiquidity += fees * 10 / 99;
1565 tokensForRewards += fees * 78 / 99;
1566 tokensFordiv += fees * 2 / 99;
1567 }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1564 tokensForLiquidity += fees * 10 / 99;
1565 tokensForRewards += fees * 78 / 99;
1566 tokensFordiv += fees * 2 / 99;
1567 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1564 tokensForLiquidity += fees * 10 / 99;
1565 tokensForRewards += fees * 78 / 99;
1566 tokensFordiv += fees * 2 / 99;
1567 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1564 tokensForLiquidity += fees * 10 / 99;
1565 tokensForRewards += fees * 78 / 99;
1566 tokensFordiv += fees * 2 / 99;
1567 }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1570 else if (automatedMarketMakerPairs[to] && totalSellFees > 0){
1571 fees = amount.mul(totalSellFees).div(feeDivisor);
1572 tokensForRewards += fees * rewardsSellFee / totalSellFees;
1573 tokensForLiquidity += fees * liquiditySellFee / totalSellFees;
1574 tokensFordiv += fees * divSellFee / totalSellFees;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1570 else if (automatedMarketMakerPairs[to] && totalSellFees > 0){
1571 fees = amount.mul(totalSellFees).div(feeDivisor);
1572 tokensForRewards += fees * rewardsSellFee / totalSellFees;
1573 tokensForLiquidity += fees * liquiditySellFee / totalSellFees;
1574 tokensFordiv += fees * divSellFee / totalSellFees;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1570 else if (automatedMarketMakerPairs[to] && totalSellFees > 0){
1571 fees = amount.mul(totalSellFees).div(feeDivisor);
1572 tokensForRewards += fees * rewardsSellFee / totalSellFees;
1573 tokensForLiquidity += fees * liquiditySellFee / totalSellFees;
1574 tokensFordiv += fees * divSellFee / totalSellFees;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1571 | fees = amount.mul(totalSellFees).div(feeDivisor);
1572 | tokensForRewards += fees * rewardsSellFee / totalSellFees;
1573 | tokensForLiquidity += fees * liquiditySellFee / totalSellFees;
1574 | tokensFordiv += fees * divSellFee / totalSellFees;
1575 | }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1571 | fees = amount.mul(totalSellFees).div(feeDivisor);
1572 | tokensForRewards += fees * rewardsSellFee / totalSellFees;
1573 | tokensForLiquidity += fees * liquiditySellFee / totalSellFees;
1574 | tokensFordiv += fees * divSellFee / totalSellFees;
1575 | }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1571 | fees = amount.mul(totalSellFees).div(feeDivisor);
1572 | tokensForRewards += fees * rewardsSellFee / totalSellFees;
1573 | tokensForLiquidity += fees * liquiditySellFee / totalSellFees;
1574 | tokensFordiv += fees * divSellFee / totalSellFees;
1575 | }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1572 tokensForRewards += fees * rewardsSellFee / totalSellFees;
1573 tokensForLiquidity += fees * liquiditySellFee / totalSellFees;
1574 tokensFordiv += fees * divSellFee / totalSellFees;
1575 }
1576
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1572 tokensForRewards += fees * rewardsSellFee / totalSellFees;
1573 tokensForLiquidity += fees * liquiditySellFee / totalSellFees;
1574 tokensFordiv += fees * divSellFee / totalSellFees;
1575 }
1576
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1572 tokensForRewards += fees * rewardsSellFee / totalSellFees;
1573 tokensForLiquidity += fees * liquiditySellFee / totalSellFees;
1574 tokensFordiv += fees * divSellFee / totalSellFees;
1575 }
1576
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1578 else if(automatedMarketMakerPairs[from] && totalBuyFees > 0) {
1579 fees = amount.mul(totalBuyFees).div(feeDivisor);
1580 tokensForRewards += fees * rewardsBuyFee / totalBuyFees;
1581 tokensForLiquidity += fees * liquidityBuyFee / totalBuyFees;
1582 tokensFordiv += fees * divBuyFee / totalBuyFees;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1578 | else if(automatedMarketMakerPairs[from] && totalBuyFees > 0) {
1579 | fees = amount.mul(totalBuyFees).div(feeDivisor);
1580 | tokensForRewards += fees * rewardsBuyFee / totalBuyFees;
1581 | tokensForLiquidity += fees * liquidityBuyFee / totalBuyFees;
1582 | tokensFordiv += fees * divBuyFee / totalBuyFees;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1578 | else if(automatedMarketMakerPairs[from] && totalBuyFees > 0) {
1579 | fees = amount.mul(totalBuyFees).div(feeDivisor);
1580 | tokensForRewards += fees * rewardsBuyFee / totalBuyFees;
1581 | tokensForLiquidity += fees * liquidityBuyFee / totalBuyFees;
1582 | tokensFordiv += fees * divBuyFee / totalBuyFees;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1579 | fees = amount.mul(totalBuyFees).div(feeDivisor);
1580 | tokensForRewards += fees * rewardsBuyFee / totalBuyFees;
1581 | tokensForLiquidity += fees * liquidityBuyFee / totalBuyFees;
1582 | tokensFordiv += fees * divBuyFee / totalBuyFees;
1583 | }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1579 | fees = amount.mul(totalBuyFees).div(feeDivisor);
1580 | tokensForRewards += fees * rewardsBuyFee / totalBuyFees;
1581 | tokensForLiquidity += fees * liquidityBuyFee / totalBuyFees;
1582 | tokensFordiv += fees * divBuyFee / totalBuyFees;
1583 | }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1579 | fees = amount.mul(totalBuyFees).div(feeDivisor);
1580 | tokensForRewards += fees * rewardsBuyFee / totalBuyFees;
1581 | tokensForLiquidity += fees * liquidityBuyFee / totalBuyFees;
1582 | tokensFordiv += fees * divBuyFee / totalBuyFees;
1583 | }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1580 | tokensForRewards += fees * rewardsBuyFee / totalBuyFees;
1581 | tokensForLiquidity += fees * liquidityBuyFee / totalBuyFees;
1582 | tokensFordiv += fees * divBuyFee / totalBuyFees;
1583 | }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1580 | tokensForRewards += fees * rewardsBuyFee / totalBuyFees;
1581 | tokensForLiquidity += fees * liquidityBuyFee / totalBuyFees;
1582 | tokensFordiv += fees * divBuyFee / totalBuyFees;
1583 | }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1580 tokensForRewards += fees * rewardsBuyFee / totalBuyFees;
1581 tokensForLiquidity += fees * liquidityBuyFee / totalBuyFees;
1582 tokensFordiv += fees * divBuyFee / totalBuyFees;
1583 }
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1587 }
1588
1589 amount -= fees;
1590 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1643 function swapBack() private {
1644 uint256 contractBalance = balanceOf(address(this));
1645 uint256 totalTokensToSwap = tokensForLiquidity + tokensFordiv + tokensForRewards;
1646
1647 if(contractBalance == 0 || totalTokensToSwap == 0) {return;}
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1643 function swapBack() private {
1644 uint256 contractBalance = balanceOf(address(this));
1645 uint256 totalTokensToSwap = tokensForLiquidity + tokensFordiv + tokensForRewards;
1646
1647 if(contractBalance == 0 || totalTokensToSwap == 0) {return;}
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1648
1649  // Halve the amount of liquidity tokens
1650  uint256 liquidityTokens = contractBalance * tokensForLiquidity / totalTokensToSwap / 2;
1651  uint256 amountToSwapForETH = contractBalance.sub(liquidityTokens);
1652
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1648
1649  // Halve the amount of liquidity tokens
1650  uint256 liquidityTokens = contractBalance * tokensForLiquidity / totalTokensToSwap / 2;
1651  uint256 amountToSwapForETH = contractBalance.sub(liquidityTokens);
1652
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1648
1649  // Halve the amount of liquidity tokens
1650  uint256 liquidityTokens = contractBalance * tokensForLiquidity / totalTokensToSwap / 2;
1651  uint256 amountToSwapForETH = contractBalance.sub(liquidityTokens);
1652
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1657  uint256 ethBalance = address(this).balance.sub(initialETHBalance);
1658
1659  uint256 ethFordiv = ethBalance.mul(tokensFordiv).div(totalTokensToSwap - (tokensForLiquidity/2));
1660  uint256 ethForRewards = ethBalance.mul(tokensForRewards).div(totalTokensToSwap - (tokensForLiquidity/2));
1661
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1657 uint256 ethBalance = address(this).balance.sub(initialETHBalance);
1658
1659 uint256 ethFordiv = ethBalance.mul(tokensFordiv).div(totalTokensToSwap - (tokensForLiquidity/2));
1660 uint256 ethForRewards = ethBalance.mul(tokensForRewards).div(totalTokensToSwap - (tokensForLiquidity/2));
1661
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1658
1659 uint256 ethFordiv = ethBalance.mul(tokensFordiv).div(totalTokensToSwap - (tokensForLiquidity/2));
1660 uint256 ethForRewards = ethBalance.mul(tokensForRewards).div(totalTokensToSwap - (tokensForLiquidity/2));
1661
1662 uint256 ethForLiquidity = ethBalance - ethFordiv - ethForRewards;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1658
1659 uint256 ethFordiv = ethBalance.mul(tokensFordiv).div(totalTokensToSwap - (tokensForLiquidity/2));
1660 uint256 ethForRewards = ethBalance.mul(tokensForRewards).div(totalTokensToSwap - (tokensForLiquidity/2));
1661
1662 uint256 ethForLiquidity = ethBalance - ethFordiv - ethForRewards;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1660 uint256 ethForRewards = ethBalance.mul(tokensForRewards).div(totalTokensToSwap - (tokensForLiquidity/2));
1661
1662 uint256 ethForLiquidity = ethBalance - ethFordiv - ethForRewards;
1663
1664 tokensForLiquidity = 0;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1660 uint256 ethForRewards = ethBalance.mul(tokensForRewards).div(totalTokensToSwap - (tokensForLiquidity/2));
1661
1662 uint256 ethForLiquidity = ethBalance - ethFordiv - ethForRewards;
1663
1664 tokensForLiquidity = 0;
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1698 function airdrop(address[] memory airdropWallets, uint256[] memory amounts) external onlyOwner {
1699 require(airdropWallets.length < 200, "Can only airdrop 200 wallets per txn due to gas limits"); // allows for airdrop
1700 for(uint256 i = 0; i < airdropWallets.length; i++){
1701 address wallet = airdropWallets[i];
1702 uint256 amount = amounts[i] * (1e18);
```

UNKNOWN

Arithmetic operation "*" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
1700 for(uint256 i = 0; i < airdropWallets.length; i++){
1701 address wallet = airdropWallets[i];
1702 uint256 amount = amounts[i] * (1e18);
1703 _transfer(msg.sender, wallet, amount);
1704 }
```

UNKNOWN

Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
765 totalDividendsDistributed[nextRewardToken] = totalDividendsDistributed[nextRewardToken].add(newBalance);
766 }
767 rewardTokenCounter = rewardTokenCounter == rewardTokens.length - 1 ? 0 : rewardTokenCounter + 1;
768 nextRewardToken = rewardTokens[rewardTokenCounter];
769 }
```

UNKNOWN

Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

DiviBase.sol

Locations

```
927
928 uint index = tokenHoldersMap.indexOf[key];
929 uint lastIndex = tokenHoldersMap.keys.length - 1;
930 address lastKey = tokenHoldersMap.keys[lastIndex];
```

LOW

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

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

SWC-115

Source file

DiviBase.sol

Locations

```
1455 function processDividendTracker(uint256 gas) external {
1456 (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) = dividendTracker.process(gas);
1457 emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, false, gas, tx.origin);
1458 }
```

LOW

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

SWC-115

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

DiviBase.sol

Locations

```
1512 if (transferDelayEnabled){
1513 if (to != address(uniswapV2Router) && to != address(uniswapV2Pair)){
1514 require(_holderLastTransferTimestamp[tx.origin] < block.number + 1, "_transfer:: Transfer Delay enabled. Only one purchase per block allowed.");
1515 _holderLastTransferTimestamp[tx.origin] = block.number;
1516 }
```

LOW

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

SWC-115

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source file

DiviBase.sol

Locations

```
1513 if (to != address(uniswapV2Router) && to != address(uniswapV2Pair)){
1514 require(_holderLastTransferTimestamp[tx.origin] < block.number + 1, "_transfer:: Transfer Delay enabled. Only one purchase per block allowed.");
1515 _holderLastTransferTimestamp[tx.origin] = block.number;
1516 }
1517 }
```

LOW

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

SWC-115

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source file

DiviBase.sol

Locations

```
1599
1600 try dividendTracker.process(gas) returns (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) {
1601 emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, true, gas, tx.origin);
1602 }
1603 catch {}
```

UNKNOWN

SWC-110

Public state variable with array type causing reacheable exception by default.

The public state variable "rewardTokens" in "DividendPayingToken" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source file

DiviBase.sol

Locations

```
694
695 mapping(address => uint256) internal magnifiedDividendPerShare;
696 address[] public rewardTokens;
697 address public nextRewardToken;
698 uint256 public rewardTokenCounter;
```

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

DiviBase.sol

Locations

```
728 rewardTokens.push(address(0x4200000000000000000000000000000000000006));
729
730 nextRewardToken = rewardTokens[0];
731 }
```

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

DiviBase.sol

Locations

```
766 }
767 rewardTokenCounter = rewardTokenCounter == rewardTokens.length - 1 ? 0 : rewardTokenCounter + 1;
768 nextRewardToken = rewardTokens[rewardTokenCounter];
769 }
770
```

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

DiviBase.sol

Locations

```
773 // generate the uniswap pair path of weth -> eth
774 address[] memory path = new address[](2);
775 path[0] = uniswapV2Router.WETH();
776 path[1] = rewardToken;
```

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

DiviBase.sol

Locations

```
774 address[] memory path = new address[](2);
775 path[0] = uniswapV2Router.WETH();
776 path[1] = rewardToken;
777
778 // make the swap
```

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

DiviBase.sol

Locations

```
844 function _increase(address account, uint256 value) internal {
845 for (uint256 i; i < rewardTokens.length; i++){
846 magnifiedDividendCorrections[rewardTokens[i]][account] = magnifiedDividendCorrections[rewardTokens[i]][account]
847 .sub((magnifiedDividendPerShare[rewardTokens[i]].mul(value)).toInt256Safe());
848 }
```

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

DiviBase.sol

Locations

```
844 function _increase(address account, uint256 value) internal {
845 for (uint256 i; i < rewardTokens.length; i++){
846 magnifiedDividendCorrections[rewardTokens[i]][account] = magnifiedDividendCorrections[rewardTokens[i]][account]
847 .sub((magnifiedDividendPerShare[rewardTokens[i]].mul(value)).toInt256Safe());
848 }
```

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

DiviBase.sol

Locations

```
845 for (uint256 i; i < rewardTokens.length; i++){
846 magnifiedDividendCorrections[rewardTokens[i]][account] = magnifiedDividendCorrections[rewardTokens[i]][account]
847 .sub((magnifiedDividendPerShare[rewardTokens[i]].mul(value)).toInt256Safe());
848 }
849 }
```

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

DiviBase.sol

Locations

```
855 function _reduce(address account, uint256 value) internal {
856 for (uint256 i; i < rewardTokens.length; i++){
857 magnifiedDividendCorrections[rewardTokens[i]][account] = magnifiedDividendCorrections[rewardTokens[i]][account]
858 .add( (magnifiedDividendPerShare[rewardTokens[i]].mul(value)).toInt256Safe() );
859 }
```

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

DiviBase.sol

Locations

```
855 function _reduce(address account, uint256 value) internal {
856 for (uint256 i; i < rewardTokens.length; i++){
857 magnifiedDividendCorrections[rewardTokens[i]][account] = magnifiedDividendCorrections[rewardTokens[i]][account]
858 .add( (magnifiedDividendPerShare[rewardTokens[i]].mul(value)).toInt256Safe() );
859 }
```

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

DiviBase.sol

Locations

```
856 for (uint256 i; i < rewardTokens.length; i++){
857 magnifiedDividendCorrections[rewardTokens[i]][account] = magnifiedDividendCorrections[rewardTokens[i]][account]
858 .add( (magnifiedDividendPerShare[rewardTokens[i]].mul(value)).toInt256Safe() );
859 }
860 }
```

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

DiviBase.sol

Locations

```
898
899 function getKeyAtIndex(uint index) private view returns (address) {
900 return tokenHoldersMap.keys[index];
901 }
```

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

DiviBase.sol

Locations

```
928 uint index = tokenHoldersMap.indexOf[key];
929 uint lastIndex = tokenHoldersMap.keys.length - 1;
930 address lastKey = tokenHoldersMap.keys[lastIndex];
931
932 tokenHoldersMap.indexOf[lastKey] = index;
```

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

DiviBase.sol

Locations

```
933 delete tokenHoldersMap.indexOf[key];
934
935 tokenHoldersMap.keys[index] = lastKey;
936 tokenHoldersMap.keys.pop();
937 }
```

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

DiviBase.sol

Locations

```
1103 }
1104
1105 address account = tokenHoldersMap.keys[_lastProcessedIndex];
1106
1107 if(canAutoClaim(lastClaimTimes[account])) {
```

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

DiviBase.sol

Locations

```
1130 bool paid;
1131 for (uint256 i; i < rewardTokens.length; i++){
1132 amount = _withdrawDividendOfUser(account, rewardTokens[i]);
1133 if(amount > 0) {
1134 lastClaimTimes[account] = block.timestamp;
```

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

DiviBase.sol

Locations

```
1358 function excludeMultipleAccountsFromFees(address[] calldata accounts, bool excluded) external onlyOwner {
1359 for(uint256 i = 0; i < accounts.length; i++) {
1360 _isExcludedFromFees[accounts[i]] = excluded;
1361 }
```

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

DiviBase.sol

Locations

```
1609 // generate the uniswap pair path of token -> weth
1610 address[] memory path = new address[](2);
1611 path[0] = address(this);
1612 path[1] = uniswapV2Router.WETH();
```

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

DiviBase.sol

Locations

```
1610 address[] memory path = new address[](2);
1611 path[0] = address(this);
1612 path[1] = uniswapV2Router.WETH();
1613
1614 _approve(address(this), address(uniswapV2Router), tokenAmount);
```

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

DiviBase.sol

Locations

```
1699 require(airdropWallets.length < 200, "Can only airdrop 200 wallets per txn due to gas limits"); // allows for airdrop
1700 for(uint256 i = 0; i < airdropWallets.length; i++){
1701 address wallet = airdropWallets[i];
1702 uint256 amount = amounts[i] * (1e18);
1703 _transfer(msg.sender, wallet, amount);
```

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

DiviBase.sol

Locations

```
1700 for(uint256 i = 0; i < airdropWallets.length; i++){
1701 address wallet = airdropWallets[i];
1702 uint256 amount = amounts[i] * (1e18);
1703 _transfer(msg.sender, wallet, amount);
1704 }
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

DiviBase.sol

Locations

```
1312 tradingActive = true;
1313 swapEnabled = true;
1314 tradingActiveBlock = block.number;
1315 }
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

DiviBase.sol

Locations

```
1512 if (transferDelayEnabled){
1513 if (to != address(uniswapV2Router) && to != address(uniswapV2Pair)){
1514 require(_holderLastTransferTimestamp[tx.origin] < block.number + 1, "_transfer:: Transfer Delay enabled. Only one purchase per block allowed.");
1515 _holderLastTransferTimestamp[tx.origin] = block.number;
1516 }
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

DiviBase.sol

Locations

```
1513 if (to != address(uniswapV2Router) && to != address(uniswapV2Pair)){
1514 require(_holderLastTransferTimestamp[tx.origin] < block.number + 1, "_transfer:: Transfer Delay enabled. Only one purchase per block allowed.");
1515 _holderLastTransferTimestamp[tx.origin] = block.number;
1516 }
1517 }
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

DiviBase.sol

Locations

```
1560 // no taxes on transfers (non buys/sells)
1561 if(takeFee){
1562 if(tradingActiveBlock + 1 >= block.number && (automatedMarketMakerPairs[to] || automatedMarketMakerPairs[from])){
1563 fees = amount.mul(90).div(100);
1564 tokensForLiquidity += fees * 10 / 99;
```