MythX

REPORT 60D9FB214C4BE900197640CC

Created	Mon Jun 28 2021 16:38:57 GMT+0000 (Coordinated Universal Time)
Number of analyses	10
User	5c54c4ab1cdad40016867258

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
6a461f25-435c-4381-965d-74fce2b404c2	contracts/GuardedLaunchUpgradable.sol	0
1c502ca2-888f-4b64-93eb-1d57047f4cc5	contracts/IdleCDOTrancheRewardsStorage.sol	0
07d07e4f-a246-427b-a336-eec4d5573882	contracts/IdleCDO.sol	13
428a63db-fdd8-4e2a-8a43-1a07727ee535	contracts/IdleCDOStorage.sol	2
fd3df4dd-7da0-4af1-afce-eb005447c239	contracts/IdleCDOTranche.sol	15
ea4af467-0a99-4bc8-b2ff-7bd1eee2b5cd	contracts/IdleStrategy.sol	2
de4514dc-fb3c-4c13-8bf1-e0b005007f93	contracts/IdleCDOTrancheRewards.sol	8
299cd8ea-ce0b-4e04-a86c-0382060c10c6	contracts/mocks/MockIdleCDO.sol	6
88b970cd-83bf-41ea-af10-295763022d07	contracts/mocks/MockERC20.sol	15
afec93cd-26d6-4c01-bd47-324bb3ce6b5e	contracts/mocks/MockIdleToken.sol	16

Analysis 6a461f25-435c-4381-965d-74fce2b404c2 MythX

Started	Mon Jun 28 2021 16:39:20 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jun 28 2021 17:24:24 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/GuardedLaunchUpgradable.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	0

ISSUES

Analysis 1c502ca2-888f-4b64-93eb-1d57047f4cc5 MythX

Started	Mon Jun 28 2021 16:39:20 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jun 28 2021 17:24:20 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/IdleCDOTrancheRewardsStorage.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	0

ISSUES

Analysis 07d07e4f-a246-427b-a336-eec4d5573882 MythX

Started	Mon Jun 28 2021 16:39:30 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jun 28 2021 17:25:26 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/IdleCDO.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	13

ISSUES

LOW

SWC-115

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.

Source file
contracts/IdleCDO.sol
Locations

```
734 /// @dev Set last caller and block.number hash. This should be called at the beginning of the first function to protect
735 function _updateCallerBlock() internal {
736 _lastCallerBlock = keccak256(abi.encodePacked(tx.origin, block.number));
737 }
```

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/IdleCDO.sol
Locations

```
739 /// @dev Check that the second function is not called in the same tx from the same tx.origin
740 function _checkSameTx() internal view {
741 require(keccak256(abi.encodePacked(tx.origin, block.number)) != _lastCallerBlock, "SAME_BLOCK");
742 }
```

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/IdleCDO.sol

Locations

```
572 /// @return liquidated amount in underlyings
573 function liquidate(uint256 _amount, bool _revertIfNeeded) external returns (uint256) {
574 require(msg.sender == rebalancer || msg.sender == owner(), "IDLE:!AUTH");
575 return _liquidate(_amount, _revertIfNeeded);
576 }
```

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/IdleCDO.sol

Locations

```
739 /// @dev Check that the second function is not called in the same tx from the same tx.origin
740 function _checkSameTx() internal view {
741 require(keccak256(abi.encodePacked(tx.origin, block.number)) != _lastCallerBlock, "SAME_BLOCK");
742 }
```

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/IdleCDO.sol

Locations

```
500 /// @param _minAmount array of min amounts for uniswap trades. Lenght should be equal to the _skipReward array
501 function harvest(bool _skipRedeem, bool _skipIncentivesUpdate, bool[] calldata _skipReward, uint256[] calldata _minAmount) external {
502 require(msg.sender == rebalancer || msg.sender == owner(), "IDLE:!AUTH");
503 // Fetch state variable once to save gas
504 address _token = token;
```

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/IdleCDO.sol

Locations

```
500 /// @param _minAmount array of min amounts for uniswap trades. Lenght should be equal to the _skipReward array
501 function harvest(bool _skipRedeem, bool _skipIncentivesUpdate, bool[] calldata _skipReward, uint256[] calldata _minAmount) external {
502 require(msg.sender == rebalancer || msg.sender == owner(), "IDLE:!AUTH");
503 // Fetch state variable once to save gas
504 address _token = token;
```

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/IdleCDO.sol

Locations

```
357 _amount = IERC20Detailed(_tranche).balanceOf(msg.sender);
358 }
359 require(_amount > 0, 'IDLE:IS_0');
360 address _token = token;
361 // get current net unlent balance
```

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/IdleCDO.sol

Locations

```
365 // to steal interest generated from rewards
366 toRedeem = _amount * _lastTranchePrice(_tranche) / ONE_TRANCHE_TOKEN;
367 if (toRedeem > balanceUnderlying) {
368 // if the unlent balance is not enough we try to redeem what's missing directly from the strategy
369 // and then add the current balanceUnderlying
370 // NOTE: there could be a difference of up to 100 wei due to rounding
371 toRedeem = _liquidate(toRedeem - balanceUnderlying, revertIfTooLow) + balanceUnderlying;
372 }
373 // burn tranche token // burn tranche token
374 IdleCDOTranche(_tranche).burn(msg.sender, _amount);
375 // send underlying to msg.sender
```

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

node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol

Locations

```
47  */
48  modifier onlyOwner() {
49  require(owner() == _msgSender(), "Ownable: caller is not the owner");
50  _;
51  }
```

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

contracts/IdleCDO.sol

Locations

```
734 /// @dev Set last caller and block.number hash. This should be called at the beginning of the first function to protect
735 function _updateCallerBlock() internal {
736 _lastCallerBlock = keccak256(abi.encodePacked(tx.origin, block.number));
737 }
```

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

contracts/IdleCDO.sol

Locations

```
739 /// @dev Check that the second function is not called in the same tx from the same tx.origin
740 function _checkSameTx() internal view {
741 require(keccak256(abi.encodePacked(tx.origin, block.number)) != _lastCallerBlock, "SAME_BLOCK");
742 }
```

LOW

SWC-120

A control flow decision is made based on The block.number environment variable.

The block.number environment variable is used to determine a control flow decision. 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

contracts/IdleCDO.sol

Locations

```
739 /// @dev Check that the second function is not called in the same tx from the same tx.origin
740 function _checkSameTx() internal view {
741 require(keccak256(abi.encodePacked(tx.origin, block.number)) != _lastCallerBlock, "SAME_BLOCK");
742 }
```

LOW

Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

SWC-123

Source file

contracts/IdleCDO.sol

Locations

```
458 /// @return last saved price for minting tranche tokens, in underlyings
459 function _tranchePrice(address _tranche) internal view returns (uint256) {
460 if (IdleCDOTranche(_tranche).totalSupply() == 0) {
461 return oneToken;
462 }
```

Source file

contracts/IdleCDO.sol

Locations

```
19 /// @author Idle Labs Inc.
20 /// @dev The contract is upgradable, to add storage slots, create IdleCDOStorageVX and inherit from IdleCDOStorage, then update the definitaion below
21 contract IdleCDO is Initializable, PausableUpgradeable, GuardedLaunchUpgradable, IdleCDOStorage {
22 using SafeERC20Upgradeable for IERC20Detailed;
23
24 // ###################
25 // Initializer
26 // ###################
27
28 /// @notice can only be called once
29 /// @dev Initialize the upgradable contract
30 /// @param _limit contract value limit
31 /// @param _guardedToken underlying token
32 /// @param _governanceFund address where funds will be sent in case of emergency
33 /// @param _owner guardian address
34 /// @param _rebalancer rebalancer address
35 /// @param _strategy strategy address
36 /// @param _trancheAPRSplitRatio trancheAPRSplitRatio value
37 /// @param _trancheIdealWeightRatio trancheIdealWeightRatio value
38 /// @param _incentiveTokens array of addresses for incentive tokens
39 function initialize(
40 uint256 _limit, address _guardedToken, address _governanceFund, address _owner, // GuardedLaunch args
41 address _rebalancer,
42 address _strategy,
43 uint256 _trancheAPRSplitRatio, // for AA tranches, so eg 10000 means 10% interest to AA and 90% BB
44 uint256 _trancheIdealWeightRatio, // for AA tranches, so eg 10000 means 10% of tranches are AA and 90% BB
45 address[] memory _incentiveTokens
46 ) public initializer {
47 // Initialize contracts
48 PausableUpgradeable.__Pausable_init();
49 GuardedLaunchUpgradable.__GuardedLaunch_init(_limit, _governanceFund, _owner);
50 // Deploy Tranches tokens
51 AATranche = address(new IdleCDOTranche("Idle CDO AA Tranche", "IDLE_CDO_AA"));
52 BBTranche = address(new IdleCDOTranche("Idle CDO BB Tranche", "IDLE_CDO_BB"));
53 // Set CDO params
54 token = _guardedToken;
55 strategy = _strategy; //@audit sanity check on the addresses + guardedToken + rebalancer
56 strategyToken = IIdleCDOStrategy(_strategy).strategyToken();
57 rebalancer = _rebalancer;
58 trancheAPRSplitRatio = _trancheAPRSplitRatio;
59 trancheIdealWeightRatio = _trancheIdealWeightRatio;
60 idealRange = 10000; // trancheIdealWeightRatio ± 10%
61 uint256 _oneToken = 10**(IERC20Detailed(_guardedToken).decimals());
62 oneToken = _oneToken;
63 uniswapRouterV2 = IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
```

```
    weth = address(0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2);
    incentiveTokens = _incentiveTokens;
    priceAA = _oneToken;
    priceBB = _oneToken;
    lastAAPrice = _oneToken;
    lastBBPrice = _oneToken;
    unlentPerc = 2000; // 2%
    // Set flags
    allowAAWithdraw = true;
    allowBBWithdraw = true;
    revertIfTooLow = true;
    // skipDefaultCheck = false is the default value
    // Set allowance for strategy
    IERC20Detailed(_guardedToken).safeIncreaseAllowance(_strategy, type(uint256).max);
    IERC20Detailed(strategyToken).safeIncreaseAllowance(_strategy, type(uint256).max);
    // Save current strategy price
    lastStrategyPrice = strategyPrice();
    // Fee params
    fee = 10000; // 10% performance fee
    feeReceiver = address(0xBecC659Bfc6EDcA552fa1A67451cC6b38a0108E4); // feeCollector //@audit hardcoded fee address
    guardian = _owner;
  }

  // ###############
  // Public methods
  // ###############

  /// @notice pausable
  /// @dev msg.sender should approve this contract first to spend `_amount` of `token`
  /// @param _amount amount of `token` to deposit
  /// @return AA tranche tokens minted
  function depositAA(uint256 _amount) external whenNotPaused returns (uint256) {
    return _deposit(_amount, AATranche);
  }

  /// @notice pausable
  /// @dev msg.sender should approve this contract first to spend `_amount` of `token`
  /// @param _amount amount of `token` to deposit
  /// @return BB tranche tokens minted
  function depositBB(uint256 _amount) external whenNotPaused returns (uint256) {
    return _deposit(_amount, BBTranche);
  }

  /// @notice pausable
  /// @param _amount amount of AA tranche tokens to burn
  /// @return underlying tokens redeemed
  function withdrawAA(uint256 _amount) external nonReentrant returns (uint256) {
    require(!paused() || allowAAWithdraw, 'IDLE:AA_!ALLOWED');
    return _withdraw(_amount, AATranche);
  }

  /// @notice pausable
  /// @param _amount amount of BB tranche tokens to burn
  /// @return underlying tokens redeemed
  function withdrawBB(uint256 _amount) external nonReentrant returns (uint256) {
    require(!paused() || allowBBWithdraw, 'IDLE:BB_!ALLOWED');
    return _withdraw(_amount, BBTranche);
  }

  // ###############
  // Views
  // ###############

```

```
/// @param _tranche tranche address
/// @return tranche price
function tranchePrice(address _tranche) external view returns (uint256) {
return _tranchePrice(_tranche);
}

/// @param _tranche tranche address
/// @return last tranche price
function lastTranchePrice(address _tranche) external view returns (uint256) {
return _lastTranchePrice(_tranche);
}

/// @notice rewards (gov tokens) are not counted. It may include non accrued fees (in unclaimedFees)
/// @return contract value in underlyings
function getContractValue() public override view returns (uint256) {
address _strategyToken = strategyToken;
// strategyTokens value in underlying + unlent balance
uint256 strategyTokenDecimals = IERC20Detailed(_strategyToken).decimals();
return (_contractTokenBalance(_strategyToken) * strategyPrice() / (10**(strategyTokenDecimals))) + _contractTokenBalance(token);
} //@audit review ^ formula

/// @param _tranche tranche address
/// @return apr at ideal trancheIdealWeightRatio balance between AA and BB
function getIdealApr(address _tranche) external view returns (uint256) {
return _getApr(_tranche, trancheIdealWeightRatio);
}

/// @param _tranche tranche address
/// @return actual apr given current ratio between AA and BB tranches
function getApr(address _tranche) external view returns (uint256) {
return _getApr(_tranche, getCurrentAARatio());
}

/// @return strategy net apr
function strategyAPR() public view returns (uint256) {
return IIdleCDOStrategy(strategy).getApr();
}

/// @return strategy price, in underlyings
function strategyPrice() public view returns (uint256) {
return IIdleCDOStrategy(strategy).price();
}

/// @return array of reward token addresses
function getRewards() public view returns (address[] memory) {
return IIdleCDOStrategy(strategy).getRewardTokens();
}

/// @return AA tranches ratio (in underlying value) considering all NAV
function getCurrentAARatio() public view returns (uint256) {
uint256 AABal = virtualBalance(AATranche);
uint256 contractVal = AABal + virtualBalance(BBTranche);
if (contractVal == 0) {
return 0;
}
// Current AA tranche split ratio = AABal * FULL_ALLOC / getContractValue()
return AABal * FULL_ALLOC / contractVal;
}

/// @notice this should always be >= of _tranchePrice(_tranche)
/// @dev useful for showing updated gains on frontends
/// @param _tranche address of the requested tranche
/// @return tranche price with current nav
```

```
function virtualPrice(address _tranche) public view returns (uint256) { //@audit HERE!
uint256 nav = getContractValue(); //@audit deep and dangerous
uint256 lastNAV = _lastNAV();
uint256 trancheSupply = IdleCDOTranche(_tranche).totalSupply();
uint256 _trancheAPRSplitRatio = trancheAPRSplitRatio;

if (lastNAV == 0 || trancheSupply == 0) {
return oneToken;
}
// If there is no gain return the current saved price
if (nav <= lastNAV) {
return _tranchePrice(_tranche);
}

uint256 gain = nav - lastNAV;
// remove performance fee
gain -= gain * fee / FULL_ALLOC;
// trancheNAV is: lastNAV + trancheGain
uint256 trancheNAV;
if (_tranche == AATranche) {
// trancheGain (AAGain) = gain * trancheAPRSplitRatio / FULL_ALLOC;
trancheNAV = lastNAVAA + (gain * _trancheAPRSplitRatio / FULL_ALLOC);
} else {
// trancheGain (BBGain) = gain * (FULL_ALLOC - trancheAPRSplitRatio) / FULL_ALLOC;
trancheNAV = lastNAVBB + (gain * (FULL_ALLOC - _trancheAPRSplitRatio) / FULL_ALLOC);
}
// price => trancheNAV * ONE_TRANCHE_TOKEN / trancheSupply
return trancheNAV * ONE_TRANCHE_TOKEN / trancheSupply;
}

/// @param _tranche address of the requested tranche
/// @return net asset value, in underlying tokens, for _tranche considering all nav
function virtualBalance(address _tranche) public view returns (uint256) {
return IdleCDOTranche(_tranche).totalSupply() * virtualPrice(_tranche) / ONE_TRANCHE_TOKEN; //@audit verify
}

/// @return array with addresses of incentiveTokens
function getIncentiveTokens() public view returns (address[] memory) {
return incentiveTokens;
}

// ###############
// Internal
// ###############

/// @notice automatically reverts on lending provider default (strategyPrice decreased)
/// Ideally users should deposit right after an `harvest` call to maximize profit
/// @dev this contract must be approved to spend at least _amount of `token` before calling this method
/// @param _amount amount of underlyings (`token`) to deposit
/// @param _tranche tranche address
/// @return _minted number of tranche tokens minted
function _deposit(uint256 _amount, address _tranche) internal returns (uint256 _minted) {
// check that we are not depositing more than the contract available limit
_guarded(_amount);
// set _lastCallerBlock hash
_updateCallerBlock();
// check if strategyPrice decreased
_checkDefault();
// interest accrued since last mint/redeem/harvest is splitted between AA and BB
// according to trancheAPRSplitRatio. NAVs of AA and BB are so updated and tranche
// prices adjusted accordingly
_updatePrices();
// NOTE: mint of shares should be done before transferring funds
```

```
// mint tranches tokens according to the current prices
_minted = _mintShares(_amount, msg.sender, _tranche);
// get underlyings from sender
IERC20Detailed(token).safeTransferFrom(msg.sender, address(this), _amount);
}

/// @dev accrues interest to the tranches (update NAVs variables) and updates tranche prices
function _updatePrices() internal {
uint256 _oneToken = oneToken;
// get last saved total net asset value
uint256 lastNAV = _lastNAV();
if (lastNAV == 0) {
return;
}
// get the current total net asset value
uint256 nav = getContractValue();
if (nav <= lastNAV) {
return;
}
// Calculate gain since last update
uint256 gain = nav - lastNAV;
// get performance fee amount
uint256 performanceFee = gain * fee / FULL_ALLOC;
gain -= performanceFee;
// and add the value to unclaimedFees
unclaimedFees += performanceFee;
// Get the current tranche supply
uint256 AATotSupply = IdleCDOTranche(AATranche).totalSupply();
uint256 BBTotSupply = IdleCDOTranche(BBTranche).totalSupply();
uint256 AAGain;
uint256 BBGain;
if (BBTotSupply == 0) {
// all gain to AA
AAGain = gain;
} else if (AATotSupply == 0) {
// all gain to BB
BBGain = gain;
} else {
// split the gain between AA and BB holders according to trancheAPRSplitRatio
AAGain = gain * trancheAPRSplitRatio / FULL_ALLOC;
BBGain = gain - AAGain;
}
// Update NAVs
lastNAVAA += AAGain;
// BBGain
lastNAVBB += BBGain;
// Update tranche prices
priceAA = AATotSupply > 0 ? lastNAVAA * ONE_TRANCHE_TOKEN / AATotSupply : _oneToken;
priceBB = BBTotSupply > 0 ? lastNAVBB * ONE_TRANCHE_TOKEN / BBTotSupply : _oneToken;
}

/// @param _amount, in underlyings, to convert in tranche tokens
/// @param _to receiver address of the newly minted tranche tokens
/// @param _tranche tranche address
/// @return _minted number of tranche tokens minted
function _mintShares(uint256 _amount, address _to, address _tranche) internal returns (uint256 _minted) {
// calculate # of tranche token to mint based on current tranche price
_minted = _amount * ONE_TRANCHE_TOKEN / _tranchePrice(_tranche);
IdleCDOTranche(_tranche).mint(_to, _minted);
// update NAV with the _amount of underlyings added
if (_tranche == AATranche) {
lastNAVAA += _amount;
} else {
```

```
lastNAVBB += _amount;
}
}

/// @notice this will be called only during harvests
/// @param _amount amount of underlyings to deposit
/// @return _currAARatio current AA ratio
function _depositFees(uint256 _amount) internal returns (uint256 _currAARatio) {
if (_amount > 0) {
_currAARatio = getCurrentAARatio();
_mintShares(_amount, feeReceiver,
// Choose the right tranche to mint based on getCurrentAARatio
_currAARatio >= trancheIdealWeightRatio ? BBTranche : AATranche
);
// reset unclaimedFees counter
unclaimedFees = 0;
}
}

/// @dev updates last tranche prices with the current ones
function _updateLastTranchePrices() internal {
lastAAPrice = priceAA();
lastBBPrice = priceBB();
}

/// @notice automatically reverts on lending provider default (strategyPrice decreased)
/// a user should wait at least one harvest before redeeming otherwise the redeemed amount
/// would be less than the deposited one due to the use of a checkpointed price at last harvest
/// Ideally users should redeem right after an `harvest` call
/// @param _amount in tranche tokens
/// @param _tranche tranche address
/// @return toRedeem number of underlyings redeemed
function _withdraw(uint256 _amount, address _tranche) internal returns (uint256 toRedeem) {
// check if a deposit is made in the same block from the same user
_checkSameTx();
// check if strategyPrice decreased
_checkDefault();
// accrue interest to tranches and updates tranche prices
_updatePrices();
// redeem all user balance if 0 is passed as _amount
if (_amount == 0) {
_amount = IERC20Detailed(_tranche).balanceOf(msg.sender);
}
require(_amount > 0, 'IDLE:IS_0');
address _token = token;
// get current net unlent balance
uint256 balanceUnderlying = _contractTokenBalance(_token);
// Calculate the amount to redeem using the checkpointed price from last harvest
// NOTE: if use _tranchePrice directly one can deposit a huge amount before an harvest
// to steal interest generated from rewards
toRedeem = _amount * _lastTranchePrice(_tranche) / ONE_TRANCHE_TOKEN;
if (toRedeem > balanceUnderlying) {
// if the unlent balance is not enough we try to redeem what's missing directly from the strategy
// and then add the current balanceUnderlying
// NOTE: there could be a difference of up to 100 wei due to rounding
toRedeem = _liquidate(toRedeem - balanceUnderlying, revertIfTooLow) + balanceUnderlying;
}
// burn tranche token
IdleCDOTranche(_tranche).burn(msg.sender, _amount);
// send underlying to msg.sender
IERC20Detailed(_token).safeTransfer(msg.sender, toRedeem);

// update NAV with the _amount of underlyings removed
```

```
379 if (_tranche == AATranche) {
380 lastNAVAA -= toRedeem;
381 } else {
382 lastNAVBB -= toRedeem;
383 }
384 }
385
386 /// @dev check if strategyPrice is decreased since last update and updates last saved strategy price
387 function _checkDefault() internal {
388 uint256 currPrice = strategyPrice();
389 if (!skipDefaultCheck) {
390 require(lastStrategyPrice <= currPrice, "IDLE:DEFAULT_WAIT_SHUTDOWN");
391 }
392 lastStrategyPrice = currPrice;
393 }
394
395 /// @dev this should liquidate at least _amount or revertIfNeeded
396 /// @param _amount in underlying tokens
397 /// @param _revertIfNeeded flag whether to revert or not if the redeemed amount is not enough
398 /// @return _redeemedTokens number of underlyings redeemed
399 function _liquidate(uint256 _amount, bool _revertIfNeeded) internal returns (uint256 _redeemedTokens) {
400 _redeemedTokens = IIdleCDOStrategy(strategy).redeemUnderlying(_amount);
401 if (_revertIfNeeded) {
402 // keep 100 wei as margin for rounding errors
403 require(_redeemedTokens + 100 >= _amount, 'IDLE:TOO_LOW');
404 }
405 }
406
407 /// @notice sends specific rewards to the tranche rewards staking contracts
408 function _updateIncentives(uint256 currAARatio) internal {
409 // Read state variables only once to save gas
410 uint256 _trancheIdealWeightRatio = trancheIdealWeightRatio;
411 uint256 _trancheAPRSplitRatio = trancheAPRSplitRatio;
412 uint256 _idealRange = idealRange;
413 address _BBStaking = BBStaking;
414 address _AAStaking = AAStaking;
415
416 // Check if BB tranches should be rewarded (is AA ratio high)
417 if (_BBStaking != address(0) && (currAARatio > (_trancheIdealWeightRatio + _idealRange))) {
418 // give more rewards to BB holders, ie send some rewards to BB Staking contract
419 return _depositIncentiveToken(_BBStaking, FULL_ALLOC);
420 }
421 // Check if AA tranches should be rewarded (is AA ratio low)
422 if (_AAStaking != address(0) && (currAARatio < (_trancheIdealWeightRatio - _idealRange))) {
423 // give more rewards to AA holders, ie send some rewards to AA Staking contract
424 return _depositIncentiveToken(_AAStaking, FULL_ALLOC);
425 }
426
427 // Split rewards according to trancheAPRSplitRatio in case the ratio between
428 // AA and BB is already ideal
429 // NOTE: the order is important here, first there must be the deposit for AA rewards
430 _depositIncentiveToken(_AAStaking, _trancheAPRSplitRatio);
431 // NOTE: here we should use FULL_ALLOC directly and not (FULL_ALLOC - _trancheAPRSplitRatio)
432 // because contract balance for incentive tokens is fetched at each _depositIncentiveToken
433 // and the balance for AA already transferred
434 _depositIncentiveToken(_BBStaking, FULL_ALLOC);
435 }
436
437 /// @notice sends requested ratio of incentive tokens reward to a specific IdleCDOTrancheRewards contract
438 /// @param _stakingContract address which will receive incentive Rewards
439 /// @param _ratio ratio of the incentive token balance to send
440 function _depositIncentiveToken(address _stakingContract, uint256 _ratio) internal {
441 address[] memory _incentiveTokens = incentiveTokens;
```

```
    for (uint256 i = 0; i < _incentiveTokens.length; i++) {
    address _incentiveToken = _incentiveTokens[i];
    // deposit the requested ratio of the current contract balance of _incentiveToken to `_to`
    uint256 _reward = _contractTokenBalance(_incentiveToken) * _ratio / FULL_ALLOC;
    if (_reward > 0) {
    IIdleCDOTrancheRewards(_stakingContract).depositReward(_incentiveToken, _reward);
    }
    }
    }

    /// @return the total saved net asset value for all tranches
    function _lastNAV() internal view returns (uint256) {
    return lastNAVAA + lastNAVBB;
    }

    /// @param _tranche tranche address
    /// @return last saved price for minting tranche tokens, in underlyings
    function _tranchePrice(address _tranche) internal view returns (uint256) {
    if (IdleCDOTranche(_tranche).totalSupply() == 0) {
    return oneToken;
    }
    return _tranche == AATranche ? priceAA : priceBB;
    }

    /// @param _tranche tranche address
    /// @return last saved price for redeeming tranche tokens (updated on harvests), in underlyings
    function _lastTranchePrice(address _tranche) internal view returns (uint256) {
    return _tranche == AATranche ? lastAAPrice : lastBBPrice;
    }

    /// @notice the apr can be higher than the strategy apr
    /// @dev returns the current apr for a tranche based on trancheAPRSplitRatio and the provided AA split ratio
    /// @param _tranche tranche token
    /// @param _AATrancheSplitRatio AA split ratio used for calculations
    /// @return apr for the specific tranche
    function _getApr(address _tranche, uint256 _AATrancheSplitRatio) internal view returns (uint256) {
    uint256 stratApr = strategyAPR();
    uint256 _trancheAPRSplitRatio = trancheAPRSplitRatio;
    bool isAATranche = _tranche == AATranche;
    if (_AATrancheSplitRatio == 0) {
    return isAATranche ? 0 : stratApr;
    }
    return isAATranche ?
    stratApr * _trancheAPRSplitRatio / _AATrancheSplitRatio :
    stratApr * (FULL_ALLOC - _trancheAPRSplitRatio) / (FULL_ALLOC - _AATrancheSplitRatio);
    }

    // ###################
    // Protected
    // ###################

    /// @notice can be called only by the rebalancer or the owner
    /// @dev it redeems rewards if any from the lending provider of the strategy and converts them in underlyings.
    /// it also deposits eventual unlent balance already present in the contract with the strategy.
    /// This method will be called by an exteranl keeper bot which will call the method sistematically (eg once a day)
    /// @param _skipRedeem whether to redeem rewards from strategy or not (for gas savings)
    /// @param _skipIncentivesUpdate whether to update incentives or not
    /// @param _skipReward array of flags for skipping the market sell of specific rewards. Lenght should be equal to the `getRewards()` array
    /// @param _minAmount array of min amounts for uniswap trades. Lenght should be equal to the _skipReward array
    function harvest(bool _skipRedeem, bool _skipIncentivesUpdate, bool[] calldata _skipReward, uint256[] calldata _minAmount) external {
    require(msg.sender == rebalancer || msg.sender == owner(), "IDLE:!AUTH");
    // Fetch state variable once to save gas
    address _token = token;
```

```
address _strategy = strategy;
// Check whether to redeem rewards from strategy or not
if (!_skipRedeem) {
// Fetch state variables once to save gas
address[] memory _incentiveTokens = incentiveTokens;
address _weth = weth;
IUniswapV2Router02 _uniRouter = uniswapRouterV2;
// Redeem all rewards associated with the strategy
IIdleCDOStrategy(_strategy).redeemRewards();
// get all rewards addresses
address[] memory rewards = getRewards();
for (uint256 i = 0; i < rewards.length; i++) {
address rewardToken = rewards[i];
// get the balance of a specific reward
uint256 _currentBalance = _contractTokenBalance(rewardToken);
// check if it should be sold or not
if (_skipReward[i] || _currentBalance == 0 || _includesAddress(_incentiveTokens, rewardToken)) { continue; }
// Prepare path for uniswap trade
address[] memory _path = new address[](3);
_path[0] = rewardToken;
_path[1] = _weth;
_path[2] = _token;
// approve the uniswap router to spend our reward
IERC20Detailed(rewardToken).safeIncreaseAllowance(address(_uniRouter), _currentBalance);
// do the uniswap trade
_uniRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(
_currentBalance,
_minAmount[i],
_path,
address(this),
block.timestamp + 1
);
}
// split converted rewards and update tranche prices for mint
// NOTE: that fee on gov tokens will be accumulated in unclaimedFees
_updatePrices();
// update last saved prices for redeems at this point
// if we arrived here we assume all reward tokens with 'big' balance have been sold in the market
// others could have been skipped (with flags set off chain) but it just means that
// were not worth a lot so should be safe to assume that those wont' be siphoned from theft of interest attacks
// NOTE: This method call should not be inside the `if finalBalance > initialBalance` just in case
// no rewards are distributed from the underlying strategy
_updateLastTranchePrices();

// Get fees in the form of totalSupply diluition
// NOTE we return currAARatio to reuse it in _updateIncentives and so to save some gas
uint256 currAARatio = _depositFees(unclaimedFees);

if (!_skipIncentivesUpdate) {
// Update tranche incentives distribution and send rewards to staking contracts
_updateIncentives(currAARatio);
}
}
// If we _skipRedeem we don't need to call _updatePrices because lastNAV is already updated

// Keep some unlent balance for cheap redeems and as reserve of last resort
uint256 underlyingBal = _contractTokenBalance(_token);
uint256 idealUnlent = getContractValue() * unlentPerc / FULL_ALLOC;
if (underlyingBal > idealUnlent) {
// Put unlent balance at work in the lending provider
IIdleCDOStrategy(_strategy).deposit(underlyingBal - idealUnlent);
}
}
```

```
568
569 /// @notice can be called only by the rebalancer or the owner
570 /// @param _amount in underlyings to liquidate from lending provider
571 /// @param _revertIfNeeded flag to revert if amount liquidated is too low
572 /// @return liquidated amount in underlyings
573 function liquidate(uint256 _amount, bool _revertIfNeeded) external returns (uint256) {
574 require(msg.sender == rebalancer || msg.sender == owner(), "IDLE:!AUTH");
575 return _liquidate(_amount, _revertIfNeeded);
576 }
577
578 // ###################
579 // onlyOwner
580 // ###################
581
582 /// @param _allowed flag to allow AA withdraws
583 function setAllowAAWithdraw(bool _allowed) external onlyOwner {
584 allowAAWithdraw = _allowed;
585 }
586
587 /// @param _allowed flag to allow BB withdraws
588 function setAllowBBWithdraw(bool _allowed) external onlyOwner {
589 allowBBWithdraw = _allowed;
590 }
591
592 /// @param _allowed flag to enable the 'default' check (whether strategyPrice decreased or not)
593 function setSkipDefaultCheck(bool _allowed) external onlyOwner {
594 skipDefaultCheck = _allowed;
595 }
596
597 /// @param _allowed flag to enable the check if redeemed amount during liquidations is enough
598 function setRevertIfTooLow(bool _allowed) external onlyOwner {
599 revertIfTooLow = _allowed;
600 }
601
602 /// @notice updates the strategy used (potentially changing the lending protocol used)
603 /// @dev it's REQUIRED to liquidate / redeem everything from the lending provider before changing strategy
604 /// it's also REQUIRED to transfer out any incentive tokens accrued if those are changed from the current ones
605 /// if the lending provider is changes
606 /// @param _strategy new strategy address
607 /// @param _incentiveTokens array of incentive tokens addresses
608 function setStrategy(address _strategy, address[] memory _incentiveTokens) external onlyOwner {
609 require(_strategy != address(0), 'IDLE:IS_0');
610 IERC20Detailed _token = IERC20Detailed(token);
611 // revoke allowance for the current strategy
612 address _currStrategy = strategy;
613 _token.safeApprove(_currStrategy, 0);
614 IERC20Detailed(strategyToken).safeApprove(_currStrategy, 0);
615 // Updated strategy variables
616 strategy = _strategy;
617 // Update incentive tokens
618 incentiveTokens = _incentiveTokens;
619 // Update strategyToken
620 address _newStrategyToken = IIdleCDOStrategy(_strategy).strategyToken();
621 strategyToken = _newStrategyToken;
622 // Approve underlyingToken
623 _token.safeIncreaseAllowance(_strategy, type(uint256).max);
624 // Approve strategyToken
625 IERC20Detailed(_newStrategyToken).safeIncreaseAllowance(_strategy, type(uint256).max);
626 // Update last strategy price
627 lastStrategyPrice = strategyPrice();
628 }
629
630 /// @param _rebalancer new rebalancer address
```

```
631  function setRebalancer(address _rebalancer) external onlyOwner {
632  require((rebalancer = _rebalancer) != address(0), 'IDLE:IS_0');
633  }
634
635  /// @param _feeReceiver new fee receiver address
636  function setFeeReceiver(address _feeReceiver) external onlyOwner {
637  require((feeReceiver = _feeReceiver) != address(0), 'IDLE:IS_0');
638  }
639
640  /// @param _guardian new guardian (pauser) address
641  function setGuardian(address _guardian) external onlyOwner {
642  require((guardian = _guardian) != address(0), 'IDLE:IS_0');
643  }
644
645  /// @param _fee new fee
646  function setFee(uint256 _fee) external onlyOwner {
647  require((fee = _fee) <= MAX_FEE, 'IDLE:TOO_HIGH');
648  }
649
650  /// @param _unlentPerc new unlent percentage
651  function setUnlentPerc(uint256 _unlentPerc) external onlyOwner {
652  require((unlentPerc = _unlentPerc) <= FULL_ALLOC, 'IDLE:TOO_HIGH');
653  }
654
655  /// @param _idealRange new ideal range
656  function setIdealRange(uint256 _idealRange) external onlyOwner {
657  require((idealRange = _idealRange) <= FULL_ALLOC, 'IDLE:TOO_HIGH');
658  }
659
660  /// @dev it's REQUIRED to transfer out any incentive tokens accrued before
661  /// @param _incentiveTokens array with new incentive tokens
662  function setIncentiveTokens(address[] memory _incentiveTokens) external onlyOwner {
663  incentiveTokens = _incentiveTokens;
664  }
665
666  /// @notice Set tranche Rewards contract addresses (for tranches incentivization)
667  /// @param _AAStaking IdleCDOTrancheRewards contract address for AA tranches
668  /// @param _BBStaking IdleCDOTrancheRewards contract address for BB tranches
669  function setStakingRewards(address _AAStaking, address _BBStaking) external onlyOwner {
670  // Read state variable once
671  address[] memory _incentiveTokens = incentiveTokens;
672  address _currAAStaking = AAStaking;
673  address _currBBStaking = BBStaking;
674
675  // Remove allowance for current contracts
676  for (uint256 i = 0; i < _incentiveTokens.length; i++) {
677  IERC20Detailed _incentiveToken = IERC20Detailed(_incentiveTokens[i]);
678  if (_currAAStaking != address(0)) {
679  _incentiveToken.safeApprove(_currAAStaking, 0);
680  }
681  if (_currAAStaking != address(0)) {
682  _incentiveToken.safeApprove(_currBBStaking, 0);
683  }
684  }
685
686  // Update staking contract addresses
687  AAStaking = _AAStaking;
688  BBStaking = _BBStaking;
689
690  // Increase allowance for new contracts
691  for (uint256 i = 0; i < _incentiveTokens.length; i++) {
692  IERC20Detailed _incentiveToken = IERC20Detailed(_incentiveTokens[i]);
693  // Approve each staking contract to spend each incentiveToken on beahlf of this contract
```

```
694 _incentiveToken.safeIncreaseAllowance(_AAStaking, type(uint256).max);
695 _incentiveToken.safeIncreaseAllowance(_BBStaking, type(uint256).max);
696 }
697 }
698
699 /// @notice can be called by both the owner and the guardian
700 /// @dev pause deposits and redeems for all classes of tranches
701 function emergencyShutdown() external {
702 require(msg.sender == guardian || msg.sender == owner(), "IDLE:!AUTH");
703 _pause();
704 allowAAWithdraw = false;
705 allowBBWithdraw = false;
706 skipDefaultCheck = true;
707 revertIfTooLow = true;
708 }
709
710 /// @notice can be called by both the owner and the guardian
711 /// @dev Pauses deposits and redeems
712 function pause() external {
713 require(msg.sender == guardian || msg.sender == owner(), "IDLE:!AUTH");
714 _pause();
715 }
716
717 /// @notice can be called by both the owner and the guardian
718 /// @dev Unpauses deposits and redeems
719 function unpause() external {
720 require(msg.sender == guardian || msg.sender == owner(), "IDLE:!AUTH");
721 _unpause();
722 }
723
724 // ###################
725 // Helpers
726 // ###################
727
728 /// @param _token token address
729 /// @return balance of `_token` for this contract
730 function _contractTokenBalance(address _token) internal view returns (uint256) {
731 return IERC20Detailed(_token).balanceOf(address(this));
732 }
733
734 /// @dev Set last caller and block.number hash. This should be called at the beginning of the first function to protect
735 function _updateCallerBlock() internal {
736 _lastCallerBlock = keccak256(abi.encodePacked(tx.origin, block.number));
737 }
738
739 /// @dev Check that the second function is not called in the same tx from the same tx.origin
740 function _checkSameTx() internal view {
741 require(keccak256(abi.encodePacked(tx.origin, block.number)) != _lastCallerBlock, "SAME_BLOCK");
742 }
743
744 /// @dev this method is only used to check whether a token is an incentive tokens or not
745 /// in the harvest call. The maximum number of element in the array will be a small number (eg at most 3-5)
746 /// @param _array array of addresses to search for an element
747 /// @param _val address of an element to find
748 /// @return flag if the _token is an incentive token or not
749 function _includesAddress(address[] memory _array, address _val) internal pure returns (bool) {
750 for (uint256 i = 0; i < _array.length; i++) {
751 if (_array[i] == _val) {
752 return true;
753 }
754 }
755 // explicit return to fix linter
756 return false;
```

```
757 }
758 }
```

Analysis 428a63db-fdd8-4e2a-8a43-1a07727ee535 MythX

Started	Mon Jun 28 2021 16:39:30 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jun 28 2021 17:24:31 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/IdleCDOStorage.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	2

ISSUES

LOW

SWC-131

Unused state variable "_lastCallerBlock".

The state variable "_lastCallerBlock" is declared within the contract "IdleCDOStorage" but its value does not seem to be used anywhere.

Source file
contracts/IdleCDOStorage.sol
Locations

```
12 uint256 public constant ONE_TRANCHE_TOKEN = 10**18;
13 // variable used to save the last tx.origin and block.number
14 bytes32 internal _lastCallerBlock;
15 // WETH address
16 address public weth;
```

LOW

SWC-131

Unused state variable "uniswapRouterV2".

The state variable "uniswapRouterV2" is declared within the contract "IdleCDOStorage" but its value does not seem to be used anywhere.

Source file
contracts/IdleCDOStorage.sol
Locations

```
26 address public rebalancer;
27 // address of the uniswap v2 router
28 IUniswapV2Router02 internal uniswapRouterV2;
29
30 // Flag for allowing AA withdraws
```

Analysis fd3df4dd-7da0-4af1-afce-eb005447c239 MythX

Started	Mon Jun 28 2021 16:39:30 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jun 28 2021 17:24:41 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/IdleCDOTranche.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	11	4

ISSUES

MEDIUM Function could be marked as external.

SWC-000 The function definition of "name" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file
node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol
Locations

```
57  * @dev Returns the name of the token.
58  */
59 function name() public view virtual returns (string memory) {
60 return _name;
61 }
62
63 /**
```

MEDIUM Function could be marked as external.

SWC-000 The function definition of "symbol" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file
node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol
Locations

```
65  * name.
66  */
67 function symbol() public view virtual returns (string memory) {
68 return _symbol;
69 }
70
71 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "decimals" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
82 * {IERC20-balanceOf} and {IERC20-transfer}.
83 */
84 function decimals() public view virtual returns (uint8) {
85 return 18;
86 }
87
88 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "totalSupply" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
89 * @dev See {IERC20-totalSupply}.
90 */
91 function totalSupply() public view virtual override returns (uint256) {
92 return _totalSupply;
93 }
94
95 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "balanceOf" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
96 * @dev See {IERC20-balanceOf}.
97 */
98 function balanceOf(address account) public view virtual override returns (uint256) {
99 return _balances[account];
100 }
101
102 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "transfer" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
108 * - the caller must have a balance of at least `amount`.
109 */
110 function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
111 _transfer(_msgSender(), recipient, amount);
112 return true;
113 }
114
115 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "allowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
116 * @dev See {IERC20-allowance}.
117 */
118 function allowance(address owner, address spender) public view virtual override returns (uint256) {
119 return _allowances[owner][spender];
120 }
121
122 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "approve" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
127 * - `spender` cannot be the zero address.
128 */
129 function approve(address spender, uint256 amount) public virtual override returns (bool) {
130 _approve(_msgSender(), spender, amount);
131 return true;
132 }
133
134 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
145 * `amount`.
146 */
147 function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
148 _transfer(sender, recipient, amount);
149
150 uint256 currentAllowance = _allowances[sender][_msgSender()];
151 require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
152 _approve(sender, _msgSender(), currentAllowance - amount);
153
154 return true;
155 }
156
157 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "increaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
167 * - `spender` cannot be the zero address.
168 */
169 function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
170 _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
171 return true;
172 }
173
174 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decreaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
186 * `subtractedValue`.
187 */
188 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
189 uint256 currentAllowance = _allowances[_msgSender()][spender];
190 require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
191 _approve(_msgSender(), spender, currentAllowance - subtractedValue);
192
193 return true;
194 }
195
196 /**
```

LOW

SWC-100

Function visibility is not set.

The function definition of "null" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source file

contracts/IdleCDOTranche.sol

Locations

```
11 /// @param _name tranche name
12 /// @param _symbol tranche symbol
13 constructor(
14 string memory _name, // eg. IdleDAI
15 string memory _symbol // eg. IDLEDAI
16 ) ERC20(_name, _symbol) {
17 // minter is msg.sender which is IdleCDO (in initialize)
18 minter = msg.sender;
19 }
20
21 /// @param account that should receive the tranche tokens
```

LOW

SWC-131

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
300 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
301 */
302 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
303 }
```

LOW

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

SWC-131

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
300 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
301 */
302 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
303 }
```

LOW

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

SWC-131

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
300 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
301 */
302 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
303 }
```

Analysis ea4af467-0a99-4bc8-b2ff-7bd1eee2b5cd MythX

Started	Mon Jun 28 2021 16:39:30 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jun 28 2021 17:24:42 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/IdleStrategy.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	2

ISSUES

LOW

SWC-107

Write to persistent state following external call.

The contract account state is accessed after an external call. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol

Locations

```
63  // By storing the original value once again, a refund is triggered (see
64  // https://eips.ethereum.org/EIPS/eip-2200)
65  _status = _NOT_ENTERED;
66  }
67  uint256[49] private __gap;
```

LOW

Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

SWC-123

Source file

node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol

Locations

```
117
118 // solhint-disable-next-line avoid-low-level-calls
119 (bool success, bytes memory returndata) = target.call{ value: value }(data);
120 return _verifyCallResult(success, returndata, errorMessage);
121 }
```

Source file

contracts/IdleStrategy.sol

Locations

```
15 /// @dev This contract should not have any funds at the end of each tx.
16 /// The contract is upgradable, to add storage slots, add them after the last `###### End of storage VXX`
17 contract IdleStrategy is Initializable, OwnableUpgradeable, ReentrancyGuardUpgradeable, IIdleCDOStrategy {
18 using SafeERC20Upgradeable for IERC20Detailed;
19
20 /// ###### Storage V1
21 /// @notice one idleToken (all idleTokens have 18 decimals)
22 uint256 public constant ONE_IDLE_TOKEN = 10**18;
23 /// @notice address of the strategy used, in this case idleToken address
24 address public override strategyToken;
25 /// @notice underlying token address (eg DAI)
26 address public override token;
27 /// @notice one underlying token
28 uint256 public override oneToken;
29 /// @notice decimals of the underlying asset
30 uint256 public override tokenDecimals;
31 /// @notice underlying ERC20 token contract
32 IERC20Detailed public underlyingToken;
33 /// @notice idleToken contract
34 IIdleToken public idleToken;
35 /// ###### End of storage V1
36
37 // ###################
38 // Initializer
39 // ###################
40
41 /// @notice can only be called once
42 /// @dev Initialize the upgradable contract
43 /// @param _strategyToken address of the strategy token
44 /// @param _owner owner address
45 function initialize(address _strategyToken, address _owner) public initializer {
46 // Initialize contracts
47 OwnableUpgradeable.__Ownable_init();
48 ReentrancyGuardUpgradeable.__ReentrancyGuard_init();
49 // Set basic parameters
50 strategyToken = _strategyToken;
51 token = IIdleToken(_strategyToken).token();
52 tokenDecimals = IERC20Detailed(token).decimals();
53 oneToken = 10**(tokenDecimals);
54 idleToken = IIdleToken(_strategyToken);
55 underlyingToken = IERC20Detailed(token);
56 underlyingToken.safeApprove(_strategyToken, type(uint256).max);
57 // transfer ownership
58 transferOwnership(_owner);
59 }
```

```
60
61  // ###################
62  // Public methods
63  // ###################
64
65  /// @dev msg.sender should approve this contract first to spend `_amount` of `token`
66  /// @param _amount amount of `token` to deposit
67  /// @return minted strategyTokens minted
68  function deposit(uint256 _amount) external override returns (uint256 minted) {
69  if (_amount > 0) {
70  IIdleToken _idleToken = idleToken;
71  /// get `tokens` from msg.sender
72  underlyingToken.safeTransferFrom(msg.sender, address(this), _amount);
73  /// deposit those in Idle
74  minted = _idleToken.mintIdleToken(_amount, true, address(0));
75  /// transfer idleTokens to msg.sender
76  _idleToken.transfer(msg.sender, minted);
77  }
78  }
79
80  /// @dev msg.sender should approve this contract first to spend `_amount` of `strategyToken`
81  /// @param _amount amount of strategyTokens to redeem
82  /// @return amount of underlyings redeemed
83  function redeem(uint256 _amount) external override returns(uint256) {
84  return _redeem(_amount);
85  }
86
87  /// @notice Anyone can call this because this contract holds no idleTokens and so no 'old' rewards
88  /// @dev msg.sender should approve this contract first to spend `_amount` of `strategyToken`.
89  /// redeem rewards and transfer them to msg.sender
90  function redeemRewards() external override {
91  IIdleToken _idleToken = idleToken;
92  // Get all idleTokens from msg.sender
93  uint256 bal = _idleToken.balanceOf(msg.sender); //@audit-info flashloan?
94  if (bal > 0) {
95  _idleToken.transferFrom(msg.sender, address(this), bal);
96  // Do a 0 redeem to get gov tokens
97  _idleToken.redeemIdleToken(0);
98  // Give all idleTokens back to msg.sender
99  _idleToken.transfer(msg.sender, bal);
100 // Send all gov tokens to msg.sender
101 _withdrawGovToken(msg.sender);
102 }
103 }
104
105 /// @dev msg.sender should approve this contract first
106 /// to spend `_amount * ONE_IDLE_TOKEN / price()` of `strategyToken`
107 /// @param _amount amount of underlying tokens to redeem
108 /// @return amount of underlyings redeemed
109 function redeemUnderlying(uint256 _amount) external override returns(uint256) {
110 // we are getting price before transferring so price of msg.sender
111 return _redeem(_amount * ONE_IDLE_TOKEN / price());
112 }
113
114 // ###################
115 // Internal
116 // ###################
117
118 /// @notice sends all gov tokens in this contract to an address
119 /// @dev only called
120 /// @param _to address where to send gov tokens (rewards)
121 function _withdrawGovToken(address _to) internal {
122 address[] memory _govTokens = idleToken.getGovTokens();
```

```
for (uint256 i = 0; i < _govTokens.length; i++) {
IERC20Detailed govToken = IERC20Detailed(_govTokens[i]);
// get the current contract balance
uint256 bal = govToken.balanceOf(address(this));
if (bal > 0) {
// transfer all gov tokens
govToken.safeTransfer(_to, bal);
}
}
}

/// @dev msg.sender should approve this contract first to spend `_amount` of `strategyToken`
/// @param _amount amount of strategyTokens to redeem
/// @return redeemed amount of underlyings redeemed
function _redeem(uint256 _amount) internal returns(uint256 redeemed) {
if (_amount > 0) {
IIdleToken _idleToken = idleToken;
// get idleTokens from the user
_idleToken.transferFrom(msg.sender, address(this), _amount);
// redeem underlyings from Idle
redeemed = _idleToken.redeemIdleToken(_amount);
// transfer underlyings to msg.sender
underlyingToken.safeTransfer(msg.sender, redeemed);
// transfer gov tokens to msg.sender
_withdrawGovToken(msg.sender);
}
}

// ###################
// Views
// ###################

/// @return net price in underlyings of 1 strategyToken
function price() public override view returns(uint256) {
// idleToken price is specific to each user
return idleToken.tokenPriceWithFee(msg.sender);
}

/// @return apr net apr (fees should already be excluded)
function getApr() external override view returns(uint256 apr) {
IIdleToken _idleToken = idleToken;
apr = _idleToken.getAvgAPR();
// remove fee
// 100000 => 100% in IdleToken contracts
apr -= apr * _idleToken.fee() / 100000;
}

/// @return tokens array of reward token addresses
function getRewardTokens() external override view returns(address[] memory tokens) {
return idleToken.getGovTokens();
}

// ###################
// Protected
// ###################

/// @notice This contract should not have funds at the end of each tx, this method is just for leftovers
/// @dev Emergency method
/// @param _token address of the token to transfer
/// @param value amount of `_token` to transfer
/// @param _to receiver address
function transferToken(address _token, uint256 value, address _to) external onlyOwner nonReentrant {
IERC20Detailed(_token).safeTransfer(_to, value);
```

```
186 }
187 }
```

Analysis de4514dc-fb3c-4c13-8bf1-e0b005007f93 MythX

Started	Mon Jun 28 2021 16:39:30 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jun 28 2021 17:24:54 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/IdleCDOTrancheRewards.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	8

ISSUES

LOW

SWC-107

Read of persistent state following external call.

The contract account state is accessed after an external call. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

contracts/IdleCDOTrancheRewards.sol

Locations

```
52 | usersStakes[msg.sender] += _amount;
53 | IERC20Detailed(tranche).safeTransferFrom(msg.sender, address(this), _amount);
54 | totalStaked += _amount;
55 | }
```

LOW

SWC-107

Read of persistent state following external call.

The contract account state is accessed after an external call. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

contracts/IdleCDOTrancheRewards.sol

Locations

```
76 | usersStakes[msg.sender] -= _amount;
77 | IERC20Detailed(tranche).safeTransfer(msg.sender, _amount);
78 | totalStaked -= _amount;
79 | }
```

LOW

SWC-107

Write to persistent state following external call.

The contract account state is accessed after an external call. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol

Locations

```
63  // By storing the original value once again, a refund is triggered (see
64  // https://eips.ethereum.org/EIPS/eip-2200)
65  _status = _NOT_ENTERED;
66  }
67  uint256[49] private __gap;
```

LOW

SWC-107

Read of persistent state following external call.

The contract account state is accessed after an external call. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

contracts/IdleCDOTrancheRewards.sol

Locations

```
118 // Get rewards from CDO
119 IERC20Detailed(_reward).safeTransferFrom(msg.sender, address(this), _amount);
120 if (totalStaked > 0) {
121 // rewards are splitted among all stakers
122 rewardsIndexes[_reward] += _amount * ONE_TRANCHE_TOKEN / totalStaked;
```

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

contracts/IdleCDOTrancheRewards.sol

Locations

```
47  /// @param _amount The amount of tranche tokens to stake
48  function stake(uint256 _amount) external whenNotPaused override {
49  usersStakeBlock[msg.sender] = block.number; //@audit adding to stake will restart the stake block, user must wait "coolingPeriod" to be able to unstake
50  // update user index for each reward
51  _updateUserIdx(msg.sender, _amount);
```

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

contracts/IdleCDOTrancheRewards.sol

Locations

```
58  /// @param _amount The amount to unstake
59  function unstake(uint256 _amount) external nonReentrant override {
60  require(usersStakeBlock[msg.sender] + coolingPeriod < block.number, "COOLING_PERIOD");
61
62  if (paused()) {
```

LOW

SWC-120

A control flow decision is made based on The block.number environment variable.

The block.number environment variable is used to determine a control flow decision. 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

contracts/IdleCDOTrancheRewards.sol

Locations

```
58  /// @param _amount The amount to unstake
59  function unstake(uint256 _amount) external nonReentrant override {
60  require(usersStakeBlock[msg.sender] + coolingPeriod < block.number, "COOLING_PERIOD");
61
62  if (paused()) {
```

LOW

Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

SWC-123

Source file

node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol

Locations

```
117
118 // solhint-disable-next-line avoid-low-level-calls
119 (bool success, bytes memory returndata) = target.call{ value: value }(data);
120 return _verifyCallResult(success, returndata, errorMessage);
121 }
```

Source file

contracts/IdleCDOTrancheRewards.sol

Locations

```
18 /// @title IdleCDOTrancheRewards
19 /// @dev Contract used for staking specific tranche tokens and getting incentive rewards
20 contract IdleCDOTrancheRewards is Initializable, PausableUpgradeable, OwnableUpgradeable, ReentrancyGuardUpgradeable, IIdleCDOTrancheRewards, IdleCDOTrancheRewardsStorage {
21 using SafeERC20Upgradeable for IERC20Detailed;
22
23 /// @notice Initialize the contract
24 /// @param _trancheToken tranche address
25 /// @param _rewards The rewards tokens
26 /// @param _owner The owner of the contract
27 /// @param _idleCDO The CDO where the reward tokens come from
28 /// @param _governanceRecoveryFund address where rewards will be sent in case of transferToken call
29 function initialize(
30 address _trancheToken, address[] memory _rewards, address _owner,
31 address _idleCDO, address _governanceRecoveryFund, uint256 _coolingPeriod
32 ) public initializer {
33 OwnableUpgradeable.__Ownable_init();
34 ReentrancyGuardUpgradeable.__ReentrancyGuard_init();
35 PausableUpgradeable.__Pausable_init();
36
37 transferOwnership(_owner);
38
39 idleCDO = _idleCDO;
40 tranche = _trancheToken;
41 rewards = _rewards;
42 governanceRecoveryFund = _governanceRecoveryFund;
43 coolingPeriod = _coolingPeriod;
44 }
45
46 /// @notice Stake _amount of tranche token
47 /// @param _amount The amount of tranche tokens to stake
48 function stake(uint256 _amount) external whenNotPaused override {
49 usersStakeBlock[msg.sender] = block.number; //@audit adding to stake will restart the stake block, user must wait "coolingPeriod" to be able to unstake
50 // update user index for each reward
51 _updateUserIdx(msg.sender, _amount);
52 usersStakes[msg.sender] += _amount;
53 IERC20Detailed(tranche).safeTransferFrom(msg.sender, address(this), _amount);
54 totalStaked += _amount;
55 }
56
57 /// @notice Unstake _amount of tranche tokens
58 /// @param _amount The amount to unstake
59 function unstake(uint256 _amount) external nonReentrant override {
60 require(usersStakeBlock[msg.sender] + coolingPeriod < block.number, "COOLING_PERIOD");
61
62 if (paused()) {
```

```
// If the contract is paused, "unstake" will skip the claim of the rewards,
// and those rewards won't be claimable in the future.
address reward;
for (uint256 i = 0; i < rewards.length; i++) {
reward = rewards[i];
usersIndexes[msg.sender][reward] = rewardsIndexes[reward];
}
} else {
// Claim all rewards accrued
_claim();
}

// if _amount is greater than usersStakes[msg.sender], the next line fails
usersStakes[msg.sender] -= _amount;
IERC20Detailed(tranche).safeTransfer(msg.sender, _amount);
totalStaked -= _amount;
}

/// @notice Sends all the expected rewards to the msg.sender
/// @dev User index is reset
function claim() whenNotPaused nonReentrant external {
_claim();
}

/// @notice Claim all rewards, used by claim and unstake
function _claim() internal {
address[] memory _rewards = rewards;
for (uint256 i = 0; i < _rewards.length; i++) {
address reward = _rewards[i];
uint256 amount = expectedUserReward(msg.sender, reward);
uint256 balance = IERC20Detailed(reward).balanceOf(address(this));
if (amount > balance) {
amount = balance;
}
// Set the user address equal to the global one
usersIndexes[msg.sender][reward] = rewardsIndexes[reward];
IERC20Detailed(reward).safeTransfer(msg.sender, amount);
}
}

/// @notice Calculates the expected rewards for a user
/// @param user The user address
/// @param reward The reward token address
/// @return The expected reward amount
function expectedUserReward(address user, address reward) public view returns(uint256) {
require(_includesAddress(rewards, reward), "!SUPPORTED");
return ((rewardsIndexes[reward] - usersIndexes[user][reward]) * usersStakes[user]) / ONE_TRANCHE_TOKEN;
}

/// @notice Called by the CDO to deposit rewards
/// @param _reward The rewards token address
/// @param _amount The amount to deposit
function depositReward(address _reward, uint256 _amount) external override {
require(msg.sender == idleCDO, "!AUTH");
require(_includesAddress(rewards, _reward), "!SUPPORTED");
// Get rewards from CDO
IERC20Detailed(_reward).safeTransferFrom(msg.sender, address(this), _amount);
if (totalStaked > 0) {
// rewards are splitted among all stakers
rewardsIndexes[_reward] += _amount * ONE_TRANCHE_TOKEN / totalStaked;
}
}
/// @notice It sets the coolingPeriod value
```

```
/// @param _newCoolingPeriod The new cooling period
function setCoolingPeriod(uint256 _newCoolingPeriod) external onlyOwner {
coolingPeriod = _newCoolingPeriod;
}

/// @notice Update user indexes based on the amount being staked
/// @param _user The user who is staking
/// @param _amountToStake The amount staked
function _updateUserIdx(address _user, uint256 _amountToStake) internal {
address[] memory _rewards = rewards;
uint256 userIndex;
address reward;
uint256 _currStake = usersStakes[_user];

for (uint256 i = 0; i < _rewards.length; i++) {
reward = _rewards[i];
if (_currStake == 0) {
// Set the user address equal to the global one
usersIndexes[_user][reward] = rewardsIndexes[reward];
} else {
userIndex = usersIndexes[_user][reward];
// Calculate the new user idx
usersIndexes[_user][reward] = userIndex + (
_amountToStake * (rewardsIndexes[reward] - userIndex) / (_currStake + _amountToStake)
);
}
}
}

/// @dev this method is only used to check whether a token is an incentive tokens or not
/// in the harvest call. The maximum number of element in the array will be a small number (eg at most 3-5)
/// @param _array array of addresses to search for an element
/// @param _val address of an element to find
/// @return flag if the _token is an incentive token or not
function _includesAddress(address[] memory _array, address _val) internal pure returns (bool) {
for (uint256 i = 0; i < _array.length; i++) {
if (_array[i] == _val) {
return true;
}
}
// explicit return to fix linter
return false;
}

// Emergency method, funds gets transferred to the governanceRecoveryFund address
function transferToken(address token, uint256 value) external onlyOwner nonReentrant returns (bool) {
require(token != address(0), 'Address is 0');
IERC20Detailed(token).safeTransfer(governanceRecoveryFund, value);
return true;
}
/// @notice can be called by both the owner and the guardian
/// @dev Pauses deposits and redeems
function pause() external onlyOwner {
_pause();
}

/// @notice can be called by both the owner and the guardian
/// @dev Unpauses deposits and redeems
function unpause() external onlyOwner {
_unpause();
}
}
```

Analysis 299cd8ea-ce0b-4e04-a86c-0382060c10c6 MythX

Started	Mon Jun 28 2021 16:39:30 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jun 28 2021 17:24:42 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Mocks/MockIdleCDO.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	1	5

ISSUES

MEDIUM

Multiple calls are executed in the same transaction.

SWC-113

This call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently. This might be caused intentionally by a malicious callee. If possible, refactor the code such that each transaction only executes one external call or make sure that all callees can be trusted (i.e. they're part of your own codebase).

Source file

contracts/mocks/MockIdleCDO.sol

Locations

```
26
27 function depositRewardWithoutApprove(address _reward, uint256 _amount) external {
28 IIdleCDOTrancheRewards(trancheRewardsContract).depositReward(_reward, _amount);
29 }
30 }
```

LOW

Function visibility is not set.

SWC-100

The function definition of "null" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source file

contracts/mocks/MockIdleCDO.sol

Locations

```
12 address public trancheRewardsContract;
13
14 constructor(address[] memory _rewards) {
15 rewards = _rewards;
16 }
17
18 function setTrancheRewardsContract(address a) external {
```

LOW

SWC-107

Read of persistent state following external call.

The contract account state is accessed after an external call. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

contracts/mocks/MockIdleCDO.sol

Locations

```
26
27 function depositRewardWithoutApprove(address _reward, uint256 _amount) external {
28 IIdleCDOTrancheRewards(trancheRewardsContract).depositReward(_reward, _amount);
29 }
30 }
```

LOW

SWC-107

A call to a user-supplied address is executed.

An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.

Source file

contracts/mocks/MockIdleCDO.sol

Locations

```
26
27 function depositRewardWithoutApprove(address _reward, uint256 _amount) external {
28 IIdleCDOTrancheRewards(trancheRewardsContract).depositReward(_reward, _amount);
29 }
30 }
```

LOW

SWC-123

Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

Source file

node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol

Locations

```
38  // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
39  // solhint-disable-next-line max-line-length
40  require((value == 0) || (token.allowance(address(this), spender) == 0),
41  "SafeERC20: approve from non-zero to non-zero allowance"
42  );
```

Source file

contracts/mocks/MockIdleCDO.sol

Locations

```
6   import "../interfaces/IIdleCDOTrancheRewards.sol";
7
8   contract MockIdleCDO {
9   using SafeERC20 for IERC20;
10
11  address[] public rewards;
12  address public trancheRewardsContract;
13
14  constructor(address[] memory _rewards) {
15  rewards = _rewards;
16  }
17
18  function setTrancheRewardsContract(address a) external {
19  trancheRewardsContract = a;
20  }
21
22  function depositReward(address _reward, uint256 _amount) external {
23  IERC20(_reward).safeApprove(trancheRewardsContract, _amount);
24  IIdleCDOTrancheRewards(trancheRewardsContract).depositReward(_reward, _amount);
25  }
26
27  function depositRewardWithoutApprove(address _reward, uint256 _amount) external {
28  IIdleCDOTrancheRewards(trancheRewardsContract).depositReward(_reward, _amount);
29  }
30  }
```

LOW

SWC-123

Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

Source file

contracts/mocks/MockIdleCDO.sol

Locations

```
26
27 function depositRewardWithoutApprove(address _reward, uint256 _amount) external {
28 IIdleCDOTrancheRewards(trancheRewardsContract).depositReward(_reward, _amount);
29 }
30 }
```

Source file

node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol

Locations

```
38 // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
39 // solhint-disable-next-line max-line-length
40 require((value == 0) || (token.allowance(address(this), spender) == 0),
41 "SafeERC20: approve from non-zero to non-zero allowance"
42 );
43 _callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, value));
44 }
```

Analysis 88b970cd-83bf-41ea-af10-295763022d07 MythX

Started	Mon Jun 28 2021 16:39:30 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jun 28 2021 17:24:48 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Mocks/MockERC20.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	11	4

ISSUES

MEDIUM Function could be marked as external.

SWC-000 The function definition of "name" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file
node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol
Locations

```
57  * @dev Returns the name of the token.
58  */
59 function name() public view virtual returns (string memory) {
60 return _name;
61 }
62
63 /**
```

MEDIUM Function could be marked as external.

SWC-000 The function definition of "symbol" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file
node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol
Locations

```
65  * name.
66  */
67 function symbol() public view virtual returns (string memory) {
68 return _symbol;
69 }
70
71 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "decimals" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
82 * {IERC20-balanceOf} and {IERC20-transfer}.
83 */
84 function decimals() public view virtual returns (uint8) {
85 return 18;
86 }
87
88 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "totalSupply" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
89 * @dev See {IERC20-totalSupply}.
90 */
91 function totalSupply() public view virtual override returns (uint256) {
92 return _totalSupply;
93 }
94
95 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "balanceOf" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
96 * @dev See {IERC20-balanceOf}.
97 */
98 function balanceOf(address account) public view virtual override returns (uint256) {
99 return _balances[account];
100 }
101
102 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "transfer" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
108 * - the caller must have a balance of at least `amount`.
109 */
110 function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
111 _transfer(_msgSender(), recipient, amount);
112 return true;
113 }
114
115 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "allowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
116 * @dev See {IERC20-allowance}.
117 */
118 function allowance(address owner, address spender) public view virtual override returns (uint256) {
119 return _allowances[owner][spender];
120 }
121
122 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "approve" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
127 * - `spender` cannot be the zero address.
128 */
129 function approve(address spender, uint256 amount) public virtual override returns (bool) {
130 _approve(_msgSender(), spender, amount);
131 return true;
132 }
133
134 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
145 * `amount`.
146 */
147 function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
148 _transfer(sender, recipient, amount);
149
150 uint256 currentAllowance = _allowances[sender][_msgSender()];
151 require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
152 _approve(sender, _msgSender(), currentAllowance - amount);
153
154 return true;
155 }
156
157 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "increaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
167 * - `spender` cannot be the zero address.
168 */
169 function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
170 _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
171 return true;
172 }
173
174 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decreaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
186 * `subtractedValue`.
187 */
188 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
189 uint256 currentAllowance = _allowances[_msgSender()][spender];
190 require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
191 _approve(_msgSender(), spender, currentAllowance - subtractedValue);
192
193 return true;
194 }
195
196 /**
```

LOW

SWC-100

Function visibility is not set.

The function definition of "null" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source file

contracts/mocks/MockERC20.sol

Locations

```
7  address public minter;
8
9  constructor(
10 string memory _name, // eg. IdleDAI
11 string memory _symbol // eg. IDLEDAI
12 ) ERC20(_name, _symbol) {
13 minter = msg.sender;
14 _mint(msg.sender, 10000000 * 10**18); // 10M to creator
15 }
16 }
```

LOW

SWC-131

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
300 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
301 */
302 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
303 }
```

LOW

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

SWC-131

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
300 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
301 */
302 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
303 }
```

LOW

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

SWC-131

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
300 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
301 */
302 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
303 }
```

Analysis afec93cd-26d6-4c01-bd47-324bb3ce6b5e MythX

Started	Mon Jun 28 2021 16:39:40 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jun 28 2021 17:25:16 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Mocks/MockIdleToken.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	12	4

ISSUES

MEDIUM Function could be marked as external.

SWC-000 The function definition of "name" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file
node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol
Locations

```
57 * @dev Returns the name of the token.
58 */
59 function name() public view virtual returns (string memory) {
60 return _name;
61 }
62
63 /**
```

MEDIUM Function could be marked as external.

SWC-000 The function definition of "symbol" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file
node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol
Locations

```
65 * name.
66 */
67 function symbol() public view virtual returns (string memory) {
68 return _symbol;
69 }
70
71 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "decimals" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
82 * {IERC20-balanceOf} and {IERC20-transfer}.
83 */
84 function decimals() public view virtual returns (uint8) {
85 return 18;
86 }
87
88 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "totalSupply" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
89 * @dev See {IERC20-totalSupply}.
90 */
91 function totalSupply() public view virtual override returns (uint256) {
92 return _totalSupply;
93 }
94
95 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "balanceOf" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
96 * @dev See {IERC20-balanceOf}.
97 */
98 function balanceOf(address account) public view virtual override returns (uint256) {
99 return _balances[account];
100 }
101
102 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "transfer" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
108 * - the caller must have a balance of at least `amount`.
109 */
110 function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
111 _transfer(_msgSender(), recipient, amount);
112 return true;
113 }
114
115 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "allowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
116 * @dev See {IERC20-allowance}.
117 */
118 function allowance(address owner, address spender) public view virtual override returns (uint256) {
119 return _allowances[owner][spender];
120 }
121
122 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "approve" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
127 * - `spender` cannot be the zero address.
128 */
129 function approve(address spender, uint256 amount) public virtual override returns (bool) {
130 _approve(_msgSender(), spender, amount);
131 return true;
132 }
133
134 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
145 * `amount`.
146 */
147 function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
148 _transfer(sender, recipient, amount);
149
150 uint256 currentAllowance = _allowances[sender][_msgSender()];
151 require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
152 _approve(sender, _msgSender(), currentAllowance - amount);
153
154 return true;
155 }
156
157 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "increaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
167 * - `spender` cannot be the zero address.
168 */
169 function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
170 _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
171 return true;
172 }
173
174 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decreaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
186 * `subtractedValue`.
187 */
188 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
189 uint256 currentAllowance = _allowances[_msgSender()][spender];
190 require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
191 _approve(_msgSender(), spender, currentAllowance - subtractedValue);
192
193 return true;
194 }
195
196 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "token" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/mocks/MockIdleToken.sol

Locations

```
51 return _tokenPriceWithFee;
52 }
53 function token() public view returns(address) {
54 return underlying;
55 }
56
57 function mintIdleToken(uint256 _amount, bool, address) external returns(uint256) {
```

LOW

SWC-100

Function visibility is not set.

The function definition of "null" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source file

contracts/mocks/MockIdleToken.sol

Locations

```
17 bool public transferGovTokens;
18
19 constructor(address _underlying)
20 MockERC20('IDLEDAI', 'IDLEDAI') {
21 underlying = _underlying;
22 _tokenPriceWithFee = 10**18;
23 }
24
25 function setGovTokens(address[] memory _govTokens) external {
```

LOW

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

SWC-131

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
300 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
301 */
302 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
303 }
```

LOW

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

SWC-131

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
300 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
301 */
302 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
303 }
```

LOW

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

SWC-131

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
300 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
301 */
302 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
303 }
```