ArthPool Smart Contract Preliminary Audit Report

Project Synopsis

Project Name	**MahaDao Audit**
Platform	Ethereum, Solidity
Github Repo	https://github.com/MahaDAO/arthcoin-v2/blob/master/contracts/Arth/Pools/ArthPool.sol
Deployed Contract	Not Deployed
Total Duration	15 Days
Timeline of Audit	**15th April 2021 to 02nd April 2021**

Contract Details

Total Contract(s)	1
Name of Contract(s)	**ArthPool**
Language	Solidity
Commit Hash	**6bac5389fbe25316614ceb10ee230aabb4dd07ff**

Contract Vulnerabilities Synopsis

Issues	Open Issues	Closed Issues
Critical Severity	1	0
Medium Severity	2	0
Low Severity	4	0
Informational	2	0
Total Found	9	0

Detailed Results

The contract has gone through several stages of the audit procedure that includes structural analysis, automated testing, manual code review, etc.

All the issues have been explained and discussed in detail below. Along with the explanation of the issue found during the audit, the recommended way to overcome the issue or improve the code quality has also been mentioned.

A. Contract Name: ArthPool

<u>High Severity Issues</u>

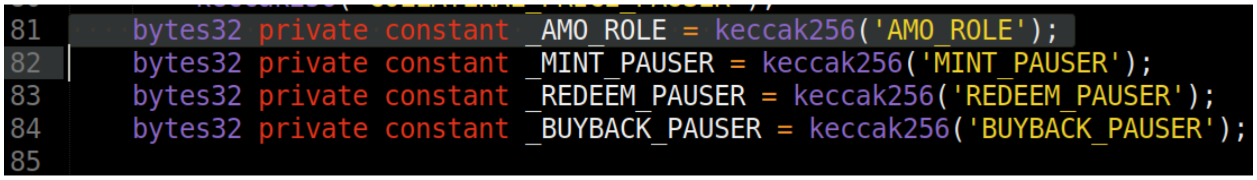
A.1 "<u>_AMO_ROLE</u>" is initialized but never assigned to any address _ Functions with <u>onlyAMOS</u> modifier are completely inaccessible

Line no: 81, 134-137

Description:

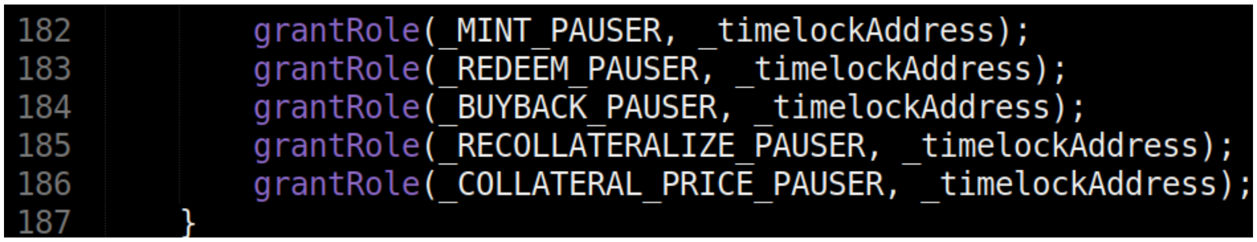
The ArthPool contract uses **AccessControl** to assign specific roles in the contract to particular addresses.

At Line 89, it initializes the **AMO_ROLE** along with other roles.



```
81      bytes32 private constant _AMO_ROLE = keccak256('AMO_ROLE');
82      bytes32 private constant _MINT_PAUSER = keccak256('MINT_PAUSER');
83      bytes32 private constant _REDEEM_PAUSER = keccak256('REDEEM_PAUSER');
84      bytes32 private constant _BUYBACK_PAUSER = keccak256('BUYBACK_PAUSER');
85
```

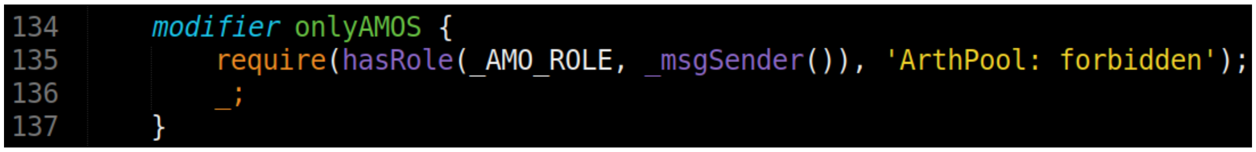
However, while assigning these roles to a specific address using **grantRole() function** in the constructor (Line *182-187*), no address is assigned for the **_AMO_ROLE.**



```
182             grantRole(_MINT_PAUSER, _timelockAddress);
183             grantRole(_REDEEM_PAUSER, _timelockAddress);
184             grantRole(_BUYBACK_PAUSER, _timelockAddress);
185             grantRole(_RECOLLATERALIZE_PAUSER, _timelockAddress);
186             grantRole(_COLLATERAL_PRICE_PAUSER, _timelockAddress);
187         }
```

Moreover, the **AMO_ROLE** is used in the modifier ***<u>onlyAMOS</u>*** which is assigned to the following the functions in the contract:

- ***borrow** at **Line 338***
- ***repay** at **Line 350***



```
134     modifier onlyAMOS {
135         require(hasRole(_AMO_ROLE, _msgSender()), 'ArthPool: forbidden');
136         _;
137     }
```

This ***onlyAMOS*** modifier ensures that only those addresses that have been assigned the **AMO_ROLE** should be able to call the function**.**

However, since the AMO_ROLE is **not assigned t**o any address in the constructor, this will lead to an extremely critical issue where the function **borrow** and **repay** will become completely inaccessible and can never be executed as they have the ***onlyAMOS modifier*** *attached to them.*

Recommendation:
The **_AMO_ROLE** must be assigned to a particular address in the constructor in order to avoid the above-mentioned scenario.

<u>Medium Severity Issues</u>

A.2 Contract State Variables are being updated after External Calls.

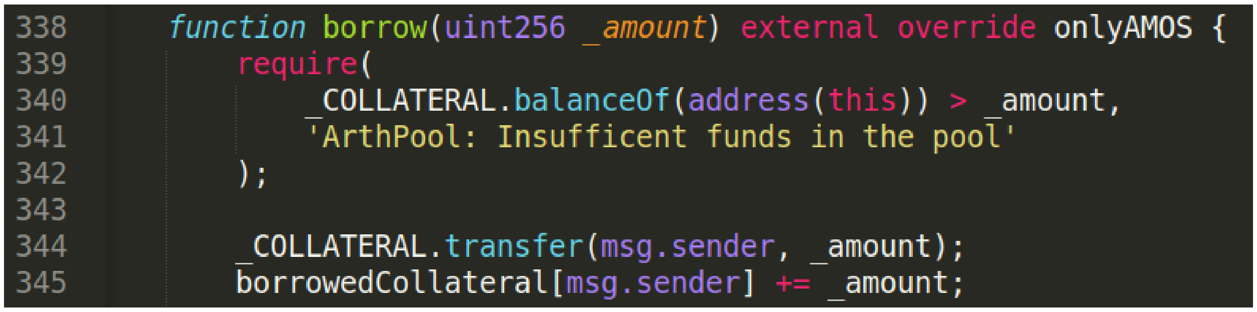
Line no - 344-345, 360-362

Explanation:
The **ArthPool** contract includes quite a few functions that update some of the very imperative state variables of the contract after the external calls are being made. According to the Check_Effects_Interaction Pattern in Solidity, external calls should be made at the very end of the function and event emission, as well as any state variable modification, must be done before the external call is made.

Therefore, as per the Solidity Guidelines, any modification of the state variables in the base contract must be performed before executing the external call.
Updating state variables after an external call might lead to a potential re-entrancy scenario. Although the call is made to the Collateral address itself, the check-effects-interaction pattern must not be violated.

The following functions in the contract updates the state variables and emits events after making an external call:

- ***borrow() function*** *at Line* ***344-345***
- ***repay() function*** *at Line* ***360-362***



```
338     function borrow(uint256 _amount) external override onlyAMOS {
339         require(
340             _COLLATERAL.balanceOf(address(this)) > _amount,
341             'ArthPool: Insufficent funds in the pool'
342         );
343
344         _COLLATERAL.transfer(msg.sender, _amount);
345         borrowedCollateral[msg.sender] += _amount;
```

Recommendation:
Modification of any State Variables must be performed before making an external call. **Check Effects Interaction Pattern** must be followed while implementing external calls in a function.

A.3 Multiplication is being performed on the result of Division

Line no - 562-584, 626-632, 821-829

***Explanation*:**
During the automated testing of the **ArthPool** contract, it was found that some of the functions in the contract are performing multiplication on the result of a Division. Integer Divisions in Solidity might truncate. Moreover,performing division before multiplication might lead to loss of precision.

The following functions involve division before multiplication in the mentioned lines:

- ***redeemFractionalARTH*** *at Line 562-584*
- ***getCollateralGMUBalance*** *at Line 626-632*
- ***estimateStabilityFeeInMAHA*** *at Line 821-829*

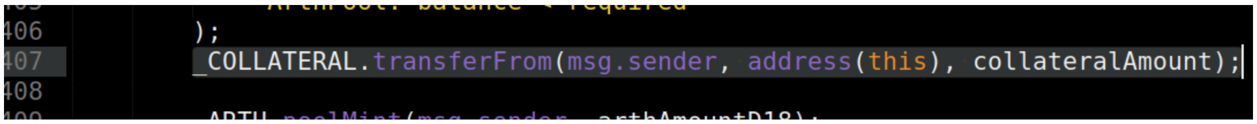
Recommendation:
Solidity doesn't encourage arithmetic operations that involve division before multiplication. Therefore the above-mentioned function should be checked once and redesigned if they do not lead to expected results.

<u>Low Severity Issues</u>

A.4 Return Value of an External Call is never used Effectively

Line no -344, 360, 407, 489, 683, 685, 732, 775
Explanation:
The external calls made in the above-mentioned lines return a boolean value that indicates whether or not the external call made was successful.
These boolean return values can be used in the function as a check to ensure that the execution of an external call was successful.

However, the **ArthPool** contract does not use these return values throughout the contract.



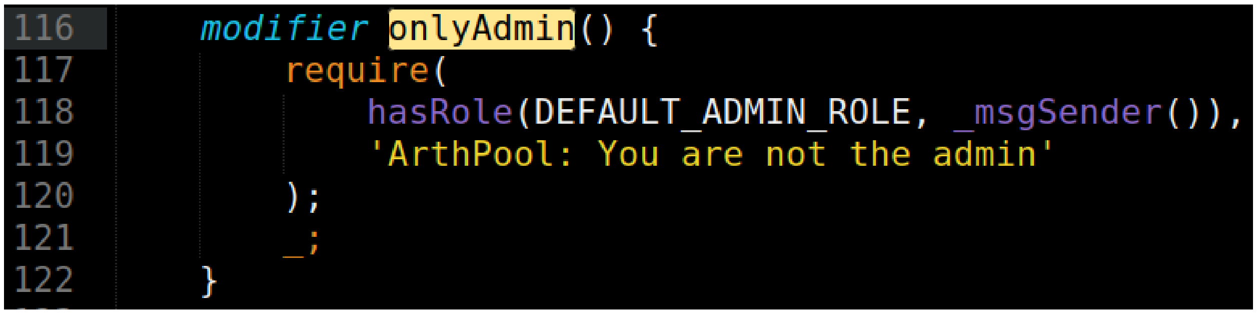
```
        );
        _COLLATERAL.transferFrom(msg.sender, address(this), collateralAmount);
```

Recommendation:
Effective use of all the return values from external calls must be ensured within the contract.

A.5 Modifier *__onlyAdmin__* never used in the ArthPool Contract

Line no - 116

Description:
The ArthPool contract includes the ***__onlyAdmin__*** modifier at Line 116 but never uses it throughout the contract.



```
116     modifier onlyAdmin() {
117         require(
118             hasRole(DEFAULT_ADMIN_ROLE, _msgSender()),
119             'ArthPool: You are not the admin'
120         );
121         _;
122     }
```

While this consumes additional space in the contract, it also adversely affects the gas optimization as well as the readability of the smart contract code.

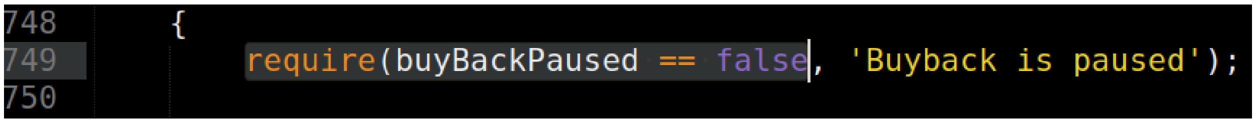
Recommendation:
Adequate use of all State Variable, modifiers, mappings etc must be ensured in the contract. If the ***onlyAdmin*** modifier holds no significance it should be removed from the contract.

A.6 Comparison to boolean Constant

Line no: 299, 683,684,697, 749, 140, 144

Description:
Boolean constants can directly be used in conditional statements or require statements.

Therefore, it's not considered a better practice to explicitly use **TRUE or FALSE** in the **require** or **IF-Else** statements.



```
    {
        require(buyBackPaused == false, 'Buyback is paused');
```

Recommendation:
The equality to boolean constants must be removed from the above-mentioned line.

A.7 External Visibility should be preferred

Explanation:

Those functions that are never called throughout the contract should be marked as ***external*** visibility instead of ***public*** visibility.
This will effectively result in Gas Optimization as well.

Therefore, the following function must be marked as **external** within the contract:

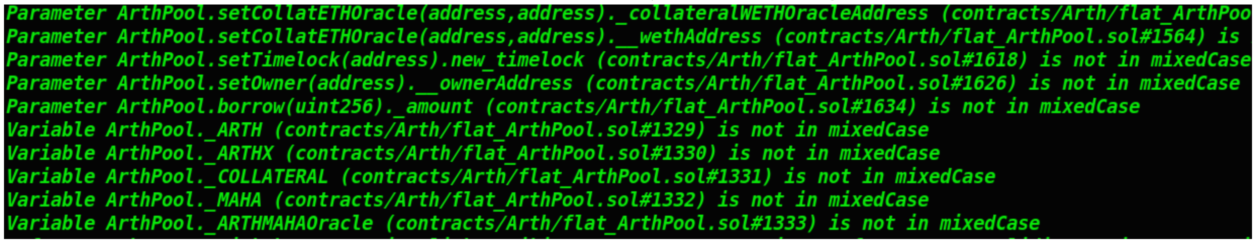
- ***getCollateralGMUBalance*** *at Line 801*

Recommendations:
If the **public** visibility of the above-mentioned is not intended, the function should be assigned an **external** keyword.

<u>Informational</u>

A.8 Coding Style Issues in the Contract

Explanation:
Code readability of a Smart Contract is largely influenced by the Coding Style issues and in some specific scenarios may lead to bugs in the future.



```
Parameter ArthPool.setCollatETHOracle(address,address)._collateralWETHOracleAddress (contracts/Arth/flat_ArthPoo
Parameter ArthPool.setCollatETHOracle(address,address).__wethAddress (contracts/Arth/flat_ArthPool.sol#1564) is
Parameter ArthPool.setTimelock(address).new_timelock (contracts/Arth/flat_ArthPool.sol#1618) is not in mixedCase
Parameter ArthPool.setOwner(address).__ownerAddress (contracts/Arth/flat_ArthPool.sol#1626) is not in mixedCase
Parameter ArthPool.borrow(uint256)._amount (contracts/Arth/flat_ArthPool.sol#1634) is not in mixedCase
Variable ArthPool._ARTH (contracts/Arth/flat_ArthPool.sol#1329) is not in mixedCase
Variable ArthPool._ARTHX (contracts/Arth/flat_ArthPool.sol#1330) is not in mixedCase
Variable ArthPool._COLLATERAL (contracts/Arth/flat_ArthPool.sol#1331) is not in mixedCase
Variable ArthPool._MAHA (contracts/Arth/flat_ArthPool.sol#1332) is not in mixedCase
Variable ArthPool._ARTHMAHAOracle (contracts/Arth/flat_ArthPool.sol#1333) is not in mixedCase
```

During the automated testing, it was found that the ArthPool contract had quite a few code style issues.

Recommendation:

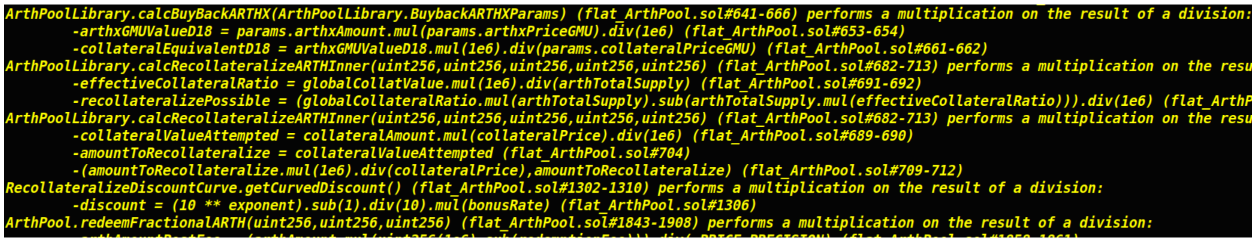
Therefore, it is highly recommended to fix the issues like naming convention, indentation, and code layout issues in a smart contract.

A.9 NatSpec Annotations must be included

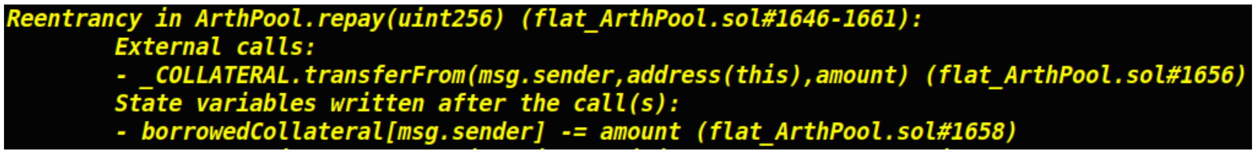
Description:
The smart contracts do not include the NatSpec annotations adequately.

Recommendation:
Cover by NatSpec all Contract methods.

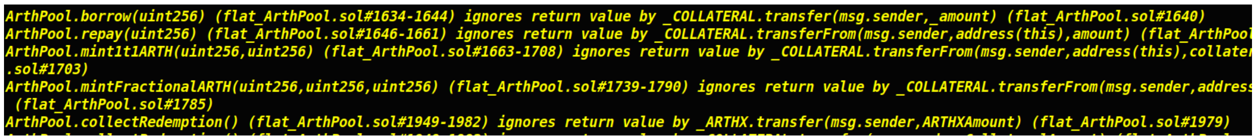
Automated Test Results



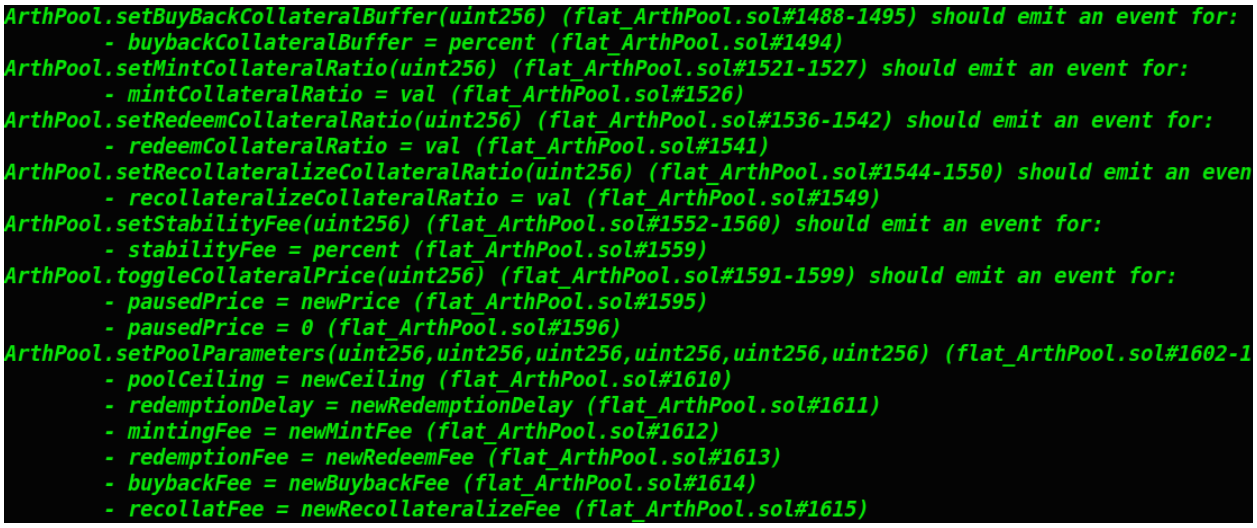
```
ArthPoolLibrary.calcBuyBackARTHX(ArthPoolLibrary.BuybackARTHXParams) (flat_ArthPool.sol#641-666) performs a multiplication on the result of a division:
        -arthxGMUValueD18 = params.arthxAmount.mul(params.arthxPriceGMU).div(1e6) (flat_ArthPool.sol#653-654)
        -collateralEquivalentD18 = arthxGMUValueD18.mul(1e6).div(params.collateralPriceGMU) (flat_ArthPool.sol#661-662)
ArthPoolLibrary.calcRecollateralizeARTHInner(uint256,uint256,uint256,uint256,uint256) (flat_ArthPool.sol#682-713) performs a multiplication on the resu
        -effectiveCollateralRatio = globalCollatValue.mul(1e6).div(arthTotalSupply) (flat_ArthPool.sol#691-692)
        -recollateralizePossible = (globalCollateralRatio.mul(arthTotalSupply).sub(arthTotalSupply.mul(effectiveCollateralRatio))).div(1e6) (flat_ArthP
ArthPoolLibrary.calcRecollateralizeARTHInner(uint256,uint256,uint256,uint256,uint256) (flat_ArthPool.sol#682-713) performs a multiplication on the resu
        -collateralValueAttempted = collateralAmount.mul(collateralPrice).div(1e6) (flat_ArthPool.sol#689-690)
        -amountToRecollateralize = collateralValueAttempted (flat_ArthPool.sol#704)
        -(amountToRecollateralize.mul(1e6).div(collateralPrice),amountToRecollateralize) (flat_ArthPool.sol#709-712)
RecollateralizeDiscountCurve.getCurvedDiscount() (flat_ArthPool.sol#1302-1310) performs a multiplication on the result of a division:
        -discount = (10 ** exponent).sub(1).div(10).mul(bonusRate) (flat_ArthPool.sol#1306)
ArthPool.redeemFractionalARTH(uint256,uint256,uint256) (flat_ArthPool.sol#1843-1908) performs a multiplication on the result of a division:
```



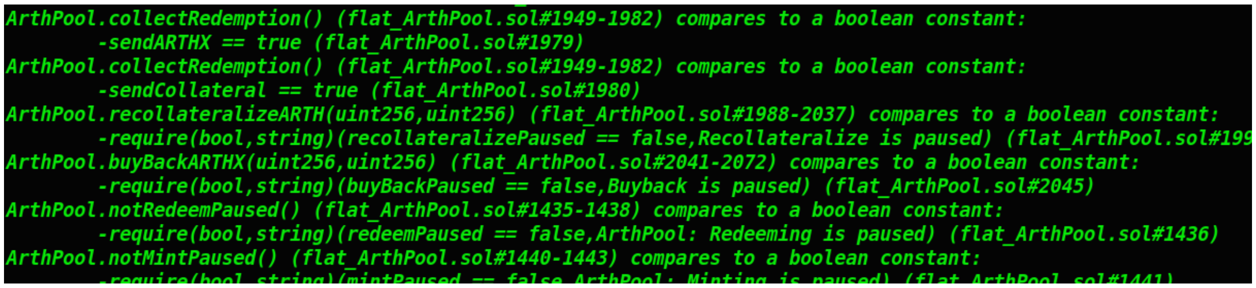
```
Reentrancy in ArthPool.repay(uint256) (flat_ArthPool.sol#1646-1661):
        External calls:
        - _COLLATERAL.transferFrom(msg.sender,address(this),amount) (flat_ArthPool.sol#1656)
        State variables written after the call(s):
        - borrowedCollateral[msg.sender] -= amount (flat_ArthPool.sol#1658)
```



```
ArthPool.borrow(uint256) (flat_ArthPool.sol#1634-1644) ignores return value by _COLLATERAL.transfer(msg.sender,_amount) (flat_ArthPool.sol#1640)
ArthPool.repay(uint256) (flat_ArthPool.sol#1646-1661) ignores return value by _COLLATERAL.transferFrom(msg.sender,address(this),amount) (flat_ArthPool
ArthPool.mint1t1ARTH(uint256,uint256) (flat_ArthPool.sol#1663-1708) ignores return value by _COLLATERAL.transferFrom(msg.sender,address(this),collater
.sol#1703)
ArthPool.mintFractionalARTH(uint256,uint256,uint256) (flat_ArthPool.sol#1739-1790) ignores return value by _COLLATERAL.transferFrom(msg.sender,address
 (flat_ArthPool.sol#1785)
ArthPool.collectRedemption() (flat_ArthPool.sol#1949-1982) ignores return value by _ARTHX.transfer(msg.sender,ARTHXAmount) (flat_ArthPool.sol#1979)
```



```
ArthPool.setBuyBackCollateralBuffer(uint256) (flat_ArthPool.sol#1488-1495) should emit an event for:
        - buybackCollateralBuffer = percent (flat_ArthPool.sol#1494)
ArthPool.setMintCollateralRatio(uint256) (flat_ArthPool.sol#1521-1527) should emit an event for:
        - mintCollateralRatio = val (flat_ArthPool.sol#1526)
ArthPool.setRedeemCollateralRatio(uint256) (flat_ArthPool.sol#1536-1542) should emit an event for:
        - redeemCollateralRatio = val (flat_ArthPool.sol#1541)
ArthPool.setRecollateralizeCollateralRatio(uint256) (flat_ArthPool.sol#1544-1550) should emit an even
        - recollateralizeCollateralRatio = val (flat_ArthPool.sol#1549)
ArthPool.setStabilityFee(uint256) (flat_ArthPool.sol#1552-1560) should emit an event for:
        - stabilityFee = percent (flat_ArthPool.sol#1559)
ArthPool.toggleCollateralPrice(uint256) (flat_ArthPool.sol#1591-1599) should emit an event for:
        - pausedPrice = newPrice (flat_ArthPool.sol#1595)
        - pausedPrice = 0 (flat_ArthPool.sol#1596)
ArthPool.setPoolParameters(uint256,uint256,uint256,uint256,uint256,uint256) (flat_ArthPool.sol#1602-1
        - poolCeiling = newCeiling (flat_ArthPool.sol#1610)
        - redemptionDelay = newRedemptionDelay (flat_ArthPool.sol#1611)
        - mintingFee = newMintFee (flat_ArthPool.sol#1612)
        - redemptionFee = newRedeemFee (flat_ArthPool.sol#1613)
        - buybackFee = newBuybackFee (flat_ArthPool.sol#1614)
        - recollatFee = newRecollateralizeFee (flat_ArthPool.sol#1615)
```



```
ArthPool.collectRedemption() (flat_ArthPool.sol#1949-1982) compares to a boolean constant:
        -sendARTHX == true (flat_ArthPool.sol#1979)
ArthPool.collectRedemption() (flat_ArthPool.sol#1949-1982) compares to a boolean constant:
        -sendCollateral == true (flat_ArthPool.sol#1980)
ArthPool.recollateralizeARTH(uint256,uint256) (flat_ArthPool.sol#1988-2037) compares to a boolean constant:
        -require(bool,string)(recollateralizePaused == false,Recollateralize is paused) (flat_ArthPool.sol#199
ArthPool.buyBackARTHX(uint256,uint256) (flat_ArthPool.sol#2041-2072) compares to a boolean constant:
        -require(bool,string)(buyBackPaused == false,Buyback is paused) (flat_ArthPool.sol#2045)
ArthPool.notRedeemPaused() (flat_ArthPool.sol#1435-1438) compares to a boolean constant:
        -require(bool,string)(redeemPaused == false,ArthPool: Redeeming is paused) (flat_ArthPool.sol#1436)
ArthPool.notMintPaused() (flat_ArthPool.sol#1440-1443) compares to a boolean constant:
```